ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING
CICLO XXXIII

Settore Concorsuale: 09/H1
Settore Scientifico Disciplinare: ING-INF/05

Deep-learning for 3D reconstruction

Presentata da: **Fabio Tosi**

Coordinatore Dottorato | Supervisore

Prof. **Davide Sangiorgi** | Prof. **Stefano Mattoccia**

Abstract

Deep-learning for 3D reconstruction

Depth perception is paramount for many computer vision applications such as autonomous driving and augmented reality. Despite active sensors (e.g., LiDAR, Time-of-Flight, structured light) are quite diffused, they have severe shortcomings that could be potentially addressed by image-based sensors. Concerning this latter category, deep learning has enabled ground-breaking results in tackling well-known issues affecting the accuracy of systems inferring depth from a single or multiple images in specific circumstances (e.g., low textured regions, depth discontinuities, etc.), but also introduced additional concerns about the domain shift occurring between training and target environments and the need of proper ground truth depth labels to be used as the training signals in network learning. Moreover, despite the copious literature concerning confidence estimation for depth from a stereo setup, inferring depth uncertainty when dealing with deep networks is still a major challenge and almost unexplored research area, especially when dealing with a monocular setup. Finally, computational complexity is another crucial aspect to be considered when targeting most practical applications and hence is desirable not only to infer reliable depth data but do so in real-time and with low power requirements even on standard embedded devices or smartphones.

Therefore, focusing on stereo and monocular setups, this thesis tackles major issues affecting methodologies to infer depth from images and aims at developing accurate and efficient frameworks for accurate 3D reconstruction on challenging environments.

Acknowledgments

I would like to acknowledge my supervisor Prof. Stefano Mattoccia for his invaluable supervision, support and patience during the course of my PhD degree. My gratitude extends to my teammates Matteo and Filippo, with whom I shared exciting years of productive research. I would like to express gratitude to all members of the CVLab for the good times spent together in the laboratory and the constructive discussions regarding computer vision. Special thanks go to Prof. Andreas Geiger for giving me the opportunity to (virtually) join and work with his fantastic team for six months in a very difficult period. I learned a lot from this great experience thanks to him and each member of the Autonomous Vision Group. My appreciation also goes out to my friends for their support all through my studies. Finally, special and profound thanks must go to my family. My parents Angelo and Ernestina for love and inspiration throughout my life, my sister Silvia for being always there for me, my uncle Valerio who encouraged me to purse my dreams, my brother-in-law Marco and my beautiful, kind, powerful nieces Gloria and Mia for their care and love.

Contents

Introduction

Since the early stages of computer vision, estimating depth from images has been one of the iconic challenges for researchers. Obtaining dense and accurate depth maps is crucial for effectively addressing higher-level tasks such as 3D reconstruction, mapping and localization, autonomous driving, and many more. Competing technologies for depth estimation rely on active sensing which comes in several forms, including structured light projection, Time-Of-Flight (ToF) measurement, Laser Imaging Detection and Ranging (LIDAR) among others. Common to these devices is the use of active illumination, required to sense depth. Although very accurate and precise, these sensors suffer from non-negligible weaknesses limiting their practical deployment for real applications. For instance, LIDAR sensors, which rely on one or more laser emitters scanning the environment through mechanical rotation, may suffer from misalignment, missing laser returns due to absorbing or reflective surfaces and multi-pathing. Moreover, they typically provide only sparse measurements of the observed scene, with density (and pricing) increasing with the number of laser emitters. For structured-light devices, such as the Microsoft Kinect, the pattern projection technology constraints the working range to a few meters and prevents usage under direct sunlight.

On the other hand, inferring depth from images acquired by a regular camera has the potential to overcome all the limitations above. Among the different techniques for this purpose, stereo matching [211] takes as input two rectified images and computes the disparity of (almost) every pixel by matching corresponding pixels along conjugate epipolar lines, thus enabling depth estimation via triangulation. Years of research proved the effectiveness of stereo, making it a viable alternative to expensive active sensors often deployed in practical applications. Although many algorithms dealt with this problem, with different degrees of effectiveness, performance in difficult environments characterized by specular or transparent surfaces, uniform regions, sun-light, etc remains an open research problems as clearly witnessed by recent datasets [66, 67, 213]. To this aim, many works focused on the formulation of meta-information capable to discriminate whether a disparity assignment has been correctly inferred by the stereo algorithm or not. Confidence measures encode this property by means of an estimated reliability score assigned to each pixel of the disparity map. The success and proliferation of machine learning and deep learning techniques in computer vision [279], led to notable improvements to both stereo matching and uncertainty estimation, even though they represent areas of computer vision in which learning was adopted relatively late. At the same time, the most recent advances in depth estimation from images have demonstrated that deep learning itself could benefit from stereo to achieve goals unimaginable just a few years ago, as in the case of single-image depth estimation, an active research topic in deep machine learning in which the goal is to learn a non-linear mapping between a single RGB image and its corresponding depth map. Even though such task represents a natural human brain activity, it is an ill-posed problem in the computer vision context, since a single 2D image may be generated from an infinite number of different 3D maps. However, unlike other multi-view scenarios, single-image depth estimation does not require any additional equipment, making it possible to apply in any devices with a single camera. While initial supervised learning approaches have enjoyed success in this task, such models typically require vast amounts of pixel-wise ground truth training annotations which are very difficult to source. Recently, unsupervised (or self-supervised) strategies have been proposed by casting depth estimation as a view synthesis problem, introducing novel photometric reconstruction loss terms to avoid the need for expensive ground truth depth, thus receiving a great deal of attention. This aim is achieved by means of extensive datasets of multi-view images depicting the same scene, i.e.stereo pairs or monocular videos. While approaches based on multiple-frame visual odometry suffers the scale ambiguity issue and need to estimate the relative camera poses, the use of stereo images constrains the scene depth to be in a real-world scale being known a priori the baseline distance between the camera. On the basis of this latter consideration, becomes clear how inferring depth without any scale ambiguity is made possible even using a single RGB image as input, but can cause issues related to occlusion and texture-copy artifacts.

Driven by these successes, in this thesis we will inquire about the application of such methodologies to low level vision problems like stereo matching and monocular depth estimation. More specifically, the topics covered in this manuscript will be a complete taxonomy of the many confidence measures present in literature exhaustively evaluated on three challenging datasets using three popular stereo algorithm (Chapter 4), an efficient implementation of some of these measures on embedded systems (Chapter 5), a deep learning methodology to locally refine already predicted confidence maps (Chapter 6), a novel multi-stage cascaded network to effectively combine global and local information of the overall image content (Chapter 7) and a deep investigation of the performance of confidence estimation methods neglecting the use of the cost volume (Chapter 8). Moreover, it will be introduced a novel confidence estimation technique for ToF data and will be shown how it can been exploited in order to guide the fusion of depth data from both passive and active sensors (Chapter 9). Furthermore, we will prove how confidence measures based on deep learning techniques can be effectively trained without the supervision of expensive ground truth depth data (Chapter 10) and in an online manner (Chapter 11). The subsequent chapters, instead, will cover a novel strategy for disparity refinement leveraging on confidence measures (Chapter 12), a novel stereo matching framework aimed at improving depth accuracy near object boundaries and suited for disparity super-resolution by means of a bimodal mixture densities as output representation combined with a continuous function formulation (Chapter 13), an unsupervised and continuous online adaptation of a deep stereo network, which allows for preserving its accuracy in any environment (Chapter 14), a deep stereo paradigm leveraging a small amount of sparse, yet reliable depth measurements retrieved from an external source (Chapter 15), a novel strategy to source reliable disparity proxy labels in order to train deep stereo networks in a self-supervised manner leveraging a monocular completion paradigm (Chapter 16) and a lightweight architecture able to infer full scene flow jointly reasoning about stereo depth and optical flow (Chapter 17). In the last part of this thesis, instead, a particular attention will be paid to the monocular depth estimation task, in which we will introduce a novel interleaved training procedure to enforce a trinocular assumption from current binocular datasets in order to train a monocular depth network and reduce depth artifacts (Chapter 18), a deep learning architecture to improve monocular depth by leveraging semantic information (Chapter 19), an unsupervised monocular depth estimation strategy within a Generative Adversarial Network (GAN) paradigm (Chapter 20), a novel architecture that exploits a more robust self-supervised training leveraging on proxy ground truth labels generated through a traditional stereo algorithm (Chapter 21) and a novel architecture capable to quickly infer an accurate depth map on a CPU, even of an embedded system (Chapter 22). Moreover, we will introduce a comprehensive design and optimization framework aimed at improving the energy efficiency of depth perception on low power devices (Chapter 23), a real-time network for comprehensive scene understanding from monocular videos (Chapter 24) and an investigation of uncertainty modelling in self-supervised monocular depth estimation (Chapter 25).

Chapter 1

Related Work

In this chapter, a thorough review of the main works relevant to this thesis will be reported.

Stereo matching. Depth from stereo images has a longstanding history in computer vision and several hand-designed methods based on some of the steps outlined in [211] have been proposed. For instance, a fast yet noisy solution can be obtained by simply matching pixels according to a robust function [276] over a fixed window (Block Matching), while a better accuracy-speed trade-off is obtained by running Semi-Global Matching (SGM) [83]. Recently, deep learning proved unpaired performance at tackling stereo correspondence [195]. Starting from matching cost computation [44, 136, 279], deep networks at first replaced single steps in the pipeline [211], moving then to optimization [217], disparity selection [219] and refinement [68]. The first end-to-end model was proposed by Mayer et al. [151], deploying a 1D correlation layer to encode pixel similarities and feed them to a 2D network. In alternative, Kendall et al. [109] stacked features to build a cost volume, processed by 3D convolutions to obtain disparity values through a differentiable *argmin* operation. These two pioneering works paved the way for more complex and effective 2D [99, 129, 167] and 3D [31, 274, 282] architectures. Finally, multi-task frameworks combining stereo with semantic segmentation [57, 262] and edge detection [225] proved to be effective as well. On the other hand, deep learning stereo methods able to learn directly from images largely alleviate the need for labels. These have been used either for domain adaptation or for training from scratch a deep stereo network. In the former case, Tonioni et al. [235, 236] leveraged traditional algorithms and confidence measures, in [238] developed a modular architecture able to be updated in real time leveraging image reprojection and in [237] made use of meta-learning for the same purpose. In the latter, an iterative schedule to train an unsupervised stereo CNN has been proposed in [291], Godard et al. [69] trained a naive stereo network using image reprojection. Zhong et al. [289] first showed the fast convergence of 3D networks when trained with image reprojection, then adopted a RNN LSTM network using stereo video sequences [290]. Wang et al. [253] improved their stereo network thanks to a rigid-aware direct visual odometry module, while in [122] the authors exploited the relationship between optical flow and stereo. Joung et al. [107] trained a network from scratch selecting good matches obtained by a pretrained model. Finally, in [224] a semi-supervised framework leveraging raw LiDAR and image reprojection has been proposed.

Confidence measures for stereo. Confidence measures were first extensively reviewed and categorized by [89], and more recently by [184] considering learning-based approaches. Both works emphasize how different cues can be taken into account to formulate a confidence score, such as: matching cost, local or global property of the cost curve, left-right consistency and others. Single confidence measures can be effectively combined with other hand-crafted features and fed to a classifier (usually, a random forest) to learn a more accurate confidence score [77]. Some works [169, 178, 194, 228] adopted this rationale deploying different features. Moreover, leveraging the learned confidence estimation, these methods enabled improvements to stereo accuracy.

Eventually, confidence estimation was tackled exploiting CNNs. Specifically, in [179] we proposed to learn from scratch a confidence measure training the CCNN network on samples extracted from the raw reference disparity map only, while [217] processing hand-crafted features extracted from the reference and target disparity maps for their *Patch Based Confidence Prediction* (PBCP) approach. In [62] the additional contextual information from the reference frame is exploited. However, this method requires a much larger training set compared to any other method discussed so far because of the increased variety of data occurring in the image domain. In [183] a comparison between random forest and CNN processing the same features is reported. Differently, arguing local consistency of confidence maps, in [116, 181] deep networks were trained to improve the overall accuracy of an input confidence map. Other effective strategies consist in combining local and global cues from both image and disparity domains as proposed by [240] or adding features computed from the cost volume as shown by [117]. An evaluation of confidence measures suited for embedded devices was proposed in [185]. Finally, [239] and [160] proposed two strategies to train confidence measures without ground truth labels.

Monocular depth estimation. At first, depth estimation was tackled as a supervised [60, 123] or semi-supervised task [121]. Nonetheless, self-supervision from image reconstruction is now becoming the preferred paradigm to avoid hard to source labels. Stereo pairs [64, 69] can provide such supervision and enable scale recovery, with further improvements achievable by leveraging on trinocular assumptions [187], proxy labels from SGM [241, 255] or guidance from visual odometry [15]. Monocular videos [293] are a more flexible alternative, although they do not allow for scale recovery and mandate learning camera pose alongside with depth. Recent developments of this paradigm deal with differentiable direct visual odometry [249] or ICP [142] and normal consistency [269]. Other works, such as [16, 43, 135, 268, 270, 296], model rigid and non-rigid components using the projected depth, relative camera transformations, and optical flow to handle independent motions, which can also be estimated independently in the 3D space [29, 261]. In [72], the authors show how to learn camera intrinsics together with depth and egomotion to enable training on any unconstrained video. In [23, 70, 292], reasoned design choices such as a minimum reprojection loss between frames, self-assembled attention modules and auto-mask strategies to handle static camera or dynamic objects proved to be very effective. Supervision from stereo and video have also been combined [70, 281], possibly improved by means of proxy supervision from stereo direct sparse odometry [265]. Uncertainty modeling for self-supervised monocular depth estimation has been studied in [190].

Finally, lightweight networks aimed at real-time performance on low-power systems have been proposed within self-supervised [173–175, 186] as well as supervised [257] learning paradigms.

Semantic segmentation. Nowadays, fully convolutional neural networks [134] are the standard approach for semantic segmentation. Within this framework, multi-scale context modules and proper architectural choices are crucial to performance. The former rely on spatial pyramid pooling [78, 287] and atrous convolutions [37, 38, 40]. As for the latter, popular backbones [79, 120, 222] have been improved by more recent designs [45, 91]. While for years the encoder-decoder architecture has been the most popular choice [17, 206], recent trends in Auto Machine Learning (AutoML) [39, 130] leverage on architectural search to achieve state-of-the-art accuracy. However, these latter have huge computational requirements. An alternative research path deals with real-time semantic segmentation networks. In this space, [170] deploys a compact and efficient network architecture, [273] proposes a two paths network to attain fast inferences while capturing high resolution details. DABNet [127] finds an effective combinations of depth-wise separable filters and atrous-convolutions to reach a good trade-off between efficiency and accuracy. [128] employs cascaded sub-stages to refine results while FCHardNet [34] leverages on a new harmonic densely connected pattern to maximize the inference performance of larger networks.

Optical flow estimation. The optical flow problem concerns estimation of the apparent displacement of pixels in consecutive frames, and it is useful in various applications such as, e.g., video editing [33, 104] and object tracking [259]. Initially introduced by Horn and Schunck [88], this problem has traditionally been tackled by variational approaches [25, 26, 204]. More recently, Dosovitskiy et al.[55] showed the supremacy of deep learning strategies also in this field. Then, other works improved accuracy by stacking more networks [97] or exploiting traditional pyramidal [94, 200, 231] and multi-frame fusion [203] approaches. Unfortunately, obtaining even sparse labels for optical flow is extremely challenging, which renders self-supervision from images highly desirable. For this reason, an increasing number of methods propose to use image reconstruction and spatial smoothness [103, 202, 221] as main signals to guide the training, paying particular attention to occluded regions [96, 102, 132, 133, 154, 267].

Semantic segmentation and depth estimation. Monocular depth estimation is tightly connected to the semantics of the scene. We can infer the depth of a scene by a single image mostly because of context and prior semantic knowledge. Prior works explored the possibility to learn both tasks with either full supervision [57, 59, 106, 111, 161, 250, 285] or supervision concerned with semantic labels only [41, 277]. Unlike previous works, we propose a compact architecture trained by self-supervision on monocular videos and exploiting proxy semantic labels.

Semantic segmentation and optical flow. Joint learning of semantic segmentation and optical flow estimation has been already explored [95]. Moreover, scene segmentation [18, 218] is required to disentangle potentially moving and static objects for focused optimizations. Differently, [201] leverages on optical flow to improve semantic predictions of moving objects. Peculiarly w.r.t. previous work, our proposal features a novel self-distillation training procedure guided by semantics to improve occlusion handling.

Scene understanding from stereo videos. Finally, we mention recent works approaching stereo depth estimation with optical flow [10] and semantic segmentation [105] for comprehensive scene understanding.

Chapter 2

Datasets

In most computer vision problems, the availability of large and diverse datasets is of paramount importance for successfully developing new algorithms and for being able to measure their effectiveness. For years, researchers in stereo matching evaluated their proposals on a few dozen stereo pairs with ground truth depth maps acquired in controlled, indoor environments [86, 211, 212]. Although these datasets allowed notable progress in the design of stereo algorithms, they did not adequately highlight many of the challenges arising in real applications. Moreover, modern machine learning algorithms are data-hungry and require much more than a few dozen stereo pairs.

In 2012, the first large-scale dataset with images of outdoor real environments was released [66] and an indoor dataset with much higher resolution [213] appeared soon after. Later, with the advent of deep learning [279] these datasets were followed by large, synthetic image sets which are ideal for training deep networks thanks to the negligible cost required to generate a multitude of training samples and by other large real-world datasets. In all cases, the datasets provide depth annotations obtained through different methodologies discussed later. The rest of this section will introduce in detail each of these datasets, summarized in Figure 2.1 where we show one reference image and the associated ground truth disparity map for each of them, respectively for a) KITTI 2015, b) Middlebury 2014, c) ETH3D and d) Freiburg SceneFlow. The first three were the

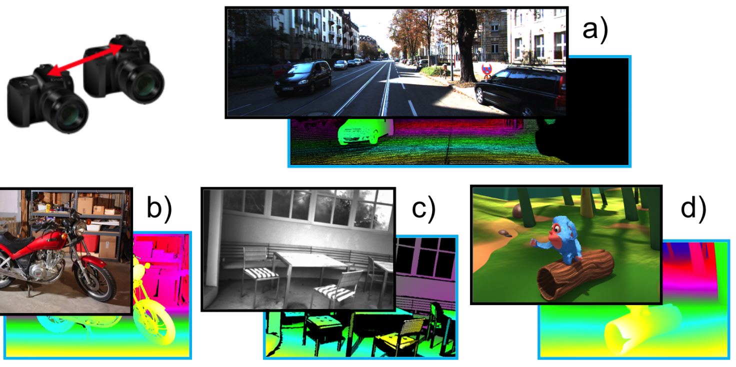


Figure 2.1: Overview of the most popular stereo datasets in literature, with examples of reference images and associated ground truth disparity. a) KITTI 2015 [155], b) Middlebury 2014 [213], c) ETH3D [215], d) Freiburg SceneFlow [151].

foundation of the stereo aspect of the Robust Vision Challenge (ROB)[1] in 2018.

2.0.1 KITTI

Acquired by Geiger et al.[67], the KITTI Vision Benchmark Suite represents the first, large-scale collection of images from a driving environment. The KITTI benchmarks have been seminal to the development of several algorithms and methods supporting autonomous driving. The data have been acquired from a car equipped with two stereo camera pairs, one greyscale and one color, a Velodyne LIDAR, GPS and inertial sensors. It consists of about 42k stereo pairs and LIDAR point clouds taken from 61 different scenes. From this extensive collection of images, appropriate benchmarks are available for key computer vision tasks such as stereo, optical flow, visual odometry, object detection and more. Two main datasets are available for stereo matching: KITTI 2012 and KITTI 2015.

KITTI 2012 [66]. This is the first dataset for stereo matching comprising outdoor images of static scenes and providing an online benchmark[2] for evaluation. It consists of 389 grayscale stereo pairs (recently made available in color format as well), split into 194 training pairs with available ground truth and 195 test pairs with withheld ground truth. Ground truth depth was obtained from LIDAR measurements as follows. A set of

[1]`robustvision.net`
[2]`cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo`

consecutive frames (5 before and 5 after) were registered using ICP, accumulated point clouds were re-projected onto the image, and finally, all ambiguous image regions such as windows and fences were manually removed. Using calibration parameters, the 3D points were projected on the images to obtain depth measurements, which were converted into disparities. This strategy yields semi-dense ground truth maps, where edges are usually not well aligned with RGB sensor, covering about one third of the pixels in each input image. The error metrics on the benchmark are the percentage of pixels with a disparity error greater than 3 and the average disparity error, measured either on all pixels or non-occluded pixels only. Metrics computed in reflective regions are also available.

KITTI 2015 [155]. A few years later, an improved dataset and benchmark for scene flow estimation [155] was proposed. In this case, the dataset consists of 400 color stereo pairs, evenly split into training and test sets. In contrast to the previous dataset, the stereo pairs are from dynamic scenes with objects (mostly cars) moving independently. The same procedure used for KITTI 2012 is followed here to obtain ground truth labels, except for moving objects whose 3D points cannot be properly accumulated over time. Hence, to obtain depth annotations for cars, 3D cad models are fitted into accumulated point clouds and re-projected onto the image. As the primary evaluation metric, the percentage of pixels with an absolute disparity error greater than 3 and a relative error larger than 5% (D1) is reported on the online benchmark[3], either considering only foreground (i.e.belonging to moving objects), background or all pixels. Moreover, masks to distinguish between non-occluded and all pixels are available.

2.0.2 Middlebury

The Middlebury Stereo Vision Page provided the first benchmark that allowed authors to submit the results of their algorithms. Over the years, the Middlebury stereo datasets have provided few indoor images with dense ground truth labels, obtained by manual annotation at first [211] and by structured light sensors later [86, 212, 213]. Three main

[3]`cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo`

versions have been proposed between 2002 [211] and 2014 [213], with varying resolution and image content. We will focus on this latter version, namely *Middlebury 2014*, since it provides an online benchmark for evaluation and still represents one of the most challenging datasets for stereo matching.

Middlebury 2014 [213]. It consists of 33 scenes, divided into *training*, *additional* and *test* splits made of respectively 13, 10 and 10 stereo pairs. Some of the data are used multiple times under different exposure and illumination conditions. A unique feature of this dataset is the very high image resolution, which reaches 6 megapixels compared to 0.3 megapixels of the KITTI images, and a disparity range between 200 and 800 pixels, representing one of the hardest challenges of this dataset. Images and ground truth disparity maps are provided at full, half and quarter resolution. An active stereo pipeline, described in detail in [213], was deployed to obtain dense and accurate ground truth depth. The limited number of training samples and the variety of content in the images make this dataset particularly challenging for deep learning methods, in particular for end-to-end models as we will set in the next sections. The online benchmark[4], reports the percentage of pixels having disparity errors larger than 0.5, 1, 2 and 4, as well as average and root mean square errors (RMSE) and other metrics, on either all or non-occluded pixels.

2.0.3 ETH3D

One of the most recent among real-world datasets, ETH3D [215]; it is a multi-view dataset for 3D reconstruction acquired in both indoor and outdoor environments at ETH Zurich. It consists of 25 high-resolution color multi-view stereo scenes divided into 13 for training and 12 for testing, 10 low-resolution grayscale many-view videos evenly divided for training and testing and finally 47 low-resolution grayscale stereo pairs, respectively split into 27 and 20 for training and testing. To obtain ground truth disparities, the authors recorded the scene geometry with a Faro Focus X 330 laser scanner, taking one or more 360°

[4]`vision.middlebury.edu/stereo/eval3`

scans with up to 28 million points each. Together with depth, the color of each 3D point captured by the laser scanner's integrated RGB camera was acquired, taking about 9 minutes to collect a single scan. The online benchmark[5], reports similar metrics to those of the Middlebury 2014 dataset.

2.0.4 Freiburg SceneFlow

The Freiburg SceneFlow dataset [151, 152] was a ground-breaking step forward in the field. As evidence, we underline that most of the proposed end-to-end networks for stereo matching are trained from scratch on this large dataset, before being fine-tuned on real data. The dataset consists of 3D scenes, from which images and dense ground truth for stereo, optical flow, and scene flow are rendered. To this end, the authors modified the internal rendering engine of the freely available Blender suite in order to produce fully dense and accurate ground truth for the two views of a virtual stereo camera with a resolution of 540×960 pixels. The dataset is organized into three subsets, named *FlyingThings3D*, *Monkaa* and *Driving*, totalling about 39000 stereo pairs overall. We briefly summarize the three datasets, referring the reader to [152] for more details.

FlyingThings3D. This set of images has been obtained fully automatically: the authors created a structured background from random geometric shapes, and overlayed on it dynamic foreground objects sampled from ShapeNet [30] and following linear trajectories in 3D space, as the camera itself does. The combination of objects and camera motions allows for complex object flows and scene settings. Even though the generated scenes are far from realistic, they allow for a large and diverse dataset. It totals 22872 stereo pairs, while 4370 additional pairs are set aside as the validation set of the entire SceneFlow dataset.

Monkaa. In contrast to FlyingThings3D, stereo pairs contained in this split are generated from an animated movie in a deterministic way. 3D artists modeled original scenes and elements, then the authors produced custom environments and rendered long

[5]`eth3d.net/low_res_two_view`

scenes to sample sufficient data. This subset contains 8591 stereo pairs.

Driving. Similarly to Monkaa, this split has been generated in a deterministic way as well. The aim of this portion of the Freiburg dataset is to provide data relevant to driving environments, as opposed to general scenes. This set contains 4392 samples.

2.0.5 Other datasets

In addition to most popular datasets discussed so far, we mention a few more which have not been used widely, partially because some of them are very recent. In terms of synthetic datasets, we mention MPI Sintel [258] and CARLA [56]. The former was extracted from short, animated movies and although it is more popular for optical flow, it also provides stereo pairs and dense disparity ground truth. The latter is a simulator enabling the synthesis of image sequences and associated ground truth labels in a virtual, urban driving environment.

Among real-world datasets, the Oxford Robotcar dataset [141] has been acquired after more than 100 km navigation with a trinocular camera, thus collecting stereo pairs with both a narrow and a wide baseline. Ground truth depth is generated from raw LIDAR measurements. Apolloscape [251] provides 5165 stereo pairs at 3-megapixel resolution, divided into 4156 pairs for training and 1009 for testing, with dense ground truth obtained by point cloud accumulation and fitting 3D CAD models, similarly to KITTI 2015. The recent DrivingStereo dataset [264] provides over 180k stereo pairs at 1.4 megapixels resolution, with semi-dense ground truth disparities obtained by interpolating LIDAR measurements and refining them with a deep stereo network. Potentially relevant are very large datasets released to support research on autonomous driving. Specifically, the Waymo Open Dataset [232], Argoverse by Argo AI [32] and the Lyft Level 5 dataset [112] are of unprecedented scale and one could imagine rectified stereo pairs with ground truth being extracted from them. We anticipate that Apolloscape and DrivingStereo, which provide binocular stereo imagery directly, will play a significant role for future developments in stereo matching.

Chapter 3

Evaluation Protocols

In this chapter, we will introduce the standard techniques used to evaluate and compare the approaches proposed in this thesis with the state-of-the-art. We report protocols for both confidence measures and depth map evaluations.

3.1 Depth Evaluation

Algorithms often have different strengths and weaknesses, such as overall accuracy or sensitivity to fine structures, which may be prioritized very differently depending on the application. Two metrics used in this thesis to evaluate the performance of a stereo algorithm are:

1. Average error (measured in disparity units) between the computed disparity map d and the ground truth disparity map $\tilde{d}$

$$EPE = \frac{1}{N} \sum_{(x,y)} |d(x,y) - \tilde{d}(x,y)| \tag{3.1}$$

2. Percentage of bad matching pixels, where τ is a disparity error tolerance.

$$bad_\tau = \frac{1}{N} \sum_{(x,y)} (|d(x,y) - \tilde{d}(x,y)| > \tau) \tag{3.2}$$

The threshold value is assigned according to dataset specifications, in particular for KITTI 2012 and 2015 τ usually it is 3 and for Middlebury 2014 it is 1 [184].

For what concern the monocular depth estimation task, we employ several metrics which have been used in prior works [60]:

1. Absolute relative distance between the computed depth map d and the ground truth depth map $\tilde{d}$
$$AbsRel = \frac{1}{N}\sum_{(x,y)} |d(x,y) - \tilde{d}(x,y)|/\tilde{d}(x,y) \tag{3.3}$$

2. Squared relative distance between the computed depth map d and the ground truth depth map $\tilde{d}$
$$SqRel = \frac{1}{N}\sum_{(x,y)} |d(x,y) - \tilde{d}(x,y)|^2/\tilde{d}(x,y) \tag{3.4}$$

3. Root mean squared distance between the computed depth map d and the ground truth depth map $\tilde{d}$
$$RMSE = \sqrt{\frac{1}{N}\sum_{(x,y)} ||d(x,y) - \tilde{d}(x,y)||^2} \tag{3.5}$$

4. Root mean squared distance in log space between the computed depth map d and the ground truth depth map $\tilde{d}$
$$RMSE(log) = \sqrt{\frac{1}{N}\sum_{(x,y)} ||\log d(x,y) - \log \tilde{d}(x,y)||^2} \tag{3.6}$$

5. Percentage of predicted depth values d such that the following equation with respect to the ground truth depth map $\tilde{d}$ is satisfied
$$Accuracy = max\left(\frac{d(x,y)}{\tilde{d}(x,y)}, \frac{\tilde{d}(x,y)}{d(x,y)}\right) = \delta < \tau \tag{3.7}$$

3.2 Confidence Measures

The ability to distinguish correct disparity assignments from wrong ones is the most desirable property of a confidence measure. To quantitatively evaluate this, [89] adopted ROC curve analysis, measuring the capability of removing errors from a disparity map according to the confidence values. More specifically, given a disparity map, a subset p of pixels is extracted in order of decreasing confidence (e.g., 5% of the total pixels) and the error rate on such sample is computed, as the percentage of points with an absolute distance from ground truth value higher than a threshold τ, varying withthe dataset. Then, the subset is increased by extracting more pixels (e.g., an additional 5%) and the error rate is computed, until all the pixels in the image are considered. Ties are solved by including all the tying pixels in the subsample. The relation between each sub-sample p and its error rate draws a ROC curve and its AUC (Area Under the Curve) measures the capability of the confidence measure to effectively distinguish good matches from wrong ones. Considering a disparity map with a portion $\epsilon \in [0, 1]$ of erroneous pixels, an optimal measure would be able to achieve a 0 error rate when extracting the first $(1 - \epsilon)$ points. Thus, the optimal AUC value [89] can be obtained as follows:

$$AUC_{opt} = \int_{1-\varepsilon}^{\varepsilon} \frac{p - (1 - \varepsilon)}{p} dp = \varepsilon + (1 - \varepsilon) \ln (1 - \varepsilon) \tag{3.8}$$

This value can be obtained when a confidence measures perfectly split pixels into correct assignments and outliers (i.e., all correct matches are subsampled before all the missmatched). The closer is the AUC to the optimum, the more effective the measure is.

Chapter 4

Confidence measures in a machine learning world

The content of this chapter has been presented at the International Conference on Computer Vision (ICCV 2017) - "Quantitative evaluation of confidence measures in a machine learning world" [184].

4.1 Introduction

Although depth from stereo still represents an open problem [67, 155, 213], in recent years this field has seen notable improvements concerning the effectiveness of such algorithms (e.g., [217, 279]) and confidence measures, aimed at detecting unreliable disparity assignments, proved to be very effective cues when plugged in stereo vision pipelines as shown in [169, 178, 217, 228]. However, shortcomings of stereo algorithms have been emphasized by the availability of very challenging datasets with ground truth such as KITTI 2012 (K12) [67], KITTI 2015 (K15) [155] and Middlebury 2014 (M14) [213]. Thus, the ability to reliably predict failures of a stereo algorithm by means of a confidence measure is fundamental and many approaches have been proposed for this purpose. Hu and Mordohai [89] exhaustively reviewed confidence measures available at that time, with two variants of a standard local algorithm, and defined a very effective metric to evaluate their effectiveness on the small and mostly unrealistic dataset [211] with ground truth available. However, since then there have been major breakthroughs in this field:

- Novel and more reliable confidence prediction methods, in particular those based on random-forests [77, 169, 178, 228] and deep learning [179, 217]
- Much larger datasets with ground truth depicting very challenging and realistic scenes acquired in indoor [213] and outdoor environments [67, 155]
- Novel and more effective stereo algorithms, some leveraging on deep learning techniques [151, 279], more and more often coupled with confidence measures [169, 178, 217]. Moreover, in recent years, SGM [83] became the preferred disparity optimization method for most state-of-the-art stereo algorithms (e.g., [217, 279])

Considering these facts, we believe that this field deserves a further and deeper analysis. Therefore, in this chapter we aim at i) extending and updating the taxonomy provided in [89] including novel confidence measure and in particular those based on machine learning techniques, ii) exhaustively assessing their performance on the larger and much more challenging datasets [155, 213] available today, iii) understanding the impact of training data on the effectiveness of confidence measures based on machine learning, iv) assessing their performance when dealing with new data and state-of-the-art stereo algorithms, v) and evaluating their behavior when plugged into a state-of-the-art stereo pipeline.

Although our focus is mostly on approaches based on machine learning, for completeness, we include in our taxonomy and evaluation any available confidence measure. Overall, we assess the performance of 52 measures, actually 76 considering their variants, providing an exhaustive evaluation of state-of-the-art in this field with three stereo algorithms on the three challenging datasets with ground truth KITTI 2012 , KITTI 2015 and Middlebury 2014 available today.

4.2 Taxonomy of confidence measures

Despite the large number of confidence measures proposed, all of them process (a subset of) information concerning the cost curve, the relationship between left and right images or disparity maps. Following [89], confidence measures can be grouped into categories according to their input cues. To better clarify which cues are processed by each single measure we introduce the following notation. Given a stereo pair made of left (L) and right (R) images, a generic stereo algorithm assigns a cost curve c to each pixel of L. We denote the minimum of such curve as c_1 and its corresponding disparity hypothesis as d_1. We refer to the second minimum of the curve as c_2 (and to its disparity hypothesis as d_2), while c_{2m} denotes the second local minimum (it may coincide with c_2). In our taxonomy we group the considered 52 confidence measures (and their variants) in the following 8 categories.

4.2.1 Minimum cost and local properties of the cost curve

These methods analyze local properties of the cost curve encoded by c_1, c_2 and c_{2m}. As confidence values for each point, the *matching score measure* **(MSM)** [89] simply assumes the negation of minimum cost c_1. *Maximum margin* **(MM)** computes the difference between c_{2m} and c_1 while its variant *maximum margin naive* **(MMN)** [89] replaces c_{2m} with c_2. *Non linear margin* **(NLM)** [76] computes a non linear transformation according to the difference between c_{2m} and c_1 while its variant *non linear margin naive* **(NLMN)** replaces c_{2m} with c_2. *Curvature* **(CUR) [89]** and *local curve* **LC** [256] analyze the behavior of the cost curve around the minimum c_1 and its two neighbors at (d_1-1) and (d_1+1) according two similar, yet different, strategies. *Peak ratio* **(PKR)** [84, 89] computes the ratio between c_{2m} and c_1. In one of its variants, *peak ratio naive* **(PKRN)** [89], c_{2m} is replaced with the second minimum c_2. In *average peak ratio* **(APKR)** [114] the confidence value is computed averaging PKR values on a patch. We include in our evaluation a further variant, based on the same patch-based average strategy adopted by APKR and referred to as *average peak ratio naive* **(APKRN)**. Similarly and respectively, *weighted peak ratio* **(WPKR)** [115] and *weighted peak ratio naive* **(WPKRN)**, average on a patch the original confidence measures PKR and PKRN with binary weights computed according to the reference image content. Finally, we include in this category two confidence measures belonging to the pool of features proposed in [77]. *Disparity ambiguity measure* **(DAM)** computes the distance between d_1 and d_2, while *semi-global energy* **(SGE)** relies on a strategy inspired by the SGM algorithm [83]. It sums, within a patch, the c_1 costs of points laying on multiple scanlines penalized, if their disparity is not the same of the point under examination, by P1 when the difference is 1 and by P2 (>P1) otherwise.

4.2.2 Analysis of the entire cost curve

Differently from previous confidence measures, those belonging to this category analyze for each point the overall distribution of matching costs. *Perturbation* **(PER)** [77] measures the deviation of the cost curve to an ideal one. *Maximum likelihood measure* **(MLM)** [89, 147] and *attainable likelihood measure* **(ALM)** [89, 157] infer from the matching costs a *probability density function* (pdf) with respect to an ideal c_1, respectively, equal to zero for MLM and to the actual c_1 for ALM. *Number of inflections* **(NOI)** [125] determines the number of local minima in the cost curve while *local minima in neighborhood* **(LMN) [114]** counts, on a patch, the number of points with local minimum at the same disparity d_1 of the examined point. *Winner margin measure* **(WMN)** [89] normalizes for each point the difference between c_{2m} and c_1 by the sum of all costs while its variant *winner margin measure naive* **(WMNN)** [89] adopts the same strategy replacing c_{2m} with c_2. Finally, *negative entropy measure* **(NEM)** [89, 210] relates the degree of uncertainty of each point to the negative entropy of its matching costs.

4.2.3 Left and right consistency

This category evaluates the consistency between corresponding points according to two different cues: one, symmetric, based on left and right maps and one, asymmetric, based only on the left map. Confidence measures adopting the first strategy are: *left-right consistency* **(LRC)** [58, 89], that assigns as confidence the negation of the absolute difference between the disparity of a point in L and its homologous point in R, and *left-right difference* **(LRD)** [89] that computes the difference between c_2 and c_1 divided by the absolute difference between c_1 and the minimum cost of the homologous point in R. We include in this category *zero-mean sum of absolute differences* **(ZSAD)** [77] that evaluates the dissimilarity between patches centered on homologous points in the stereo pair. It is worth pointing out that for LRC and ZSAD the full cost volume is not required. On the other hand, confidence measures based only on the analysis of the reference disparity map exploit the *uniqueness constraint*. *Asymmetric consistency check* **(ACC)** [158] and *uniqueness constraint* (**UC**) [54] detect the pool of multiple *colliding* points at the same coordinate in the right image. ACC verifies, according to a binary strategy, whether the candidate with the largest disparity in the pool has the smallest cost with respect to any other one while UC simply selects as valid the candidate with the minimum cost. Moreover, we consider two further non binary variants of this latter strategy. One referred to as *uniqueness constraint cost* **(UCC)**, that assumes as confidence the negative of c_1, and one referred to as *uniqueness constraint occurrences* **(UCO)**, that assumes that confidence is inversely proportional to the number of collisions. For the latter four outlined strategies the other candidates in the pool of colliding points are always set to invalid.

4.2.4 Disparity map features

Confidence measures belonging to this group are obtained by extracting features from the reference disparity map. Therefore they are potentially suited to infer confidence for any 3D sensing device. *Distance to discontinuity* **(DTD)** [169, 228] determines for each point the distance to the supposed closest depth boundary while, for the same purpose, *disparity map variance* **(DMV)** computes the disparity gradient module [77]. Remaining confidence measures belonging to this category extract features on a patch centered on the examined point. *Variance of disparity* **(VAR)** [169, 178] computes the disparity variance, *disparity agreement* **(DA)** [178] counts the number of points having the same disparity of the central one, *median deviation of disparity* **(MDD)** [169, 178, 228] computes the difference between disparity and its median and *disparity scattering* **(DS)** [178] encodes the number of different disparity assignments on the patch.

4.2.5 Reference image features

Confidence measures belonging to this category use as domain only the reference image. *Distance to border* **(DB)** [169, 228] aims at detecting invalid disparity assignments often originated in the image border due to the stereo setup. Assuming the left image as reference a more meaningful variant of DB, referred to as *distance to left border* **(DLB)**, deploys the distance to the left border. Both measures rely on prior information and not on image content. The last two confidence measure of this category extract features from the reference image: *horizontal gradient measure* **(HGM)** [77, 169] analyses the response to horizontal gradients in order to detect image texture while *distance to edge* **(DTE)** attempts to detect depth boundaries, sometimes unreliable for stereo algorithms, according to the distance to the closest edge.

4.2.6 Image distinctiveness

The idea behind these confidence measures is to exploit the notion of distinctiveness of the examined point within its neighborhoods along the horizontal scanline of the same image. *Distinctiveness* **(DTS)** [89, 144] exactly leverages on such definition by assuming as confidence for a given point the lowest *self-matching* cost computed within a certain prefixed range excluding the point under examination. *Distinctive similarity measure*

(DSM) [89, 272] assigns as confidence value to a given point the product of two DTSs, one computed on the reference image and the other one on the right image in the location of the assumed homologous point, divided by the square of c_1 [89] or c_1 [272]. For a given point the *self-aware matching measure* **(SAMM)** [89, 159] computes the zero mean normalized correlation between the left-right cost curve, appropriately translated according to the assumed disparity, and the left-left cost curve.

4.2.7 Learning-based approaches

Recently, some authors proposed to infer confidence measures exploiting machine learning frameworks. A common trend in such approaches consists in feeding a random forest classifier with multiple confidence measures [77, 169, 178, 228] or deploying for the same purpose deep learning architectures [179, 217]. A notable difference with conventional confidence measures reviewed so far, is that learning-based approaches require a training phase, on datasets with ground truth or by means of appropriate methodologies [160, 239], to infer the degree of uncertainty of disparity assignments.

Random forest approaches

In this category a seminal approach is represented by *ensemble learning* **(ENS$_c$)** [77]. This method infers a confidence measure by feeding to a random forest, trained for classification, a feature vector made of 23 confidence measures extracted from the original stereo pair, the left and right disparity maps and the cost volumes computed on the stereo pair at different scales. Then, the resulting features are up-sampled to the original resolution. The feature vector consists of the following measures: PKR^{1,2,3}, NEM^{1,2,3}, PER^{1,2,3}, LRC^{1}, HGM^{1,2,3}, DMV^{1,2,3}, DAM^{1,2,3}, ZSAD^{1,2,3} and SGE^{1}. The superscript refers to the scale: 1 original resolution, 2 half-resolution and 3 quarter-resolution. The authors advocate to train the random-forest with such feature vector for classification *"as confidence measures do not contain matching error magnitude information"*, by extracting the posterior probability of the predicted class at inference time. However, the average response over all the trees in the forest can be used as well by training in *regression*. Therefore, we also include in our evaluation *ensemble learning in regression mode* (**ENS**$_r$) that to the best of our knowledge has not been considered before. In *ground control point* (**GCP**) **[228]** the confidence measure is inferred by feeding to a random forest, trained in regression mode, a feature vector containing 8 measures computed at the original scale. The features extracted from left image, left and right disparity maps and the cost volume are: MSM, DB, MMN, AML, LRC, LRD, DTD and MDD. In *leveraging stereo* (**LEV**) **[169]** a feature vector containing 22 measures extracted from the left image, left and right disparity maps and cost volume is fed to a random forest trained for regression. The feature vector, superscript encodes the patch size, consists of: PKR, PKRN, MSM, MM, WMN, MLM, PER, NEM, LRC, LRD, LC, DTD, VAR^{1,2,3,4}, MDD^{1,2,3,4}, HGM and DLB. Differently from previous approaches, *O(1) disparity features* (**O1**) **[178]** proposes a method entirely based on features extracted in constant time from the left disparity map. The feature vector, superscript encodes the patch size, consists of: DA^{1,2,3,4}, DS^{1,2,3,4}, MED^{1,2,3,4}, MDD^{1,2,3,4} and VAR^{1,2,3,4}, being MED the median of disparity. As for ENS$_r$, GCP and LEV the feature vector is fed to a random forest trained in regression mode. We conclude this section observing that ENS [77] and LEV [169] also propose variants of the original method with a reduced number of features, respectively 7 and 8. For LEV, the features are selected analyzing the importance of variable once trained the random forest with the full 22 feature vector and then retraining the network. However, as reported in [77] and [169], being higher the effectiveness of full feature vectors, we consider in our evaluation such versions of ENS, in classification and regression mode, and LEV.

CNN approaches

As for many other computer vision fields, convolutional neural networks have recently proven to be very effective also for confidence estimation. In *patch based confidence prediction* (**PBCP**) [217] the input of a CNN consists of two channels p_1 and p_2 computed, on a patch basis, from left and right disparity maps. Being patch values strictly related to their central pixel, confidence map computation is pretty demanding. A faster solution, made of patches no longer related to central pixels, allows for a very efficient confidence map prediction according to common optimization techniques in deep learning, with a minor reduction of effectiveness. However, being the full-version more effective we consider this one in our experiments.

A step towards a further abstraction is represented by *confidence CNN* **(CCNN) [179]**. In fact, in this approach confidence prediction is regressed by a CNN without extracting any cue from the input data. The deep network, trained on patches, learns from scratch a confidence measure by processing only the left disparity map. This property, shared with O1, makes these methods potentially suited to any 3D sensor [178, 179].

4.2.8 SGM specific

This category groups two approaches intrinsically related to SGM [83]. The idea behind these approaches is to exploit intermediate results available in such stereo algorithm to infer a confidence map. Specifically, the *local-global relation* **(PS)** [145] combines the cues available in the cost curve before and after semi-global optimization, while *sum of consistent scanlines* **(SCS)** [85] counts for each pixel the number of scanlines voting for the same disparity assigned by the full SGM pipeline.

4.3 Experimental results

In this section, we report exhaustive experimental results concerning different aspects related to the examined confidence measures on the following datasets K12 (194 images), K15 (200 images) and M14 (15 images). For each dataset we consider the stereo pairs belonging to the *training set* being the ground truth available. We include in the evaluation all the measures previously reviewed including any variant. Moreover, for patch-based ones (i.e., APKR, APKRN, WPKR, WPKRN, DA, DS, MED, VAR) we consider patches of different size (i.e., 5×5, 7×7, 9×9 and 11×11 corresponding to superscript 1,2,3,4

(a)

	K12 ($\varepsilon = 38.82\%$)			K15 ($\varepsilon = 35.41\%$)			M14 ($\varepsilon = 37.78\%$)		
Category	measure	rank	AUC	measure	rank	AUC	measure	rank	AUC
4.2.1	APKR$_{11}$	4^{12}	0.1806	APKR$_{11}$	4^{12}	0.1541	APKR$_{11}$	4^{7}	0.1355
4.2.2	WMNN	7^{34}	0.2215	WMN	7^{34}	0.2024	WMN	6^{23}	0.1579
4.2.3	LRD	5^{20}	0.1946	LRD	6^{28}	0.1825	LRD	5^{21}	0.1519
4.2.4	DA$_{11}$	3^{8}	0.1668	DA$_{11}$	3^{7}	0.1399	DA$_{11}$	3^{4}	0.1294
4.2.5	DB	8^{65}	0.3446	DB	8^{66}	0.3103	DLB	8^{69}	0.3333
4.2.6	SAMM	6^{25}	0.2030	SAMM	5^{20}	0.1715	DSM	7^{40}	0.1798
4.2.7	O1	2^{3}	0.1309	O1	2^{3}	0.1128	O1	2^{3}	0.1211
4.2.7	**CCNN**	$\mathbf{1^{1}}$	**0.1223**	**CCNN**	$\mathbf{1^{1}}$	**0.1041**	**CCNN**	$\mathbf{1^{1}}$	**0.1128**
Optimal			0.1067			0.0884			0.0899

Categories 4.2.7 and 4.2.7			
Measure	K12	K15	M14
ENS$_c$	7	11	44
ENS$_r$	5	5	33
GCP	6	6	8
LEV	4	4	5
O1	3	3	3
PBCP	2	2	2
CCNN	**1**	**1**	**1**

(b)

(c)

	K12 ($\varepsilon = 17.10\%$)			K15 ($\varepsilon = 15.37\%$)			M14 ($\varepsilon = 26.70\%$)		
Category	measure	rank	AUC	measure	rank	AUC	measure	rank	AUC
4.2.1	APKR$_{11}$	4^{11}	0.0566	APKR$_{11}$	4^{11}	0.0508	APKR$_{11}$	3^{5}	0.0728
4.2.2	WMN	6^{30}	0.0748	WMN	6^{31}	0.0654	WMN	4^{13}	0.0763
4.2.3	LRD	7^{31}	0.0748	LRD	7^{32}	0.0712	UCC	5^{22}	0.0896
4.2.4	DS$_{9}$	3^{8}	0.0542	DS$_{9}$	3^{8}	0.0477	DS$_{11}$	6^{35}	0.1061
4.2.5	DLB	8^{66}	0.1543	HGM	8^{67}	0.1439	DLB	8^{68}	0.2260
4.2.6	SAMM	5^{16}	0.0598	SAMM	5^{21}	0.0557	DSM	7^{40}	0.1228
4.2.7	O1	2^{2}	0.0317	O1	2^{2}	0.0324	O1	2^{3}	0.0680
4.2.7	**CCNN**	$\mathbf{1^{1}}$	**0.0297**	**CCNN**	$\mathbf{1^{1}}$	**0.0297**	**CCNN**	$\mathbf{1^{1}}$	**0.0637**
Optimal			0.0231			0.0213			0.0459

Categories 4.2.7 and 4.2.7			
Measure	K12	K15	M14
ENS$_c$	7	7	24
ENS$_r$	5	5	17
GCP	6	6	14
LEV	4	4	4
O1	2	2	3
PBCP	3	3	2
CCNN	**1**	**1**	**1**

(d)

(e)

	K12 ($\varepsilon = 16.78\%$)			K15 ($\varepsilon = 13.68\%$)			M14 ($\varepsilon = 25.91\%$)		
Category	measure	rank	AUC	measure	rank	AUC	measure	rank	AUC
4.2.1	APKR$_{11}$	3^{7}	0.0492	APKR$_{11}$	3^{7}	0.0457	APKR$_{9}$	2^{2}	0.0739
4.2.2	WMN	4^{11}	0.0554	WMN	5^{12}	0.0502	WMN	4^{8}	0.0.779
4.2.3	UCC	6^{21}	0.0735	UCC	6^{19}	0.0640	UCC	6^{23}	0.0959
4.2.4	DS$_{11}$	5^{12}	0.0554	DS$_{11}$	4^{11}	0.0501	DS$_{11}$	5^{13}	0.0884
4.2.5	DB	9^{67}	0.1378	DB	9^{68}	0.1265	DLB	9^{70}	0.2157
4.2.6	DSM	7^{36}	0.0811	DSM	7^{28}	0.0679	DSM	7^{32}	0.1041
4.2.7	LEV	2^{2}	0.0358	O1	2^{2}	0.0323	O1	3^{6}	0.0777
4.2.7	**CCNN**	$\mathbf{1^{1}}$	**0.0358**	**CCNN**	$\mathbf{1^{1}}$	**0.0302**	**CCNN**	$\mathbf{1^{1}}$	**0.0736**
4.2.8	SCS	8^{41}	0.0851	SCS	8^{48}	0.0790	SCS	8^{36}	0.1080
Optimal			0.0227			0.0184			0.0431

Categories 4.2.7 and 4.2.7			
Measure	K12	K15	M14
ENS$_c$	27	31	44
ENS$_r$	5	5	11
GCP	6	6	28
LEV	2	4	19
O1	3	2	6
PBCP	4	3	7
CCNN	**1**	**1**	**1**

(f)

Table 4.1: Detection of correct matches with three stereo algorithms - top (a,b) AD-CENSUS, middle (c,d) MC-CNN and bottom (e,f) SGM - and three datasets K12, K15 and M14. For each algorithm there are two tables. On the left the best confidence measure for each category (e.g., 4.2.1 refers to measures belonging to the category reviewed in Section 4.2.1), the ranking (within categories and, in superscript, absolute) and the AUC. On the right, the absolute ranking of learning-based confidence measures. We also report average error rate ε for each dataset on the top labels. Concerning categories 4.2.7 and 4.2.7 we trained each confidence measure on the first 20 images of K12 with the considered algorithm (i.e., (a,b) with AD-CENSUS, (c,d) with MC-CNN and (e,f) with SGM).

in LEV and O1 features) being the scale effective according to [169, 178]. Of course, we consider state-of-the-art methods based on random forests, including variant ENS$_r$, and the two approaches based on CNNs. Overall, we evaluate 76 confidence measures. We assess with three stereo algorithms the performance of such measures when dealing with the selection of correct matches by means of the ROC curve analysis proposed in [89] and widely adopted in this field [77, 169, 178, 179, 217, 228]. Moreover, since machine learning is the key technology behind most recent approaches, in Section 4.3.1 we report how training affects their effectiveness focusing in particular on the amount of training samples and the capability to generalize across different data (i.e., datasets). Finally, being confidence measures often employed to improve stereo accuracy [169, 178, 217, 228], in Section 4.3.2 we assess the performance of the most effective confidence measures when plugged in one of such state-of-the-art methods [169].

We evaluate the 76 confidence measures on K12, K15 and M14 with three popular stereo algorithms adopting the *winner takes all* strategy for disparity selection: AD-CENSUS, MC-CNN and SGM.

Concerning confidence measures based on machine learning, for each stereo algorithm, we train each one on a subset of images from the K12 dataset (the first 20 images, extracting a sample from each pixel with available ground truth, for a total of 2.7 million samples) and evaluate it on all the datasets (for K12 excluding the training images), in order to assess their performance on very different scenes. For approaches based on random forests we train on 10 trees as suggested in [169] and adopting a fixed number of iteration as termination criteria (e.g., proportional to the number of trees), while we train CNN based measures for 25 epochs (resulting in about 1 million iterations), with a batch of size 64, *learning rate* of 0.001 and *momentum* of 0.9, by minimizing the loss functions reported in [179, 217]. Different training sets (e.g., datasets, number of samples and so on) may lead to different performance. This fact will be thoroughly evaluated in Section 4.3.1. For the evaluation reported in this section we trained only on K12 in order to assess how much a confidence measure is able to generalize its behavior across different datasets which is an important and desirable feature in most practical applications. We adopt as error bound $\tau = 3$ for K12 and K15 and $\tau = 1$ for M14[1] as suggested in the corresponding papers.

In Table 10.2 we summarize results in terms of AUC averaged on each dataset (K12, K15 and M14) for AD-CENSUS (a,b), MC-CNN (c,d) and SGM (d,e), reporting the average error rate ε for each dataset. For each algorithm we report on the left table the best measure for each category described in Section 4.2 and its absolute ranking and, on the right table, the absolute ranking for confidence measures based on machine learning. Observing tables 10.2 (a,c,e), we can notice that these latter measures always yield the best results, with CCNN systematically the top-performing one in terms of AUC, and the ones based on random forest following very close (with O1 the best in its category in 7 out of 9 experiments). Focusing on categories 4.2.7 and 4.2.7, we can notice that in most cases PBCP, O1 and LEV perform very well with the exception of the SGM algorithm and M14 (Table 10.2(f)). In this specific case, excluding CCNN, APKR$_{11}$ performs better than approaches based on machine learning. Anyway, in this case too, the effectiveness of O1 and PBCP seems acceptable. This fact highlights that some confidence measure based on learning approaches (in particular CCNN but also O1 and PBCP) have excellent performance across different data. Interestingly, such measures use as input cue only the disparity maps. Tables 10.2 (b,d,f) also show that for other measures such as ENS$_c$, ENS$_r$, GCP and LEV this behavior is not always verified, in particular with M14. Finally, we observe that ENS$_r$ always (and sometimes significantly) outperforms ENS$_c$. Concerning other categories, we can notice that APKR yields good results in all the experiments and not only with M14 and SGM as already highlighted. Other interesting confidence measures are those belonging to category 4.2.4 and in particular DA with AD-CENSUS and DS with MC-CNN and SGM. Such results confirm that processing cues from the disparity map only, as done by best learning-based approaches, yields reliable confidence

[1]Middlebury frames have been processed at quarter resolution to level out the original disparity range with other datasets (800 vs 228 for KITTIs).

estimation. Other categories do not seem particularly effective, especially those based only on left image cues have always the overall worst performance. For measures belonging to category 4.2.2, though not very effective excluding experiments with SGM, WMN always achieves the best results. Besides, it's worth pointing out that naive versions of traditional strategies produce worse AUC values than their original counterparts. Regarding SGM-specific methods, SCS always outperforms PS but with AUC values quite far from the top-performing approaches. Finally, concerning categories 4.2.3 and 4.2.6, such measures on the three datasets do not grant reliable confidence prediction.

4.3.1 Impact of training data

Having assessed the performance of the confidence measure with different algorithms and datasets, this section aims at analyzing the impact of training data on the effectiveness of learning-based measures. To quantitatively compare the results between different training configuration, we define Δ_k as the ratio between the AUC value achieved by the measure k and the AUC$_{opt}$ as,

$$\Delta_k = \frac{AUC_k}{AUC_{opt}} \tag{4.1}$$

The lower the Δ_k, the better the training configuration.

The first issue we are going to evaluate is the amount of training samples required and how it affects the overall effectiveness of each confidence measure. We carried out multiple trainings with a different number of samples obtained from 5, 10, 15, 20 and 25 stereo pairs of K12 dataset starting from the first image. These subsets provide, respectively, about 0.7, 1.5, 2, 2.7 and 3.5 million samples with available ground truth for training. By using more data we can deploy more complex random forests as well. Nevertheless, we keep the same parameters and termination criteria described so far to compare the behavior of the same forest fed with different feature vectors when more samples are available. Figure 4.1 reports Δ_k, as a function of the number of training samples, for the best six measures

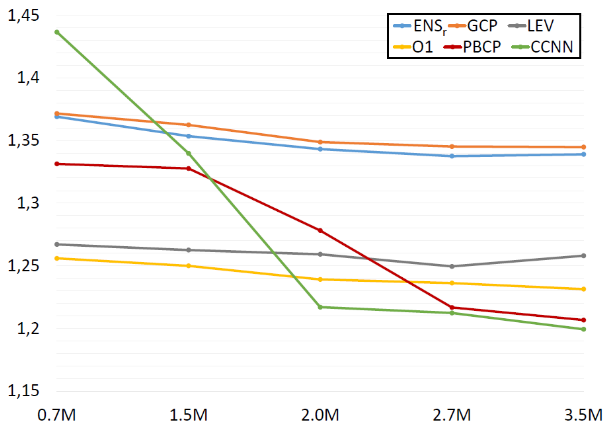


Figure 4.1: Ratio between the average AUC achieved by learning-based confidence measures trained with different number of samples from K12 and the optimal AUC. Evaluated on the rest of K12 with AD-CENSUS algorithm.

based on machine learning (i.e., ENS$_r$, GCP, LEV, O1, CCNN and PBCP) trained on AD-CENSUS algorithm. We can notice how the amount of training data slightly changes the effectiveness of the methods based on random forest (less than 0.05 Δ_k improvement), highlighting how the best AUC is obtained starting from 2.7 million samples. Conversely, measures based on CNNs improve their effectiveness by a significant margin only when trained on a sufficiently larger amount of data, but such improvement almost saturates at 2.7 million samples. In particular, we can observe how CCNN achieves the worst results when trained with the smallest subset of images, resulting to be the best measure with a larger training set (with a Δ_k margin of about 0.25). Excluding LEV and ENS$_r$ at 3.5M, all the measures show a monotonic improvement in terms of AUC by increasing the number of samples.

The second issue evaluated concerns how much a confidence measure can generalize across different environments/scenes (i.e., datasets). To quantitatively evaluate this behavior, we trained with AD-CENSUS the confidence measures on a subset of M14, processing an almost equivalent amount of training samples with respect to the training configuration adopted in so far. Then, we compared the results achieved with this configuration to the one used previously with AD-CENSUS on the remaining data from M14, computing Δ_k as defined in Equation 4.1. A confidence measure achieving similar Δ_k in

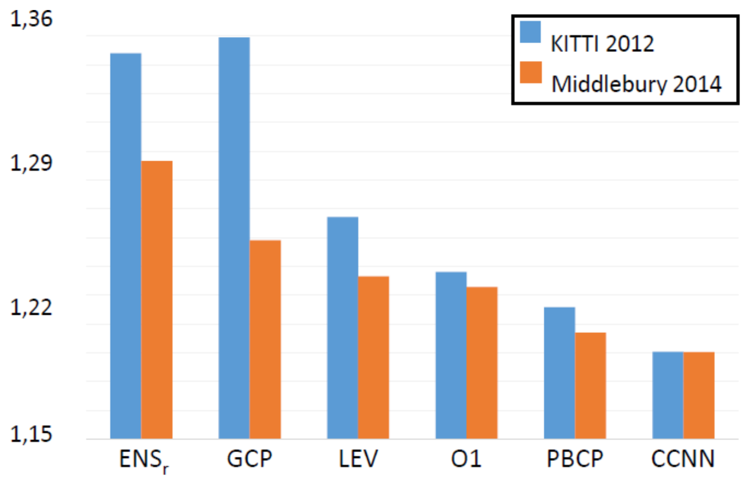


Figure 4.2: Experimental results on M14. Ratio between the average AUC achieved by each confidence measure, trained on K12 (blue) and M14 (orange), and the optimal AUC evaluated on the rest of M14 with AD-CENSUS algorithm.

the two configuration is able to generalize well between the two very different scenarios. Figure 4.2 plots the two values for the six confidence measures. We can clearly notice how measures based on CNNs better generalize with respect to random forest approaches, with CCNN being more effective in this sense than PBCP. Moreover, O1 appears to better adapt to different data, achieving a lower margin between the two Δ_k with respect to ENS$_r$, GCP and LEV. This experiment highlights once again that confidence measures using as input cue the disparity map(s) (i.e., CCNN, PBCP and O1) seem less prone to under-fitting.

4.3.2 Improvements to stereo accuracy

The final issue we investigated is the impact of confidence measures on stereo accuracy, a topic that recently gained a lot of attention (e.g., [169, 178, 217, 228]). For this evaluation we choose the cost modulation proposed by Park and Yoon [169]. The reason is that differently from [178], which is specific for SGM algorithm, and [217, 228], based on parameters potentially different from measure to measure, [169] is suited for any stereo algorithm and parameter-free. We plugged in [169] the machine learning based measures, as well as three standalone measures (i.e., APKR, SAMM and DA$_{11}$). On the three

	K12		K15		M14	
	bad3	avg	bad3	avg	bad1	avg
SGM	16.53	7.40	13.68	6.13	25.91	7.11
APKR$_{11}$	11.26^{10}	3.60^{10}	9.57^{10}	2.94^{10}	23.79^{8}	5.15^{10}
SAMM	10.95^{6}	3.15^{6}	9.13^{6}	2.58^{6}	24.07^{10}	4.94^{4}
DA$_{11}$	11.18^{9}	3.40^{9}	9.50^{9}	2.77^{9}	23.98^{9}	5.10^{9}
ENS$_c$	10.42^{2}	2.71^{4}	9.02^{4}	2.33^{4}	23.49^{4}	5.00^{8}
ENS$_r$	10.63^{5}	2.95^{5}	9.08^{5}	2.46^{5}	23.74^{7}	4.96^{6}
GCP	11.05^{8}	3.26^{8}	9.28^{7}	2.67^{7}	23.54^{5}	4.97^{7}
LEV	10.97^{7}	3.22^{7}	9.34^{8}	2.72^{8}	23.67^{6}	4.94^{5}
O1	**10.41**^{1}	**2.36**^{1}	8.79^{2}	1.84^{2}	23.18^{3}	4.07^{2}
PBCP	10.63^{4}	2.60^{3}	8.86^{3}	1.91^{3}	22.92^{2}	**3.95**^{1}
CCNN	10.61^{3}	2.41^{2}	**8.79**^{1}	**1.80**^{1}	**22.86**^{1}	4.12^{3}

Table 4.2: Error rate (percentage) and average pixel error on the three datasets achieved by vanilla SGM (first row) and the confidence modulation proposed in [169] plugging: APKR$_{11}$, SAMM, DA$_{11}$, ENS$_c$, ENS$_r$, GCP, LEV (the one proposed in [169]), O1, PBCP and CCNN. Learning-based confidence measures trained, with AD-CENSUS, on the first 20 images of K12.

datasets K12, K15 and M14, from Table 4.2 we can notice that confidence measures based on machine learning are overall more effective than other ones. In particular, O1 achieves the lowest error rate with K12 and CCNN and PBCP outperforms other ones in K15 and M14. This experiment highlights that there is not a direct relationship with the effectiveness of the confidence measure in terms of AUC. However, most effective confidence measures (i.e.,, CCNN, PBCP and O1) according to this metric achieve the best results. Finally we point out that in this experiments, ENS$_c$ and ENS$_r$, frequently perform better than others confidence measures, conventional and learning-based ones. Moreover, for their deployment in cost modulation ENS$_c$ outperforms ENS$_r$ most of the times, conversely to what observed in terms of AUC.

4.4 Conclusions

In this chapter we have reviewed and evaluated state-of-the-art confidence measures focusing our attention on recent ones based on machine learning techniques. Our exhaustive evaluation, with three stereo algorithms and three large and challenging datasets, clearly highlights that learning-based ones are much more effective than conventional approaches. In particular, those using as input cue the disparity maps achieve better results in terms of detection of correct match, capability to adapt to new data and effectiveness to improve stereo accuracy. In such methods training is certainly an additional issue but, as reported in our evaluation, the overall amount of training data required is limited and best learning-based confidence measures much better generalize to new data. More recently, a more updated review about confidence measures has been proposed which includes the latest advances in the field of confidence estimation and considering also state-of-the-art 3D end-to-end stereo networks and experiments on realistic synthetic datasets [182].

Chapter 5

Efficient confidence measures for embedded stereo

The content of this chapter has been presented at the International Conference on Image Analysis and Processing (ICIAP 2017) - “Efficient confidence measures for embedded stereo” [185].

5.1 Introduction

The recent availability of embedded depth sensors paved the way to a variety of computer vision applications for autonomous driving, robotics, 3D reconstruction and so on. In these application depth is crucial and several approaches have been proposed to tackle this problem following two main strategies. On one hand *active* sensors infer depth by enhancing the sensed scene by means of structured light, laser projection and so on. On the other hand, *passive* depth sensors infer depth not altering at all the sensed environment. Although sensors based on active technologies are quite effective they have some limitations. In particular, some of them (e.g., Kinect) are not suited for outdoor environments during daytime while others (e.g., LIDAR) provide only sparse depth maps and are quite expensive, cumbersome and containing moving mechanical parts.

Many stereo algorithms have been proposed to solve the stereo correspondence problem, some of them particularly suited for hardware implementation, thus enabling the design of compact, low-powered and real-time depth sensors [87], [150], [20], [65]. Despite the vast literature in this field, challenging conditions found in most practical applications represent a major challenge for stereo algorithms. Therefore, regardless of the stereo algorithm deployed, it is essential to detect its failures to filter-out wrong unreliable points that might lead to a wrong interpretation of the sensed scene. Some recent confidence measures combine multiple features within random forest frameworks to obtain more reliable confidence scores while an even more recent trend aims to infer confidence prediction leveraging on Convolutional Neural Networks (CNN) [179], [217]. Despite their effectiveness, the latter strategies are often not compatible with the computing resources available inside the depth sensor, typically a low cost FPGA or a System-On-Chip (SoC) based on ARM CPU cores and an FPGA (e.g., Xilinx Zynq). Moreover, the features required by most of these machine-learning frameworks are not available as output of the embedded stereo cameras being in most cases computed from the cost volume (often referred to as disparity space image (DSI) [211]).

Therefore, in this chapter we consider a subset of confidence measures compatible with embedded devices evaluating their effectiveness, on two popular challenging datasets and two algorithms typically deployed for real-time stereo for embedded systems, focusing our attention on issues related to their FPGA implementation. Our study highlights that some of the considered confidence measures, appropriately modified to fit with typical hardware constraints found in the target architectures, clearly outperform those currently deployed in most embedded stereo cameras.

5.2 Hardware strategies for confidence implementation

When dealing with conventional CPU based systems confidence measures are generally implemented in C, C++ and to maintain the whole dynamic range single or double floating point data types are deployed. However, floating point arithmetic is sometimes not available in embedded CPU and generally unsuited to FPGAs. In particular, transcendental functions and divisions represent major issues when dealing with such devices. To overcome these limitations, fixed point arithmetic is usually deployed [19]. Fixed point represents an efficient and hardware-friendly way to express and manipulate fractional numbers with a fixed number of bits [19]. Indeed, fixed-point math can be represented with an integer number split into two distinct parts: the integer content (I), and the fractional content (F). Through the simple use of integer operations, the math can be efficiently performed with little loss of accuracy taking care to use a sufficient number of bits. The steps required to convert a floating point value to the corresponding fixed representation with F bits - the higher, the better in terms of accuracy - are the following:

1. Multiply the original value by 2^F
2. Round the result to the closest integer value
3. Assign this value into the fixed-point representation

Fixed point encoding greatly simplifies arithmetic operations with non-integer values, but integer divisions can be demanding - in particular on FPGAs - except when dealing with divisors which are powers of 2. In fact, in this case division requires almost negligibly hardware resources being carried out by means of a simple right shift. Thus, a simplified method to avoid integer divisions consists in rounding the dividing value to the closest power of 2, then shifting right according to its $\log_2$. This strategy will be referred to as *pow*.

Although fixed point increases the overall efficiency, some confidence measures rely on transcendental functions (in particular, exponentials and logarithms) which represent an a further major issue even when dealing with CPU based systems. An effective strategy to deal with such functions consists in deploying Look-Up Tables (LUTs) to store precomputed results encoded with fixed point arithmetic. That is, given a function $\mathcal{F}(x)$, with x assuming n possible values, a LUT of size n can store all the possible outcome of such function. Of course, this approach is feasible only when the size of the LUT (proportional to n) is compatible with the memory available in the device.

5.3 Confidence measures suited for hardware implementation

In this section we describe the pool of confidence measures suited for implementation on target embedded devices. Figure 5.1 shows the matching cost curve for a pixel of the reference image. Given a pixel $\mathbf{p}(x, y)$, we will refer to its minimum cost as c_1, the second minimum as c_2 and the second local minimum as c_{2m}. The matching cost for any disparity hypothesis d will be referred to as c_d while the disparity corresponding to c_1 as d_1, the one corresponding to c_2 as d_2 and so on. If not specified otherwise, costs and disparities are referred to the reference left image (L) of the stereo pair. When dealing with right image (R), we introduce the ^{R} symbol on costs (e.g., c_1^R) and disparities. We denote as $\mathbf{p}'(x', y')$ the homologous point of $\mathbf{p}$ according to d_1 (i.e., $x' = x - d_1$, $y' = y$). It is worth to note that, assuming the right image as reference, the matching costs can be easily obtained by scanning in diagonal the cost volume computed with reference the left image without any further new computation. Nevertheless, adopting this strategy would require an additional buffering of $\frac{d_{max} \cdot (d_{max}+1)}{2}$ matching costs with d_{max} the disparity range deployed by the stereo algorithm.

We distinguish the pool of confidence measures in two, mutually exclusive, categories:

- *Hardware friendly*: confidence measures whose standard implementation is fully compliant with embedded systems.
- *Hardware challenging*: confidence measures involving transcendental functions and/or floating point divisions not well suited for embedded systems in their conventional formulation.

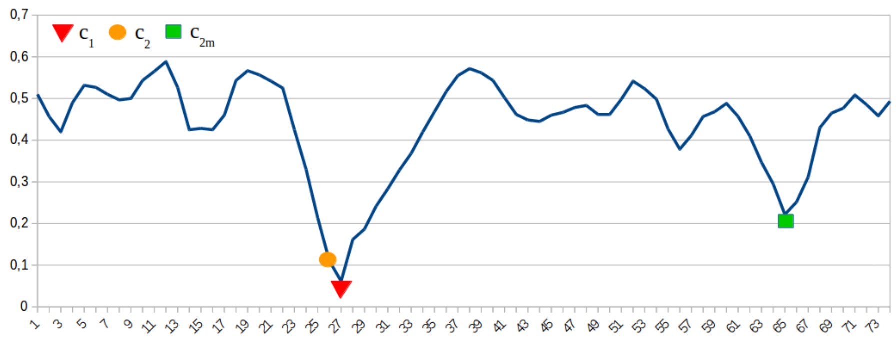


Figure 5.1: Example of cost curve, showing the matching cost c_1, the second minimum c_2 and the second local minimum c_{2m}. On x axis the disparity range, on y magnitude of the costs.

5.3.1 Hardware friendly

This category groups confidence measures involving simple math operations that do not represent issues when dealing with implementation on embedded systems. The *matching score measure* (MSM) [89] negates the minimum cost c_1 assuming it related to the reliability of a disparity assignment. *Maximum margin* (MM) estimates match uncertainty by computing the difference between c_{2m} and c_1 while its variant *maximum margin naive* (MMN) [89] replaces c_{2m} with c_2. Given two disparity maps computed by a stereo algorithm assuming as reference L and R, the *left-right consistency* (LRC) [89] sets as confidence the negation of the absolute difference between the disparity of a point in L and its homologous point in R. Another popular and more efficient strategy based on a single matching phase is the *uniqueness constraint* (UC) [54]: it assumes as poorly confident those pixels colliding on the same point of the target image (R) with the exception of the one having the lowest c_1. *Curvature* (CUR) [89] and *local curve* (LC) [256] analyze the behavior of the matching costs in proximity of the minimum c_1 and its two neighbors at (d_1-1) and (d_1+1) according to two similar strategies. Finally, *number of inflections* (NOI) [89] counts the number of local minima in the cost curve assuming that the lower, the more confident is the disparity assignment.

5.3.2 Hardware challenging

Confidence measures belonging to this category can not be directly implemented in embedded systems following their original formulation. We consider *peak ratio* (PKR) [89] which computes the ratio between c_{2m} and c_1 and its variant *peak ratio naive* (PKRN) [89] which replaces c_{2m} with the second minimum c_2. According to the literature, these measures are quite effective but seldom deployed in embedded stereo cameras. Another popular measure is *winner margin measure* (WMN) [89] which normalizes the difference between c_{2m} and c_1 by the sum of all costs. Its variant *winner margin measure naive* (WMNN) [89] follows the same strategy replacing c_{2m} with c_2. The *left-right difference* measure (LRD) [89] computes the difference between c_2 and c_1 divided by the absolute difference between c_1 and the minimum cost of the homologous point in R (c_1^R). For these confidence measures the major implementation issue on embedded systems is represented by the division. For the remaining confidence measures the main problem is represented by transcendental functions: exponentials and logarithms. *Maximum likelihood measure* (MLM) [89] and *attainable maximum likelihood* (AML) [89] infer from the cost curve a *probability density function* (pdf) related to an ideal c_1, respectively, equal to zero for MLM and to c_1 for AML. A more recent and less computational demanding approach *perturbation* (PER) [77], encodes the deviation of the cost curve from a Gaussian function ant its implementation requires a division by a constant value suited for a LUT-based strategy. Finally, we also mention two very effective confidence measures based on distinctiveness, namely *distinctive similarity measure* (DSM) and *self-aware matching measure* (SAMM) and one *negative entropy measure* (NEM) [89] that infers the degree of uncertainty of each disparity assignment from the negative entropy of c_1. However, they require additional cues (e.g., self-matching costs on both reference and target images for SAMM) not well suited to embedded systems and thus not included in our evaluation.

5.4 Experimental results

In this section we evaluate the 16 confidence measures previously reviewed and implemented following the design strategies outlined so far. We test their effectiveness with the output of two popular stereo algorithms well-suited for implementation on embedded systems: AD-CENSUS and SGM.

We encode matching costs with, respectively, 6 and 8 bit integer values, being this amount enough to encode the entire ranges. Regarding parameters of the confidence measures: for LC, we set the normalization factor γ to 1 to avoid division, while for PER, MLM and AML we set s_{PER} to 1.2 and σ_{aml}, σ_{mlm} to 2 before initializing the LUTs. The other 12 confidence measures do not have parameters.

For CUR, LRC, LC, MM, MMN, MSM, NOI and UC we provide experimental results with the conventional implementation since their mapping on embedded devices is totally equivalent. Moreover, regarding PER, we do not report results concerned with division by the closest power of two being the divisor a constant value and thus such operation can be addressed with a LUT. Finally, it is worth observing that most embedded stereo vision systems rely on LRC [20, 87] and UC [20, 150] for confidence estimation.

measure	standard
Opt.	0.08891
CUR	0.24377 (14)
LRC	0.19933 (7)
LC	0.24377 (15)
MM	0.17765 (6)
MMN	0.19933 (8)
MSM	0.23182 (13)
NOI	0.39053 (16)
UC	0.20974 (10)

measure	standard
Opt.	0.08891
AML	0.21173 (11)
LRD	0.17004 (3)
MLM	0.22413 (12)
PER	0.20687 (9)
PKR	0.16250 (1)
PKRN	0.17185 (5)
WMN	0.16503 (2)
WMNN	0.17169 (4)

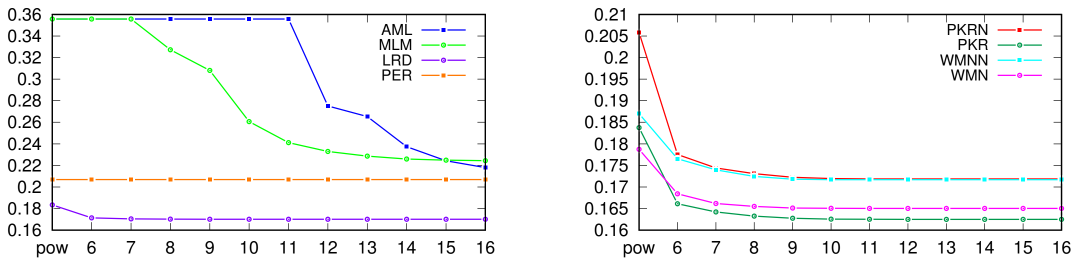
(a)

measure	standard
Opt.	0.04367
CUR	0.11602 (11)
LRC	0.16853 (15)
LC	0.11602 (12)
MM	0.09371 (5)
MMN	0.12920 (14)
MSM	0.10181 (7)
NOI	0.32028 (16)
UC	0.10347 (9)

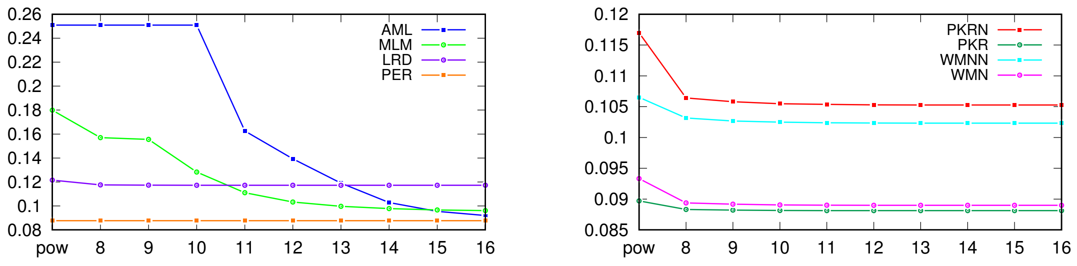
measure	standard
Opt.	0.04367
AML	0.08843 (3)
LRD	0.11725 (13)
MLM	0.09567 (6)
PER	0.08766 (1)
PKR	0.08813 (2)
PKRN	0.10527 (10)
WMN	0.08898 (4)
WMNN	0.10232 (8)

(b)

Table 5.1: Experimental results, in terms of AUC, on Middlebury 2014 dataset with AD-CENSUS (a) and SGM (b) algorithms for the 16 confidence measures using a conventional software implementation. In red, top-performing measure. We also report the absolute ranking.



(a)



(b)

Figure 5.2: Average AUC values on the Middlebury 2014 dataset for hardware challenging measures, varying the implementation settings (i.e., *pow* and number of bits of fixed-point arithmetic). (a) AD-CENSUS, (b) SGM algorithm.

5.4.1 Experiments

In this section we report results on Middlebury 2014 and KITTI 2015 datasets in terms of average AUC values achieved by confidence measures implemented in software. For hardware challenging measures of section 5.3.2 we also report multiple AUC obtained with increasing number of bits dedicated to fixed point operations (i.e., from 6 to 16 for AD-CENSUS and from 8 to 16 for SGM, so as to handle the whole cost range). Moreover, for such measures, we also report the results obtained by rounding to the closest power of 2 and, then, shifting right (referred to as *pow* in the charts).

Table 5.1 shows for Middlebury 2014 that LRC and UC, confidence measures typically deployed in embedded stereo cameras, are less effective than MM, LRD, PKR, PKRN, WMN, WMNN with AD-CENSUS and MM, MSM, AML, MLM, PER, PKR, WMN, WMN with SGM. We can notice that LRC provides poor confidence estimation with SGM but achieves better results with AD-CENSUS while UC has average performance with both algorithms. Considering the more effective confidence measures in the table,

measure	standard
Opt.	0.08055
CUR	0.30692 (14)
LRC	0.20018 (2)
LC	0.30692 (15)
MM	0.20601 (4)
MMN	0.24588 (11)
MSM	0.25571 (13)
NOI	0.31160 (16)
UC	0.22324 (8)

measure	standard
Opt.	0.04367
AML	0.23053 (10)
LRD	0.20706 (5)
MLM	0.25180 (12)
PER	0.22575 (9)
PKR	0.19821 (1)
PKRN	0.20931 (7)
WMN	0.20221 (3)
WMNN	0.20795 (6)

(a)

measure	standard
Opt.	0.01618
CUR	0.08585 (11)
LRC	0.10377 (15)
LC	0.08585 (12)
MM	0.06374 (8)
MMN	0.09549 (14)
MSM	0.05999 (5)
NOI	0.16308 (16)
UC	0.06310 (7)

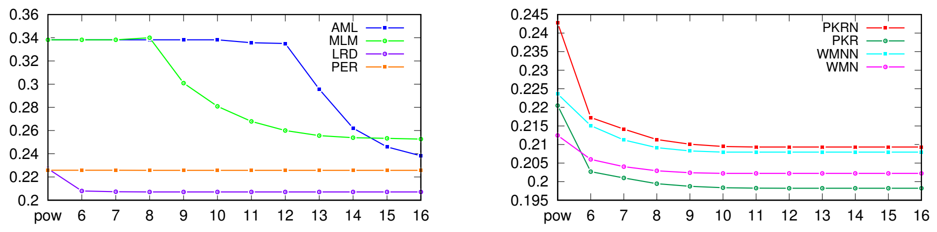
measure	standard
Opt.	0.01618
AML	0.05738 (2)
LRD	0.08744 (13)
MLM	0.05889 (3)
PER	0.05657 (1)
PKR	0.06003 (6)
PKRN	0.07611 (10)
WMN	0.05970 (4)
WMNN	0.07149 (9)

(b)

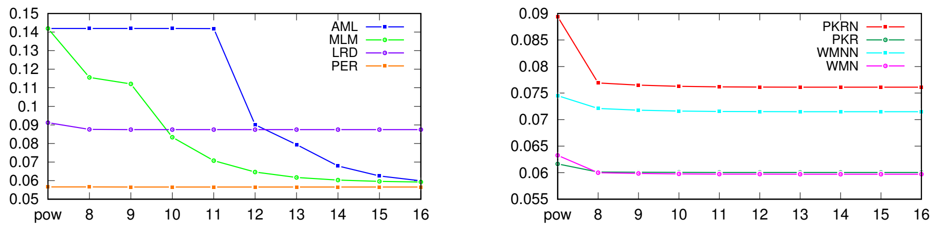
Table 5.2: Experimental results, in terms of AUC, on KITTI 2015 dataset with AD-CENSUS (a) and SGM (b) algorithms for the 16 confidence measures using a conventional software implementation. In red, top-performing measure. We also report the absolute ranking.

we can notice that PKR and WMN, as well as their naive formulations, performs pretty well with both algorithms clearly providing much more accurate confidence estimation compared to LRC and UC. Moreover, we can notice that PER achieves the best performance with SGM but it does not perform as well with AD-CENSUS, yielding slightly better confidence predictions with respect to UC. Specularly, LRD provides very reliable predictions with AD-CENSUS but poor results with SGM. Finally, we point our that top-performing confidence measures always belong to the hardware challenging category.

Therefore, in Figure 5.2 we report the performance of hardware challenging confidence measures, on Middlebury 2014 with AD-CENSUS and SGM, with multiple simplification settings. Observing the charts, PER is independent of the adopted strategy, being based on a LUT. Moreover, excluding PER, we can notice that the best performing ones (PKR, PKRN, WMN and WMNN at the right side of the figure) are those less affected by the number of bits deployed for fixed-point computations, thus resulting in reduced computational resources. In particular, we can observe that with only 8 bits, PKR and WMN achieve with both algorithms results almost comparable to their conventional software implementation. A similar behavior can be observed, with slightly worse performance, for their naive formulation PKRN and WMNN and for LRD that, excluding PER, is the approach less dependent of the number of bits. On the other hand, AML e MLM with both algorithms are significantly affected by the number of bit deployed for their implementa-



(a)



(b)

Figure 5.3: Average AUC values on the KITTI 2015 dataset for hardware challenging measures, varying the implementation settings (i.e., *pow* and number of bits of fixed-point arithmetic). (a) AD-CENSUS, (b) SGM algorithm.

tion achieving results comparable to their traditional software formulation, respectively, only with 13 and 16 bits. Finally, excluding PER, we can observe that dividing by a power of 2 always provides poor results with respect to other simplifications. However, we highlight that even with this very efficient implementation strategy, PKR, WMN outperform LRC and UC with both stereo algorithms. Thus, trading simplified computations with memory footprint leads to design better alternatives to standard confidence measures for embedded systems.

Table 17.2 reports the average AUCs for the two considered stereo algorithms on KITTI 2015 for software implementation of the 16 confidence measures. Compared to Table 5.1 we can notice a similar behavior with a notable difference. In fact, observing Table 17.2 we highlight that LRC achieves almost optimal results on AD-CENSUS but yields very poor performance with SGM. Looking at the behavior of the hardware challenging measures, reported in Figure 5.3, we observe on KITTI 2015 a substantially similar behavior with respect to Figure 5.2 concerned with Middlebury 2014.

5.5 Conclusions

In this chapter we have evaluated confidence measures suited for embedded stereo cameras. Our analysis shows that conventional approaches, LRC and UC, are outperformed by other considered solutions, whose implementation on embedded devices enables to achieve more accurate confidence predictions with a negligible amount of hardware resources and/or computations. In particular, according to our evaluation on Middlebury 2014 and KITTI 2015, PKR and WMN represent the overall best choice when dealing with two popular algorithms, AD-CENSUS and SGM, frequently deployed for embedded stereo systems.

Chapter 6

Even more confident predictions with deep machine-learning

The content of this chapter has been presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (EVW 2017) - "Even More Confident predictions with deep machine-learning" [183].

6.1 Introduction

Several measures obtained by processing different cues from the cost volume, disparity maps or input images have been proposed. Following this observation, state-of-the-art approaches focused on combining multiple, possibly *orthogonal*, confidence measures by means of machine-learning frameworks based on random-forests.

These results and the effectiveness of deep machine-learning applied to computer vision problems motivated us to inquire about the opportunity to achieve more accurate confidence estimation leveraging on Convolutional Neural Networks (CNNs). Figure 6.1, considering a sample from the KITTI 2015 dataset, shows the disparity map computed by a local stereo algorithm and two confidence maps obtained processing the same input features, respectively, by means of a state-of-the-art approach [169] based on a random-

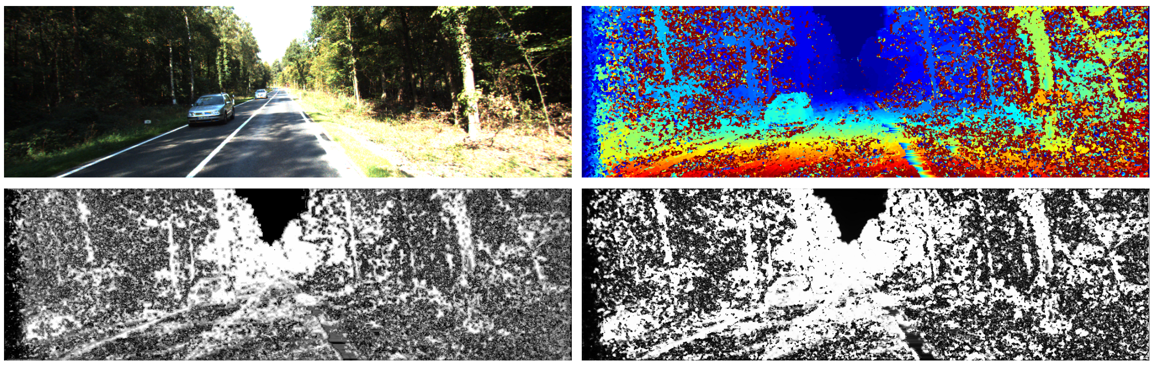


Figure 6.1: Comparison between confidence measures obtained by [169] and by our proposal processing the same input features. From left to right, in the first row left image and the corresponding disparity map, in the second row confidence map computed by a random forest and confidence map computed by our CNN-based method. In disparity maps, warm colors encodes closer points. In confidence maps brighter values encode more confident disparity.

forest and our CNN-based proposal. We can observe from the figure how the confidence map obtained with deep-learning provides "*Even More Confident*" (EMC) predictions. In particular, the random-forest approach in (c) sets a large amount of points to intermediate scores being not sure enough about their actual reliability. On the other hand, our proposal (d) clearly depicts much more polarized scores. In section 13.3 we'll report quantitative results confirming the advantages yielded by our strategy.

Differently from approaches relying on random-forest classifiers that infer, for each point, an estimated match reliability by processing a 1D input feature vector made of point-wise confidence measures and features, our proposal relies on a more distinctive 3D input domain. Such input domain, for the point under analysis, is made of patches extracted from multiple input confidence and feature maps around the examined point as shown in Figure 6.2. Leveraging on a CNN, our proposal is able to infer more meaningful confidence estimations with respect to a random forest fed with the same input data. Doing so, our approach moves from the single pixel confidence strategy adopted by most state-of-the-art methods to a patch-based domain in order to exploit more meaningful local information.

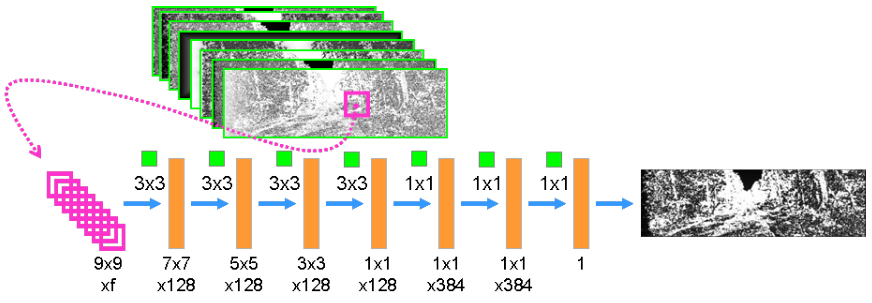


Figure 6.2: Architecture of CNN with highlighted in purple the confidence measures and features processed in a 3D domain by our method.

We validate our method as follows. Once selected a subset of stereo pairs from the KITTI 2012 [67] training dataset, we run a fast local stereo algorithm, using as matching cost the census transform plus Hamming distance, a cost function common to previous works [169, 178]. From the outcome of the previous phase we compute a pool of confidence measures and features training a random forest and our CNN framework on such data. In particular, we choose as input confidence measures and features the same adopted by state-of-the-art methods [228], [169] and [178] based on random-forest frameworks. Then, we evaluate the effectiveness of our proposal with respect to [228], [169] and [178] by means of ROC curve analysis [89], on the remaining portion of KITTI 2012. Moreover, we cross-validate without re-training on KITTI 2015 and Middlebury 2014.

6.2 Deep learning for confidence measures

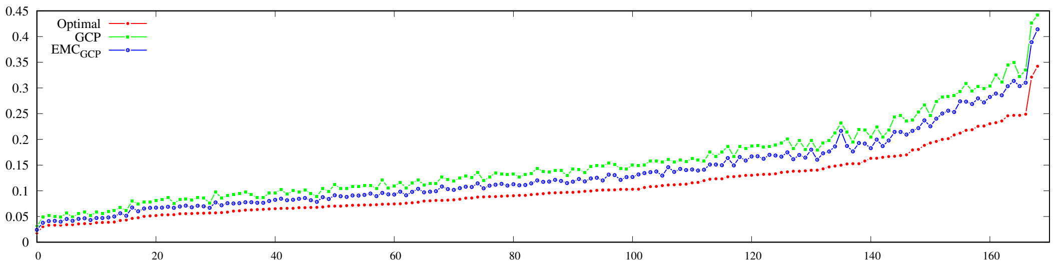
In this chapter, we follow the successful strategy of combining multiple confidence measures through supervised learning, by exploiting CNN. Such solution greatly increases the amount of information processed when predicting confidence with respect to conventional random-forest classifiers. In particular, by processing confidences and other hand-crafted features as images, our approach moves from the 1D features domain of the random forest classifiers to a more distinctive 3D domain, encoding local behavior of features and, thus, going beyond single pixel confidence analysis . Two dimensions are given by the image domain and one by the features domain as shown in Figure 6.2.

In [77] the random-forest classifier is fed with a feature vector F containing f different features, obtained according to f functions (e.g., multiple confidence measures computed at different scales). Although this strategy and the others inspired by this method [169, 178, 228] enabled remarkable improvements, the random forest classifier takes as input a 1D feature domain made of elements of F, encoding pixel-wise properties.

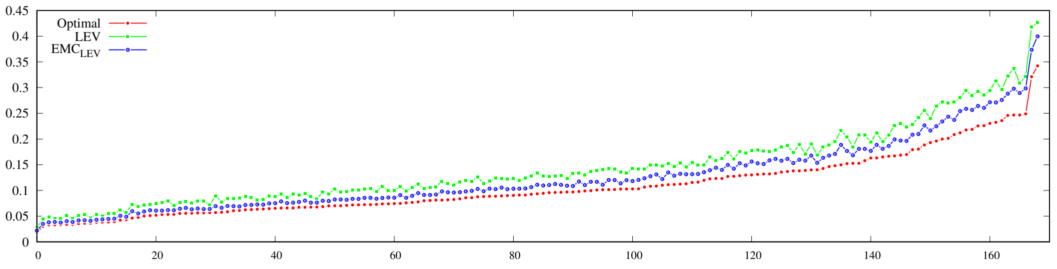
By moving into the deep learning domain, we can imagine this feature vector F as a set of f general purpose feature maps that might be generated by a generic convolutional layer C_i and fed as input to the following one C_{i+1}. According to this observation, we model our framework as a CNN with a first layer H in charge of extracting a set of hand-crafted feature maps. Excluding the front-end layer H, the remaining portion of the deep architecture is trained according to the number input feature maps provided by such layer. For example, adopting the same input features of [228] in our framework, the H front-end would provide to the first convolutional layer of the deep network the following eight feature maps described in [228]: MSM, MMN, AML, LRC, LRD, distance to border, distance to discontinuities and median deviation of disparity.

6.3 Experimental Results

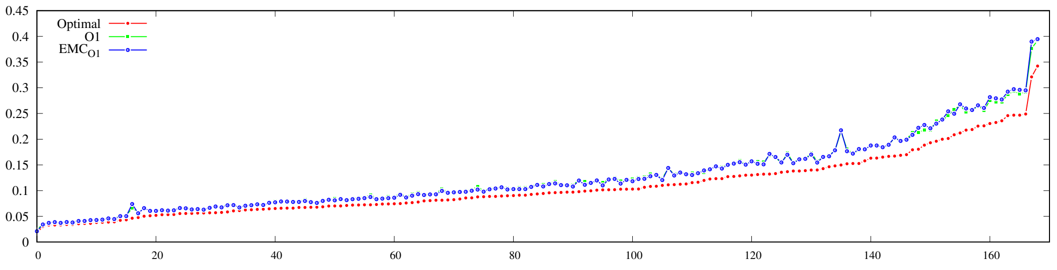
To evaluate our proposal, we feed our network with multiple stand-alone confidence measures and hand-crafted features comparing the results with state-of-the-art confidence measures [169, 178, 228] based on random-forest frameworks. We perform a single training on a portion of the KITTI 2012 dataset (25 out of 194 total images), then we test the methods on the remaining stereo pairs available, deployed as evaluation set. Moreover, we further cross-validate the confidence measures on KITTI 2015 (200 images) and Middlebury 2014 datasets (15 images).



(a)



(b)



(c)

Figure 6.3: AUC values on the KITTI dataset. Each value on the plot represent the AUC on a single image of the dataset, sorted in non-descending order according to their optimal values. We report, from top to bottom, comparison between GCP and EMC_{GCP} (a), LEV and EMC_{LEV} (b), O1 and EMC_{O1} (c). Cost volumes obtained by census based fixed window algorithm.

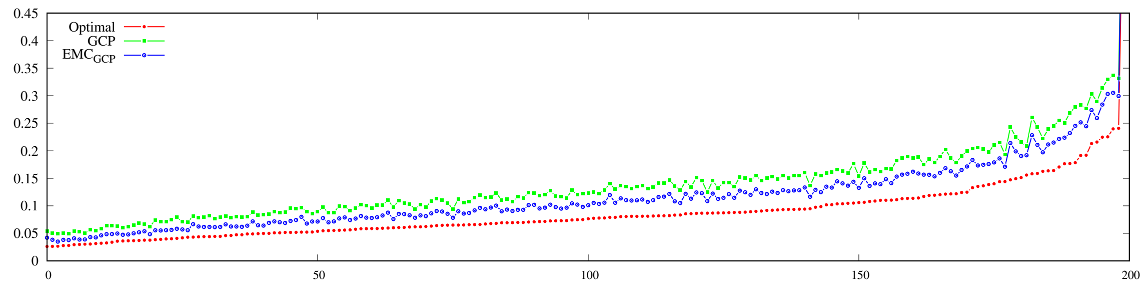
6.3.1 Training phase

We trained our network according to *stochastic gradient descend*, we choose the *binary cross entropy* as loss function, according to the regression problem we are dealing with. We trained on nearly 3.5 million samples, obtained from the first 25 stereo pairs of the KITTI 2012 training dataset. Each sample corresponds to a volume of $9 \times 9 \times f$ patches output of the H layer, each one centered on a pixel with provided ground truth available in the dataset. We define a batch size of 128 training samples, training for 5 *epochs*, corresponding to nearly 135 thousand iterations, with a 0.002 learning rate and 0.8 momentum. We applied training samples shuffling.

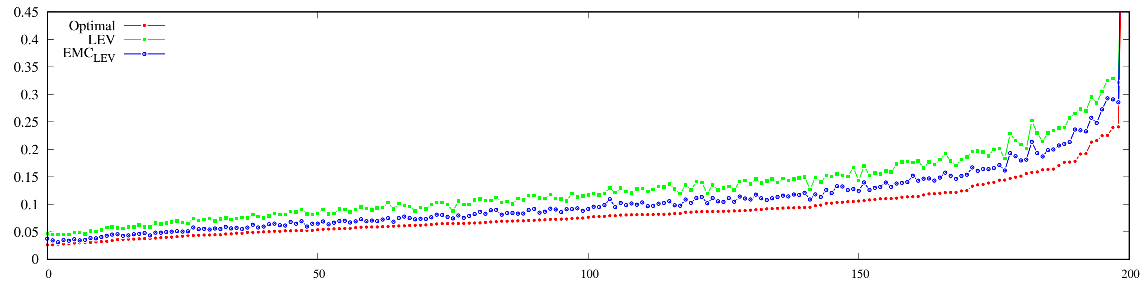
The stereo algorithm used to generate matching costs for the training phase consists of a 5×5 census based data term, aggregated on a fixed local window of size 5×5. We set as error threshold the value 3, commonly adopted to compute the error rate of the stereo algorithms on the most popular datasets [67, 155]. Samples concerning pixels with a disparity assigned by the fixed window aggregation lower than the threshold are labeled with high confidence (1 values). For a fair evaluation, we compare the proposed methodology with random-forests trained on the same amount of data. In our experiments, we choose [228], [169] and [178], representing state-of-the art confidence measures inferred by random-forest frameworks. During the validation, these three methods will be referred to as, respectively,

6.3.2 EMC vs random-forest

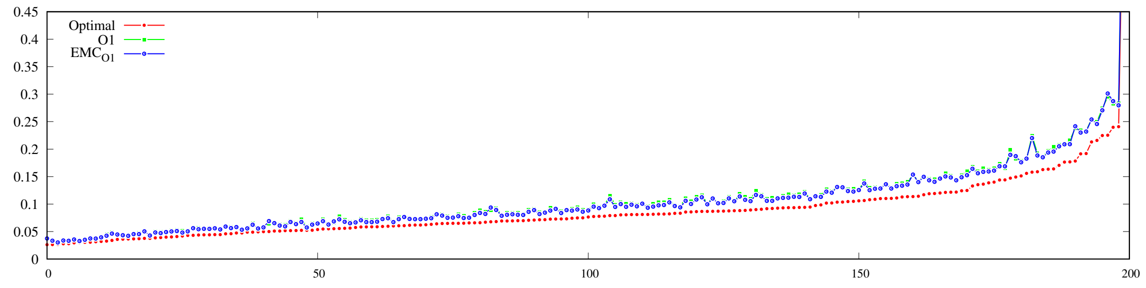
Figure 6.3 depicts three plots, containing the AUC values computed over the entire KITTI 2012 (excluding the images processed during training) of both the EMC approach and the corresponding random forest counterpart, for GCP [228], LEV [169], O1 [178]. The curves are plotted in non-descending order according to optimal values (red), together with curves related to random forest implementation (referred to as GCP, LEV and O1, plotted in green) and our method processing the same inputs (referred to as EMC_{GCP},



(a)



(b)



(c)

Figure 6.4: AUC values on the KITTI 2015 dataset. Each value on the plot represent the AUC on a single image of the dataset, sorted in non-descending order according to their optimal values. We report, from top to bottom, comparison between GCP and EMC_{GCP} (a), LEV and EMC_{LEV} (b), O1 and EMC_{O1} (c). Cost volumes obtained by census based fixed window algorithm.

	KITTI 2012			KITTI 2015			Middlebury 2014		
	GCP	LEV	O1	GCP	LEV	O1	GCP	LEV	O1
Optimal	0.107802			0.088357			0.068375		
RF	0.152764	0.144077	0.127645	0.139611	0.131662	0.108812	0.109302	0.104146	0.090908
EMC	0.133684	0.125211	0.126898	0.117551	0.107969	0.106523	0.091749	0.088473	0.089928
Δ_k	-42.44%	-52.01%	-3.76%	-43.04%	-54.71%	-11.19%	-42.88%	-43.81%	-4.35%

Table 6.1: Average AUC values on the three dataset, KITTI 2012, KITTI 2015 and Middlebury 2014 from left to right respectively. First row reports optimal AUC values according to [89], second row shows values concerning the random forest implementation of GCP [228], LEV [169] and O1 [178], third row shows results achieved by EMC implementation. Final row shows the improvement Δ_k led by EMC with respect to optimal AUC values. Cost volumes obtained by census based fixed window algorithm.

EMC$_{LEV}$ and EMC$_{O1}$, plotted in blue). In particular, from top to bottom, (a) concerns with GCP versus EMC$_{GCP}$, (b) with LEV versus EMC$_{LEV}$, (c) with O1 vs EMC$_{O1}$. As we can observe, for the first two experiments the EMC implementations achieves lower AUC values, thus closer to optimal values. From the AUC curve, it's evident how the EMC framework outperforms the random forest on each image of the dataset. Concerning O1, our implementations performs very similarly to the original proposal [178], but on average it achieves a better AUC on the entire dataset.

Figure 6.4 depicts the three plots for the entire KITTI 2015, comparing the EMC approach with the corresponding random forest counterpart, for GCP [228], LEV [169], O1 [178]. Optimal values are plotted in red, curves related to random forest implementation (referred to as GCP, LEV and O1, plotted in green) and our method processing the same inputs (referred to as EMC$_{GCP}$, EMC$_{LEV}$ and EMC$_{O1}$, plotted in blue). In particular, top graph (a) concerns with GCP versus EMC$_{GCP}$, the second one (b) with LEV versus EMC$_{LEV}$, the final (c) with O1 vs EMC$_{O1}$. The behavior observed on KITTI 2012 is confirmed, GCP and LEV features achieve major improvements when processed within EMC framework with respect to random forest, while we can observe a minor improvement concerning O1.

Figure 6.5 shows three plots concerning the evaluation on the Middlebury 2014 dataset. As for the previous figures, optimal values are plotted in red, curves related to random

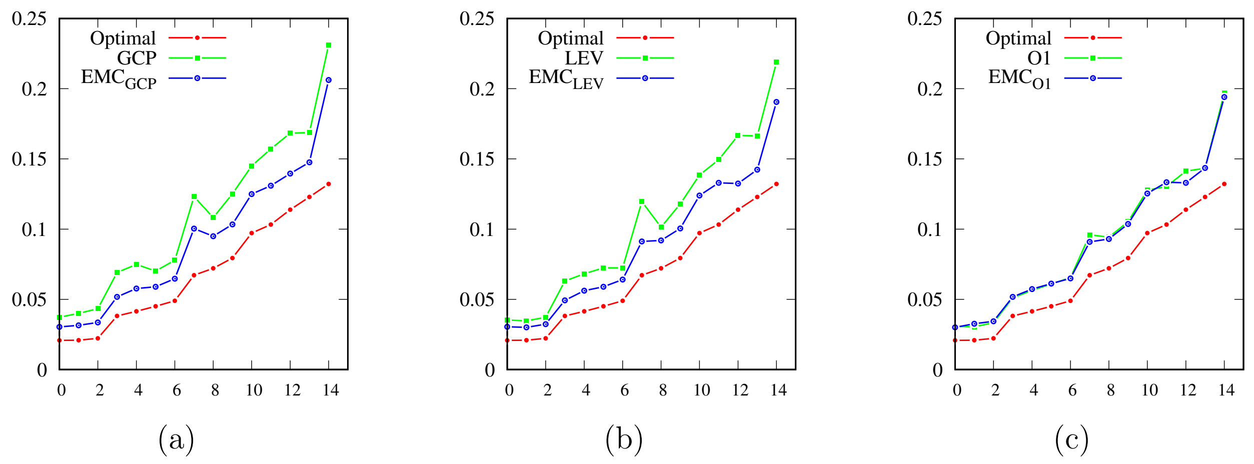


Figure 6.5: AUC values on the Middlebury dataset. Each value on the plot represent the AUC on a single image of the dataset, sorted in non-descending order according to their optimal values. We report, from top to bottom, comparison between GCP and EMC_{GCP} (a), LEV and EMC_{LEV} (b), O1 and EMC_{O1} (c). Cost volumes obtained by census based fixed window algorithm.

	KITTI 2012		
	GCP	LEV	O1
EMC win rate	169/169	169/169	122/169
	KITTI 2015		
	GCP	LEV	O1
EMC win rate	200/200	200/200	181/200
	Middlebury 2014		
	GCP	LEV	O1
EMC win rate	15/15	15/15	8/15

Table 6.2: EMC win rate on the three dataset, KITTI 2012, KITTI 2015 and Middlebury (i.e., number of images per dataset on which EMC outperforms the random forest) from top to bottom respectively. First row reports optimal AUC values according to [89], second row shows values concerning the random forest implementation of GCP [228], LEV [169] and O1 [178], third row shows results achieved by EMC implementation. Final row shows the improvement Δ_k led by EMC with respect to optimal AUC values. Cost volumes obtained by census based fixed window algorithm.

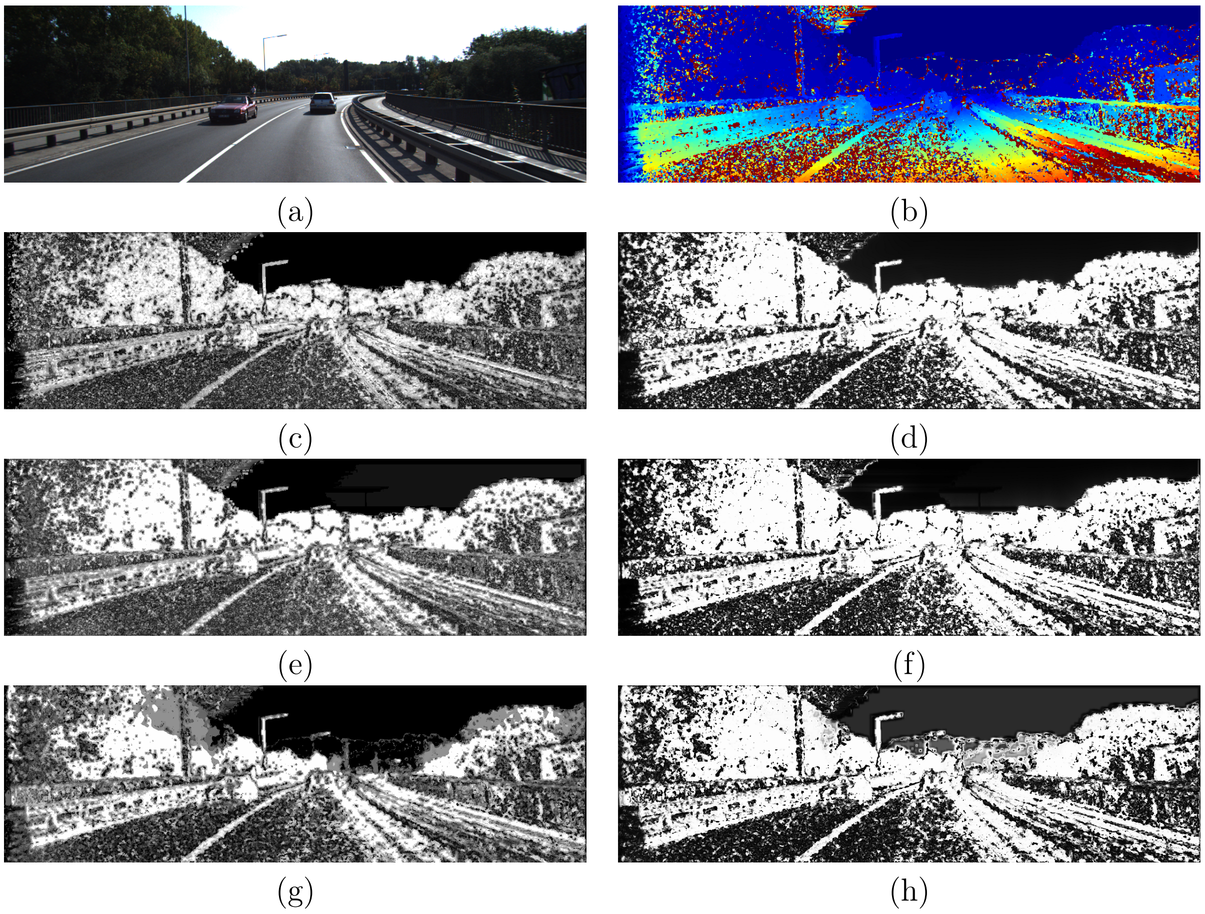


Figure 6.6: Confidence maps obtained by random forest and EMC. The disparity map is concerned with the considered stereo algorithm on pair 000176 of the KITTI 2015 dataset. Reference image (a) , disparity map (b), confidence map obtained by GCP [228] using a random forest (c) and EMC (d), confidence map obtained by LEV [169] using a random forest (e) and EMC (f), confidence map obtained by O1 [178] using a random forest (g) and EMC (h).

forest implementation are in green (referred to as GCP, LEV and O1) and those related to EMC processing the same inputs (referred to as EMC_{GCP}, EMC_{LEV} and EMC_{O1}). In particular, from left to right, (a) concerns with GCP versus EMC_{GCP}, (b) with LEV versus EMC_{LEV}, (c) with O1 vs EMC_{O1}. The three confidence measures confirm the behaviors already highlighted on the KITTI datasets.

To further perceive the improvements lead by our framework (and, concerning O1, to highlight its behavior more clearly), we report AUC values averaged over each of the three datasets for the three confidence measures, for both random forest and EMC implementations. We report two aspects allowing for such comparison. The first is the variation of average AUC achieved by EMC implementation of confidence measure k with respect to its random forest counterpart and optimal value, referred to as Δ_k and obtained as:

$$\Delta_k = \frac{AUC_k - AUC_{EMC_k}}{AUC_k - AUC_{opt}} \tag{6.1}$$

Negative values of this variation reflects an improvement achieved by EMC, while positive stand for a worse confidence prediction. The second is the win rate, as the number of images on which EMC achieves a lower AUC with respect to its random forest counterpart. Table 6.1 reports average AUC for each confidence measure (GCP, LEV, O1) on the three datasets KITTI 2012, KITTI 2015 and Middlebury. The first row reports optimal AUC, according to [89], averaged over each dataset, then AUC concerning both implementations (referred to as, respectively, RF for random forest, EMC for our approach). Finally, Δ_k highlights the effectiveness of the CNN with respect to the random forest. We can observe how on the KITTI 2012 dataset the improvement yielded by our method is, concerning GCP and LEV, higher than 40%, respectively, 42.44% with respect to GCP and 52.01% with respect to LEV. These results are confirmed on the KITTI 2015 dataset, reporting Δ_k very close to the previous ones, and on Middlebury 2014, on which LEV achieve a lower, yet important Δ_k value. Focusing on O1, the improvement is lower, between 3% and 12% (the higher is on KITTI 2015, -11.19%) on the three datasets. This may be caused by the higher accuracy of the random forest implementation compared to GCP and LEV solutions, or to the nature of the features extracted by O1, all processed from the disparity map only and, probably, encoding less different behaviors with respect to GCP and LEV features. Nonetheless, on average with O1, EMC is more effective than the random forest counterpart. Table 6.2 reports the win rate achieved by EMC for each confidence measure on the three datasets. While EMC outperforms random forests on all the stereo pairs of the three datasets for GCP and LEV (i.e., 100% win rate), it wins 122 out of 169 times on KITTI 2012, 181 out of 200 on KITTI 2015 (confirming to be more effective on this dataset) and 8 out of 15 on Middlebury for O1, confirming to be less effective, but still outperforming random forest implementation on average. We would like to point-out that the training procedure did not take into account any of the KITTI 2015 nor Middlebury 2014 data for random forest approaches and EMC. This evaluation proves how the effectiveness of the CNN-based proposal implementation result is kept processing different data. This fact (i.e., the capability to generalize to new data) represents a notable result for a machine-learning framework. Finally, Figure 6.6 reports a qualitative comparison of confidence maps obtained by random forest and EMC, respectively, with GCP (c,d), LEV (e,f) and O1 (g,h), for a stereo pair from KITTI 2015 dataset.

6.4 Conclusions

In this chapter we tackled the confidence prediction problem exploiting a deep network to combine multiple confidence and feature maps. Differently from state-of-art approaches based on random-forest framework processing input features in a 1D domain, our proposal relies on more distinctive features in the 3D domain enabling to extract more effective confidence predictions. Extensive experimental results show that our proposal improves the effectiveness of top-performing approaches based on random-forest when fed with the same input features and trained on the same amount of data.

Chapter 7

Beyond local reasoning for stereo confidence estimation with deep learning

The content of this chapter has been presented at the European Conference on Computer Vision (ECCV 2018) - "Beyond local reasoning for stereo confidence estimation with deep learning" [240].

7.1 Introduction

Among the many confidence estimators proposed in the literature methods using as input cue information extracted from the disparity domain only [178, 179, 217] proved to be particularly effective. Compared to approaches relying on cues extracted from the cost volume or other strategies known in the literature, these methods currently represent state-of-the-art.

Regardless of the strategy adopted, all these methods estimate confidence with a relatively small receptive field intrinsic in their local patch-based nature. Increasing such parameter in these methods does not enable significant improvements and may also lead

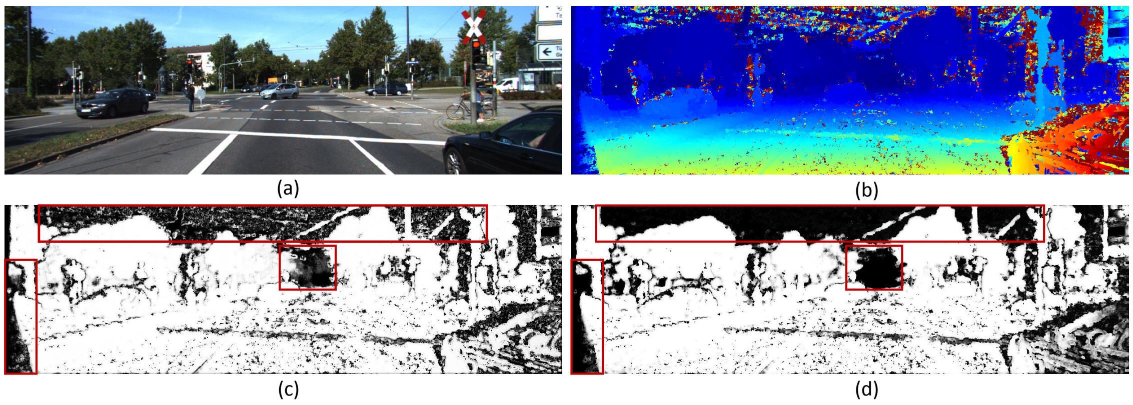


Figure 7.1: Example of confidence estimation. (a) Reference image from KITTI 2015 dataset [155], (b) disparity map obtained with MC-CNN [279], (c) confidence estimated with a local approach (CCNN [179]) and (d) the proposed local-global framework, highlighting regions on which the latter method provides more reliable predictions (red bounding boxes).

to poor results. Thus, state-of-the-art methods do not take advantage of the whole image and disparity content. Although this strategy is undoubtedly valid, on the other hand, it seems clear that by looking at the whole reference image and disparity map matters for uncertainty estimation. This fact can be readily perceived by observing Figure 7.1 in the highlighted areas.

In particular, considering more global reasoning on the whole image and disparity content can improve the prediction for disparity values more unlikely to occur (e.g., objects extremely close to the camera), at the cost of a smoother prediction. This task can be undertaken by architectures with a large receptive field such as encoder-decoder models thus less accurate in the presence of high-frequency noise (e.g., outliers on the output of stereo algorithms such as AD-CENSUS or other matching functions). On the other hand, networks working on patches detect very well this kind of outliers but they are not able to capture farther information.

Therefore, in this chapter, we propose to overcome this limitation by combining the best of the two worlds (i.e., networks based on small and large receptive fields). We do this by deploying a CNN-based architecture able to extract nearby and far-sighted cues, in the RGB and disparity domains, and to merge them to obtain a more accurate confidence estimation. By training a multi-modal cascaded architecture we first obtain two confidence predictions by reasoning respectively on local and farther cues, then we further elaborate on them to obtain a final, more accurate prediction. Figure 7.1 shows qualitatively how this strategy enables to estimate more reliable confidence scores.

To the best of our knowledge, our proposal is the first one enabling to i) exploit more global context for learning confidence predictions and ii) combine this novel technique with local approaches to design an effective local-global confidence measure. From now on, we will define as *global*, with abuse of language, a strategy going beyond traditional neighboring boundaries usually adopted in the field of confidence estimation. We extensively evaluate the proposed framework on three popular datasets, KITTI 2012 [66], KITTI 2015 [155] and Middlebury 2014 [213] using three popular algorithms used in this field, respectively, AD-CENSUS [276], MC-CNN-fst matching cost [279] and SGM [82]. Such exhaustive evaluation clearly highlights that our proposal is state-of-the-art.

7.2 Method overview

In this section, we introduce our local-global framework for confidence estimation. Driven by the recent success of confidence measures obtained by processing cues in the disparity domain only, and in particular those based on deep learning [63, 179, 217], we look beyond the small local neighborhood taken into account for each pixel by these methods and we analyze global context from both RGB and disparity domains to obtain a more consistent confidence estimation. Being local and global approaches characterized by complementary strengths, respectively the formers are very effective at detecting high-frequency patterns while the latter can incorporate much more cues from the surrounding pixels, we argued that combining them can further improve confidence estimation by overcoming the specific limitations of the single approaches. To do so, we will deploy two main architectures, respectively in charge of process local and global context. Then, the output of these two networks is combined to obtain the final prediction. In Section 7.2.1 we describe the local

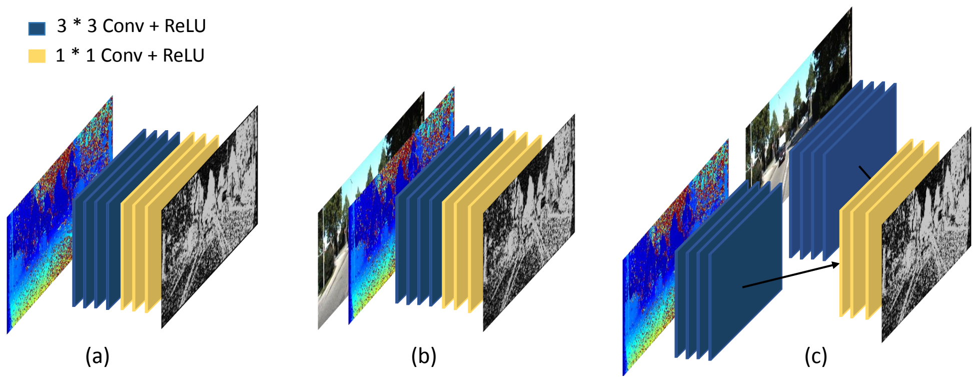


Figure 7.2: Local architectures, respectively (a) CCNN [179], (B) EFN [63] and (c) LFN [63]. The networks uses 3×3 (blue) and 1×1 convolutional layers, all followed by ReLUs except the last one.

network, for which we choose state-of-the-art CCNN measure [179] and its extensions proposed in [63]. In Section 7.2.2 we introduce a novel architecture for *global* confidence estimation referred to as *ConfNet*, inspired by works concerning end-to-end stereo matching [151]. Finally, in Section 7.2.3 we outline our overall local-global framework combining cues generated by local and global approaches.

7.2.1 Local approaches

With local approaches, we refer to methodologies aimed at estimating the confidence score for a single pixel by looking at nearby pixels laying on a small local neighborhood. PBCP [217], CCNN [179] and multi-modal approaches [63] belongs to this category. We use the two latter techniques in our framework, depicted in Figure 7.2, because of the superior outliers detection performance achieved by the first [184] further improved, in some circumstances, by multi-modal networks [63]. Another reason to use CCNN-based networks is that both can be computed without requiring the right disparity map, required by PBCP [217], not always available in some circumstances as previously highlighted.

CCNN.

This confidence measure is obtained by processing the disparity map through a shallow network, made of 4 convolutional layers with 3×3 kernels producing 64 features map at each level, followed by 2 convolutional layers with 1×1 kernels producing 100 features map and a final 1×1 convolution followed by Sigmoid activation to obtain confidence scores in $[0, 1]$ interval. All the other layers are followed by ReLU non-linearities. The first 4 layers do not apply any explicit padding to its input, thus reducing input size by 2 pixels on both height and width (i.e., 1 pixel on each side). This makes the single pixel confidence prediction bound to a 9×9 local patch, the receptive field of the network, centered on it. The fully convolutional nature of this model allows for training on image patches and then performs a single forward of a full resolution disparity map if properly padded (i.e., applying 4 pixel padding on each side).

Multi-modal networks.

In [63] the authors propose to improve CCNN [179] by feeding to the network additional information from the RGB reference image. To this aim Fu et al. propose two strategies, respectively, the Early Fusion Network (EFN) and the Late Fusion Network (LFN). In the EFN, RGB and disparity patches are concatenated to form a 4-channel input, processed by a shallow network with the same structure of CCNN, but different number of channels at each layer (i.e., 112 for 3×3 and 384 for 1×1 convolutions). In the LFN, the information from the two domain is processed into two different streams, obtained by building two towers made of four 3×3 convolutional kernels without sharing the weights between them, in order to learn domain specific features representations. The outputs of the two towers are then concatenated and processed by the final 1×1 convolutions. Final outputs pass through a Sigmoid activation as for CCNN. The number of channels are the same as for EFN model. Both models have been trained and compared with CCNN, proving to perform better when trained with a much larger amount of samples compared to the

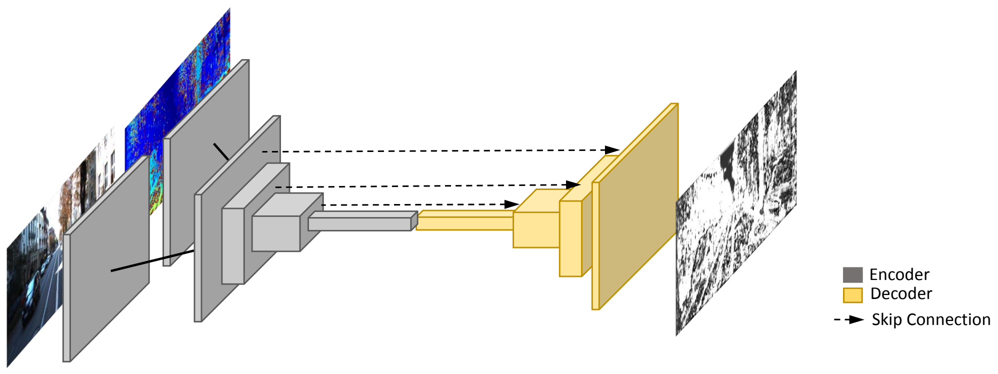


Figure 7.3: ConfNet architecture. Encoding blocks (light blue) are made by 3×3 convolutions followed by batch normalization, ReLU and max-pooling. Decoding blocks (yellow) contains 3×3 deconvolutions and 3×3 convolutions to reduce grid artifacts.

amount (i.e., 94 stereo pairs versus 20) typically deployed in this field [184]. The receptive field of both networks is the same of CCNN (9×9).

7.2.2 Proposed global approach

In this section, we describe the network architecture designed to infer confidence prediction by looking at the whole image and disparity content.

ConfNet.

Inspired by recent works in stereo matching [109, 151, 167], we design an encoder/decoder architecture enabling a large receptive field and at the same time maintaining the same input dimensions for the output confidence map. Figure 7.3 shows an overview of the ConfNet architecture. After concatenating features computed by 3×3 convolutional layers from both RGB reference image and disparity map, they are forwarded to the first part of the network, made of 4 encoding blocks. Each of them is made of a 3×3 convolutional layer ReLU activations and a 2×2 max-pooling used to decimate the input dimension and thus to increase the receptive field. More precisely, after the fourth block the original resolution is reduced by a factor 16, making a 3×3 convolution actually processing a 48×48 receptive field of the initial input. The number of channels of the

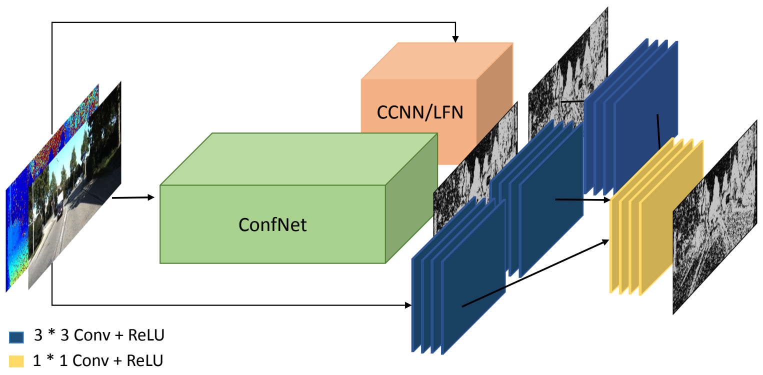


Figure 7.4: LGC-net architecture. Given the input reference image and its disparity map, they are forwarded to both local (CCNN or LFN, in orange) and global (ConfNet, green) networks, whose outputs and disparity are processed by 3 independent towers, concatenated to finally infer the output confidence map.

convolutional layers in different blocks are respectively 64, 128, 256 and 512, doubling after each max-pooling operator. Then, four decoding block follow in order to restore the original resolution of the input before obtaining the final confidence map. Each block uses a 3×3 deconvolutional layer with stride 2, followed by a 3×3 convolutional layer processing deconvolutional outputs concatenated with features taken from the encoding part at the same resolution. This reduces grid artifacts introduced by deconvolutional layers as suggested in [151], as well as enables to keep fine details present before down-sampling in the encoding part and missing after up-sampling from lower resolutions. The number of channels in each block for both deconvolutional and convolutional layers are respectively 256, 128, 64 and 32. A final 3×3 convolutional layer produces the final, full resolution confidence map followed by a Sigmoid operator to obtain normalized confidence values. The much larger receptive field enables to include much more information when computing per-pixel scores, but also acts as a *regularizer* yielding smoother confidence estimations and this leads to poor accuracy when dealing with high frequency patterns.

7.2.3 Local-global approach

To effectively combine both local and global cues, we introduce a final module acting in cascaded manner after the first two networks by processing their outputs and the initial disparity map. The module in charge of combining these cues is made of three towers processing respectively the local map, the global map and the disparity map. Weights are not shared between towers to extract distinct features from the three domains. Each tower is made of four convolutional layers with kernels 3×3 and 64 channels, their output are then concatenated and forwarded to two final 1×1 convolutional layers producing 100 features map each and a final 1×1 convolution in charge of the final confidence estimation, passed through a Sigmoid layer. Figure 7.4 describes the overall framework, referred to as Local Global Confidence Network (LGC-Net).

7.3 Implementation details and training protocol

We implemented our models using the TensorFlow framework. In particular, we deployed CCNN, EFN and LFN using the same configuration proposed in [179]: 64 and 100 channels respectively for 3×3 and 1×1 convolution, for which we report extensive experimental results in the next section. While the entire framework is fully differentiable from the input to the output, thus trainable in end-to-end manner, we first train the local and global networks separately, then we train the cascaded module. As already highlighted in [167], training cascaded models in end-to-end fashion may lead the network to converge at a local minimum, while a reasoned training of each module may enable better overall performance.

Local networks training schedule. Following the guidelines provided in [184], we extract 9×9 image patches from the first 20 stereo pairs in the KITTI 2012 training dataset [66] centered on pixels with available ground truth disparity used to obtain confidence ground truths, resulting into about 2.7 million samples. We trained for 14 epochs as proposed in [63, 179] using a batch of dimension 128, resulting into nearly 300k iterations. We used Stocastic Gradient Descent optimizer (SGD) to minimize the Binary Cross Entropy (BCE) [63, 179], a learning rate of 0.003 decreased by a factor 10 after 11 epochs and a momentum of 0.9.

ConfNet training schedule. We train ConfNet on 256×512 images estimating a confidence value for each pixel differently from local approaches that estimate confidence only for the central one in a patch (thus requiring to center the neighborhood on a pixel with available ground truth). Despite training complex architectures like DispNet requires a large amount of data usually obtained from synthetic datasets [151], we found out that training the same 20 images from KITTI is enough to effectively learn a confidence measure. This is probably due to the simpler task the network is faced with. In fact, finding outliers in a disparity map (i.e., a binary classification of the pixels) is much easier compared to infer depth from a stereo pair. Moreover, the disparity domain is less variegated than its RGB counterpart. Despite RGB data being processed jointly with disparity inside ConfNet, it plays a minor role compared to the latter. Cross-validation on Middlebury 2014 dataset [213], with indoor imagery extremely different from outdoor environments observed at training time will confirm this fact. We train ConfNet for 1600 epochs extracting random crops from the training stereo pairs, for a total of about 32k iterations. It is worth to note that, at training time, local networks produce a single pixel prediction versus the 256×512 available from ConfNet. For a single iteration, the minimized loss function encodes the contribution from 128 pixels for local networks (i.e., one for each sample in the batch) and 2^{16} for ConfNet, processing $512\times$ the amount of data. For this reasons only 32k iterations are enough for ConfNet to converge compared to the 300k of local methods. Pixels whose disparity ground truth is not available are masked when computing the loss function. We used SGD and BCE as for local networks, with an initial learning rate of 0.003, divided by a factor 10 after 1k epochs.

LGC-Net final training schedule. Finally, we train the cascaded module after freezing the weights of the local and global networks. We run additional 14 epochs processing image patches extracted from both disparity, local and global confidence estimations. The same 20 images, SGD, BCE loss, learning rate schedule and momentum are used for this training as well.

7.4 Experimental results

In this section, we report extensive experimental results supporting the superior accuracy achieved by the proposed LGC-Net compared to state-of-the-art. We evaluate the newly proposed framework estimating confidence for disparity maps obtained from three popular algorithms standard in this field [184], respectively AD-CENSUS [276], MC-CNN-fst matching cost [279] and SGM [82]. For this latter algorithm, compared to [184], we tuned better P1 and P2 penalties to 3 and 0.03, obtaining more accurate disparities on KITTI datasets slightly reducing accuracy on Middlebury 2014 dataset. In Section 7.4.1 we report results on both KITTI 2012 dataset [66] (i.e., on images not involved in training) and KITTI 2015, while in Section 15.3.3 we cross-validate on Middlebury 2014 [213] as commonly done by most recent works [184] to measure how well the confidence measures perform on data quite different from the one deployed for training.

7.4.1 Evaluation on KITTI datasets

To assess the effectiveness of LGC-Net, we train the networks on the first 20 images of the KITTI 2012 dataset and we report extensive experimental results on the remaining 174 images of the same stereo dataset [66] as well as on the entire KITTI 2015 dataset [155]. This second dataset depicts outdoor environments similar to the first dataset but with the addition of dynamic objects not present in the other. We evaluate confidence measures provided by standalone modules (i.e., CCNN, EFN, LFN and the global architecture ConfNet) as well as those produced by the full local-global framework in two configurations obtained respectively by deploying CCNN [179] or multi-modal architectures [63] as local network. For a fair comparison, all the evaluated models have been trained from scratch following the same protocol described in Section 7.3.

Table 7.1 reports experimental results on KITTI 2012. Each row refers to one of the three considered algorithms, respectively AD-CENSUS, MC-CNN and SGM and each column to a confidence measure, reporting AUC values averaged on the entire dataset. In bold, the best AUC for each algorithm. Considering at first single networks, we observe that multi-modal network LFN perform similarly to CCNN being this latter method outperformed by a small margin only with AD-CENSUS. The EFN network has always worse performance compared to CCCN and LFN. These results highlight that, with LFN and EFN networks in this configuration, processing the RGB image does not provide additional information compared to the one inferred from the disparity domain. Looking at ConfNet we can observe how processing global information only leads, as expected, to less accurate results with noisy disparity maps provided by AD-CENSUS but it performs reasonably well, and better than EFN, with smoother disparity maps generated by SGM and MC-CNN. In particular it is always outperformed by CCNN and LFN. According to these results, confirmed also by following evaluations, the global approach alone loses accuracy when dealing with fine details, despite the deployment of skip-connection between encoder and decoder sections, while local approaches performs very well in these cases. Observing LGC-Net results, both configurations outperform all the other evaluated techniques, highlighting how the two complementary cues from local and global networks can be effectively combined to improve confidence estimation moving a step forward optimality for all the three stereo algorithms. By directly comparing the two configurations of LGC-Net, using respectively CCNN or LFN as local network, there is no clear winner highlighting how the contribution given by the RGB image on a small neighborhood seems negligible. In fact, it yields a 0.0001 difference in terms of average AUC between the two versions, in favor of the first configuration on AD-CENSUS and the second one on MC-CNN and SGM. These experiments highlights that the major benefit is obtained by the proposed strategy exploiting local and global context information.

Table 7.2 reports experimental results on the KITTI 2015 dataset [155], with AUC values averaged over the available 200 stereo pairs with ground truth. First of all, we

KITTI 2012 [66] (174 images)	CCNN [179]	EFN [63]	LFN [63]	ConfNet	LGC-Net (CCNN)	LGC-Net (LFN)	Optim.
AD-CENSUS [276]	0.1207	0.1261	0.1201	0.1295	**0.1174**	0.1176	0.1067
MC-CNN [279]	0.0291	0.0316	0.0294	0.0311	0.0279	**0.0278**	0.0231
SGM [82]	0.0194	0.0229	0.0198	0.0199	0.0176	**0.0175**	0.0088

Table 7.1: Experimental results on KITTI 2012 dataset [66]. From top to bottom, evaluation concerning AD-CENSUS [276], MC-CNN [279] and SGM [82] algorithms. For each column, average AUC achieved on the entire dataset (i.e., 174 out of 194 stereo pairs) for different confidence measures.

KITTI 2015 [155] (200 images)	CCNN [179]	EFN [63]	LFN [63]	ConfNet	LGC-Net (CCNN)	LGC-Net (LFN)	Optim.
AD-CENSUS [276]	0.1045	0.1087	0.1026	0.1128	**0.0999**	0.1004	0.0883
MC-CNN [279]	0.0289	0.0319	0.0292	0.0315	0.0281	**0.0278**	0.0213
SGM [82]	0.0201	0.0239	0.0209	0.0216	0.0193	**0.0190**	0.0091

Table 7.2: Experimental results on KITTI 2015 dataset [155]. From top to bottom, evaluation concerning AD-CENSUS [276], MC-CNN [279] and SGM [82] algorithms. For each column, average AUC achieved on the entire dataset (i.e., 200 stereo pairs) for different confidence measures.

observe that the same trend observed for KITTI 2012 is confirmed also in this case, with CCNN being slightly outperformed by LFN only on AD-CENSUS. CCNN and LFN always provide more accurate estimation accuracy compared to EFN while ConfNet outperforms this latter method on smoother MC-CNN and SGM disparity maps as in previous experiment. Finally, the two LGC-Net versions achieve overall best performance on this dataset, as for KITTI 2012, confirming the effectiveness of the proposed method. Moreover, the same results also highlight once again the negligible margin brought in by using the RGB image with CCNN.

7.4.2 Cross-validation on Middlebury 2014

Having proved the effectiveness of the proposed LGC-Net on KITTI datasets, we conduct a more challenging evaluation by cross-validating on Middlebury 2014 imagery [213] confidence measures trained on the first 20 images of the KITTI 2012 dataset. As done in [184], assessing the performance on a validation dataset quite different from the one used

Middlebury 2014 [213] (15 images)	CCNN [179]	EFN [63]	LFN [63]	ConfNet	LGC-Net (CCNN)	LGC-Net (LFN)	Optim.
AD-CENSUS [276]	0.1131	0.1263	0.1146	0.1206	**0.1099**	0.1109	0.0899
MC-CNN [279]	0.0668	0.0781	0.0645	0.0755	0.0624	**0.0616**	0.0458
SGM [82]	0.0794	0.1005	0.0856	0.0886	**0.0703**	0.0709	0.0431

Table 7.3: Experimental results on Middlebury 2014 dataset [213]. From top to bottom, evaluation concerning AD-CENSUS [276], MC-CNN [279] and SGM [82] algorithms. For each column, average AUC achieved on the entire dataset (i.e., 15 stereo pairs) for different confidence measures.

during the training phase effectively measures how robust a confidence measure is with respect to circumstances very likely to occur in practical applications. Being our models trained on KITTI images, depicting outdoor environments concerned with autonomous driving applications, the indoor scenes included in the Middlebury 2014 dataset represent a completely different scenario ideal for the kind of cross-validation outlined.

Table 7.3 quantitatively summarizes the outcome of this evaluation. First and foremost, as in previous experiments, LGC-Net outperforms with both configurations all standalone confidence measures confirming the negligible difference, lower or equal than 0.001, between the two local networks. The trend between single architectures is substantially confirmed with respect to previous experiments, with ConfNet performing always better than EFN in this cross-evaluation even with the noisy AD-CENSUS maps. CCNN and LFF, as for previous experiments, performs quite similarly confirming once again the small impact of RGB cues in local networks with our training configuration.

In Figure 7.5 we report a qualitative comparison between local, global (ConfNet) and LGC-Net for two images of the the Middlebury 2014 dataset processed with SGM and MC-CNN stereo algorithms. The quantitative advantages reported for LGC-Net in the previous evaluations can be clearly perceived qualitatively by looking, for instance, at texture-less regions on the wall in *PianoL* stereo pair and at the occluded area on the background in *Pipes* stereo pair.

To summarize, exhaustive experiments on three datasets and three stereo algorithms proved that the proposed framework always outperforms both local and global standalone

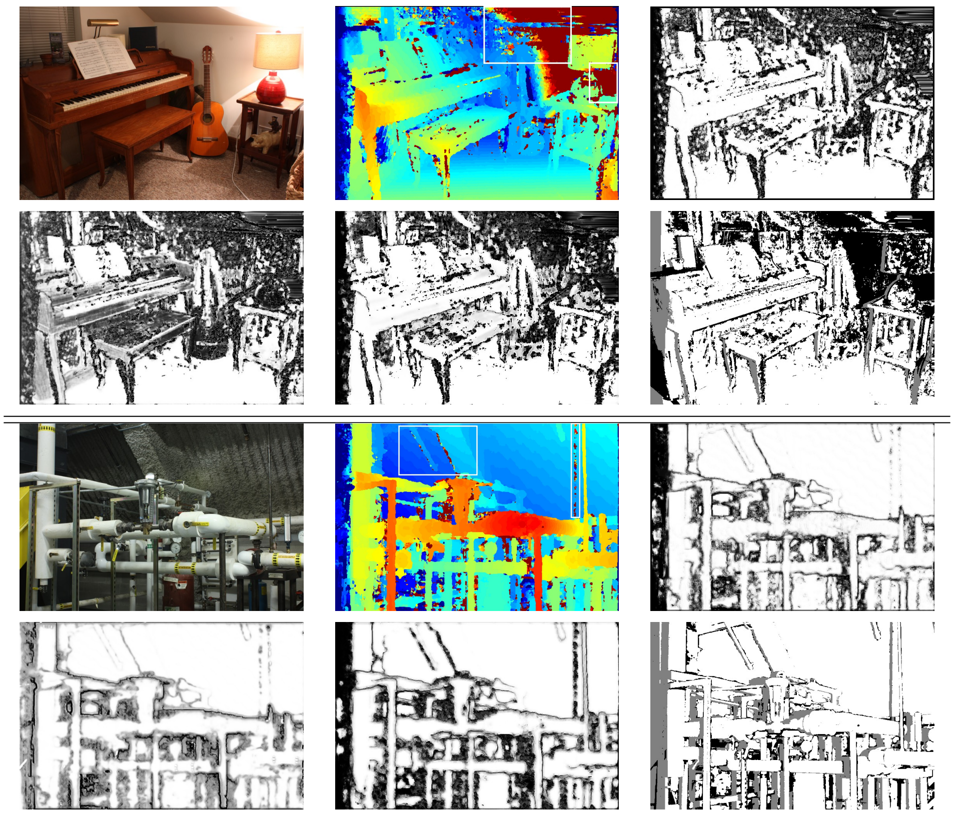


Figure 7.5: Qualitative comparison of confidence maps on selected images from Middlebury 2014 dataset [213]. For each sample, we report from top left to bottom right reference image, disparity map, confidence map respectively for CCNN, ConfNet and LGC-net and ground truth confidence labels. On top *PianoL* pair processed by MC-CNN-fst, on bottom *Pipes* pair processed by SGM.

strategy by a significant margin, thus effectively learning to combine local and global cues to obtain more accurate confidence estimation. This trend is also confirmed moving to very different data as reported in the cross evaluation, proving that LGC-Net is more capable to generalize to completely different image contents. Overall, the proposed method always outperforms state-of-the-art methods for confidence estimation.

7.5 Conclusions

In this chapter we propose, for the first time to the best of our knowledge, to leverage on global and local context to infer a confidence measure for stereo. Driven by the outstanding results achieved by CNN-based confidence measures, in this work we argue that their effectiveness can be improved by changing their intrinsic local nature. To this aim we propose to combine with a CNN cues inferred with two complementary strategies, based on two very different receptive fields. The proposed LGC-Net, a multi-modal cascaded network, merges the outcome of the two complementary approaches enabling more accurate confidence estimation. We extensively evaluated the proposed method on three datasets and three algorithms following standard protocols in this field proving that our proposal outperforms state-of-the-art confidence measures and further moves a step forward optimality.

Chapter 8

Good cues to learn from scratch a confidence measure for passive depth sensors

The content of this chapter has been published at the IEEE Sensors - "Good cues to learn from scratch a confidence measure for passive depth sensors" [193].

8.1 Introduction

In [179], authors showed for the first time that a *Confidence Convolutional Neural Network* (CCNN) could be trained for state-of-the-art confidence estimation from a single disparity map. Currently, such a network also represents the basic building block of top-performing methods, processing either the disparity map [181, 240] or the cost volume [117].

Following this path, in this chapter, we deeply evaluate which features traditionally available from any depth camera, i.e. disparity map(s) and the RGB image(s), are relevant to estimate confidence when fed to a deep network trained for this purpose. To assess the importance of such features, we carry out an exhaustive evaluation with three standard datasets and three popular stereo algorithms [184]. Moreover, since end-to-end

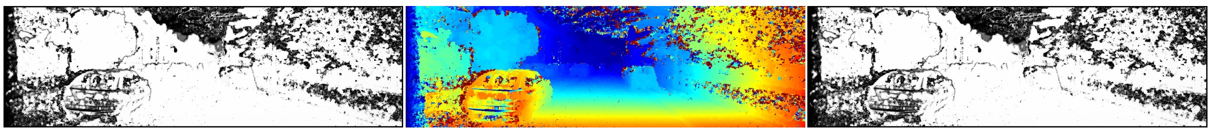


Figure 8.1: Confidence estimation in the disparity domain. By visually inspecting reference image (left) and corresponding disparity map (center) computed by a stereo algorithm, outliers can be easily identified. On the same principle, a CNN can predict a reliable confidence map (right) processing the same cues.

stereo architectures represent the state-of-the-art for stereo, we show that the considered methods can infer, even in this case, a meaningful confidence estimation whereas other known techniques based on cost volume processing could not. Finally, we move beyond stereo matching and evaluate the considered confidence estimation methods with the maps generated by deep networks for monocular depth perception.

Our evaluation highlights that CCNN is not only the most versatile method, being suited for any depth sensing device, but also, in most cases, the most effective confidence estimation approach for depth data. Nonetheless, in some specific circumstances, adding additional cues such as RGB image(s) and the target disparity map, despite this latter cue is not always available, yields slightly better accuracy.

8.2 Learning from scratch confidence measures

A recent trend concerning confidence estimation proves that it can be reliably inferred in the disparity domain. The primary rationale behind this strategy is that, by visual inspection, several outliers can be easily spotted from the disparity assignments of the neighboring pixels, as evident from Fig. 8.1.

Purposely, in this chapter, we thoroughly investigate strategies to learn from scratch a confidence measure with deep learning by relying on visual cues only and neglecting the use of the cost volume, seldom available outside of the conventional stereo context. For instance, nowadays, the outlined circumstance frequently occurs in very relevant cases, such as when dealing with custom stereo cameras, deep stereo [151] and monocular [69]

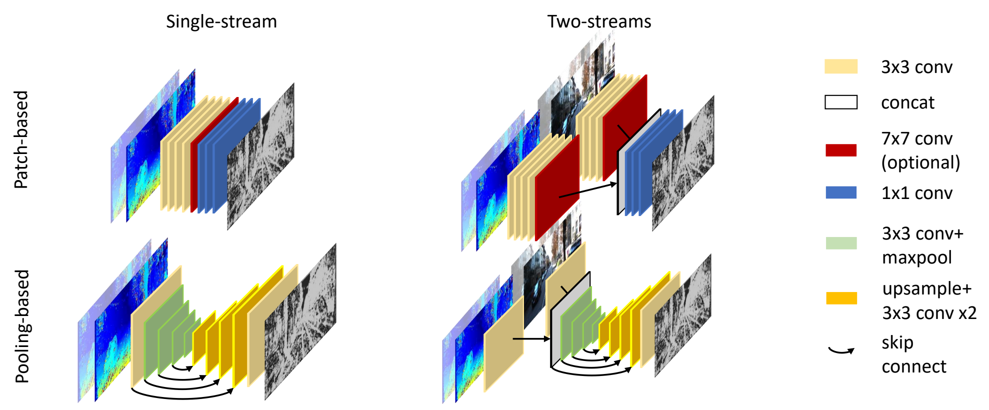


Figure 8.2: Confidence estimation networks.. On the left, the single-stream models processing only disparity cues. On the right, two-streams models processing both RGB image and disparity cue separately. On top, patch-based networks, on bottom pooling-based. 7×7 convolutional layer is used only in case of 15×15 receptive field.

depth estimation networks.

8.2.1 Input cues

In order to avoid the need for any intermediate representation, such as the cost volume, we leverage only on the visual cues potentially available with a stereo or monocular setup. Specifically, in the stereo case, given the left and right images I_L and I_R and assuming the former as the reference one, we can always obtain a disparity map for the reference view and, possibly, a map for the target view, respectively D_L and D_R. Nonetheless, it is worth observing that D_R might be not available in some cases, for instance, when dealing with off-the-shelf stereo cameras (e.g., the Intel RealSense stereo camera).

Thus, in this chapter, we consider an exhaustive combination of such input cues including configurations already evaluated and some never explored before. We compare with the same CNN baseline network different combinations of input features enabling to highlight which cues are truly useful for confidence estimation:

Reference disparity map D_L: the disparity map aligned with the input reference frame (typically, the left image). In [179] this cue proved to be sufficient to learn a confidence measure.

Reference image I_L: RGB input reference frame. [62] extended CCNN to account for this additional cue.

Warped target disparity map D'_R: obtained by warping target disparity map D_R into the left reference frame according to D_L. The absolute difference between D'_R and D_L encodes the *left-right difference* exploited by [217] with PBCP. Purposely, pixel at coordinates (x, y) is sampled at $(x - D_L(x, y), y)$ from D_R as $D'_R(x, y) = D_R(x - D_L(x, y), y)$. This configuration, referred to as *fast* in [217], is suited for processing in fully convolutional manner and thus compatible with both network architectures adopted in our experiments and detailed in the reminder, conversely to other configurations in [217] that require independent processing of each single patch.

Warped target image I'_R: image obtained by warping I_R into the left reference frame according to D_L. To the best of our knowledge, this cue has never been considered before for confidence prediction. Pixel at coordinates (x, y) is sampled at $(x - D_L(x, y), y)$ from I_R as $I'_R(x, y) = I_R(x - D_L(x, y), y)$.

By designing fully-convolutional architectures, in a single forward pass, we can estimate confidence for all image pixels.

8.2.2 Network architectures and configurations

In the literature, different CNN architectures have been deployed for confidence estimation. In the remainder, to adequately assess the contribution given by the different input cues to the final confidence estimation, we focus on two main categories:

Patch-based [62, 179, 217], made by convolutional layers only. Spatial resolution is reduced by convolving over valid pixels (i.e., without padding). For instance, 3×3 convolutions reduce the resolution by 2 pixels on each dimension, thus processing a single output value from a 3×3 patch.

Pooling-based [240], decimating the resolution by means of pooling operations. The original resolution is restored for the output through deconvolutions or upsampling operations.

For both, we deploy a baseline architecture regarding the number of convolutional layers, channels, activation layers and input dimensions. Precisely, we deploy the architecture from [179] for patch-based family and ConfNet [240] for pooling-based. In this latter case, we replace deconvolutions with nearest neighbour upsampling followed by convolutions to improve accuracy. The final outputs are normalized in [0, 1] by a sigmoid layer.

Figure 22.1 depicts the architectures outlined so far, respectively patch-based (top) and pooling-based (bottom). Regarding patch-based models, we consider two variants with different receptive field as proposed in the literature: 9×9 [179] and 15×15 [217]. For this latter, a 7×7 layer (red in figure) is added to reduce features dimensions to 1×1 as well[1].

We consider different combinations of the input cues mentioned above, leading overall to the following six network configurations:

- **CCNN** – our *Confidence Convolutional Neural Network* processing D_L only [179]
- **LF** – *Late Fusion* of D_L and I_L [62]
- **PBCP** – *Patch Based Confidence Prediction* from D_L and D'_R [217]
- **LF-PBCP** – a model mixing information from D_L, D'_R and the reference image I_L
- **LRLF** – a late fusion model combining I_L, I'_R with reference disparity map D_L
- **LRLF-PBCP** – a network processing all the information available from a stereo setup: D_L, D'_R, I_L and I'_R

CCNN and PBCP rely on a single-stream architecture while the others on two streams. The two resulting variants, for each family, are depicted respectively at the left and right of Figure 22.1. The two streams are combined using concatenation (white layers). While

[1]While 9×9 patches are reduced by 4 layers to a single pixel, 15×15 patches are reduced to 7×7 as in [217]. The 7×7 layer reduces these latter to a single pixel as well

CCNN, LF and PBCP are known in the literature, LF-PBCP, LRLF and LRLF-PBCP are new proposals. Therefore, given the 6 configurations, the 2 patch-based and the pooling-based models, a total of 18 networks will be evaluated.

8.3 Experimental results

In this section, we report an exhaustive evaluation concerning the models introduced above.

8.3.1 Algorithms and networks for depth from passive sensors

We consider in our evaluation traditional (AD-CENSUS, SGM) and learning-based stereo algorithms (MC-CNN), an end-to-end model to infer depth from stereo pairs (DispNetC) and a monocular depth estimation network (Monodepth).

8.3.2 Implementation details and training procedure

Patch-based models have been trained on batches of 128 image patches, while and pooling-based models on batches of 4 crops of size 256×512, both with Binary Cross Entropy (BCE) loss between estimated confidence c_i and ground truth confidence label y_i, for central pixel i in each patch in the former case or for all the pixels in each crop in the latter. Labels y_i are set to 0 if, in the case of stereo algorithms, the difference between estimated and ground truth disparity is higher than a threshold τ and 1 otherwise. For Monodepth, we follow a slightly different protocol, described in detail in Sec. 8.3.5.

We used Stochastic Gradient Descent (SGD) as the optimizer and a learning rate of 3×10^{-3}. Training samples have been generated inferring disparity maps for each of the five considered depth sensing methods on the KITTI 2012 dataset. In particular, we sampled two different training splits out of the total 200 stereo pairs with ground truth depth labels available:

Table 8.1: Runtime on KITTI images (375 × 1242) on a Titan Xp GPU.

Architecture	Single-Stream			Two-Streams		
Variant	9×9	15×15	Pool	9×9	15×15	Pool
Runtime	0.07 s	0.10s	0.02	0.09s	0.13s	0.03

KITTI-small, made of the first 20 frames [184], providing about 2.7 million pixels with available ground truth labels.

KITTI-large, made of the first 94 frames [62]. This configuration yields about 8 million depth samples.

In order to assess the performance of confidence prediction across different domains, we train on both splits and test on remaining samples from KITTI 2012, as well as on KITTI 2015 and Middlebury 2014 without re-training the networks [184] to highlight how each input feature or their combinations are robust to domain shift. Indeed, since all the networks have been designed starting from the same baseline structure, such evaluation will be able to assess the impact of each input cue. Given the six combinations of input cues, the two patch-based and the pooling-based models and the two training portions described so far we trained: 12 models for the Monodepth algorithm, 18 for DispNetC and 36 for each stereo algorithms. Overall, we trained 138 networks in about one week with an NVIDIA Titan Xp GPU. Concerning runtime, Table 8.1 reports the time required on the same GPU to estimate a confidence map at KITTI resolution (about 375 × 1242), showing almost equivalent runtime for the two patch-based models. Pooling-based models are much faster, thanks to the reduction of spatial resolution, but less accurate as we will see through the evaluation reported next.ù

8.3.3 Evaluation protocol

To quantitatively measure the effectiveness of confidence prediction, we use the standard protocol adopted in this field [89, 184].

In our experiments, in order to compute AUC scores we sampled pixels at intervals of 5% of the overall amount and we label as outliers pixels with an absolute disparity

Table 8.2: Average AUC margin (%) on disparity maps by different stereo methods.

	KITTI 2012						KITTI 2015						Middlebury 2014					
	Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool	
	small	large	small	large	small	large	small	large	small	large	small	large	small	large	small	large	small	large
CCNN	12.84	13.14	17.71	13.33	20.52	19.03	15.97	16.76	21.97	17.21	28.88	26.39	25.03	22.36	30.03	24.92	43.27	37.82
PBCP	16.03	14.11	22.21	17	20.62	24.64	22.2	20.16	29.11	23.56	28.88	33.18	32.26	22.47	32.7	30.92	40.6	40.49
LF	13.78	14.88	21.93	14.88	19.96	18.65	17.33	18.01	27.97	20.16	27.07	25.71	30.59	26.03	40.16	31.37	39.49	37.26
LF-PBCP	15.09	13.62	23.62	16.43	22.68	20.1	21.74	19.48	32.16	23.33	30.58	27.29	32.93	28.92	43.6	42.83	47.27	44.83
LRLF	14.06	14.2	20.15	13.72	21.74	25.6	17.1	18.35	25.59	18.35	28.09	36.24	26.81	27.7	41.71	28.59	45.38	45.94
LRLF-PBCP	15.93	15.17	24.46	16.62	20.43	30.53	22.76	21.52	34.43	23.56	27.18	37.49	29.25	34.15	51.61	33.93	39.27	60.18

(a) CENSUS algorithm

	KITTI 2012						KITTI 2015						Middlebury 2014					
	Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool	
	small	large	small	large	small	large	small	large	small	large	small	large	small	large	small	large	small	large
CCNN	26.41	22.32	39.39	28.57	41.99	41.52	37.26	35.38	50.47	40.57	54.25	56.13	39.08	36.03	46.29	35.37	41.7	41.92
PBCP	29.44	22.77	42.42	28.57	35.93	33.04	42.45	37.26	53.3	41.98	50.94	46.7	40.83	37.34	49.78	34.93	40.39	37.77
LF	28.14	22.77	39.83	31.25	48.05	51.79	38.68	39.15	51.42	42.45	68.87	69.81	48.47	46.94	56.99	51.53	89.96	89.74
LF-PBCP	30.3	23.21	39.83	30.36	43.29	34.38	41.51	38.21	58.96	46.23	58.49	50.94	50.66	42.14	68.56	50.66	79.69	59.83
LRLF	29.44	23.21	35.93	35.27	44.16	41.96	43.87	36.32	50.47	49.06	61.32	59.43	40.17	42.79	48.91	50.44	83.19	79.26
LRLF-PBCP	29.87	23.66	38.1	29.02	36.36	36.16	43.4	38.21	54.72	42.92	51.89	52.36	46.07	55.46	62.88	46.51	63.1	63.76

(b) MC-CNN algorithm

	KITTI 2012						KITTI 2015						Middlebury 2014					
	Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool	
	small	large	small	large	small	large	small	large	small	large	small	large	small	large	small	large	small	large
CCNN	117.05	103.49	119.32	125.58	119.32	115.12	116.48	115.38	115.38	136.26	123.08	126.37	69.38	63.44	73.13	75.33	76.65	81.06
PBCP	115.91	105.81	110.23	102.33	128.41	110.47	117.58	117.58	114.29	119.78	135.16	116.48	72.25	68.28	77.31	68.5	85.02	75.99
LF	126.14	101.16	126.14	118.6	176.14	129.07	126.37	116.48	137.36	125.27	238.46	153.85	76.87	81.94	90.97	75.55	177.97	116.96
LF-PBCP	102.27	95.35	110.23	109.3	198.86	162.79	114.29	112.09	131.87	125.27	249.45	216.48	83.7	68.06	87.44	73.57	203.52	176.21
LRLF	129.55	106.98	246.59	146.51	234.09	109.3	124.18	126.37	260.44	148.35	258.24	143.96	83.04	88.55	135.68	74.89	231.28	129.07
LRLF-PBCP	112.5	96.51	128.41	111.63	303.41	163.95	116.48	113.19	140.66	124.18	327.47	198.9	79.3	75.99	80.4	70.26	281.94	163

(c) SGM algorithm

	KITTI 2012						KITTI 2015						Middlebury 2014					
	Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool	
	small	large	small	large	small	large	small	large	small	large	small	large	small	large	small	large	small	large
CCNN	549.4	473.26	655.42	636.05	1213.25	838.37	442.06	382.24	601.87	572.9	700	670.09	198.26	198.98	178.53	186.3	266.87	218.71
LF	628.92	515.12	603.61	702.33	706.02	605.81	502.8	412.15	478.5	619.63	506.54	584.11	215.13	201.64	207.16	202.04	236.3	219.63
LRLF	721.69	538.37	687.95	697.67	687.95	677.91	599.07	384.11	574.77	612.15	570.09	575.7	208.18	209.71	203.07	183.13	232.21	208.9

(d) DispNetC network

Table 8.3: Average AUC margin (%) on disparity maps by Monodepth.

	KITTI 2012						KITTI 2015					
	Patch (9x9)		Patch (15x15)		Pool		Patch (9x9)		Patch (15x15)		Pool	
	small	large	small	large	small	large	small	large	small	large	small	large
CCNN	299.12	267.57	769.91	927.93	1251.33	359.46	304.46	264.97	893.63	1027.39	1317.83	325.48
LF	361.06	270.27	488.5	654.95	446.02	554.95	347.77	320.38	645.22	648.41	382.8	499.36

error larger than 3 for KITTI datasets and 1 for Middlebury, this latter processed at quarter resolution. For training, we set τ to 3. As already pointed out, when dealing with a monocular network, we change the method to detect outliers, as will be explained in detail later.

Considering the vast amount of data collected, before reporting the outcome of our evaluation we describe in detail how to correctly parse the information provided. In particular, for each depth estimation method, we will report tables organized into three main blocks for the three datasets, respectively KITTI 2012, KITTI 2015 and Middlebury 2014 from left to right. Each block is divided into three groups of two columns regarding patch-based models (9×9 and 15×15) and the pooling-based model (Pool) as third. In each group, the two columns concern the KITTI-small and KITTI-large splits. Each row reports averaged AUC values for a specific combination of input cues. Thus, each score refers to a features configuration tested on a particular dataset after being trained on one of the two possible training splits. For each single dataset, we apply a heatmap to better distinguish top-performing configurations (in green) from those less effective (in red).

8.3.4 Evaluation with stereo algorithms

Table 25.2 reports results concerned with confidence estimation from disparity maps computed by the aforementioned stereo algorithms. We will now discuss on each one.

AD-CENSUS. Table 25.2 (a) reports results concerned AD-CENSUS algorithm [276]. Such a method achieves quite high error rate on the three datasets, respectively, about 39% bad-3, 35% bad-3 and 37% bad-1 on KITTI 2012, KITTI 2015 and Middlebury 2014. Nonetheless, its cost volume is often deployed inside well-known pipelines such as SGM

[82]. Thus, inferring an effective confidence scoring in this case allows for deployment of techniques such as [169, 217, 228].

Concerning patch-based methods, we can notice from the table how the reference disparity map alone contains enough information to detect outliers with this stereo algorithm nearly optimally. Indeed, CCNN always achieves the lowest average AUC margin. The reason is that the network effectively learns to detect from this cue glaring error patterns, clearly visible in the disparity map, as well as high-frequency noise often present in regions supposed to be smooth. Adding other cues reduces confidence estimation capability slightly, probably because they are scarcely informative (e.g., the algorithm often fails where the original stereo images lack information, as in textureless regions). Processing 9×9 patches catches enough context with all datasets, while including more training samples from KITTI 2012 (large vs small splits) improves the results on the remaining of KITTI 2012 and on Middlebury. Nonetheless, this strategy yields slightly worse accurate confidence prediction on KITTI 2015.

Pooling-based models, although faster, typically perform worse than patch-based methods confirming the findings in [240], because of pooling operations losing high-frequency details. Nonetheless, additional cues, e.g. the LF configuration, enable to reduce the gap on all the datasets partially.

MC-CNN. Table 25.2 (b) reports the outcome of the evaluation with MC-CNN-fst. This algorithm almost halves the number of outliers compared to AD-CENSUS, leading respectively to about 17%, 15% and 26% error rates on KITTI 2012, KITTI 2015 and Middlebury. Nevertheless, its local nature yields error patterns similar to those observed with AD-CENSUS.

Therefore for patch-based models, in most cases, the D_L features alone still leads to the best overall performance. The most notable exception is on Middlebury; in fact, using a 15×15 receptive field and KITTI-large for training, the warped disparity D'_R processed by PBCP enables to achieve slightly better confidence prediction accuracy compared to CCNN. This outcome might be the consequence of the stereo method itself processing larger patches (i.e., 11×11) on Middlebury. Moreover, in this case the large split is always beneficial to obtain the best accuracy, while the size of the receptive fields impacts differently according to the dataset.

Again, pooling-based models perform consistently worse than patch-based approaches, although they benefit more of additional information and achieve the best results with PBCP.

SGM. Table 25.2 (c) collects results concerning SGM, with error rates of about 9%, 10% and 27%.

For patch-based models, we can notice that even with much more accurate disparity maps, CCNN is still effective. Nonetheless, in this case, it is no longer the best overall solution to detect outliers as in the previous two experiments. In particular, PBCP improves confidence prediction of patch-based models on KITTI 2012 for three out of four cases, while adding the left image to CCNN seems effective only for 15×15 models. On KITTI 2015, mixed results are achieved by the three variants. However, LF-PBCP yields optimal performance on both KITTI datasets when training on the large split with a 9×9 receptive field. On the other hand, adding the warped right images seems not effective at all. Conversely, for Middlebury 2014 we can notice the best overall results are achieved by CCNN trained on KITTI-large with receptive field 9×9. Although the disparity cue alone is not optimal on data similar to the training images, it achieves better generalization across datasets, proving to be more robust when deployed in totally unseen environments.

Concerning pooling-based methods, in this case, more input cues seem useful only when more training data are available, with mixed configurations outperforming CCNN, still not matching the performance of patch-based networks.

DispNetC. Table 25.2 (d) reports the outcomes of experiments on DispNetC. Although this deep stereo network achieves about 6% and 7% bad-3 error rate on KITTI 2012 and 2015, its performance dramatically drops with the more heterogeneous indoor scenes depicted in the Middlebury 2014 dataset, falling to more than 30% bad-1 score.

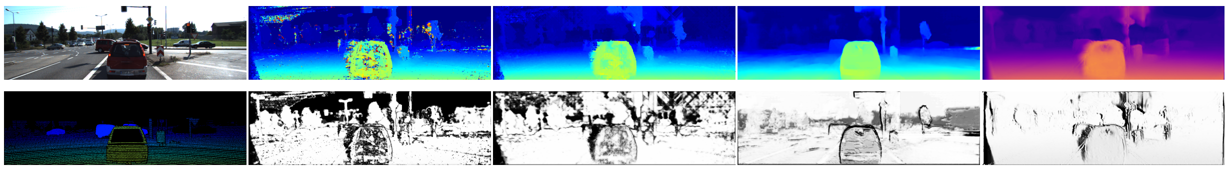


Figure 8.3: Qualitative results on KITTI 2015. Leftmost side: reference frame (top) and ground truth disparity (bottom). Then, from left to right: (top) disparity maps for for MC-CNN-fst, SGM, DispNetC and inverse depth map for Monodepth, (bottom) corresponding best confidence estimation method for each one (respectively, CCNN, LF-PBCP, CCNN and CCNN). Disparity is encoded with colormap jet, inverse monocular depth map with colormap plasma, confidence with grayscale values.

In this latter case, although the produced disparity maps look visually consistent, depth prediction is often inaccurate and detecting outliers becomes of paramount importance, yet very challenging. Being no cost volume available in this case, visual cues are crucial for the purpose. Since it provides a single disparity map aligned with the reference image without processing any traditional cost volume, only configurations CCNN, LF and LRLF are suitable. First of all, we can notice how AUC margins are much higher compared to what observed on previous evaluations. This fact can be explained considering the very accurate results yielded by the DispNetC network on KITTI datasets and the erroneous, yet visually consistent maps obtained on Middlebury.

For patch-based models, CCNN achieves the best results in most cases. In general, on KITTI the best results are achieved by 9×9 models, while on Middlebury 2014 the 15×15 architectures are more effective. LF seldom yields better performance (only in 2 out of 12 cases) and never enables to achieve the best accuracy. A possible cause is the fact that disparity maps produced by end-to-end models are particularly consistent with the reference image shape, thus adding such cue as input to detect outliers does not add particular information for the purpose. Interestingly, on Middlebury the best performance are achieved by training on the small split using a receptive field of 15×15.

As for traditional algorithms, pooling-based models cannot compete with patch-based ones and processing D_L alone leads most times to the worst results.

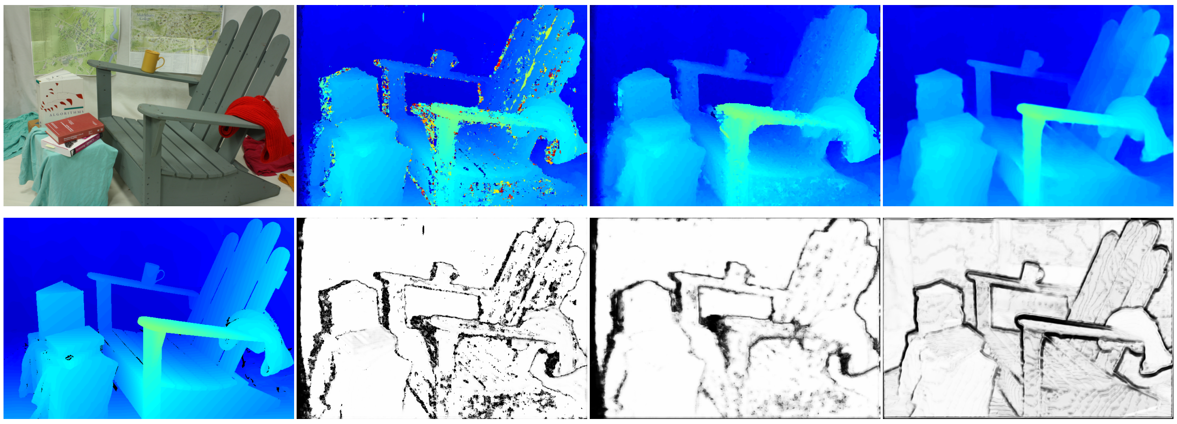


Figure 8.4: Qualitative results on Middlebury 2014. Leftmost side: reference frame (top) and ground truth disparity (bottom). Then, from left to right, (top) disparity maps for MC-CNN-fst, SGM, DispNetC and (bottom) outcome of the best confidence estimation method for each one (CCNN). Disparity is encoded with colormap jet, confidence with grayscale values.

8.3.5 Evaluation with monocular depth estimation network

Finally, we inquire about the effectiveness of confidence measures initially conceived for stereo when dealing with monocular depth estimation networks. As a representative method in this field, we choose Monodepth [69], trained in a self-supervised manner on stereo pairs, for historical reasons and reproducibility. Given its input and output cues, only CCNN and LF can be deployed with this network. Moreover, we also point out that the authors have trained Monodepth on KITTI raw sequences [67], and thus it performs quite well in similar environments. However, it performs poorly when deployed in entirely different environments, such as the indoor scenes depicted in Middlebury 2014. For this reason, we evaluate the performance of confidence prediction frameworks for Monodepth only on the KITTI 2012 and 2015 datasets.

Metrics. In contrast to stereo, monocular depth estimation is an ill-posed problem enabling to infer depth up to a scale factor. Thus, we change the criterion to label pixels as outliers following the methodology used to evaluate monocular depth estimation methods. Explicitly, we follow [60] and consider outliers all pixels having $\max(\frac{D_L}{G}, \frac{G}{D_L}) > 1.25$, being D_L estimated depth and G ground truth depth. The same criterion is used at training time. Accordingly, Monodepth produces respectively 10% and 16% outliers on KITTI 2012 and 2015.

Evaluation. Table 8.3 collects the results regarding this experiment. Similarly to the evaluation with DispNetC, we can notice larger AUC margins compared to traditional stereo algorithms. Nonetheless, we can notice how CCNN always achieves the best accuracy for outliers detection among all considered measures with a 9×9 receptive field. In particular, trained on a smaller amount of images, it achieves the best results on KITTI 2015, granting better generalization capability. On the other hand, using 94 pairs for training yields optimal results on KITTI 2012 but at the same time, reduces confidence estimation accuracy on KITTI 2015. Adding the left image to the 9×9 networks does not increase accuracy except on KITTI 2015 when training on the more extensive training set. By enlarging the receptive field, CCNN loses accuracy. Processing the left image attenuates this effect, but still does not vouch for the same performance obtained by processing only the inverse depth map with a 9×9 receptive field. Concerning pooling-based strategy, this time it outperforms 15×15 patch-based networks when trained on a broader training set, but still cannot compete with 9×9. Surprisingly, CCNN configurations perform better than LF when trained on more samples.

8.3.6 Qualitative analysis

Finally, we report qualitative examples to highlight both the different nature of noise in the estimated disparity/depth maps and the effectiveness of confidence measures at detecting outliers. Figure 8.3 shows an example from the KITTI 2015 dataset. It reports the disparity map, for stereo, and the inverse depth maps, for the monocular network, generated by some of the method considered in our evaluation and the corresponding best confidence map for each one. We can notice how, with different degrees of reliability, a low confidence score (black) generally corresponds to an erroneous depth estimation on the top map. Figure 8.4 collects results on the *Adirondack* stereo pair from Middlebury 2014. We point out that, for the reasons outlined before, the confidence scores are more likely to fail in this case, for instance on the disparity map inferred by DispNetC (right), where part of the armchair is missing and the corresponding confidence values are high instead.

8.4 Conclusions

In this chapter, we studied the importance of input cues processed to infer confidence scores from depth data generated by passive depth sensors with a CNN. Considering the same baseline architectures, we extensively assessed the performance of six models processing different input cues with different stereo and mono depth sensing strategies, including learning-based approaches.

Our in-depth evaluation yields the following insights. 1) Despite slower, patch-based models outperform pooling-based ones. 2) The D_L cue, i.e. CCNN configuration, allows for the best results when dealing with disparity maps generated by local approaches either conventional (CENSUS) or learning-based (MC-CNN). 3) For algorithms generating smoother disparity maps like SGM, the most effective configuration is PBCP coupled with the reference image. Nonetheless, CCNN is still competitive and the right disparity map required by PBCP is not always available, especially when dealing with end-to-end depth sensing networks. 4) In such a case, experiment on DispNetC maps highlight once again that CCNN is the best option among the considered ones, stressing the low contribution given by reference image, already exploited at its best to estimate the disparity map. 5) The same behaviour is also confirmed when tackling confidence estimation from monocular depth estimation models such as Monodepth.

In summary, processing the disparity map alone, as done by the original CCNN network [179], turns out the most versatile and overall effective strategy across all algorithms and datasets.

Chapter 9

Confidence estimation for ToF and stereo sensors and its application to depth data fusion

The content of this chapter has been published at the IEEE Sensors - "Confidence Estimation for ToF and Stereo Sensors and its Application to Depth Data Fusion" [188].

9.1 Introduction

Among the various possible solutions for depth estimation, two techniques are increasing their popularity due to their simplicity, low cost and capability of handling dynamic scenes: stereo vision systems and matricial Time-of-Flight (ToF) sensors. Even if widely deployed in several practical applications, both approaches have their specific shortcomings. Stereo vision uniquely rely on images framed by standard imaging devices and thus provides unreliable depth measurements when the matching of corresponding points is ambiguous/challenging, like for instance when dealing with low-textured regions or edges. On the other hand, ToF sensors have a limited resolution and suffer from mixed pixels and Multi-Path Interference (MPI) artifacts [5, 61, 162, 278] although they give perfectly aligned silhouettes.

Regardless of the depth sensing technique deployed, for the reasons outlined, extracting reliable confidence metrics for depth data is a relevant task. For stereo vision, confidence estimation has been a widely explored research field and recently traditional techniques have been outperformed by a large margin by machine learning approaches. On the other hand, inferring confidence estimation from ToF sensors is an almost unexplored field. Thus, in this chapter we will present different methods for this purpose, ranging from simple traditional strategies to the adaptation of machine learning (ML) approaches developed for stereo vision and *ad-hoc* ML strategies explicitly targeted to ToF sensors.

Among the various applications, confidence data proved to be very useful when depth data coming from the two approaches needs to be combined together. For this task it is important to ensure that the two confidence metrics are consistent and we will show that by using deep networks able to jointly estimate both confidence metrics yields the best results. Finally, we will exploit the proposed confidence estimation strategies into different depth data fusion frameworks. An extensive experimental evaluation on three different datasets, including both synthetic and real world scenes, has been carried out for the confidence measures and for the data fusion algorithms. The results show how the proposed deep learning approaches, specially if jointly trained on both devices, allow to obtain a very accurate confidence information. Furthermore, state-of-the-art results on stereo-ToF fusion can be obtained by exploiting the computed confidence data to guide the fusion algorithm.

In this work, we refer to as d and z for disparity and depth values sourced by stereo and ToF. We denote with $\hat{C}_S$ and $\hat{C}_T$ the estimated confidence maps for stereo and ToF respectively, and as C_S and C_T the corresponding ground truth confidence labels that range from 1 (when the error estimated on d or z is 0) to 0 (when the estimated d or z is not reliable).

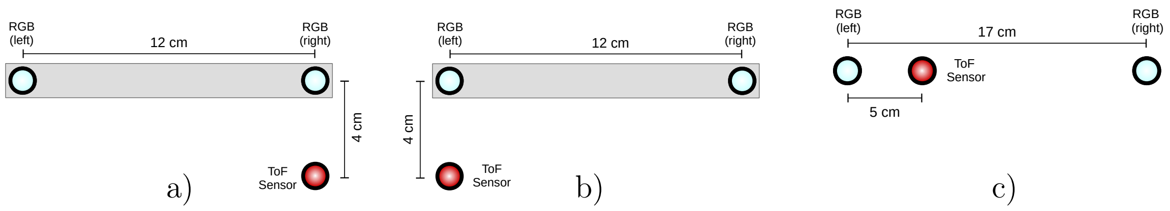


Figure 9.1: Sensor arrangements: a) setup used in the SYNTH3 dataset; b) setup used in the REAL3 dataset; c) setup used in the LTTM5 dataset.

9.2 Joint Estimation Of Stereo And ToF Confidence

In order to combine stereo and ToF information, it is paramount to have two consistent ways of evaluating confidence. Even if the two confidence maps can be estimated separately, we will show in Section 15.3 that the best results are achieved by jointly estimating the two maps.

The first work to introduce this idea was [6], where a CNN taking in input a multi-channel representation containing stereo and ToF data, projected on the reference camera of the stereo system, was in charge of estimating confidence scores for both sensors. This approach has been refined in [7]: we will refer to it as ST-CNN* in the results section. From now on the * is used to highlight methods jointly estimating confidence for both sensors together. In [7] the CNN takes in input multiple clues, i.e., the two disparity maps, the ToF amplitude information (used also by [145] but not by the other strategies considered in this work) and the difference between the left image of the stereo system and the right one warped over it using the stereo disparity. As shown in [7], the two additional channels allow to obtain a slight improvement in the confidence estimation w.r.t. using only disparity information. This network has two outputs and can be trained on synthetic data by minimizing a loss containing two components, one for each sensor. More in detail, the confidence is computed as a negative exponential function of the error, i.e., $C(p) = e^{-|\hat{d}(p)-d(p)|}$ and the loss to be minimized is:

$$\mathcal{L} = \sum \left(\hat{C}_T(p) - C_T(p)\right)^2 + \sum \left(\hat{C}_S(p) - C_S(p)\right)^2 \tag{9.1}$$

where $C_S(p)$ and $C_T(p)$ are the ground truth confidence values for stereo and ToF while $\hat{C}_S(p)$ and $\hat{C}_T(p)$ are the ones estimated from the network.

Consistently, to account for both sensors, the state-of-the-art CNN-based confidence measures CCNN [179] and LGC [240] originally proposed for stereo can be modified to jointly infer confidence scores for stereo and ToF by doubling the inputs and outputs of the network. In the next of this chapter we will refer to their joint confidence estimation respectivelly with CCNN* and LGC*. Nonetheless, in contrast to ST-CNN* [8], their training relies only on depth data provided by the two sensors without any additional feature. Moreover, since approaches developed for stereo traditionally minimize a binary cross-entropy loss and, in this case, both modalities could be correct in terms of inlier vs outlier classification, we convert the task to a multi-labeling classification problem [180] and minimize for the following objective loss:

$$\begin{aligned}\mathcal{L} = &\sum \Big(C_T(p) \log \hat{C}_T(p) + (1 - C_T(p))) \log(1 - \hat{C}_T(p)\Big) \\ &+ \sum \Big(C_S(p) \log \hat{C}_S(p) + (1 - C_S(p))) \log(1 - \hat{C}_S(p)\Big)\end{aligned} \tag{9.2}$$

where in this case $C_T(p)$ and $C_S(p)$ are binary labels that can be 0 or 1 depending if the corresponding sensor has an error smaller than a pre-defined threshold, while $\hat{C}_T(p)$ and $\hat{C}_S(p)$ are the ones estimated from the network.

Concerning O1 [178], the joint training is not feasible without significant modifications, and hence, it has been trained to infer confidence estimation independently for stereo and ToF.

9.3 Fusion of stereo and ToF data

Data fusion is a widely adopted strategy in many application fields such as wireless sensor networks [46], remote sensing [42], and network traffic analysis [4] to name a few. Concerning depth sensor fusion these two sensor technologies have rather complementary strengths and drawbacks making data fusion based on such setup quite appealing to obtain reliable depth information. Specifically, we will consider a trinocular setup like the ones depicted in Figure 9.1.

We will assume that the setup has been calibrated (we used the approach of [49] for this purpose) and that ToF data has been reprojected to the stereo viewpoint and interpolated to the same resolution of the stereo information. For the interpolation, we used the method of [50] based on an extended version of the cross bilateral filter.

In the considered setup, two different depth (or disparity) fields relating to the same viewpoint and at the same resolution are available. Different strategies can be exploited to combine the output of the two sensors taking into account confidence estimation. Purposely, we consider two simple approaches and a more advanced fusion strategy.

A first straightforward solution, referred to as *Highest Hypothesis (HH)*, consists of selecting at each pixel location, the disparity source (stereo or ToF) that has the highest confidence. Despite its simplicity, this strategy is fast, and, provided that confidence information is reliable, allows to significantly reduce artifacts when one of the two approaches is entirely unreliable (e.g., in case of wrong matches for the stereo system).

A second strategy, referred to as *Weighted Average (WA)*, consists of a weighted average of the two disparity values d_T and d_S, computed according to the estimated confidence values as follows:

$$d = \frac{(\hat{C}_T + \epsilon) * d_T + (\hat{C}_S + \epsilon) * d_S}{\hat{C}_T + \hat{C}_S + 2\epsilon} \tag{9.3}$$

where ϵ is a small constant introduced to avoid issues when both acquisition systems have confidence close to 0. Compared to the previous one, this strategy is more flexible and can output any depth value in the middle between the 2 measures. It typically yields better results when the two depth values are both reliable. However, if one sensor provides a wrong value and its confidence score is low but not close to 0 can easily lead to artifacts. It is worth observing that although not very reliable, as reported next, an additional strategy, referred to as *Average*, can be obtained by neglecting the confidence contribution in WA (i.e., assuming the same weight for both sensors).

Finally, we consider a more advanced fusion strategy referred to as *LC*, based on the Locally Consistent fusion framework, introduced in [148] for stereo disparity refinement and extended to stereo-ToF fusion in [50, 145]. In its original formulation [148], the Locally Consistent framework aimed at inferring depth from a stereo pair exploiting a patch-based strategy and assuming piece-wise smooth surfaces in the sensed scene. Specifically, it analyzes the multiple depth hypotheses enforced for each point during the local processing in order to determine the most likely one accordingly. When tackling sensor fusion, the rationale behind this strategy can be exploited for reasoning about multiple depth maps, for instance, obtained by stereo and ToF as in [50]. Moreover, such an approach for sensor fusion can be further improved by taking into account confidence estimation as done in [145] and in the experiments reported in the next section.

9.4 Experimental Results

The experimental evaluation has been carried out on three different datasets, a synthetic dataset and two smaller sets of real-world scenes. Since this work proposes both a new set of confidence measures and various data fusion strategies, we divide the experimental evaluation into two parts: we firstly assess confidence estimation and then analyze the data fusion results according to standard evaluation protocols.

9.4.1 Datasets

The first dataset is the SYNTH3 dataset [6]: it contains 55 synthetic scenes created using the Blender 3D rendering software and the Sony ToF Explorer simulator from Sony EUTec (that is based on the work of [153]). The parameters of the virtual cameras and their arrangement have been chosen in order to resemble an acquisition system composed by a Kinect v2 ToF sensor below a ZED stereo camera. Fig. 9.1a shows the camera setup: the stereo system has a baseline of 12 cm and the ToF sensor is placed below the right stereo camera at a distance of 4 cm. The data is split into a training set with 40 scenes and a test set with the remaining 15 scenes. The scenes have a large variability and include indoor and outdoor environments of very different sizes with objects of various shapes, material and color. Notice that this is the only dataset large enough to perform training of ML-based approaches in a ToF-stereo fusion framework. Hence, all the learning-based confidence measures have been trained on this dataset.

The second dataset is the REAL3 dataset [7]; it contains 8 real-world scenes, and due to its small size has been used only for testing purposes. In contrast to the previous one, it contains real-world data. The scenes have been acquired with a Kinect v2 ToF sensor and a ZED stereo camera, deploying the SGM algorithm [83], while ground truth information has been obtained using two synchronized color cameras and a line laser (see [7] for more details on how the dataset has been created). The ToF and stereo camera placement is depicted in Fig. 9.1b, it is as in the previous dataset but with the ToF sensor below the left camera. The scenes are all indoor scenarios and include both simple flat surfaces and objects with a more complex geometry made of different materials.

The last dataset is LTTM5 [52]: it is a real-world dataset containing only 5 scenes all depicting various objects put on a table acquired with a stereo system and a ToF camera arranged as in Fig. 9.1c, i.e., with a larger baseline of 17 cm and the ToF sensor placed between the two color cameras (closer to the left one). Despite its small size, it is interesting since it has been used to evaluate many stereo-ToF fusion approaches and allows to compare with the state-of-the-art in the field.

9.4.2 Training of Learning-based Approaches

Some of the stereo and ToF confidence estimators employed in this work rely on machine learning techniques. In particular, the deep learning approaches ST-CNN, CCNN and LGC have been trained using the training split of the SYNTH3 dataset and the hyper-parameters shown in Table 9.1. Please notice that the different methods have different

	ST-CNN*	CCNN / CCNN*	LGC / LGC*
Learning rate	10^{-1}	10^{-3}	10^{-3}
Epochs	500	14	14/1600/14
Optimizer	AdaDelta [280]	SGD	SGD
Regularization	l_2 ($\lambda = 10^{-2}$)	-	-

Table 9.1: ST-CNN, CCNN and LGC training hyper-parameters.

	SYNTH3		REAL3		LTTM5	
	Stereo	ToF	Stereo	ToF	Stereo	ToF
ST-D	12.97	14.71	53.53	79.35	10.00	31.56
DA	4.95	12.68	44.52	72.15	4.29	29.25
DS	5.46	18.05	45.46	70.30	4.67	30.53
O1	4.10	10.45	42.91	72.08	3.85	22.81
ST-CNN*	3.26	11.05	45.02	**66.85**	4.47	22.16
CCNN	5.24	20.63	44.35	69.94	3.20	20.84
CCNN*	**2.59**	**10.41**	**40.19**	75.10	2.84	**15.28**
LGC	3.34	16.40	43.88	67.33	3.13	18.82
LGC*	2.75	12.08	41.03	76.41	**2.25**	18.50
Opt.AUC	1.54	3.84	34.96	48.09	0.77	8.76
Err.rate (%)	15.16	21.10	67.04	78.11	12.09	37.08

Table 9.2: Confidence evaluation: AUC values ($\times 10^2$) with threshold 1.

parameters, but the networks jointly estimating ToF and stereo confidences (ST-CNN*, CCNN* and LGC*) share the same hyper-parameters of their base implementation. For what concerns the LGC method, CCNN, ConfNet and the final module have been trained for 14, 1600 and 14 epochs, respectively.

9.4.3 Confidence Evaluation

We start from evaluating confidence measures on stereo and ToF data according to the standard protocol used in this field [89, 184] on the 3 datasets. Tables 9.2, 9.3 and 9.4 show the AUC values of the different considered confidence metrics for both stereo and ToF with the error threshold set to 1, 2 and 4 respectively. All tables report, in different columns, results on the three datasets mentioned above. On the bottom, we also report both the optimal AUC obtained according to [89, 184]. The scores have been multiplied by a factor 10^2 to ease readability.

Starting from the SYNTH3 dataset, it is possible to see how learning-based approaches (O1 [178], CCNN [179], LGC [240], ST-CNN* [7]) have in general better performance

	SYNTH3		REAL3		LTTM5	
	Stereo	ToF	Stereo	ToF	Stereo	ToF
ST-D	9.96	5.26	47.18	51.91	6.72	14.13
DA	3.46	4.69	37.92	42.19	2.69	11.67
DS	3.87	11.02	38.82	39.34	3.00	14.84
O1	2.60	2.39	36.12	41.61	2.33	7.24
ST-CNN*	1.95	7.12	35.25	36.00	2.06	4.67
CCNN	3.56	8.96	37.74	37.20	2.04	5.73
CCNN*	1.38	2.21	**30.63**	44.22	0.90	**3.50**
LGC	2.28	6.00	37.47	**30.79**	1.91	4.88
LGC*	**1.35**	**2.20**	31.13	44.12	**0.73**	4.18
Opt.AUC	0.94	0.82	26.91	18.22	0.39	2.06
Err.rate (%)	11.85	11.91	60.24	49.31	8.60	18.52

Table 9.3: Confidence evaluation: AUC values ($\times 10^2$) with threshold 2.

	SYNTH3		REAL3		LTTM5	
	Stereo	ToF	Stereo	ToF	Stereo	ToF
ST-D	7.79	3.58	40.27	3.95	5.67	3.68
DA	2.35	3.56	32.10	3.35	2.39	7.22
DS	2.76	10.01	32.75	3.25	2.68	11.69
O1	1.59	1.56	29.89	3.44	2.04	1.84
ST-CNN*	1.31	6.09	25.92	3.63	1.58	2.11
CCNN	2.15	6.43	31.65	3.85	1.78	1.74
CCNN*	0.86	1.48	**21.64**	3.60	0.62	1.23
LGC	1.47	4.22	31.65	2.79	1.63	1.51
LGC*	**0.78**	**1.41**	21.88	**2.72**	**0.51**	**1.08**
Opt.AUC	0.58	0.45	19.05	0.54	0.30	0.64
Err.rate (%)	9.36	9.01	51.87	7.43	7.41	11.03

Table 9.4: Confidence evaluation: AUC values ($\times 10^2$) with threshold 4.

than traditional ones (ST-D, DA and DS although these latter two methods are rather effective).

Focusing on CCNN and LGC approaches: they have been trained both independently on stereo and ToF and jointly on the two sensors (tagged in this case, respectively, as LGC* and CCNN* in the tables). ST-CNN* has instead always been trained jointly as initially proposed in [7]. We can note that the joint training on the stereo and ToF data consistently yields much better performance. Moreover, as reported later, such a strategy will be particularly helpful when dealing with the fusion problem where the consistency between the two confidence metrics is a fundamental requirement for achieving high performance.

According to tables 9.2, 9.3 and 9.4, on the SYNTH3 dataset the two best performing approaches, for both stereo and ToF, are CCNN* and LGC* jointly trained on both modalities. However, it is worth noticing that these two approaches are trained to minimize a classification loss function that is ideal for reducing the AUC, while ST-CNN* is trained with a regression loss where the ground truth confidence measure has been computed as a function of the sensor disparity error thorugh Equation (9.1). The CCNN* approach is the best when the AUC threshold is set to 1 while LGC* leads to better performance on both stereo and ToF data when considering larger thresholds. Thus, LGC is less effective with smaller errors but better when dealing with higher magnitude outliers. In particular, in these latter cases (Tables 9.3 and 9.4) LGC gets quite close to the optimal AUC. The ST-CNN* approach has a relatively good performance on this dataset, especially for what concerns the stereo data, demonstrating again that learning-based approaches jointly trained on stereo and ToF are the best family of solutions. Although O1 performs relatively well, it is always outperformed by LGC* and CCNN* with all thresholds.

A fundamental problem for deep learning approaches is the risk of focusing too much on the training dataset, that in this case, for the reasons outlined before, is entirely composed of synthetic data. Therefore, it is essential to assess how they can generalize to real-world scenarios represented by datasets REAL3 and LTTM5. The tests on the REAL3 dataset show how the learned confidence measure keep excellent performance even if the gap with traditional ones gets smaller compared to the synthetic dataset.

On the REAL3 dataset, as on synthetic data, the competition is still between LGC* and CCNN* jointly trained on the stereo and ToF data. Nonetheless, considering the smallest threshold, ST-CNN* turns out to be the best on ToF data. Moreover, the gap between traditional DA and DS confidence measures and learning-based ones is reduced if compared to SYNTH3.

Finally, on the other real-world LTTM5 dataset LGC* and CCNN* approaches are overall the best ones. In particular, LGC* is always the best on stereo data and with the ToF sensor with threshold 4. CCNN* is the best in the other 2 cases with ToF data.

	SYNTH3			REAL3			LTTM5		
	LC	WA	HH	LC	WA	HH	LC	WA	HH
ST-D	2.02	2.13	2.59	2.97	3.04	3.33	2.79	2.78	3.04
DA	2.40	2.01	2.33	3.91	5.12	3.66	3.23	2.91	3.58
DS	2.39	2.15	2.46	3.96	5.42	4.99	3.46	3.36	3.94
O1	1.99	1.77	1.98	3.23	4.60	3.90	3.25	2.97	3.49
ST-CNN*	1.97	**1.66**	**1.80**	2.88	4.05	3.78	**2.70**	**2.70**	3.12
CCNN	2.04	1.80	2.03	3.24	4.00	3.25	3.44	2.90	3.42
CCNN*	1.92	1.91	2.03	**2.40**	2.75	**2.60**	2.74	2.91	3.00
LGC	2.00	1.74	1.95	3.21	3.77	2.87	3.53	3.13	3.53
LGC*	**1.83**	1.89	2.03	2.49	**2.65**	**2.60**	2.75	2.85	**2.95**
Average	2.34			7.50			3.04		
Stereo	3.67			14.19			4.47		
ToF	2.18			3.28			3.40		

Table 9.5: Fusion accuracy measured with RMSE.

ST-CNN* and O1 have overall good performance and again DA and DS are less reliable but with a smaller gap compared to the synthetic case.

From this exhaustive evaluation we can notice how results are consistent on all the experiments, showing that learning approaches jointly trained for estimation of ToF and stereo confidences are the best solution and that, on average, LGC* is the best technique.

9.4.4 Fusion of Stereo and ToF data

Once assessed the performance of confidence measures, we leverage this cue for the fusion of the two disparity fields generated by stereo and ToF sensors. Specifically, we evaluated all the confidence measures existing for stereo and ToF with the fusion strategies outlined in Section 9.3. The outcome concerning the Root Mean Square Error (RMSE) between the fused disparity maps and ground truth data is shown in Table 9.5. We also report in the table the RMSE for raw ToF and stereo data and the simple fusion scheme averaging the two disparity values at each location.

On the SYNTH3 dataset, the ST-CNN* approach allows obtaining excellent results deploying simple fusion strategies, especially with WA. On the other hand, with LC, the LGC* approach performs better. This behavior changes on the REAL3 dataset where LC achieves the best results with CCNN* although straightforward fusion strategies coupled with LGC* and CCNN* do pretty well. A similar trend is observed on the LTTM5

SYNTH3	$\epsilon = 1$			$\epsilon = 2$			$\epsilon = 4$		
	LC	WA	HH	LC	WA	HH	LC	WA	HH
ST-D	14.05	13.96	13.57	4.93	4.99	4.72	2.26	2.23	2.24
DA	9.89	8.95	7.58	4.74	4.57	4.50	2.54	2.41	2.47
DS	9.88	8.94	8.49	4.79	4.72	4.41	2.57	2.57	2.47
O1	9.98	8.94	8.86	4.35	4.19	4.08	2.25	2.12	2.13
ST-CNN*	**8.26**	8.13	6.88	**3.77**	4.02	**3.50**	1.94	1.81	1.72
CCNN	9.79	9.35	10.36	4.63	4.69	4.03	2.40	2.30	1.72
CCNN*	9.61	8.94	7.54	4.20	3.52	3.69	1.94	1.71	1.72
LGC	9.84	9.62	**6.69**	4.67	4.79	3.74	2.40	2.32	1.81
LGC*	9.35	**7.78**	9.69	3.92	**3.15**	3.75	**1.71**	**1.65**	**1.65**
Average		11.30			6.30			3.42	
Stereo		10.12			6.80			4.22	
ToF		14.92			5.35			2.25	

Table 9.6: Fusion accuracy measured as the *PERCENTAGE OF WRONG PIXELS* on the SYNTH3 dataset.

dataset, with LC and WA coupled with ST-CNN* yielding the best results although CCNN* and LGC* allow to obtain slightly worse accuracy. A first thing to notice from the table, and experiments on other datasets reported next, is that the simple average of the two disparity fields is often unable to improve the accuracy of the output disparity map compared to the raw depth maps coming from the ToF and stereo devices, resulting in between the two in terms of accuracy. Moreover, the use of confidence maps in the fusion process helps to reliably combine the two depth data sources. By using accurate confidence maps, provided by learning-based methods, and very accurate data like the synthetic one available in SYNTH3 even the straightforward fusion strategies WA and HH allow getting excellent results. However, the more complex LC fusion strategy turns out quite useful on real-world datasets (especially with REAL3). Moreover, a second thing to notice is that training ST-CNN* by minimizing a regression loss turns out not optimal concerning AUC evaluation, but it helps to obtain a better granularity when dealing with small errors in the LC framework. ST-CNN* however also exploits additional features that help in driving the fusion process.

Tables 9.6, 9.7 and 9.8 report for the three datasets the fusion outcome in term of percentage of wrong pixels. The error bound set to discriminate the goodness of the disparity estimation is respectively set to 1, 2 and 4 pixels as for confidence evaluation.

REAL3	$\epsilon = 1$			$\epsilon = 2$			$\epsilon = 4$		
	LC	WA	HH	LC	WA	HH	LC	WA	HH
ST-D	74.07	81.77	81.58	47.11	49.24	53.13	6.04	6.49	7.06
DA	57.93	70.41	51.16	33.11	38.76	27.98	3.68	22.41	**4.17**
DS	55.56	66.53	51.61	31.27	38.27	28.90	3.77	23.63	6.88
O1	57.89	69.53	55.91	33.06	39.91	32.27	3.94	22.13	6.68
ST-CNN*	**54.75**	**61.06**	51.22	**30.35**	37.53	30.03	3.93	16.89	8.45
CCNN	60.13	77.00	76.42	35.52	39.10	49.83	4.70	17.74	6.64
CCNN*	57.55	69.63	**48.35**	32.54	29.26	**26.63**	**3.11**	6.79	5.46
LGC	60.49	77.24	58.71	36.00	38.97	35.26	5.26	15.21	6.13
LGC*	57.14	66.04	48.58	32.15	**27.73**	26.96	3.25	**5.65**	5.50
Average		81.33			50.12			34.17	
Stereo		57.44			50.13			42.17	
ToF		83.67			55.37			7.93	

Table 9.7: Fusion accuracy measured as the *PERCENTAGE OF WRONG PIXELS* on the REAL3 dataset.

LTTM5	$\epsilon = 1$			$\epsilon = 2$			$\epsilon = 4$		
	LC	WA	HH	LC	WA	HH	LC	WA	HH
ST-D	28.94	33.56	33.85	10.69	13.74	14.78	3.87	5.18	5.03
DA	18.36	18.07	13.54	6.84	8.73	7.32	4.53	5.62	5.27
DS	**15.13**	**14.70**	13.37	6.56	9.00	7.72	5.09	7.01	6.12
O1	16.55	19.49	16.46	6.88	9.55	8.53	4.59	6.08	5.20
ST-CNN*	17.24	20.68	20.05	**5.40**	9.11	7.50	3.59	6.01	5.24
CCNN	19.94	23.69	31.01	7.77	10.84	13.50	4.96	6.64	5.50
CCNN*	17.54	18.07	18.04	5.82	**6.57**	5.99	3.54	4.59	4.34
LGC	20.64	25.17	23.02	8.01	10.15	10.56	5.19	6.62	5.95
LGC*	17.48	18.28	**10.79**	5.75	6.77	**5.70**	**3.53**	**4.58**	**4.28**
Average		24.03			12.46			8.85	
Stereo		13.72			9.31			7.67	
ToF		37.14			17.27			6.66	

Table 9.8: Fusion accuracy measured as the *PERCENTAGE OF WRONG PIXELS* on the LTTM5 dataset.

Concerning the evaluation on the SYNTH3 dataset, from Table 9.6 we can notice that the trend is substantially the same observed when evaluating the RMSE. The simple fusion strategies WA and HH yield the best performance when coupled with LGC with threshold 1 and with LGC* with the other thresholds. On the REAL3 dataset (Table 9.7), HH with CCNN* is the best fusion method regarding the percentage of pixels with error less than 2 pxl, instead LC in combination with the same confidence measure is able to reduce the most the error higher than 4 pxl. In all cases, CCNN* is the confidence measure yielding the best results. Regarding the LTTM5 dataset (Table 9.8), results are more variegate: LC achieves the lowest percentage of bad pixels with threshold 2 and

	LC ST-CNN*	WA ST-CNN*	HH LGC*	[50]	[266]	[294]	[52]
RMSE	**2.70**	**2.70**	2.95	3.17	3.31	3.34	3.49

Table 9.9: Comparison with state-of-the-art fusion methods on the SYNTH3 dataset.

	ST-D	DA	DS	O1	ST-CNN*	CCNN*	LGC*
Runtime [ms]	480.6	868.9	868.9	3534.2	102.6	26.7	131.9

Table 9.10: Runtime of the confidence estimation techniques.

4, respectively coupled with ST-CNN* and CCNN*, while HH confirms to be the best method with threshold 1, especially when coupled with LGC*.

From the results reported so far, we can notice that the confidence measure with the best AUC is not necessarily the best method for depth fusion. On the other hand, it is quite evident that the best results are typically obtained by confidence measures showing good performance according to the AUC metric. Moreover, the joint training of confidence measures turns out to be very useful as done for ST-CNN*, CCNN* and LGC*.

In Table 9.9 we also report the evaluation on the LTTM5 dataset of 4 state-of-the-art approaches: namely, [50] that was the first to use LC for stereo-ToF fusion, the MAP-MRF Bayesian frameworks proposed in [294] and [52], and the approach based on bilateral filtering of the cost volume introduced in [266]. The results clearly show how the best strategies proposed in this chapter outperform previous approaches known in the literature.

9.4.5 Qualitative Results

Figures 9.2, 9.3 and 9.4 report qualitative experimental results on a sample scene extracted respectively from the SYNTH3, REAL3 and LTTM5 datasets. The first row contains the disparity maps coming from the ToF and stereo sensors and the results of the LC fusion using the ST-CNN* and the LGC* confidences. We selected these fusion strategies since they are on average the best performing on the 3 datasets. The second row contains the disparity error maps computed as the true disparity minus the target disparity. The color

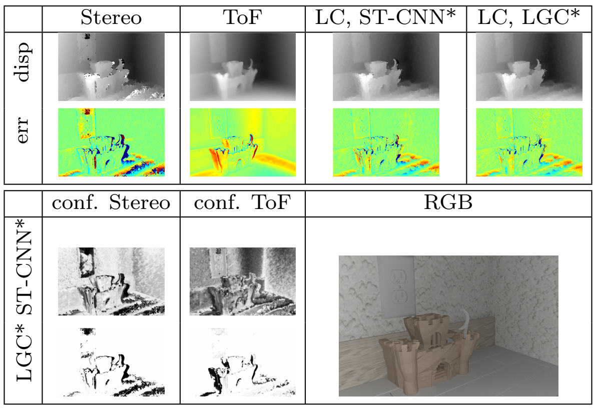


Figure 9.2: Qualitative results on the SYNTH3 dataset.

map encodes the correct estimation with green, the disparity overestimation with colder colors and disparity underestimation with warmer colors. From the figures, it is possible to notice well known issues of ToF and stereo. Specifically, concerning stereo, we can observe poor performance on textureless regions and in case of repeating patterns like the green box with the white grid in Fig. 9.4. On the other hand, the main issues of the ToF sensor arise on the object sides, due to the original low spatial resolution of the sensor, and the disparity underestimation near to corners, due to the multi-path interference. The error maps related to the two fusion strategies show a large overall reduction of the error: the proposed approaches are able to take the best from the 2 depth sources thus avoiding the stereo artifacts on un-textured regions and greatly reducing the MPI corruption in the ToF data. The third and fourth column contain the ST-CNN* and LGC* confidence maps which have values in the range from 0 (not reliable pixels) to 1 (highly reliable pixels). Both of of them follow quite accurately the error distribution, but with different behaviour. LGC* is more binarized, since it is trained using a classification loss, instead ST-CNN* has a smoother transition, since it is trained with a regression loss.

9.4.6 Runtime analysis

In order to asses the computational complexity of various fusion methods considered, we report the runtime of all the steps of the various fusion methods on the REAL3 dataset.

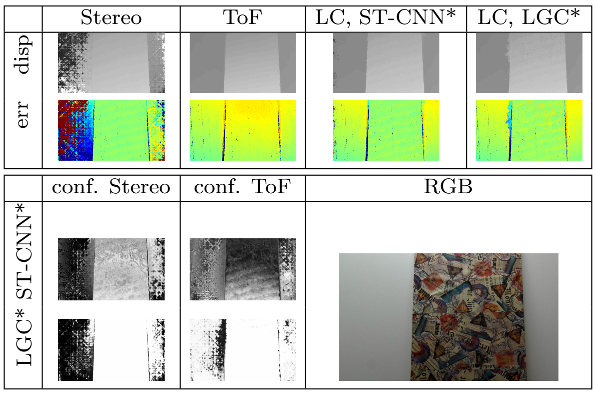


Figure 9.3: Qualitative results on the REAL3 dataset.

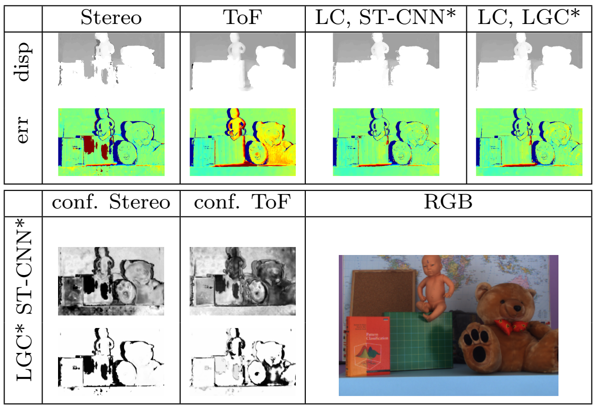


Figure 9.4: Qualitative results on the LTTM5 dataset.

These tests have been carried out on a PC with an Intel i7-4790 CPU and an NVIDIA Titan X GPU (used only for CNN-based techniques). Starting from pre-processing steps, the stereo disparity map computation with SGM [83] takes 1.49 s and the reprojection and interpolation of the ToF depth on the reference camera of the stereo system takes 4.47 s. These two operations are carried out for all the fusion techniques.

Table 9.10 collects the runtime for confidence estimation. They range from just 26 ms for CCNN* to more than 3 s for O1. For methods allowing the joint confidence estimation of ToF and stereo, the separate estimation of the two confidences roughly doubles the computation time compared to the joint estimation since two inferences on two different networks are required.

Concerning the final fusion step, the simplicity of the average, HH and WA schemes allows us to perform these tasks in less than 5 ms. In contrast, LC is much more complex and requires about 35 s.

9.5 Conclusions

Time-of-Flight and stereo are two popular depth sensing technologies with quite complementary strengths and limitations. For this reason, they are often combined to infer more accurate depth maps. Therefore, inspired by recent advances in stereo confidence estimation in this work we introduce and evaluate learning-based confidence estimation strategies suited for depth data generated by ToF and stereo sensors showing how a joint training of such methods yields in general better performance. Moreover, deploying three fusion frameworks, we report how confidence estimation can effectively guide the fusion of data generated by the two depth sensing technology. Exhaustive experimental results show how the accurate confidence cues obtained allow to outperform state-of-the-art data fusion schemes even deploying straightforward fusion strategies.

Chapter 10

Learning confidence measures in the wild

The content of this chapter has been presented at the British Machine Vision Conference (BMVC 2017) - “Learning confidence measures in the wild” [239].

10.1 Introduction

Regardless of their specific deployment purpose, confidence estimation techniques based on machine-learning require a significant amount of training samples obtained from *ground truth* data. In general, the higher amount and variety of labeled data available, the more effective the confidence estimation is. However, excluding a tedious and time consuming manual labeling, accurate ground truth labels require either not trivial setup based on structured light, as described in [213], or expensive and appropriately registered active sensors, typically LIDAR, as done in [67, 155]. The first strategy provides dense (i.e., available for each point) ground truth labels but it is only suited for still scenes acquired in indoor environments while the latter one enables to determine sparse ground truth data from any indoor and outdoor environment. To overcome these issues, synthetic datasets have been recently deployed to train end-to-end stereo methods based on CNNs [151] with

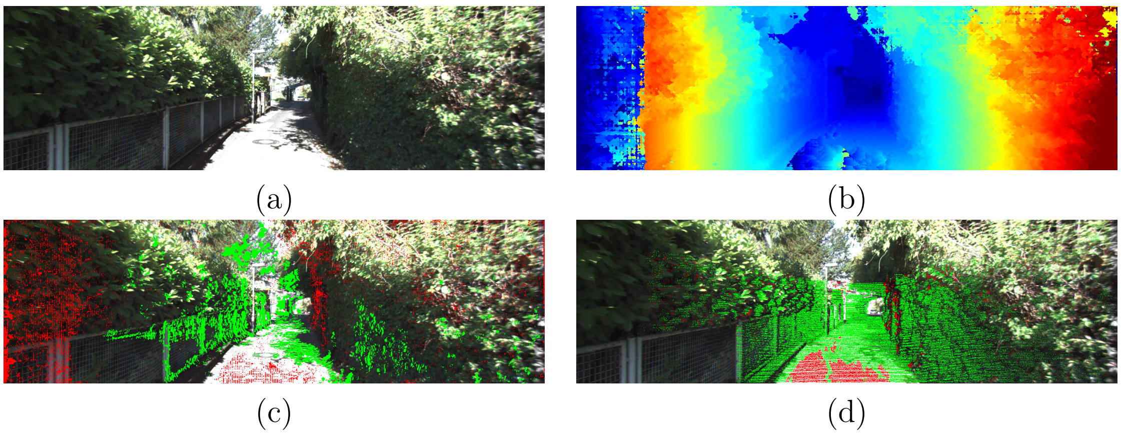


Figure 10.1: Overview of our proposal. (a) Reference frame 000122 from KITTI 12 dataset [67], (b) disparity map generated by SGM algorithm [83], (c) labels inferred by our method and (d) labels assigned comparing the output of SGM with ground truth data. In green and red, respectively, *correct* and *wrong* labels.

satisfactory results. However, such method requires an additional fine tuning on large labeled real data (e.g., the whole 194 images of the KITTI 2012 training dataset in [151]) to achieve top performance on standard datasets. Thus, regardless of the desired goal, self-supervised and accurate labeling of disparity is crucial when dealing with machine-learning algorithms that require, for a specific application domain, a large amount of training samples as would occur in most practical circumstances.

To this aim Mostegel et al. [160] proposed an automatic technique, referred to as SELF, capable to automatically assign labels to train confidence measures by leveraging on contradictions and consistencies between disparity maps generated by the same stereo algorithm from multiple view points. This self-supervised approach proved to be very effective but it intrinsically suffers of two strong limitations. Firstly, it requires image sequences which are not always available. For instance, the Middlebury 2014 dataset [213] does not provide such data at all. Moreover, this method accounts for camera ego-motion but it does not enable to detect labels belonging to moving subjects, such as cars or pedestrians, in the sensed environment.

Therefore, to overcome these issues we propose an approach to automatically generate, in a self-supervised manner, labels for training confidence measures without any of the aforementioned constraints. Our method, given a disparity map generated by a stereo algorithm, assigns a *correct* label to highly confident points and a *wrong* label to poorly reliable disparity measurements leveraging on the joint estimation provided by a pool of conventional confidence measures which do not require any training phase. Figure 10.1 summarizes our proposal. Given a stereo pair (a) and the disparity map generated by a stereo algorithms (b), we determine (c) training labels by assigning correct (green) or wrong (red) labels according to the joint confidence estimation carried out by means of conventional measures. In this very image, compared to ground truth, our method correctly estimates 97.57% of correct and wrong labels. In the same figure, (d) shows for the same disparity map the intersection with ground truth points.

We assess the performance of our self-supervised labeling approach on three challenging datasets (KITTI 12 [67], KITTI 15 [155] and Middlebury 2014 [213]) with three stereo algorithms characterized by different accuracy (block-matching, MC-CNN [279] and SGM [83]) by training on labels inferred by our method three state-of-the-art confidence measures [178, 179, 217] based on machine-learning. Our experimental evaluation with three state-of-the-art confidence measures clearly highlights that, using the same images for training, the proposed method not only provides an unconstrained labeling strategy with respect to SELF [160] but it also yields much more accurate confidence estimation.

10.2 Self-supervised labeling

In this section we outline our proposal to automatically determine training labels from stereo pairs in order to obtain a distribution of training labels as much as possible similar to GT data. The fundamental underlying assumption made by our method concerns the capability of a combination of *hand-crafted* confidence measures to discriminate between correct and wrong disparity assignments generated by a stereo algorithm. This selection procedure allows us to obtain two distinct labels, *correct* and *wrong*, that can be used as training samples for state-of-the-art confidence measures based on machine-learning. The primary goal of this method is to find a set of values as accurate as possible with the aim of reducing the number of false positive and false negative labels which could negatively affect training and consequently inference. Since we want to avoid a *chicken-and-egg* situation we can't rely on machine-learning confidence estimation for label selection and thus a careful choice of traditional confidence measures is mandatory.

The effectiveness of a specific confidence measure is quantitatively assessed by means of a ROC curve analysis [89, 279] according to a standard procedure in this field [77, 169, 178, 179, 181, 183, 217, 228].

Our strategy relies on a set of conventional, yet according to the literature [54, 89, 114, 169, 228, 279] reliable, confidence measures to automatically generate classification labels with a distribution as much as possible similar to GT data required to train state-of-the-art measures based on machine-learning. Differently from [160], our proposal does not enforce any constraint on the input data being it suited for image sequences, for uncorrelated stereo pairs as well as for scenes containing moving objects.

10.2.1 Confidence measures for label selection

In this section we review the confidence measures adopted by our method. We carefully selected them according to the voting technique deployed to generate labels, explained in detail in section 10.2.2. Given the cost curve provided by a stereo algorithm for a pixel $\mathbf{p}(x, y)$, the chosen confidence measures process (a subset of) cues such as the minimum cost $c_1(\mathbf{p}) \equiv c_1(\mathbf{p}, d_1(\mathbf{p}))$ at disparity hypothesis $d_1(\mathbf{p})$, the second smallest local minimum as $c_{2m}(\mathbf{p}) \equiv c_{2m}(\mathbf{p}, d_{2m}(\mathbf{p}))$ at disparity hypothesis d_{2m} (and, in general, the cost for a certain disparity hypothesis d as $c_d(\mathbf{p})$), the disparity value $\mathcal{D}(\mathbf{p})$ assigned by *winner-takes-all* strategy to $\mathbf{p}$ and its corresponding pixel on the right image referred to as $\mathbf{p}'$, having disparity $\mathcal{D}^R(\mathbf{p}')$. We denote as $N_\mathbf{p}$ a squared patch centered on pixel $\mathbf{p}$ (of size 25×25 in our experiments).

- **Average Peak Ratio (APKR)** [114]: computed by processing the ratio between

$c(\mathbf{q}, d_{2m}(\mathbf{p}))$ and $c(\mathbf{q}, d_1(\mathbf{p}))$, averaged on a squared neighborhood.

$$APKR(\mathbf{p}) = \frac{1}{|N_{\mathbf{p}}|} \sum_{\mathbf{q} \in N_{\mathbf{p}}} \frac{c(\mathbf{q}, d_{2m}(\mathbf{p}))}{c(\mathbf{q}, d_1(\mathbf{p}))} \tag{10.1}$$

- **Left-Right Consistency** (**LRC**) [89, 279]: obtained by comparing the disparity of pixel $\mathbf{p}$ with the corresponding point $\mathbf{p}'$ on right disparity map.

$$LRC(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathcal{D}(\mathbf{p}) \neq \mathcal{D}^R(\mathbf{p}') \\ 1, & \text{otherwise} \end{cases} \tag{10.2}$$

- **Median deviation of disparity** (**MED**) [228]: represents the difference between disparity D on pixel $\mathbf{p}$ and the median disparity computed on a squared neighborhood:

$$MED(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathcal{D}(\mathbf{p}) \neq median_{N_{\mathbf{p}}}(\mathcal{D}(\mathbf{p})) \\ 1, & \text{otherwise} \end{cases} \tag{10.3}$$

- **Uniqueness Constraint** (**UC**) [54]: a binary measure that encodes with low confidence points colliding on the same pixel $\mathbf{p}'$ in the right image thus violating the *uniqueness* constraint:

$$UC(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathbf{p} \in Q \\ 1, & \text{otherwise} \end{cases} \tag{10.4}$$

  being Q the set of pixels matching the same point on the right image.

- **Winner Margin** (**WMN**) [89, 279]: obtained by processing the difference between local minimum c_{2m} and minimum cost c_1, normalized by the sum of costs over the entire disparity range.

$$WMN(\mathbf{p}) = \frac{c_{2m}(\mathbf{p}) - c_1(\mathbf{p})}{\sum_d c_d(\mathbf{p})} \tag{10.5}$$

- **Distance to Left Border** (**DLB**) [169]: distance from the left border of the image, thresholded to the maximum disparity value $\mathcal{D}_{max}$ set for the stereo algorithm:

$$DLB(\mathbf{p}) = \begin{cases} 0, & \text{if } x < \mathcal{D}_{max} \\ 1, & \text{otherwise} \end{cases} \tag{10.6}$$

10.2.2 Label selection strategy

Given a disparity map D generated by a stereo algorithm, we want to reliably assign on subset of points labels $\mathcal{L} = \{L_0, L_1\}$ standing, respectively, for wrong and correct. From each of the confidence measures previously described, we obtain a map $\mathcal{C}$ assigning values $\in [0,1]$ to each point $\in D$. We define two sets of points $\mathcal{C}_0$ and $\mathcal{C}_1$ one for each label L_0 and L_1. For binary confidence measures we simply assume as correct points $\mathbf{p}$ with $\mathcal{C}(\mathbf{p}) = 1$ and as wrong those with $\mathcal{C}(\mathbf{p}) = 0$ while for the others the choice is made by sorting all points $\in D$ in ascending order of confidence and then defining the two sets as:

$$\mathcal{C}_0 = \{\mathbf{p} \in D | 0 \leq \mathcal{C}(\mathbf{p}) \leq \delta_0\}, \quad \mathcal{C}_1 = \{\mathbf{p} \in D | 1 - \delta_1 \leq \mathcal{C}(\mathbf{p}) \leq 1\} \tag{10.7}$$

with (δ_0, δ_1) representing portions of the entire disparity map, corresponding to the least ($\mathcal{C}_0$) and most ($\mathcal{C}_1$) confident pixels. For example, with $(\delta_0, \delta_1) = (0.2, 0.2)$, $\mathcal{C}_0$ will group the 20% pixels having lowest confidence value and $\mathcal{C}_1$ the 20% having highest scores.

By following this strategy for each $\mathcal{C}$ in a pool $\mathcal{P} = \{\mathcal{C}', \mathcal{C}'', ..\}$ of confidence measures, we obtain two ensembles $\mathcal{P}_0 = \{\mathcal{C}'_0, \mathcal{C}''_0, ...\}$ and $\mathcal{P}_1 = \{\mathcal{C}'_1, \mathcal{C}''_1, ...\}$ for the two labels L_0 and L_1. We combine the different labeling hypothesis $\in \mathcal{P}$ provided by the measures to obtain the final sets $\mathcal{G}_0$, $\mathcal{G}_1$ as follows:

$$\mathcal{G}_0 = \bigcap_{\mathcal{C}^k \in \mathcal{P}_0} \mathcal{C}_0^k, \qquad \mathcal{G}_1 = \bigcap_{\mathcal{C}^k \in \mathcal{P}_1} \mathcal{C}_1^k \tag{10.8}$$

According to this strategy, in order to reduce false positives and negatives originated by each single measure, only pixels classified by all the confidence measures as either correct or wrong are used for labeling. On the other hand, this conservative strategy also reduces the amount of pixels for which our method provides labels. Our conservative selection strategy aims at obtaining very accurate labels comparable to those provided by GT data.

10.3 Experimental Results

In this section, we assess[1] the effectiveness of our proposal with three datasets and three stereo algorithms by training three state-of-the-art confidence measures with the labels generated by our method, the ones generated by SELF [160] as well as using ground truth data and comparing their performance by means of ROC analysis. Regarding the datasets, we consider KITTI 12 [67], KITTI 15 [155] and Middlebury 2014 [213]. As confidence measures we choose the three top-performing methods known in literature: O1 [178], CCNN [179] and PBCP [217]. The choice of these measures was driven by their effectiveness with respect to all other machine learning approaches. In particular, all of them proved to outperform the work of [169]. Concerning the stereo algorithms, we consider three approaches characterized by different performance. The popular, yet not very effective, block matching algorithm, referred to as CENSUS, aggregating costs (computed by means of Hamming distance on census transformed images) with a 5×5 box-filter. As representative of algorithms with high accuracy we use MC-CNN [279], considering the matching costs computed on patches (9×9 on KITTI 12 and KITTI 15

[1]For SELF [160], O1 [178], CCNN [179] and MC-CNN [279] we used the code available in the authors' web site while for PBCP [217], CENSUS and SGM we implemented them following the description available in each paper.

KITTI 12	CENSUS		MC-CNN		SGM	
Method	A	D / D∩GT	A	D / D∩GT	A	D / D∩GT
SELF [160]	88.9%	33.8% / 38.0%	85.4%	29.4% / 30.7%	81.3%	21.5% / 23.2%
Prop.	98.5%	8.4% / 12.5%	97.0%	12.4% / 13.3%	88.6%	12.5% / 14.6%

Table 10.1: Analysis of training labels inferred on 8 sequences of KITTI 12. For SELF [160] and our proposal we report the accuracy A for the predicted labels (computed for points with available ground truth), the average density D on the 8 sequences, the intersection between the density of labels inferred by the two methods and the 8 images with ground truth (D∩GT). The average density of KITTI 12 ground truth data on the 8 images is 19.5%.

and 11×11 on Middlebury 2014 and using the weights provided by the authors), and SGM [83] in a eight scanlines implementation using for data term the same CENSUS aggregated costs and for parameters P1 and P2, respectively, 0.03 and 3 (being matching costs normalized).

10.3.1 Training data

Confidence measures are trained in most works in this field [77, 169, 178, 217, 228] by selecting eight stereo pairs from KITTI 12 dataset: 43, 71, 82, 87, 94, 120, 122 and 180^{th}. These images with ground truth labels provide about 724K training samples. According to SELF [160], on the extended eight sequences available on KITTI 12 corresponding to the 8 stereo pairs 43, 71, 82, 87, 94, 120, 122 and 180^{th}, we generate training labels following the protocol described by the authors. For all considered sequences there are available 21 stereo pairs, excluding 82^{th} containing only 16. On such 163 stereo pairs SELF extracts a huge amount of training labels: about 25M for CENSUS, 22M for MC-CNN and 16M for SGM. For a fair comparison, we generate labels with our method from the same sequences. However, differently from SELF, we point out that our method is not constrained to sequences but we use for the aforementioned reason the same input data to generate our training labels. In fact, taking the same number of stereo pairs from different scenes would favour our approach making the comparison unfair. Overall, our framework provides from the eight sequences about 6M training labels for CENSUS and 9M for MC-CNN and SGM.

Despite the significantly lower amount of labels generated by our proposal with respect to SELF, observing Table 10.1 we can notice that our training samples are always more accurate. This fact highlights that our proposal significantly reduces the percentage of wrong assignments to $\mathcal{G}_0$ and $\mathcal{G}_1$ trading accuracy for density. Moreover, it is worth to note that KITTI 12 provides, on the 8 images, ground truth labels only for 19.5% of points. On the 8 sequences SELF always generates a larger percentage of labels, parameter D in the table, compared to our method. We can also notice from D∩GT that our method selects a larger percentage of points not overlapping with ground truth data with respect to SELF. This fact potentially allows us to include more points in regions not covered by LIDAR as shown in Figure 10.1 in the left and upper side of the disparity map. Moreover as reported in Figure 10.2, we observed that with respect to our proposal SELF provides a limited amount of correct samples for farther points in the disparity map. All these facts might explain the overall best performance of our strategy and why, in some circumstances, it allows us to achieve more accurate results with respect to deploy ground truth labels for training confidence measures as will be detailed in the next section.

10.3.2 Quantitative evaluation and analysis of training data

In this section we exhaustively compare our proposal with SELF [160] on three datasets KITTI 12, KITTI 15 and Middlebury 2014 and three algorithms for training the three state-of-the-art confidence measures O1 [178], CCNN [179] and PBCP [217] trained on labels inferred from eight sequences belonging to KITTI 12.

Moreover, we compare the performance of the same confidence measures trained on labels extracted from the corresponding eight stereo pairs with ground truth data available in KITTI 12. Detailed experimental results are reported in Table 10.2. We include in our evaluation APKR [114], the most effective confidence measure within the pool of confidence measures deployed for selecting labels as described in Section 10.2.1. Being

KITTI 12	CENSUS (ϵ=38.6%)			MC-CNN (ϵ=16.9%)			SGM (ϵ=9.1%)		
measure	GT	[160]	Prop.	GT	[160]	Prop.	GT	[160]	Prop.
O1 [178]	0.116	0.165	0.163	0.025	0.046	0.042	0.016	0.031	0.022
CCNN [179]	0.118	0.250	0.128	0.028	0.089	0.029	0.032	0.084	0.023
PBCP [217]	0.125	0.201	0.138	0.029	0.044	0.040	0.029	0.037	0.035
APKR [114]		0.166			0.048			0.030	
opt.		0.094			0.017			0.005	

KITTI 15	CENSUS (ϵ=35.4%)			MC-CNN (ϵ=15.4%)			SGM (ϵ=13.7%)		
measure	GT	[160]	Prop.	GT	[160]	Prop.	GT	[160]	Prop.
O1 [178]	0.109	0.172	0.147	0.031	0.059	0.046	0.021	0.038	0.027
CCNN [179]	0.113	0.266	0.120	0.036	0.102	0.035	0.044	0.072	0.029
PBCP [217]	0.122	0.209	0.151	0.035	0.053	0.047	0.031	0.035	0.037
APKR [114]		0.147			0.049			0.036	
opt.		0.083			0.019			0.007	

Middlebury 2014	CENSUS(ϵ=37.8%)			MC-CNN (ϵ=26.7%)			SGM (ϵ=26.9%)		
measure	GT	[160]	Prop.	GT	[160]	Prop.	GT	[160]	Prop.
O1 [178]	0.126	0.180	0.154	0.073	0.125	0.097	0.085	0.133	0.102
CCNN [179]	0.128	0.254	0.123	0.072	0.179	0.069	0.122	0.216	0.088
PBCP [217]	0.119	0.169	0.123	0.067	0.084	0.078	0.145	0.148	0.148
APKR [114]		0.137			0.074			0.100	
opt.		0.090			0.046			0.045	

Table 10.2: Average AUCs on the 3 datasets (from top to bottom: KITTI 12, KITTI 15 and Middlebury 2014). Evaluation of the 3 confidence measures with 3 algorithms (CENSUS, MC-CNN, SGM), trained on ground truth data (GT), on labels obtained by SELF [160] and by our proposal. We also include in the table a single AUC for each algorithm concerned with APKR not affected at all by training labels. We also report the average error ϵ on each dataset computed with error bound set to 3, for KITTI datasets, and set to 1 for Middlebury 2014.

such method independent of the training labels we report in the table a single AUC for APKR. On KITTI 12, our proposal always enables more effective training of confidence measures with respect to SELF. In particular, with CCNN and in most cases with PBCP, our method performs much better. Confidence measures trained with our method are more reliable than APKR in 8 out of 9 times while SELF yields better results only in 3 out of 9 times. Compared to training confidence measures on GT labels, SELF is always less reliable while our proposal with SGM and CCNN yields significantly better results. It is worth to note that, although the accuracy of our labels is higher compared to SELF, the amount of samples provided by our method for training is much lower.

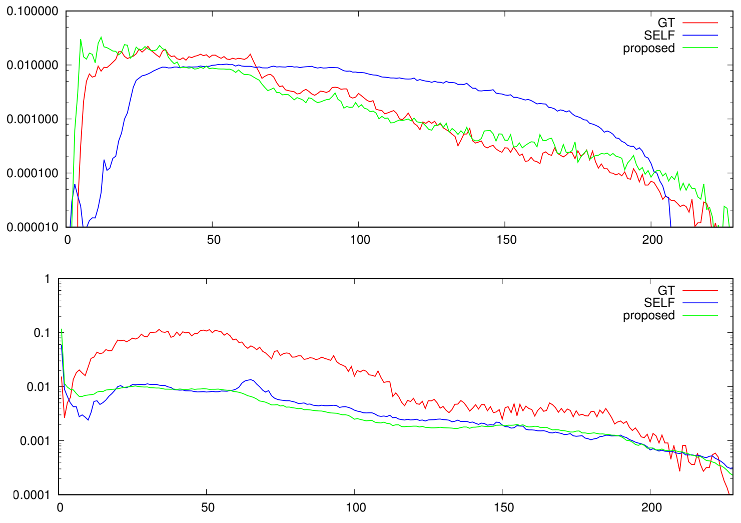


Figure 10.2: Distribution of training labels with SGM for SELF [160] in blue and the proposed method in green. In red the distribution of GT labels, independent of the stereo algorithm. (Top) Distribution of *correct* and (bottom) *wrong* labels within the disparity range.

The cross validation on KITTI 15 shows that our method is always more effective than SELF. Similarly to the results reported for KITTI 12, the validation on KITTI 15 highlights that CCNN has better performance when trained with our labels with respect to train on SELF. This trend is also confirmed with PBCP in many cases. APKR achieves better AUCs compared to our method in 2 out of 9 cases while SELF in 8 out of 9 cases. Compared to training on GT labels, our proposal enables to achieve better results in two cases (with CCNN) while SELF never yields better confidence estimation. The cross validation on Middlebury 2014 highlights, once more, that our self-labeling approach outperforms SELF excluding the test with CCNN trained on labels generated with SGM where the two methods have equivalent performance very similar to the AUC obtained training on GT labels. Compared to APKR, our method is better in 4 out of 9 situations (with any stereo algorithm training CCNN and, with CENSUS, training PBCP) while SELF is always outperformed by this method. Moreover, we point out that CCNN trained with our proposal yields always to more accurate results with respect to training on GT labels while this fact never holds for SELF. The experimental results reported in Table

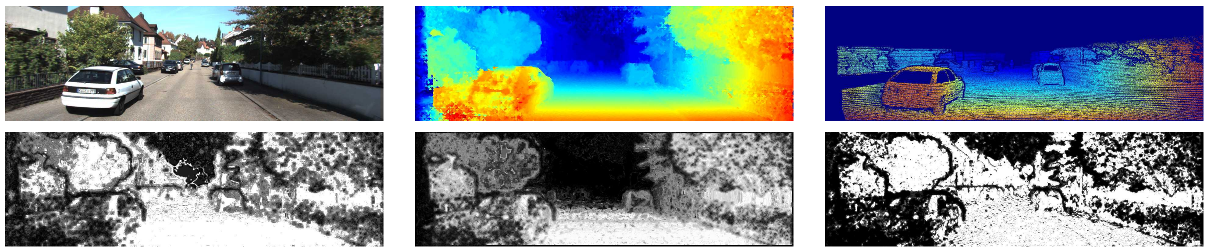


Figure 10.3: Qualitative results with the SGM algorithm on frame 000006 belonging to KITTI 12. At the top, from left to right, reference image, disparity computed by SGM and GT. At the bottom, we report for O1 [178] confidence maps obtained, from left to right, training on GT data, SELF [160] and the proposed method.

10.2 confirm that our proposal enables more effective training of confidence measures with respect to SELF as well as to a better generalization to new data. Moreover, training on labels generated by our method allows us, in most cases, to obtain confidence measures (in particular with those based on CNNs, CCNN and PBCP) with performance comparable, and sometimes even better, than training the same measures on ground truth labels. In Figure 10.2 we compare the distribution of *correct* and *wrong* training labels obtained by SELF and our proposal with KITTI 12. We also report the distribution of GT data. Observing the figures we can observe that our method generates training labels more similar to GT data. Moreover, we can notice how SELF provides very few positive labels for higher and lower disparity values especially dealing with correct labels. Figure 10.3 shows qualitative results for O1 confidence measure and SGM algorithm, obtained by training the measure on data from GT, SELF and our method. Finally, excluding disparity and confidence computation, on a i7 CPU, with our method we automatically extracted the training samples from 163 images of KITTI 12 in 76 seconds.

10.4 Conclusions

In this chapter we have proposed a novel self-supervised strategy to train confidence measures based on machine-learning. Compared to state-of-the-art methods our proposal is more general and neither constrained to image sequences nor to scene content. It generates training labels by leveraging on a pool of appropriately combined conventional confidence measures. The experimental results reported confirm that our strategy improves state-of-the-art by selecting more accurate labels thus enabling better confidence estimation when training confidence measures based on machine-learning on self-generated data. Moreover, in particular with CNN-based confidence measures, it also provides competitive results with respect to ground truth. This fact confirms our method can be deployed to train confidence measures from unlabeled stereo pairs, a circumstance frequently occurring in practical applications. Future work is aimed at further improving the proposed labeling selection strategy.

Chapter 11

Self-adapting confidence estimation for stereo

The content of this chapter has been presented at the European Conference on Computer Vision (ECCV 2020) - "Self-adapting confidence estimation for stereo"[191].

11.1 Introduction

In this chapter, inspired by recent works performing continuous learning [29, 238] for depth estimation, we propose the first-ever solution for self-adaptation of a confidence measure unconstrained to the target stereo system. For this purpose, we deploy a novel loss function built upon cues available from the input stereo pair and the output disparity only, needing no additional information to learn/adapt to the sensed environment. Our solution is comparable, and often better, w.r.t known strategies requiring full access to the cost volume [239] or static scenes for training [160].

Extensive experimental results on KITTI, Middlebury 2014, ETH3D and Driving-Stereo datasets support the following main claims of our novel confidence estimation paradigm: 1) competitive (often, better) with state-of-the-art when trained in a conventional, offline manner and tested on KITTI; 2) superior generalization capability on other datasets (e.g., Middlebury and ETH3D) compared to known self-supervised methods; 3) capable of online adaptation, outperforming competitors in unseen environments (e.g., DrivingStereo).

11.2 Learning a confidence measure out-of-the-box

This work aims at proposing a self-supervised paradigm suited for learning a confidence measure, unconstrained from the specific stereo method deployed and capable of self-adaptation. We first classify stereo systems into different categories according to the data they make available, and then we introduce a novel strategy compatible with all of them.

11.2.1 Taxonomy of stereo matching systems

In this section, we define three main broad categories of stereo matching solutions, each one characterized by different data made available during deployment. From now on, we will refer to a generic rectified stereo pair as ($\mathcal{I}_L$,$\mathcal{I}_R$), respectively made of left and right images, and to a generic stereo algorithm or deep network as $\mathcal{S}$. In the remainder, to simplify notation, we omit (x, y) coordinates if not strictly necessary.

Black-box models. Given any stereo algorithm processing a stereo pair ($\mathcal{I}_L$,$\mathcal{I}_R$), we define the output disparity map, computed assuming $\mathcal{I}_L$ as the **reference** image, as $\mathcal{D}_L = \mathcal{S}(\mathcal{I}_L, \mathcal{I}_R)$. This image triplet is the minimum amount of data available out of any stereo method, and we define as **black-box** all the systems making available only these cues. Such systems are highly representative of off-the-shelf stereo cameras (e.g., Stereolabs ZED 2) or stereo methods implemented in consumer devices (e.g., Apple iPhones). They neither allow end-users to access the implementation nor provide explicit ways (APIs) to call for it. For each ($\mathcal{I}_L$,$\mathcal{I}_R$) acquired in the field by the device, they provide the corresponding disparity map typically with undisclosed approaches based either on conventional stereo algorithms or deep networks. Hence, learning confidence measures for these systems is particularly challenging, yet appealing.

Gray-box models. Although black-box systems provide cues available in any stereo system, when explicit calls to the algorithm APIs are exposed, additional cues can be retrieved. Hence, we define a second family of systems for which, although it is given no access to the algorithm implementation or its intermediate data, explicit calls to the method itself are possible (e.g. stereo algorithms provided by pre-compiled libraries). Most deep stereo networks prevent the deployment of their internal representation since too abstract and substantially unintelligible, e.g. 2D architectures [99, 129, 151, 225, 262, 271]. We define systems belonging to this class as **gray-box**, since multiple calls to $\mathcal{S}$ allow for retrieving additional cues. For instance, it is straightforward to compute the Left to Right Consistency (LRC) of the disparity maps, a popular strategy to obtain a confidence estimator, even if not explicitly provided by $\mathcal{S}$ itself in its original implementation. Given the possibility to call $\mathcal{S}$ two times, consistency checking can be performed analyzing $\mathcal{D}_L$ and a second disparity map, namely $\mathcal{D}_R$ obtained by assuming $\mathcal{I}_R$ as the reference images. Defining $\leftarrow$ the horizontal flipping operator, $\mathcal{D}_R$ is obtained as follows:

$$\mathcal{D}_R = \overleftarrow{\mathcal{S}(\overleftarrow{\mathcal{I}_R}, \overleftarrow{\mathcal{I}_L})} \tag{11.1}$$

Once obtained $\mathcal{D}_R$, the consistency between the two can be checked as

$$\text{LRC} = |\mathcal{D}_L - \pi(\mathcal{D}_L, \mathcal{D}_R)| < \delta \tag{11.2}$$

with $\pi(a, b)$ a sampling operator, collecting values at coordinate a from b, and δ a threshold value (usually 1) above which $\mathcal{D}_L$ and $\mathcal{D}_R$ are considered inconsistent. Although less effective than other measures [89], it comes at a lower price.

White-box models. Finally, if the implementation of $\mathcal{S}$ is accessible, additional cues can be sourced by processing intermediate data structures, if meaningful. The preferred one is the cost volume $\mathcal{V}$, containing matching costs. This class of systems, referred to as **white-boxes**, enables computation of any confidence measure, either conventional [89] or learning-based [73, 117, 184]. Popular traditional confidence measures obtained from $\mathcal{V}$ are the Peak-Ratio (PKR) and Left-Right Difference (LRD).

Motivations and challenges. Indeed, for the reasons outlined so far, black-box models represent the most challenging, yet general and appealing target when dealing with confidence estimation since their constraints prevent the deployment of most state-of-the-art measures [73, 117], as well as self-supervised strategy existing in the literature [160, 239]. Hence, first and foremost, we aim at devising a general-purpose strategy enabling self-supervised confidence estimation in such constrained settings. As a notable consequence, this fact paves the way to tackle the same task even for state-of-the-art CNNs. Finally, having achieved this goal, out-of-the-box learning of confidence estimation with any stereo setup and self-adaptation in any environment is at hand.

11.2.2 Self-supervision cues for black-box models

In order to develop a self-supervised strategy suited for any stereo system, it is crucial to identify cues that are effective to source a robust supervision signal. According to the previous discussion, in the case of black-box models, we can rely on $(\mathcal{I}_L, \mathcal{I}_R)$ and $\mathcal{D}_L$ only. In this circumstance, although relevant information is not available compared to other models, we introduce three terms to obtain the desired self-supervised signal from the meagre cues available.

Image reprojection error. In recent literature, several works proved how the reprojection across the two viewpoints available in a rectified stereo pair could be a powerful source of supervision, either for monocular [69, 70, 186] or stereo [238, 285] depth estimation. Specifically, we can reproject $\mathcal{I}_R$ on the reference image coordinates as $\tilde{\mathcal{I}_R} = \pi(\mathcal{D}_L, \mathcal{I}_R)$ Then, the difference between $\mathcal{I}_L$ and warped right view $\tilde{\mathcal{I}_R}$ appearance encodes how correct the reprojection is. To this aim, the most popular choice is a weighted sum between two terms, respectively SSIM [254] and absolute difference.

$$\Delta_{(\mathcal{I}_L, \tilde{\mathcal{I}_R})} = \alpha \cdot (1 - \text{SSIM}(\mathcal{I}_L, \tilde{\mathcal{I}_R})) + (1 - \alpha)|\mathcal{I}_L - \tilde{\mathcal{I}_R}| \qquad (11.3)$$

with α usually tuned to 0.85 [69]. The higher it is, the more likely $\mathcal{D}_L$ is wrong. By definition, matching pixels is particularly challenging in ambiguous regions, such as textureless portions of the image. To this aim, we first aim at detecting regions with rich texture, being more likely to be correctly estimated by $\mathcal{S}$, by comparing Δ computed between $(\mathcal{I}_L, \mathcal{I}_R)$ with the one after reprojection as $\mathcal{T} = \Delta_{(\mathcal{I}_L,\mathcal{I}_R)} > \Delta_{(\mathcal{I}_L,\tilde{\mathcal{I}_R})}$. In large ambiguous regions, $\Delta_{(\mathcal{I}_L,\mathcal{I}_R)}$ will result equal (or even minor) than the reprojection error [70], thus identifying pixels on which stereo is prone to errors.

Agreement among neighboring matches. Since most regions of a disparity map should be smooth, variations in nearby pixels should be small except at depth boundaries. As highlighted in [178, 194], $\mathcal{D}_L$ itself allows for the extraction of meaningful cues to assess the quality of disparity assignments. Purposely, we rely on the **disparity agreement** between neighbouring pixels, defined as

$$\text{DA} = \frac{\mathcal{H}_{N\times N}(d_1)}{N \times N} \tag{11.4}$$

$\mathcal{H}_{N\times N}$ is a histogram encoding, for each pixel (x, y), the number of neighbours in a $N \times N$ window having the same disparity d (in case of subpixel precision, within 1 pixel). In the absence of depth discontinuities, the majority of pixels in the neighbourhood should share the same, or very similar, disparity hypothesis. Hence, we define a second criterion to identify reliable stereo correspondences as $\mathcal{A} = \text{DA} > 0.5$, assuming that more than half of the pixels in the neighbourhood share the same disparity. It is worth noting that this criterion is often not met in the presence of depth boundaries, even in case of correct disparities.

Uniqueness constraint. In an ideal frontal-parallel scene observed by a stereo camera in standard form, for each pixel in $\mathcal{I}_L$ exists at most one match in $\mathcal{I}_R$ and vice-versa. Leveraging this property, known as uniqueness, is particularly useful [54] to detect outliers in occluded regions and represents a reliable alternative to LRC and LRD measures, not

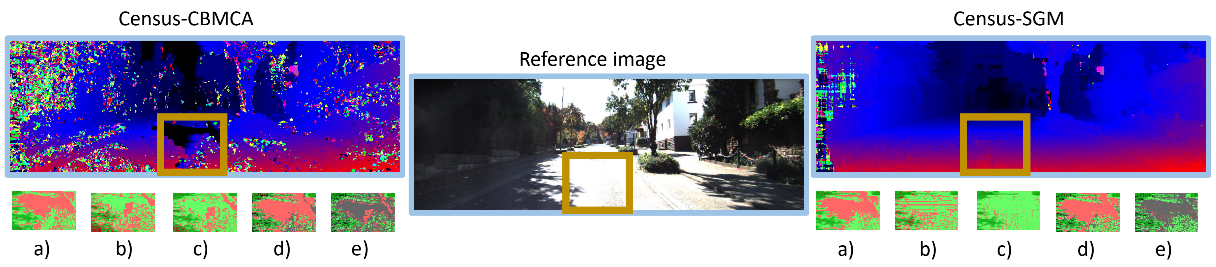


Figure 11.1: Effects of different criteria. Given the highlighted region, we show inliers (green) and outliers (red) guesses by using the following cues in multi-modal binary cross-entropy: a) $\mathcal{T}^p, \mathcal{T}^q$ b) $\mathcal{A}^p, \mathcal{A}^q$ c) $\mathcal{U}^p, \mathcal{U}^q$ d) $\mathcal{T}^p, \mathcal{A}^p, \mathcal{U}^p, \mathcal{T}^q$ e) $\mathcal{T}^p, \mathcal{A}^p, \mathcal{U}^p, \mathcal{T}^q, \mathcal{A}^q, \mathcal{U}^q$. For black pixels, the considered configuration gives no guesses.

usable when dealing with black-box models. Uniqueness Constraint (UC) is encoded as

$$\text{UC} = [x - \mathcal{D}_L(x,y)] \notin \bigcup_k [(x+k) - \mathcal{D}_L(x+k,y)] \tag{11.5}$$

with $k \in [-d^*_{max}, -1] \cup [1, d^*_{max}]$ and $d^*_{max} = d_{max} - \mathcal{D}_L(x,y)$. In other words, the uniqueness for any pixel in $\mathcal{I}_L$ holds if it does not collide in the target image with any other pixel, i.e., not matching the same pixel in $\mathcal{I}_R$ matched by any other. We exploit this property to define our third criterion as $\mathcal{U}$ =UC. We conclude observing that, although effective at detecting mostly occlusions, the uniqueness constraint is often violated in the presence of slanted surfaces.

11.2.3 Multi-modal Binary Cross Entropy

Given the three criteria outlined above, we revise the traditional binary cross entropy loss to take into account multiple label hypotheses. We refer to this variant as **Multi-modal Binary Cross Entropy** (MBCE), defined as

$$\mathcal{L}_{\text{MBCE}} = -\left[\left(\prod_{p\in\mathcal{P}} p\right)\cdot log(o) + \left(\prod_{q\in\mathcal{Q}} q\right)\cdot log(1-o)\right] \tag{11.6}$$

with o the output of the neural network $\in [0,1]$, i.e. passed through a sigmoid activation, $\mathcal{P}$ and $\mathcal{Q}$ two sets of **proxy labels** derived respectively by a criterion being met or not. For instance, pixels satisfying the first criterion on image reprojection will have labels $\mathcal{T}^p = 1$, $\mathcal{T}^q = 0$ and vice versa when they do not. Unlike traditional binary cross entropy, where a single label y and its counterpart $(1-y)$ are used, we define disjoint sets of proxies allowing for a flexible configuration of the loss function according to the three criteria described so far. For instance, by setting $\mathcal{P} = [\mathcal{T}^p, \mathcal{A}^p]$ and $\mathcal{Q} = [\mathcal{T}^q]$ we will train the network to detect good matches using image reprojection plus agreement and outliers using the former only. Adding elements to the sets $\mathcal{P}$ and $\mathcal{Q}$ reduces progressively the number of pixels considered correct or wrong, respectively. Fig. 11.1 shows this, highlighting how combining multiple guesses as in d) and e) for some pixels no supervision is given when criteria do not match. We will report the impact of this and the different configurations in a thorough ablation study.

11.3 Experimental results

In this section, we report an exhaustive evaluation to assess the effectiveness of our strategy, referred to as *Out-of-The-Box* (OTB), by conducting three main experiments, respectively: 1) ablation study on the MBCE loss, 2) comparison with self-supervised approaches [160, 239] in a conventional offline training and 3) an evaluation concerning online adaptation of OTB.

11.3.1 Implementation details

We now report all the details to understand and reproduce our experiments fully.

Evaluation Protocol. To measure the effectiveness of the learned confidence measures, we compute the Area Under Curve (AUC) of the sparsification plots [89, 117, 184, 240].

Confidence networks. Since the goal of this work is to define an effective self-supervised strategy suited for online learning rather than proposing a novel architecture, in our experiments, we test our proposal to train existing networks. Purposely, we consider three architectures: CCNN [179], ConfNet and LGC [240] to carry out our experiments because 1) they process only disparity map and reference image, thus are suited to all methods from white-box to black-box, 2) according to recent works [73, 117], the most accurate one (LGC) is on par with state-of-the-art networks processing the cost volume and 3) the source code is fully available, conversely to [73, 117]. Moreover, in ConfNet we replaced deconvolutions with bilinear upsampling followed by 3×3 convolutions and process $\mathcal{D}_L$ only, significantly improving its performance and thus filling most of the gap with CCNN and LGC. We defined a training schedule for each network, kept constant in all experiments. For CCNN, we use batches of 128 patches for 1M iterations, for ConfNet batches of single, 320×1216 crops for 25K iterations, finally for LGC batches of 128 patches for 300K iterations, starting from pre-trained CCNN and ConfNet models. We trained all networks with SGD optimizer and a constant learning rate of 0.001. For patch-based methods, proxy signals are computed offline on the full resolution image.

Datasets. We consider five standard datasets: KITTI 2012 [66], KITTI 2015 [155], Middlebury 2014[1] [213], ETH3D [215] and DrivingStereo [264], setting τ respectively to 3, 3, 1, 1 and 3. Being ground truth required to assess performance, we refer to the training set of such datasets. To train confidence estimation networks, we select the first 20 images from KITTI 2012 as in [184, 240] for supervised training and the 400 images from the first 20 sequences of the KITTI 2012 multiview extension used in [160, 239] for self-supervised ones. To evaluate the trained confidence networks, we use the remaining 174 images from KITTI 2012 as the validation set and the totality of images available from KITTI 2015 for experiments on environments similar to the training set. Moreover, we also assess their generalization performance on the whole Middlebury 2014 and ETH3D datasets. In these experiments, only the KITTI 2012 images listed above are used for training, thus the networks are transferred without any fine-tuning. Finally, to test self-adaptation peculiar of OTB we use a sequence from the DrivingStereo dataset, namely 2018-10-25-07-37, made of about 7K frames.

[1]We use the quarter resolution split as in previous works [117, 184, 240]

Stereo algorithms. Following the recent literature [117, 184, 240], we evaluate the effectiveness of our strategy on a variety of stereo algorithms with different degrees of accuracy, in order to highlight how strong is our self-supervised paradigm in the presence of heterogenous disparity maps. We consider four main stereo algorithms deploying the code provided Zbontar and LeCun [279] under different settings. Specifically: Census-CBCA, Census-SGM, MCCNN-fst-CBCA and MCCNN-fst-SGM. The first two rely on a census-based matching cost computation, respectively, optimized by a Cross Based Cost Aggregation (CBCA) strategy [283] and SGM [82]. The latter two replace the census-based matching costs with predictions obtained by MCCNN-fst, for which we use pre-trained weights on KITTI 2012, 2015 and Middlebury provided by the authors and tested on the same datasets. For ETH3D, Middlebury weights have been used. No post-processing is applied to any output. Furthermore, to evaluate the impact of self-adaptation made possible by OTB with a real black-box method, we also consider two recent deep stereo network. We choose MADNet [238] and GANet [282], both trained on synthetic images [151] and then fine-tuned with ground truth on KITTI 2015, because of the availability of trained model and its accuracy-speed trade-off. Since fine-tuned on KITTI, we conduct experiments with MADNet and GANet on DrivingStereo only.

Competitors. We compare the proposed OTB strategy with existing methods proposed by Mostegel et al. [160] (named SELF) and by Tosi et al. [239] (named WILD). The former reasons about contradictions on observations from multiple viewpoints: given a stereo sequence framing a static scene with a moving camera, $\mathcal{D}_L$ and $\mathcal{D}_R$ are computed for each pair, registered and checked for inconsistencies. Since it requires both $\mathcal{D}_L$ and $\mathcal{D}_R$ disparity maps, SELF is suited only for systems belonging to gray-box and white-box categories. Concerning WILD, it requires a pool of six confidence measures extracted from the cost volume to identify inliers and outliers according to heuristic thresholding on the measures. Since it requires access to the cost volume, WILD is suited for white-box algorithms only. In contrast, among other advantages discussed next, it worth stressing that our OTB approach is suited for black-box systems and agnostic to the scene content, KITTI 2012

Match cost						Census		MCCNN-fst		Census		MCCNN-fst		Census		MCCNN-fst	
Aggregation						CBCA	SGM	CBCA	SGM	CBCA	SGM	CBCA	SGM	CBCA	SGM	CBCA	SGM
$\Delta_{\mathcal{I}}(\mathcal{I}_L, \tilde{\mathcal{I}_R})$						0.210	0.086	0.165	0.044	0.210	0.086	0.165	0.044	0.210	0.086	0.165	0.044
DA						0.112	0.047	0.063	0.023	0.112	0.047	0.063	0.023	0.112	0.047	0.063	0.023
UC						0.165	0.063	0.123	0.034	0.165	0.063	0.123	0.034	0.165	0.063	0.123	0.034
$\mathcal{T}^p$	$\mathcal{A}^p$	$\mathcal{U}^p$	$\mathcal{T}^q$	$\mathcal{A}^q$	$\mathcal{U}^q$	CCNN				ConfNet				LGC			
✓			✓			0.080	0.045	0.047	0.018	0.077	0.033	0.045	0.014	0.082	0.058	0.046	0.026
	✓			✓		0.105	0.045	0.073	0.023	0.087	0.035	0.049	0.017	0.110	0.040	0.074	0.022
		✓			✓	0.111	0.035	0.087	0.022	0.101	0.038	0.065	0.020	0.114	0.035	0.077	0.020
✓	✓		✓			0.078	0.033	0.050	0.019	0.072	0.030	0.038	0.014	0.075	0.034	0.049	0.023
✓	✓		✓	✓		0.089	0.035	0.059	0.023	0.071	0.029	0.038	0.014	0.082	0.031	0.066	0.020
✓		✓	✓			0.072	0.038	0.053	0.019	0.074	0.029	0.040	0.013	0.070	0.036	0.042	0.016
✓		✓	✓		✓	0.088	0.032	0.075	0.020	0.076	0.030	0.041	0.013	0.084	0.031	0.071	0.017
✓	✓	✓	✓			**0.068**	0.034	**0.046**	0.018	**0.070**	0.029	**0.037**	0.013	**0.068**	0.032	**0.041**	0.016
✓	✓	✓	✓	✓	✓	0.085	**0.029**	0.057	**0.017**	0.071	**0.028**	0.038	**0.012**	0.081	**0.028**	0.050	**0.015**

Middlebury

Match cost						Census		MCCNN-fst		Census		MCCNN-fst		Census		MCCNN-fst	
Aggregation						CBCA	SGM	CBCA	SGM	CBCA	SGM	CBCA	SGM	CBCA	SGM	CBCA	SGM
$\Delta_I(\mathcal{I}_L, \mathcal{I}_R)$						0.190	0.180	0.179	0.134	0.190	0.180	0.179	0.134	0.190	0.180	0.179	0.134
DA						0.161	0.168	0.099	0.087	0.161	0.168	0.099	0.087	0.161	0.168	0.099	0.087
$\mathcal{U}$						0.193	0.188	0.192	0.145	0.193	0.188	0.192	0.145	0.193	0.188	0.192	0.145
$\mathcal{T}^p$	$\mathcal{A}^p$	$\mathcal{U}^p$	$\mathcal{T}^q$	$\mathcal{A}^q$	$\mathcal{U}^q$	CCNN				ConfNet				LGC			
✓	✓	✓	✓			**0.116**	**0.123**	**0.087**	**0.077**	**0.133**	**0.112**	**0.087**	**0.067**	**0.127**	**0.111**	**0.090**	**0.064**
✓	✓	✓	✓	✓	✓	0.153	0.146	0.095	0.081	0.134	0.122	0.095	0.069	0.138	0.142	0.099	0.080

Table 11.1: Ablation study on the proposed multi-modal binary cross entropy. We report AUC scores for networks trained on KITTI 2012 (20 or 400 images) and tested on KITTI 2012 (174 images, top) and Middlebury (15 images, bottom).

in contrast to SELF that requires static scenes.

11.3.2 Ablation study

At first, we study the impact of the different terms in the proposed self-supervised loss function. To this aim, on KITTI 2012 and as for other experiments, we train 9 variants of each network for each of the four stereo algorithms. Then, we evaluate confidences on the KITTI 2012 dataset and, without retraining, on Middlebury 2014. Table 19.1 collects the outcome of this evaluation, reporting on top results on KITTI 2012 and, at the bottom, on Middlebury. We report as baselines the performance of $\Delta_{(\mathcal{I}_L, \tilde{\mathcal{I}_R})}$, DA and UC. DA is computed on 5×5 windows.

On KITTI (top of the table), we first report the results achieved by training the three networks selecting only one of the three cues used to distinguish between correct

	Match cost	Census				MCCNN-fst			
	Aggregation	CBCA		SGM		CBCA		SGM	
	KITTI split	2012	2015	2012	2015	2012	2015	2012	2015
	Badτ %	27.193	22.281	10.330	8.998	18.875	16.926	6.084	6.028
Traditional	LRD	0.096	0.080	0.033	0.032	0.080	0.077	0.017	0.023
	PKR	0.106	0.089	0.028	0.029	0.065	0.062	0.010	0.017
	LRC	0.142	0.113	0.062	0.056	0.103	0.092	0.036	0.041
	$\Delta_{(\mathcal{I}_L,\tilde{\mathcal{I}}_R)}$	0.210	0.175	0.086	0.079	0.165	0.150	0.044	0.041
	DA	0.112	0.090	0.047	0.046	0.063	0.059	0.023	0.028
	UC	0.165	0.131	0.063	0.058	0.123	0.111	0.034	0.037
CCNN	Supervised	0.059	0.046	0.018	0.017	0.031	0.032	0.009	0.012
	WILD [239]	0.076	0.065	**0.026**	**0.026**	0.052	0.047	**0.012**	**0.017**
	SELF [160]	0.076	0.065	0.047	0.046	**0.038**	**0.041**	**0.012**	0.018
	OTB (Ours)	**0.068**	**0.055**	0.029	0.031	0.046	0.048	0.017	0.022
ConfNet	Supervised	0.061	0.049	0.017	0.016	0.033	0.034	0.006	0.010
	WILD [239]	0.089	0.067	**0.024**	**0.020**	0.054	0.050	**0.010**	**0.016**
	SELF [160]	0.075	0.066	**0.024**	0.024	0.041	0.044	0.014	**0.016**
	OTB (Ours)	**0.070**	**0.058**	0.028	0.032	**0.037**	**0.040**	0.012	0.017
LGC	Supervised	0.056	0.044	0.016	0.016	0.029	0.030	0.007	0.010
	WILD [239]	0.089	0.065	**0.026**	**0.025**	0.049	0.045	**0.011**	**0.017**
	SELF [160]	0.089	0.081	**0.026**	0.026	0.056	0.057	0.020	0.021
	OTB (Ours)	**0.068**	**0.055**	0.028	0.032	**0.041**	**0.044**	0.015	0.019
	Optimal	0.047	0.034	0.008	0.008	0.024	0.022	0.003	0.005

Table 11.2: Evaluation on KITTI. We report AUC scores for networks trained on KITTI 2012 (20 or 400 images) and tested on 2012 (174 images) and 2015 (200 images).

and wrong matches, i.e. $[\mathcal{T}^p, \mathcal{T}^q]$, $[\mathcal{A}^p, \mathcal{A}^q]$ and $[\mathcal{U}^p, \mathcal{U}^q]$ configurations. We can notice that each of them outperforms the performance of the corresponding baseline used for supervision. This trend occurs on all the algorithms and for each network, showing the surprisingly robust capacity of the networks to learn how to estimate confidence better than a noisy supervision signal used for training. In general, the models trained on $[\mathcal{T}^p, \mathcal{T}^q]$ outperforms the others, except rare cases (i.e. CCNN and LGC on Census-SGM, outperformed by $[\mathcal{U}^p, \mathcal{U}^q]$ setting). Although effective at detecting textureless and ambiguous regions, the reprojection fails at filtering outliers due to slanted surfaces and occlusions. Thus, we incrementally add a single criterion, i.e. $\mathcal{A}^p$ or $\mathcal{U}^p$ to filter out false positives obtained by $[\mathcal{T}^p, \mathcal{T}^q]$ configuration. We incrementally add, on another configuration, the corresponding negative criterion to remove pixels wrongly categorized as outliers by $\mathcal{T}^q$. In most cases, adding a single criterion to $\mathcal{P}$ is beneficial, while we can notice how introducing negative criteria degrades the performance on CBCA algorithms. This occurs because adding $\mathcal{A}^q$ or $\mathcal{U}^q$ makes textureless regions no longer labelled as

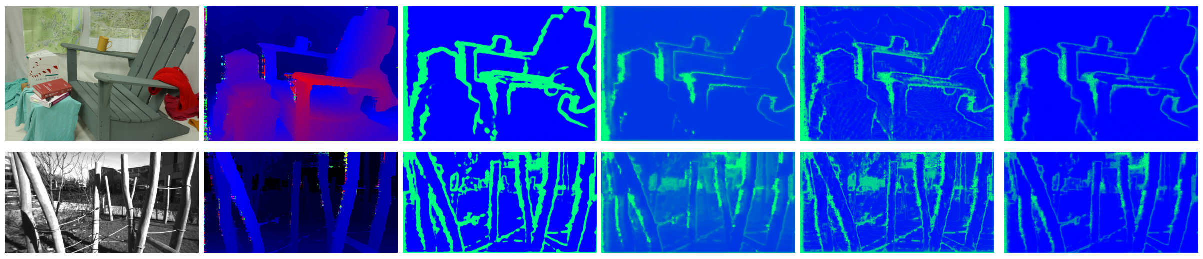


Figure 11.2: Qualitative results for generalization. From left: reference image, disparity by MCCNN-fst-SGM, ConfNet trained with [239], [160], our method and ground truth. On top, Adirondack (Middlebury), at the bottom, Playground_3l (ETH3D).

outliers, as shown in Fig. 11.1 left comparing patches d) and e). Finally, adding both $\mathcal{A}^p$ and $\mathcal{U}^p$ produces the best overall results for CBCA methods. By introducing $\mathcal{A}^q$ and $\mathcal{U}^q$ too we obtain better results only on SGM methods, since much more accurate than CBCA ones and thus more false outliers are introduced if $\mathcal{A}^q$ and $\mathcal{U}^q$ are not used, as shown in Fig. 11.1 right, comparing d) and e). On the other hand, by testing the best configurations on Middlebury 2014, enabling all the positive criteria and only $\mathcal{T}^q$ for negative allows for better generalization to unseen environments.

11.3.3 Comparison with offline methods

Having found the best configuration for the $\mathcal{L}_{\text{MBCE}}$ loss, we compare our supervision paradigm with known self-supervised approaches [160, 239]. In our experiments, we obtain proxy labels for SELF and WILD using the code provided by the respective authors. We collect the outcome of these experiments in Tables 17.2 and 11.3. We label with different colors methods ranging from **stronger** constraints (need for ground truth) to weaker (ours). For each architecture, stereo algorithm and evaluation set triplet we label in **bold** the best self-supervision approach, while in **red** the couple architecture/self-supervision on an entire evaluation set.

KITTI datasets. Table 17.2 collects evaluations on the KITTI 2012 and 2015 datasets, respectively, using the 174 validation set from 2012 and the full 2015 set. We point out that all self-supervised strategies outperform traditional measures, reported on

	Match cost	Census				MCCNN-fst			
	Aggregation	CBCA		SGM		CBCA		SGM	
	Dataset	Midd	ETH	Midd	ETH	Midd	ETH	Midd	ETH
	Bad1 %	28.701	21.270	26.682	15.471	29.799	34.279	21.799	12.594
Traditional	LRD	0.117	0.082	0.113	0.059	0.107	0.185	0.075	0.051
	PKR	0.124	0.086	0.112	0.056	0.095	0.181	0.059	0.042
	LRC	0.189	0.135	0.197	0.114	0.188	0.239	0.149	0.091
	$\Delta_{(\mathcal{I}_L,\tilde{\mathcal{I}}_R)}$	0.190	0.162	0.180	0.119	0.179	0.257	0.134	0.097
	DA	0.161	0.119	0.168	0.093	0.099	0.159	0.087	0.047
	UC	0.193	0.148	0.188	0.114	0.192	0.264	0.145	0.096
CCNN	Supervised	0.110	0.096	0.118	0.076	0.079	0.138	0.068	0.046
	WILD [239]	0.136	0.114	0.140	0.086	0.095	0.154	0.081	0.046
	SELF [160]	0.163	0.174	0.217	0.174	0.090	0.147	0.081	0.076
	OTB (Ours)	**0.116**	**0.084**	**0.123**	**0.070**	**0.087**	**0.137**	**0.077**	**0.042**
ConfNet	Supervised	0.121	0.086	0.104	0.063	0.086	0.138	0.062	0.036
	WILD [239]	**0.122**	0.101	0.117	**0.063**	0.091	0.160	0.073	0.037
	SELF [160]	0.154	0.120	0.121	0.067	0.096	0.172	0.084	0.048
	OTB (Ours)	0.133	**0.093**	**0.112**	0.067	**0.087**	**0.138**	**0.067**	**0.035**
LGC	Supervised	0.111	0.080	0.111	0.061	0.083	0.136	0.065	0.040
	WILD [239]	0.136	0.104	0.133	0.082	0.098	0.156	0.084	0.050
	SELF [160]	0.128	0.105	0.117	0.066	0.091	0.154	0.086	0.060
	OTB (Ours)	**0.127**	**0.084**	**0.111**	**0.056**	**0.090**	**0.139**	**0.064**	**0.035**
	Optimal	0.053	0.041	0.046	0.022	0.057	0.103	0.030	0.014

Table 11.3: Generalization on Middlebury and ETH3D. We report AUC scores for networks trained on KITTI 2012 (20 or 400 images) and tested on Middlebury (15 images) and ETH3D (27 images) without retraining or adaptation.

top as baselines, such as LRD, PKR, LRC and the cues used in our $\mathcal{L}_{\text{MBCE}}$ loss, struggling only when dealing with the very accurate MCCNN-fst-SGM algorithm. Comparing the different architectures, we can notice how the self-supervised paradigms break the hierarchy (i.e., self-supervised LGC is often outperformed by ConfNet). On Census-CBCA, our strategy always outperforms SELF and WILD when used to train any architecture. The same behaviour is confirmed on MCCNN-fst-CBCA, except for CCNN resulting better with SELF but with the best performance achieved by ConfNet trained with OTB. This outcome highlights the outstanding performance of OTB with noisy algorithms (about 27 and 19% error rates on the validation set), close to **full supervision**. On SGM algorithms, OTB results comparable with SELF and WILD, although sourcing supervision only from images and $\mathcal{D}_L$, thus in a much weaker form compared to the competitors. On three out of four algorithms, ConfNet results to be the most effective architecture when trained in a self-supervised manner.

	Traditional			Supervision						
Algorithm	$\Delta_{(\mathcal{I}_L,\tilde{\mathcal{I}_R})}$	DA	UC	Supervised	WILD [239]	SELF [160]	OTB	**OTB (online)**	Opt.	Badτ %
Census-SGM	0.179	0.106	0.162	0.061	0.067	0.074	0.072	**0.064**	0.029	21.007
MADNet [238]	0.133	0.147	0.155	0.116	-	0.135	0.139	**0.126**	0.021	16.226
GANet [282]	0.019	0.018	0.025	0.017	-	0.021	0.019	**0.015**	0.001	2.897

Table 11.4: Self-adaptation. We report AUC scores for networks trained on KITTI 2012 (20 or 400 images) and tested on a DrivingStereo sequence (6905 frames).

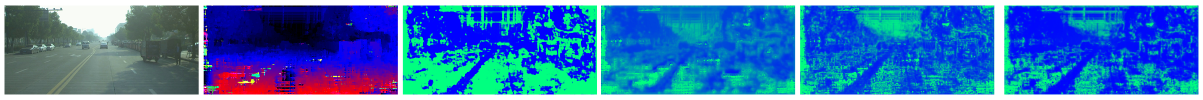


Figure 11.3: Qualitative results on DrivingStereo. From left: reference image, disparity by Census-SGM, ConfNet trained with [239], [160], OTB and online-adapted OTB.

Generalization on Middlebury and ETH3D. Table 11.3 reports results on the Middlebury 2014 and ETH3D datasets. The same networks evaluated so far (trained on KITTI 2012 images) are transferred here without retraining or adaptation, enabling to assess the generalization properties of each network/supervision configuration. Not surprisingly, the margin between learned and traditional measures is much smaller because of the domain shift. Nonetheless, in many cases the performance is still in favor of learned approaches, with some exceptions. We point out that networks trained with OTB self-supervision always outperform SELF and WILD, except for ConfNet in two cases. Moreover, networks trained with OTB generalize better than their fully supervised counterparts in some cases, mostly on the ETH3D dataset (e.g., CCNN with all algorithms, ConfNet with MCCNN-fst-SGM and LGC with both SGM methods). Fig. 11.2 shows qualitative examples of this test.

11.3.4 Self-adapting in-the-wild

Finally, we conduct experiments aimed at assessing how effective our strategy is for self-adaptation of a confidence measure in unseen environments. Purposely, we simulate deployment in an autonomous driving scenario, selecting a sequence from the DrivingStereo dataset [264]. We use sequence 2018-10-25-07-37, containing 6905 stereo pairs acquired in unconstrained (i.e., dynamic) environment. For this evaluation, we choose Census-SGM, MADNet and GANet. The former because it represents the preferred choice for hardware implementation on custom stereo cameras [20, 65, 87, 150, 198, 199, 214]. The remaining two because well representing modern end-to-end CNNs that are fast (MADNet) or yield state-of-the-art accuracy (GANet). For confidence networks we select ConfNet, because effective with accurate algorithms and well-suited for online adaptation.

In this experiment, we assume to have pre-trained versions of ConfNet with the different self-supervision paradigms, again on KITTI 2012. For OTB, we use $[\mathcal{T}^p, \mathcal{A}^p, \mathcal{U}^p, \mathcal{T}^q, \mathcal{A}^q, \mathcal{U}^q]$ for offline training and $[\mathcal{T}^p, \mathcal{A}^p, \mathcal{U}^p, \mathcal{T}^q]$ during adaptation. When performing online adaptation (**online** entry), for each stereo pair the confidence is estimated and evaluated **before** loss computation (thus, supervision only acts on the upcoming frames as in [238]). This way, ConfNet runs at 0.09 seconds (11 FPS) against the 0.02 (50 FPS) without adaptation on Titan Xp, measuring network computations only. The learning rate is set to 0.0001 during adaptation. Table 11.4 collects the outcome of this evaluation. We point out that WILD can not be deployed for MADNet and GANet since a meaningful cost volume is not available for the former or cannot be used straightforwardly for the latter. On the other hand, SELF would require $(\mathcal{D}_L, \mathcal{D}_R)$ for supervision, while MADNet and GANet provide only the former. By assuming networks as gray-boxes, we get rid of this issue at training time obtaining $\mathcal{D}_R$ as shown in Eq. 11.1. Concerning SGM, OTB performs in between WILD and SELF. Nevertheless, keeping continuous adaptation active on the whole sequence makes it outperform both. Concerning MADNet, SELF results more effective than OTB. Again, performing online adaptation makes OTB the best solution in this case as well. Finally, concerning GANet, learned measures perform worse than $\Delta_{(\mathcal{I}_L, \tilde{\mathcal{I}_R})}$ and DA. Online adaptation results crucial for OTB to obtain the best results. To conclude, Fig. 11.3 shows qualitative examples for the SGM algorithm.

On-the-fly learning with black-box sensors. Finally we report, as qualitative results, the outcome obtained by learning on-the-fly a confidence measure on disparity map sourced by an Apple iPhone XS, without any pre-training. Fig. 11.4 shows examples

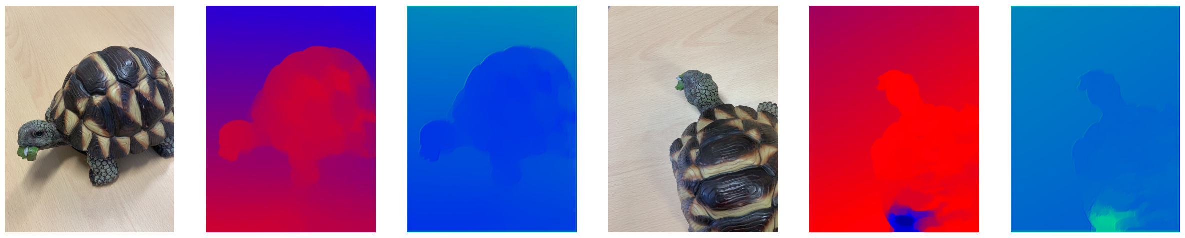


Figure 11.4: Qualitative results with Apple iPhone XS. We show two examples of reference image and disparity map acquired with the iPhone XS, followed by estimated confidence map after few iterations of on-the-fly learning.

of acquired disparity and estimated confidence maps by ConfNet adapted online, detecting gross errors like on turtle's shell.

11.4 Conclusion

In this chapter, we have introduced a novel self-supervised paradigm aimed at learning from scratch a confidence measure for stereo. We leverage few, principled cues from the input stereo pair and the estimated disparity in order to source supervision signals in place of disparity ground truth labels. Being such cues available during deployment in-the-wild, our solution is suited for continuous online adaptation on any black-box framework. Experimental results proved that our strategy is equivalent or superior to existing self-supervised approaches and, conversely to them, allows for further improvements during deployment by leveraging the online self-adaptation process.

Chapter 12

Leveraging confident points for accurate depth refinement on embedded systems

The content of this chapter has been presented at the the IEEE Conference on Computer Vision and Pattern Recognition Workshops (EVW 2019) - "Leveraging confident points for accurate depth refinement on embedded systems" [242].

12.1 Introduction

State-of-the-art stereo algorithms [31, 129] require expensive and power-hungry GPUs to run in a reasonable amount of time, making them unsuited for many practical applications constrained by hardware resources or energy consumption. Conventional (i.e.pre-deep learning) algorithms still achieve accurate results leveraging on multi-step pipelines, each one contributing to increasing the overall effectiveness with different degrees of reliability. A notable example is the Semi-Global Matching algorithm (SGM) [83], implemented in many variants thanks to its trade-off between accuracy and complexity, that usually deploy interpolation and refinement steps on estimated disparity maps. In their seminal work, Zbontar and LeCun [279] showed how plugging deep learning into a conventional stereo SGM pipeline yielded very accurate results on KITTI and Middlebury datasets not

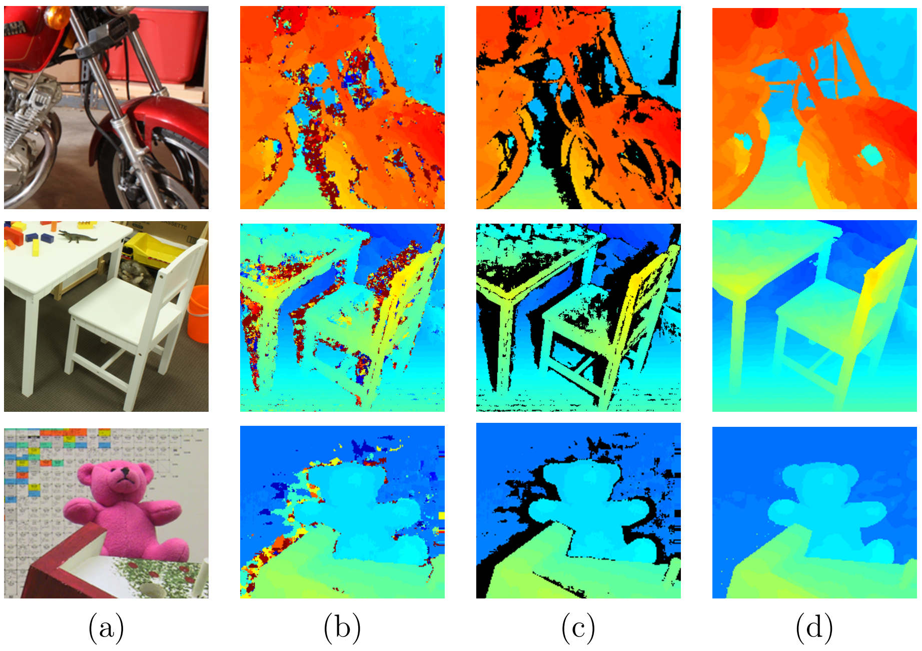
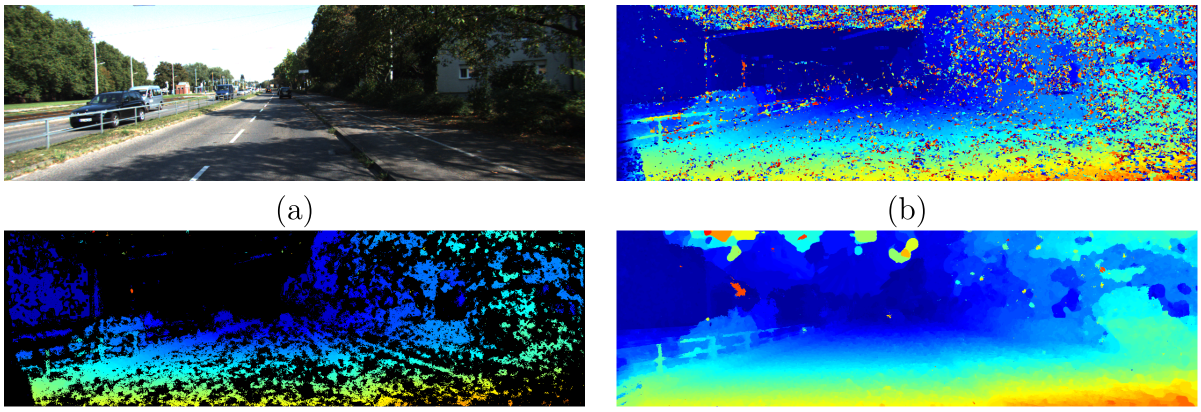


Figure 12.1: Non-Local Anchoring framework applied to three Middlebury 2014 stereo pairs. From top to bottom: *MotorcycleE*, *PianoL*, *Teddy*. (a) Detail of left image, (b) raw disparity map, (c) set of reliable pixels according to an ideal confidence measure, (d) refined disparity map.

far from end-to-end networks [31, 129].

One of the steps involved, referred to as *disparity refinement*, attempts to recover errors from the disparity map. While some refinement procedures rely on simple filters (e.g., median or bilateral filters) others exploit cues from the disparity map and the input stereo pair. Confidence measures allow to detect unreliable matches produced by stereo algorithms and, recently, strategies based on machine-learning achieved state-of-the-art results [184]. Confidence measures have been deployed in different steps of stereo pipelines, with the aim to further improve the overall accuracy. In this chapter, we propose *Non-Local Anchoring* (NLA), a novel disparity refinement method relying on confidence measures, outlined in Figure 12.1. Given a disparity map generated by any stereo algorithm, the confidence measure allows us to detect and removing erroneous pixels. Then, for each discarded pixel, among the remaining *reliable points* (RP) a subset of *anchors*, not necessarily in a close neighborhood of the examined pixel, is chosen to infer, according to both



(a) (b) (c) (d)

Figure 12.2: NLA in action on KITTI 2015 dataset [155]. (a) Left frame from stereo pair 000027, (b) raw disparity map computed by the Block-Matching algorithm (BM) algorithm, (c) sparse disparity map containing assumed reliable points RP, (d) refined disparity map with our proposal.

spatial and color information from the reference image, a new disparity value. Moreover, a CPU-friendly machine-learning framework based on a random forest classifier is proposed to deal with automatic identification of unreliable disparity assignments by analyzing local and global properties of the confidence on the whole image. This novel strategy allows us to remove the need for a heuristic selection of a confidence threshold often carried-out in this field [217, 228].

To assess the effectiveness of our proposal, we report an extensive evaluation on the Middlebury 2014 dataset comparing our framework to conventional disparity refinement methodologies as well as with recently proposed confidence-based approaches, acting on the Disparity Space Image (DSI) [211] also referred to as the *cost-volume*. Differently, NLA acts in the disparity domain hence does not require at all the cost volume that might be not available in some circumstances, e.g.when dealing with a commercial off-the-shelf (COTS) stereo camera. Factors like the number of anchors deployed and a further local aggregation strategy included in our proposal will be discussed and compared. Moreover, we evaluate our framework also on KITTI 2015 dataset [155] to further confirm the effectiveness of our method on indoor and outdoor data. Figure 12.2 shows the outcome of our proposal

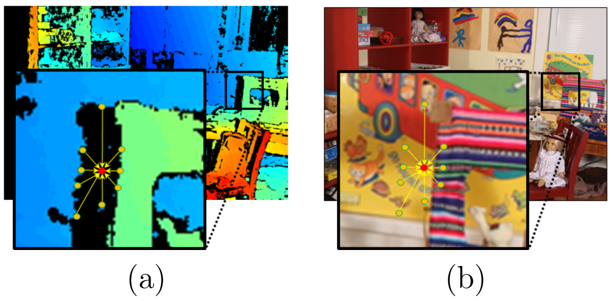


Figure 12.3: Overview of NLA on *Playtable* image from Middlebury 2014. (a) Disparity map containing reliable RP points only, (b) reference image. For each unreliable pixel (red), anchors (yellow) are selected as the closest RPs along different directions.

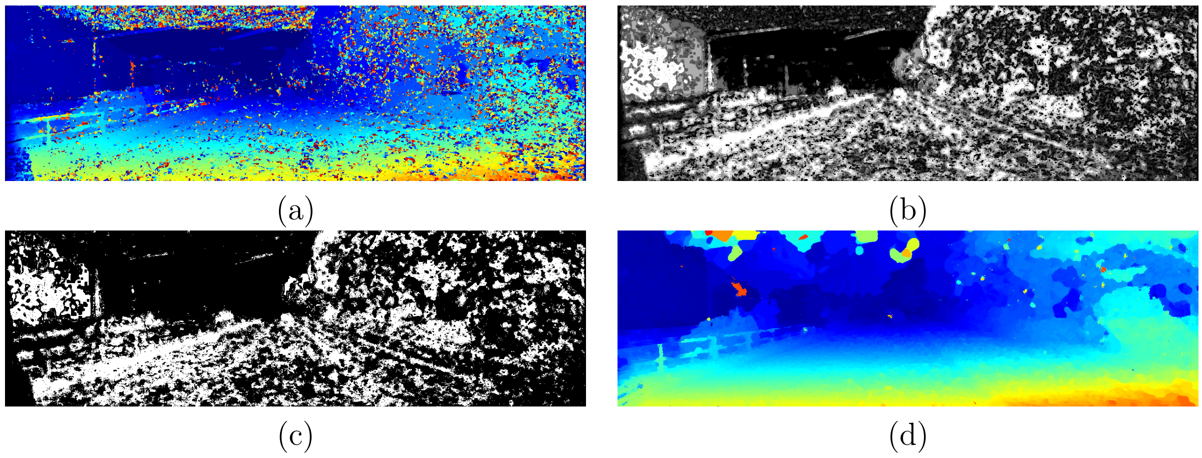


Figure 12.4: RP selection. (a) Noisy disparity map computed by Block-Matching algorithm (BM) on stereo pair 000027 of the KITTI dataset, (b) O1 [178] confidence map, (c) set of RP (white) and UP (black) according to O1, (d) refined disparity map with our proposal.

on the frame 166 of the KITTI 2015 dataset deploying the disparity map generated by the popular Block-Matching (BM) algorithm.

Finally, we point out how the proposed strategy works by acting in the disparity domain only with reduced computational complexity, fitting very well with COTS stereo cameras and in general with embedded devices, such as nVidia Jetson TX2 used to measure runtime in our experiments.

12.2 Non-Local Anchoring

In this section, we introduce the proposed NLA framework, that given a disparity map $\mathcal{D}$ and a confidence map $\mathcal{C}$ encoding the uncertainty of each pixel (the higher the confidence, the better the assumed reliability), infers a completely dense and more accurate map. It starts by classifying each disparity point belonging to $\mathcal{D}$ in two categories: *reliable* and *unreliable* points, for short RP and UP respectively. In literature [217, 228], this task is accomplished by setting a threshold value ξ and considering as RP the points with a confidence value higher than ξ. That is,

$$RP = \{p \in \mathcal{D}, \mathcal{C}(p) \geq \xi\} \tag{12.1}$$

consequently, the remaining ones are considered UP,

$$UP = \{p \in \mathcal{D}, \mathcal{C}(p) < \xi\} \tag{12.2}$$

A new disparity map $\mathcal{D}'$ is then obtained by removing from $\mathcal{D}$ the UP set. The resulting $\mathcal{D}'$ map is characterized by a lower error rate, ideally 0, at the cost of a sparser distribution of pixels compared to $\mathcal{D}$.

Afterward, the full density of $\mathcal{D}'$ is restored by looking at reliable information within the RP set. To do so, given a pixel p and a 2D vector d, we first define a subset of pixels $P(p,d)$ as the *path* on which p lays according to the direction of d:

$$P(p,d) = \{q \in \mathcal{D}, \alpha \in N, q = p + \alpha d\} \tag{12.3}$$

For a pixel $u \in$ UP, we define its *anchor* along direction d, as the closest pixel to u laying on path $P(u,d)$:

$$a(u,d) = \{v \in RP, \min_{v} |u - v|\} \tag{12.4}$$

Given a set of paths on which u lays, a set $\mathcal{A}(u)$ of anchors will contribute to computing the new disparity value for such pixel. In particular, each anchor $a \in \mathcal{A}(u)$ spreads its disparity to u, weighting it according to a similarity function between features $\mathcal{I}(u)$ and $\mathcal{I}(a)$ as follows:

$$w(u,a) = \mathcal{G}(|\mathcal{I}(u) - \mathcal{I}(a)|) \cdot \mathcal{G}(|u-a|) \tag{12.5}$$

The cues collected by each anchor $\mathcal{A}(u)$ are used to build a weighted histogram, on which each $w(u,a)$ increases the index corresponding to disparity hypothesis of pixel a. Finally, the weighted median is computed among the collected contributions:

$$\mathcal{D}(u) = \min_k \sum_{i=0}^{k} w(u,a_i) \geq \frac{1}{2} \sum_{i=0}^{n} w(u,a_i) \tag{12.6}$$

We rely on a Gaussian function $\mathcal{G}$ to encode the similarity between the unreliable pixel and one of its anchor points and on color intensity $\mathcal{I}(u)$ in the reference image. This strategy, coupled with the weighted median, enables edge-preserving disparity propagation. Figure 12.3 shows an example of anchoring for an unreliable pixel (red), receiving the contribution from a set of anchors (yellows).

Computational complexity for NLA is extremely low, as all the anchors of each unreliable pixel and their corresponding weights can be processed on a single image scan for each path in constant time. It only depends on the size of the image and the number of paths deployed for anchoring. It is worth observing that our proposal, conversely from other methods, is not constrained to a restricted area (i.e., local patches). Moreover, differently from recent methodologies exploiting confidence to improve stereo accuracy [169, 178, 217, 228], our framework acts on the disparity domain hence not requiring any information from the DSI thus enabling, for instance, its deployment with COTS devices.

Optionally, before replacing the unreliable pixel u according to the outlined strategy, a further local aggregation step can improve the effectiveness of the information gathered from nearby points. This optional phase can be carried out by building a DSI with the

$w(u, a_k)$ weights and filtering it according to the same similarity function $\mathcal{G}$. This step enables the collection of additional contributions from nearby UP pixels q_k, which set of anchors $\mathcal{A}(q_k)$ is different from $\mathcal{A}(u)$.

Stereo	All				Non-occ			
algorithm	bad 1(%)	bad 2(%)	RMSE	MAE	bad 1(%)	bad 2(%)	RMSE	MAE
BM	35.13	32.32	14.32	6.85	26.37	23.54	10.94	4.49
+ FBS [21]	33.47	28.60	12.79	5.28	24.67	19.59	8.32	2.93
+ MF [176]	27.43	23.99	10.14	4.32	18.42	14.95	5.82	2.17
+ WMF [286]	26.22	22.92	10.08	4.18	17.22	13.91	5.56	2.00
+ WMF + GF [275]	26.33	22.92	11.27	4.75	17.41	14.04	7.59	2.86
+ WMF + JBF [275]	28.03	24.93	10.95	4.68	18.86	15.75	6.87	2.44
+ LRI [279]	27.99	24.99	19.10	6.97	20.64	17.81	14.93	4.51
+ LRI + MF + BF [279]	26.02	21.53	15.78	5.94	18.68	14.23	11.18	3.58
+ LC [148]	24.23	20.00	11.68	4.74	16.30	12.23	7.93	2.65
+ NLA + O1	**22.90**	**19.86**	**9.36**	**3.63**	**14.08**	**11.20**	**5.33**	**1.71**
+ NLA + opt.	*6.23*	*4.07*	*3.06*	*0.85*	*2.20*	*1.21*	*1.77*	*0.44*

Table 12.1: Experimental results averaged on Middlebury 2014 with BM algorithm. Best results are in bold.

12.3 Threshold-free RP selection

According to the description reported in Section 12.2, classifying the disparity values in UP and RP plays a key-role for NLA to achieve optimal performance. Thus, choosing the confidence threshold ξ is of paramount importance. This strategy is common to other successful attempts to exploit confidence measures inside stereo algorithms [217, 228] or, in general, when we want to remove erroneous matches from the disparity map. For such tasks, proper tuning of the threshold ξ is required to achieve the best results.

To address this issue, we propose a second level framework to effectively distinguish pixels into RP and UP according to features extracted from the confidence map without any manual tuning. To this aim, we fed to a random forest, trained in classification mode, the following local and global features computed from the confidence map:

- $\mathcal{C}_p$, the confidence value for pixel p

- $\mu_N(\mathcal{C}_p)$, the average confidence computed on a local window $\mathcal{N}$, centered in p and made of $\bar{\bar{N}}$ pixels

$$\mu^{\mathcal{C}}_{\mathcal{N}}(p) = \frac{1}{\bar{\bar{\mathcal{N}}}} \sum_{q \in \mathcal{N}} \mathcal{C}_q \tag{12.7}$$

- $\sigma_N(\mathcal{C}_p)$, the variance of confidence on a local window $\mathcal{N}$, centered in p and made of $\bar{\bar{\mathcal{N}}}$ pixels

$$\sigma^{\mathcal{C}}_{\mathcal{N}}(p) = \frac{1}{\bar{\bar{\mathcal{N}}}} \sum_{q \in \mathcal{N}} [\mathcal{C}_q - \mu_{\mathcal{N}}(\mathcal{C}_p)]^2 \tag{12.8}$$

- $\delta_\mu(p)$, or *deviation from average confidence*, the absolute difference between $\mathcal{C}(p)$ and the average confidence over the entire disparity map $\mathcal{D}$ (i.e., $\mu_{\mathcal{D}}(\mathcal{C})$)

$$\delta_\mu(p) = |\mathcal{C}_p - \mu^{\mathcal{C}}_{\mathcal{D}}(\mathcal{C})| \tag{12.9}$$

- $\delta_\sigma(p)$, or *deviation from variance of confidence*, the absolute difference between $\mathcal{C}(p)$ and the average confidence over the entire disparity map $\mathcal{D}$ (i.e., $\sigma_{\mathcal{D}}(\mathcal{C})$)

$$\delta_\sigma(p) = |\mathcal{C}_p - \sigma^{\mathcal{C}}_{\mathcal{D}}(\mathcal{C})| \tag{12.10}$$

Concerning μ and σ, we process these features three times with increasing size of the local window $\mathcal{N}$, respectively $\Omega = 3 \times 3$, $\Theta = 7 \times 7$ and $\Gamma = 11 \times 11$. As result, we obtain the following feature vector $f_9(p)$

$$\begin{multline} f_9(p) = \{\mathcal{C}_p, \mu^{\mathcal{C}}_{\Omega}(p), \mu^{\mathcal{C}}_{\Theta}(p), \mu^{\mathcal{C}}_{\Gamma}(p), \sigma^{\mathcal{C}}_{\Omega}(p), \sigma^{\mathcal{C}}_{\Theta}(p), \\ \sigma^{\mathcal{C}}_{\Gamma}(p), \delta_\mu(p), \delta_\sigma(p)\} \end{multline} \tag{12.11}$$

We train on such feature vector a random forest, made of 10 trees, maximum depth equal to 15 and a minimum number of samples in each node to split equal to 12, in order to achieve an automatic RP selection without any hand-chosen threshold. Figure 12.4 shows a qualitative example of RP selection. Given a disparity map (a) and a confidence map (b), the reliable pixels are selected (c) and plugged into the NLA framework to obtain the final map (d).

12.4 Experimental results

In this section, we evaluate the effectiveness of the proposed NLA framework with disparity maps obtained, on challenging datasets, by two stereo algorithms: Block Matching (BM) and Semi-Global Matching (SGM). The choice was driven by the fast inference enabled by the two algorithms. Embedded stereo cameras with onboard processing (e.g., [1] or [149]) can run both BM and SGM at more than 30 FPS, sourcing disparity estimates in real-time with limited power consumption. In such a scenario, NLA can further improve the overall accuracy with low complexity, making it suited for embedded systems.

To exhaustively assess the effectiveness of our proposal, we compare it to state-of-the-art disparity refinement methods acting in the disparity domain. Moreover, since NLA relies on a confidence measure, we also compare it with recent methodologies exploiting confidence prediction to improve stereo accuracy [169, 178, 217] acting in the DSI domain. We also evaluate for NLA the effect yielded by a different number of anchors and by the optional aggregation step outlined. Moreover, we validate the effectiveness of UP/RP selection module by reporting comparison with the manual optimal choice of the ξ value by cross-validation. We evaluate all these aspects on the Middlebury 2014 [213] training dataset and then we evaluate the effectiveness of the overall NLA framework also on KITTI 2015 [155].

Stereo algorithm	All bad 1%	All bad 2%	All RMSE	All MAE	Non-occ bad 1%	Non-occ bad 2%	Non-occ RMSE	Non-occ MAE
SGM [83]	24.38	22.00	13.18	5.33	14.52	12.14	7.96	2.49
+ FBS [21]	25.06	21.55	12.05	4.51	15.46	11.93	6.80	2.10
+ MF [176]	23.13	20.44	11.20	4.43	13.45	10.74	5.95	1.91
+ WMF [286]	21.88	19.29	11.32	4.34	12.26	9.67	5.69	1.74
+ WMF + GF [275]	22.22	19.56	12.54	4.96	12.72	10.10	7.96	2.68
+ WMF + JBF [275]	22.25	19.80	11.94	4.61	12.46	10.02	6.45	1.91
+ LRI [279]	21.46	18.77	14.12	4.84	13.24	10.74	8.63	2.34
+ LRI + MF + BF [279]	22.01	17.68	13.26	4.81	14.09	9.81	7.80	2.40
+ LC [148]	20.39	16.60	10.56	3.98	12.59	9.05	6.56	1.95
+ Lev.stereo* [169]	22.22	19.52	12.45	4.60	13.38	10.73	7.39	2.20
+ Lev.stereo [169]	21.69	18.66	13.63	3.74	13.63	10.05	6.13	1.96
+ Smart-SGM [178]	22.67	19.71	11.51	4.33	13.57	10.78	6.54	2.05
+ PBCP* [217]	23.97	21.56	19.36	6.54	14.12	11.72	12.07	3.10
+ PBCP [217]	23.72	21.31	18.79	6.34	13.89	11.49	11.47	2.97
+ NLA + O1	**18.68**	**15.44**	**7.16**	**2.65**	**11.94**	**9.29**	**4.59**	**1.45**
+ NLA + opt.	*7.72*	*5.18*	*3.50*	*0.99*	*3.24*	*1.81*	*2.01*	*0.49*

Table 12.2: Experimental results averaged on Middlebury 2014 with SGM [83] algorithm. Best results are in bold. Algorithms marked with * use the original confidence measure proposed in the paper.

12.4.1 Evaluation on Middlebury 2014

In this section, we provide exhaustive experimental results concerning the full NLA framework (i.e., deploying the random forest for threshold-free RP selection and local aggregation step) and other refinement methods on the Middlebury 2014 training dataset[1].

Stereo algorithm	All bad 1(%)	All bad 2(%)	All RMSE	All MAE	Non-occ bad 1(%)	Non-occ bad 2(%)	Non-occ RMSE	Non-occ MAE
SGM [83]	24.38	22.00	13.18	5.33	14.52	12.14	7.96	2.49
+ NLA + O1 ($\xi = 0.4$)	18.90	15.86	8.06	2.99	**11.44**	**8.77**	4.63	**1.44**
+ NLA + O1 (ξ-less)	**18.68**	**15.44**	**7.16**	**2.65**	11.94	9.29	**4.59**	1.45

Table 12.3: Experimental results on Middlebury 2014 with SGM, comparing results obtained by NLA when using a threshold or the random forest classification of the RP. Best results in bold.

Comparison with other refinement strategies. In tables 12.1 and 12.2 we report results achieved by the following disparity refinement methods: fast bilateral solver (FBS [21]), median filter (MF [176]), weighted median filter (WMF [286]), weighted median

[1]We process Middlebury 2014 stereo pairs at quarter resolution. All the results reported in this work have been computed at such resolution on training split.

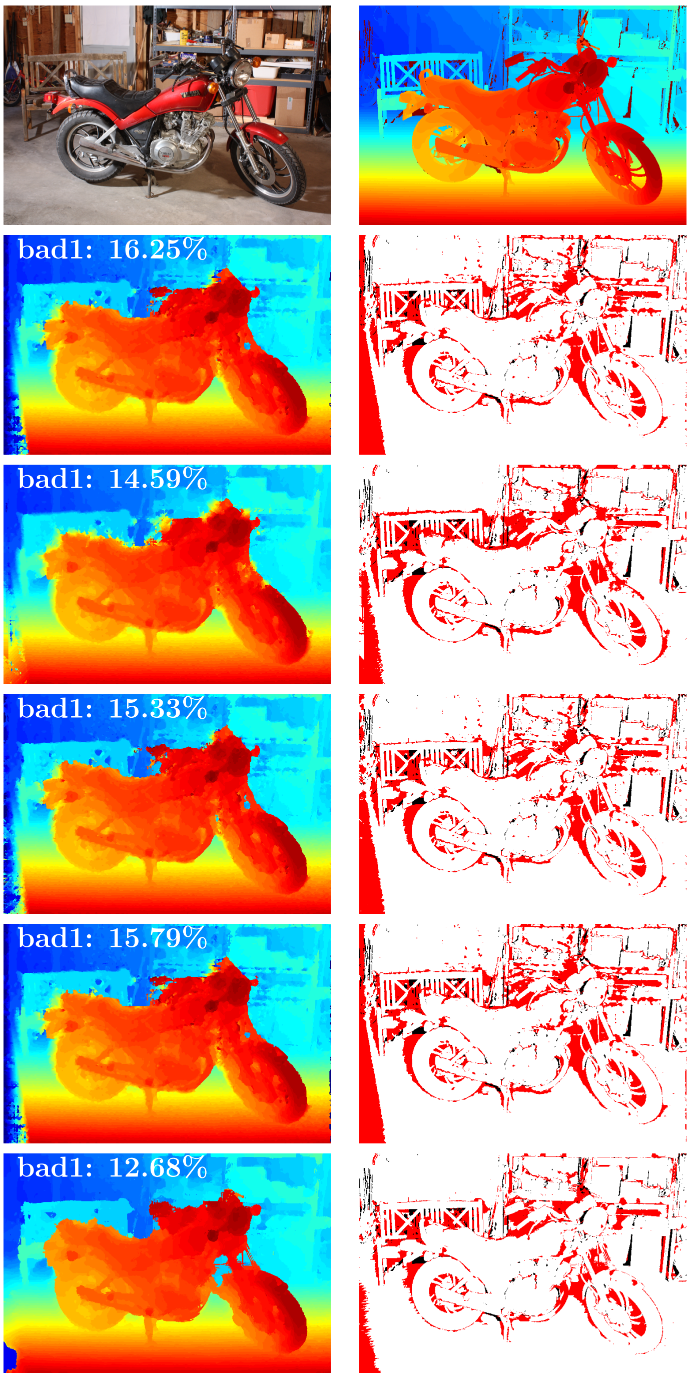


Figure 12.5: Qualitative results on Motorcycle stereo pair. First row: reference image and ground truth disparity. Then, from top to bottom, disparity maps with overimposed bad1 rate and error maps for, respectively, SGM [83], SGM+Lev.stereo [169], Smart-SGM [178], SGM+PBCP [217] and SGM+NLA. All methods use O1 as confidence measure.

filter together plus guided filter (WMF + GF [275]), weighted median filter plus joint bilateral filter (WMF + JBF [275]) and local consistency filter (LC [148]). All of these methods process only disparity map and the reference image. For each of these methods the patch size is set to 15×15. Moreover, we include left right interpolation (LRI) and the full refinement pipeline deployed in [279] (LRI + MF + BF) using authors'. We report, for each method, the amount of pixels having a disparity error larger than 1 and 2 (bad 1% and bad2%), as well as root mean square error (RMSE) and mean average error (MAE). In the same tables, we show results concerning the NLA framework with 16 anchors (i.e., from horizontal, vertical, diagonal and half-diagonal directions) using the state-of-the-art O1 [178] confidence measure. It is obtained by training a random forest framework to process 20 features extracted from the disparity map, that are Disparity Agreement (DA), Disparity Scattering (DS), Median Deviation of Disparity (MDD), Median disparity (MED) and Variance of disparity (VAR) on four windows of size $5 \times 5, 7 \times 7, 9 \times 9$ and 11×11 [178]. Its effectiveness drove the choice of this measure in the estimation of correct matches and by the aim of our framework, working in the disparity domain only and possibly running on constrained architectures, for which deep learning approaches would not be suited. We followed implementation notes, hyper-parameters tuning and code provided by the authors [178], training on a subset of images from KITTI 2012 dataset (the first 20 images) as in [184]. Since the effectiveness of the confidence measure is crucial for our method, we also report in the final row the results achieved by NLA processing an optimal confidence measure, capable of ideally distinguish between RP and UP. This represents the lower bound for the error rate with NLA. The automatic selection method proposed was trained on the 13 additional images available in Middlebury 2014 dataset [213] for each of the two considered algorithms. Table 12.1 report the effectiveness of disparity refinement methods with the BM algorithm. We can notice how the proposed NLA outperforms all of the considered refinement methods. In particular, compared to the second best method LC, NLA is more effective by nearly 2% on both all pixels and non-occluded. The last row highlights how, if an ideal confidence measure is deployed, our framework is capable of reducing the error rate from over 35% of wrong pixels in the image to almost 6%. Table 12.2 shows the results with SGM [83]. Since our SGM implementation is based on BM algorithm to obtain the data term, we first highlight how the results obtained by processing maps by NLA are very similar (even better in this case) to those obtained by running SGM optimization on the entire DSI (without applying any additional post-processing step, not deployed on our baseline SGM). This proves the effectiveness of our proposal when compared to more complex approaches such as SGM. Moreover, the DSI of the filtering map is not required with NLA, while SGM necessarily needs such information. In these experiments, we also deploy three additional methodologies relying on confidence measures to improve the results of SGM. The first one is a confidence-based modulation of the DSI carried-out before the SGM optimization, referred to as *Lev.stereo* [169]. The second one is a weighted sum of the contribution of the different scanlines, according to confidence, referred to as *Smart-SGM* [178]. The last one consists of a dynamic setting of the smoothness terms P1 and P2 according to confidence, referred to as *PBCP* [217]. We included them as representative state-of-the-art methodologies relying on confidence measures to improve the accuracy of stereo and we report results obtained when processing the confidence measures they were proposed with (marked with * in the table) as well as with the same one deployed by NLA for a fair comparison. We can observe how the NLA framework outperforms all of them, obtaining its best accuracy deploying the O1 measure. Moreover, our proposal works in the disparity domain, not requiring intermediate results from the SGM pipeline and it is thus a general-purpose technique suited for any stereo algorithm. Figure 12.5 shows a qualitative comparison between considered approaches and NLA.

Evaluation of RP selection. Once confirmed the superiority of the full NLA framework, in this section we inquire about the effectiveness of the threshold-free RP selection enabled by the random forest classifier. Table 12.3 shows comparison between the results achieved by the manually selected threshold through k-fold cross-validation, highlighting how the random forest selection strategy increases, on average, the accuracy of the

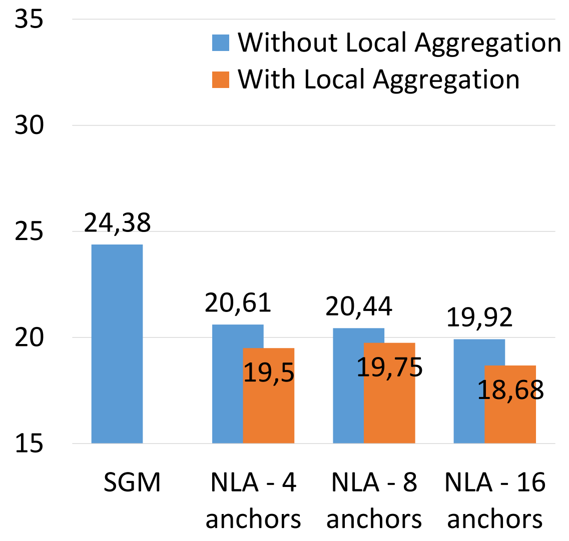


Figure 12.6: Experimental results on the entire Middlebury 2014 dataset, varying the number of anchors and enabling/disabling local aggregation with NLA framework, SGM algorithm + O1.

refined disparity maps when considering all pixels, while it performs slightly worse on non-occluded pixels, thus mainly improving selection and refinement occluded regions.

Ablation studies on NLA and runtime. To better understand the key factors enabling for such improvements, we report results concerning the use of a different amount of anchors as well as without the optional local aggregation step, deployed during the previous evaluations. Figure 12.6 plots the error rate as a function of the number of anchors (4, 8 and 16) of the vanilla NLA framework (blue) and NLA with local aggregation (orange). It shows how the aggregation step enables a notable improvement, reducing the error rate by about 1% on SGM. About runtime, on a Jetson TX2 CPU (Arm v8), NLA runs in 1.82s without aggregation, rising up to 6.39s with full (not optimized) aggregation.

12.4.2 Evaluation on KITTI 2015

In this section, we report experimental results concerned with the KITTI 2015 training dataset [155], depicting outdoor environments very different from the Middlebury indoor scenes. We deploy for these experiments our full pipeline with 16 anchors, local aggre-

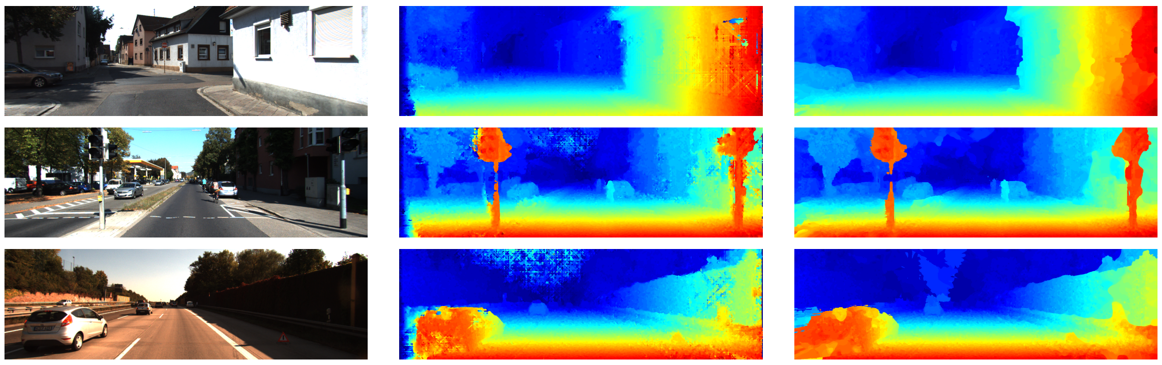


Figure 12.7: Qualitative results on KITTI 2015 dataset [155]. From top to bottom, stereo pairs 085, 186 and 197. From left to right, reference frame and disparity maps from SGM [83] or refined with NLA.

gation and threshold-free selection of RP. Table 12.4 reports experimental results when refining disparity maps obtained by BM and SGM algorithms. We report the amount of pixels having a disparity error larger than 3 (bad 3%). Since KITTI 2015 dataset is very different compared to Middlebury 2014, we tuned P1 and P2 smoothing penalties to 0.3 and 3 in order to obtain the most accurate results from the original SGM algorithm. We compare our results with best methods MF, WMF and LC approaches. We can observe how, even on this very different dataset, the NLA framework can reduce the error rate of the raw disparity maps by nearly 26% (BM) and by more than 3% (SGM), notably outperforming the other refinement techniques. Since the scene contents depicted by KITTI 2015 are more smooth compared to indoor scenes considered before (e.g., large road planes), the smoothing constraint enforced by SGM is stronger than the non-local refinement processed by NLA, being nonetheless capable of reaching with BM a comparable degree of accuracy with significantly lower computational efforts. Focusing entirely on SGM results, we report, as for the Middlebury 2014 evaluation, the improvements yielded by state-of-the-art confidence-based cost modulations proposed in [169, 178, 217]. Similarly to Middlebury 2014, we evaluated the three previous strategies with their originally proposed confidence measures as well as with the same plugged into NLA for a fair comparison. The trend previously highlighted is confirmed on KITTI 2015 as well. Figure 12.7 shows additional qualitative results on KITTI 2015.

Stereo algorithm	bad 3% - All	
	BM	SGM
Baseline	37.30	10.78
MF [176]	19.95	8.73
WMF [286]	21.03	8.81
LRI [279]	25.29	10.12
LRI +MF + BF [279]	18.90	9.11
LC [148]	14.92	9.72
+ Lev.stereo* [169]	-	10.10
+ Lev.stereo [169]	-	9.52
+ Smart-SGM [178]	-	8.47
+ PBCP* [217]	-	10.63
+ PBCP [217]	-	10.62
NLA + O1	**11.42**	**7.68**

Table 12.4: Experimental results averaged on KITTI 2015 with BM and SGM [83] algorithms. Best results are in bold.

12.5 Conclusions

In this chapter, we proposed a fast, yet accurate, non-local disparity refinement technique based on confidence measures. It jointly enables the benefits of techniques acting in the disparity domain and the power of confidence measures extracted from the same domain. Conversely from other similar techniques, leveraging on confidence measures and designed for specific algorithms, our proposal acts outside the stereo pipeline, making it a general purpose alternative, hence totally agnostic to the stereo algorithm generating disparity maps. Experimental results on popular datasets confirmed the superiority of NLA compared to known techniques when dealing with disparity maps obtained from algorithms suited for deployment on embedded devices.

0 0 k

Chapter 13

SMD-Nets: Stereo Mixture Density Networks

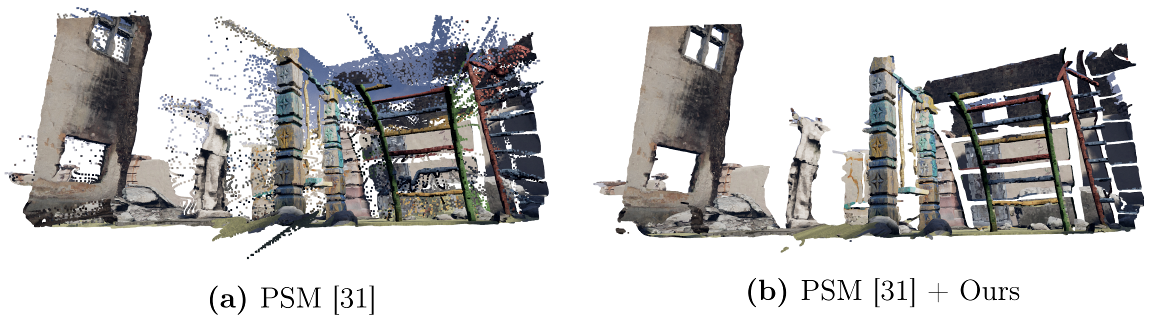
The content of this chapter has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2021) - "SMD-Nets: Stereo Mixture Density Networks" [244].

13.1 Introduction

Although deep stereo matching produce compelling results, two major issues remain unsolved: predicting accurate depth boundaries and generating high-resolution outputs with limited memory and computation.

The first issue is shown in Fig.13.1a: As neural networks are smooth function approximators, they often poorly reconstruct object boundaries, causing "bleeding" artifacts (i.e., flying pixels) when converted to point clouds. These artifacts can be detrimental to subsequent applications such as 3D reconstruction or 3D object detection. Thus, while being ignored by most commonly employed disparity metrics, accurate 3D reconstruction of contours is a desirable property for any stereo matching algorithm.

Furthermore, existing methods are limited to discrete predictions at pixel locations of



(a) PSM [31]

(b) PSM [31] + Ours

Figure 13.1: Point Cloud Comparison between the stereo network PSM [31] and our Stereo Mixture Density Network (SMD-Net) on the UnrealStereo4K dataset. Notice how SMD-Net notably alleviates bleeding artifacts near object boundaries, resulting in more accurate 3D reconstructions.

a fixed resolution image grid, while geometry is a piecewise continuous quantity where object boundaries may not align with pixel centers. Increasing the output resolution by adding extra upsampling layers partially addresses this problem as this leads to a significant increase in memory and computation.

In this chapter, we address both issues. Our key contribution is to learn a representation that is precise at object boundaries and scales to high output resolutions. In particular, we formulate the task as a continuous estimation problem and exploit bimodal mixture densities [24] as output representation. Our simple formulation lets us (1) avoid bleeding artifacts at depth discontinuities, (2) regress disparity values at arbitrary spatial resolution with constant memory and (3) provides a measure for aleatoric uncertainty.

We illustrate the boundary bleeding problem and our solution to it in Fig.13.2. While classical deep networks for stereo regression suffer from smoothness bias and are incapable of representing sharp disparity discontinuities, the proposed Stereo Mixture Density Networks (SMD-Nets) effectively address this issue. The key idea is to alter the output representation adopting a mixture distribution such that sharp discontinuities can be regressed *despite* the fact that the underlying neural networks are only able to make smooth predictions (note that all curves in Fig. 13.2b are indeed smooth while the predicted disparity is discontinuous).

Furthermore, the proposed model is capable of regressing disparity values at arbitrary continuous locations in the image, effectively solving a stereo super-resolution task. In combination with the proposed representation, this allows for regressing sharp discontinuities at sub-pixel resolution while keeping memory requirements constant.

In summary, we present: (i) A novel learning framework for stereo matching that exploits compactly parameterized bimodal mixture densities as output representation and can be trained using a simple likelihood-based loss function. (ii) A continuous function formulation aimed at estimating disparities at arbitrary spatial resolution with constant memory footprint. (iii) A new large-scale synthetic binocular stereo dataset with ground truth disparities at 3840×2160 resolution, comprising photo-realistic renderings of indoor and outdoor environments. (iv) Extensive experiments on several datasets demonstrating improved accuracy at depth discontinuities for various backbones on binocular stereo, monocular and active depth estimation tasks.

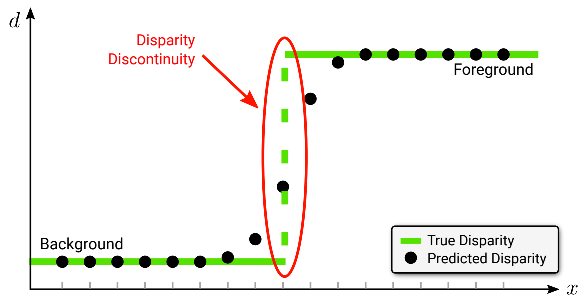
13.2 Method

Fig.13.3 illustrates our model. We first encode a stereo pair into a feature map using a convolutional backbone (left). Next, we estimate parameters of a mixture density distribution at any continuous 2D location via a multi-layer perceptron head, taking the bilinearly interpolated feature vector as input (middle). From this, we obtain a disparity as well as uncertainty map (right). We now explain our model, loss function and training protocol in detail.

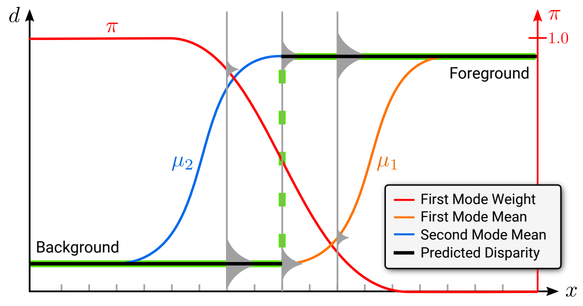
13.2.1 Problem Statement

Let $\mathbf{I} \in \mathbb{R}^{W \times H \times 6}$ denote an RGB stereo pair for which we aim to predict a disparity map $\mathbf{D}$ at arbitrary resolution. As shown in Fig.13.2, classical stereo regression networks suffer from over-smoothing due to the smoothness bias of neural networks. In this work, we exploit a mixture distribution as output representation [24] to overcome this limitation.

More specifically, we propose to use a bimodal Laplacian mixture distribution with



(a) Stereo Regression Network



(b) Stereo Mixture Density Network

Figure 13.2: Overcoming the Smoothness Bias with Mixture Density Networks. For clarity, we visualize the disparity d only for a single image row. (a) Classical deep networks for stereo regression suffer from smoothness bias and hence continuously interpolate object boundaries. In addition, disparity values are typically predicted at discrete spatial locations. (b) In this work, we propose to use a bimodal Laplacian mixture distribution (illustrated in gray) with weight π as output representation which can be queried at any continuous spatial location x. This allows our model to accurately capture uncertainty close to depth discontinuities while at inference time recovering sharp edges by selecting the mode with the highest probability density. In this example, the first mode (μ_1, b_1) models the background and the second mode (μ_2, b_2) models the foreground disparity close to the discontinuity. When the probability density of the foreground mode becomes larger than the probability density of the background mode, the most likely disparity sharply transitions from the background to the foreground value.

weight π and two modes (μ_1, b_1), (μ_2, b_2) to model the continuous probability distribution over disparities at a particular pixel. Using two modes allows our model to capture both the foreground as well as the background disparity at object boundaries. At inference time, we recover sharp object boundaries by selecting the mode with the highest density value. Thus, our model is able to transition from one disparity to another in a discontinuous fashion while at the same time relying only on the regression of functions $(\pi, \mu_1, b_1, \mu_2, b_2)$ which are smooth with respect to the image domain and which therefore can easily be represented using neural networks.

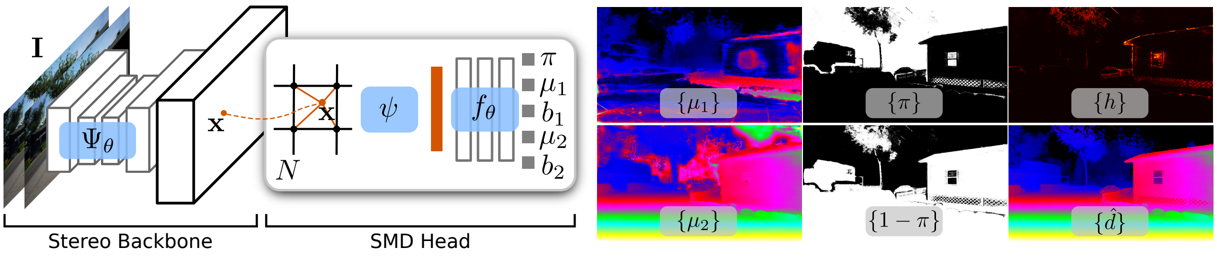


Figure 13.3: Method Overview. We assume a 2D or 3D stereo backbone network Ψ_θ which takes as input a stereo pair $\mathbf{I}$ (either concatenated or processed by siamese towers), and outputs a D-dimensional feature map in the domain of the reference image. Given any continuous 2D location $\mathbf{x}$, we query its feature from the feature map via bilinear interpolation as denoted by ψ. The interpolated feature vector is then fed into a multi-layer perceptron f_θ to estimate a five-dimensional vector $(\pi, \mu_1, b_1, \mu_2, b_2)$ which represents the parameters of a bimodal distribution. N denotes the number of points randomly sampled at continuous 2D locations during training and the number of pixels during inference. On the right we show maps of μ_1, μ_2, π, $1-\pi$, uncertainty h and predicted disparity $\hat{d}$.

13.2.2 Stereo Mixture Density Networks

We now formally describe our model. Let

$$\Psi_\theta : \mathbb{R}^{W\times H\times 6} \to \mathbb{R}^{W\times H\times D} \tag{13.1}$$

denote a *stereo backbone* network with parameters θ as shown in Fig.13.3 (left). Ψ_θ takes as input the stereo pair $\mathbf{I}$ and outputs a D-dimensional feature map, represented in the domain of the reference image (e.g.the left image of a stereo pair). Examples for such networks are standard 2D convolutional networks, or networks which perform 3D convolutions. For the 2D networks, the stereo pair can be concatenated as input or processed by means of siamese towers with shared weights as typically done for 3D architectures. Similarly, this generic formulation also applies to the structured light setting (e.g., Kinect setting where $\mathbf{I} \in \mathbb{R}^{W\times H}$) and the monocular depth estimation problem ($\mathbf{I} \in \mathbb{R}^{W\times H\times 3}$).

As geometry is a piecewise continuous quantity, we apply a deterministic transformation to obtain feature points for any continuous location in $\mathbb{R}^{W\times H}$. More specifically, for every continuous 2D location $\mathbf{x} \in \mathbb{R}^2$, we bilinearly interpolate the features from its four nearest pixel locations in the feature map $\mathbb{R}^{W\times H\times D}$. More formally, we describe this transformation as:

$$\psi : \mathbb{R}^2 \times \mathbb{R}^{W\times H\times D} \to \mathbb{R}^D \tag{13.2}$$

Finally, we employ a multi-layer perceptron to map this abstract feature representation to a five-dimensional vector $(\pi, \mu_1, b_1, \mu_2, b_2)$ which represents the parameters of a univariate bimodal mixture distribution:

$$f_\theta : \mathbb{R}^D \to \mathbb{R}^5 \tag{13.3}$$

Note that we have re-used the parameter symbol θ to simplify notation. In the following, we use θ to denote all parameters of our model. We refer to $f_\theta(\psi(\cdot,\cdot))$ as *SMD Head*, see Fig. 13.3 for an illustration.

To robustly model a distribution over disparities which can express two modes close to disparity discontinuities, we choose a bimodal Laplacian mixture as output representation:

$$p(d) = \frac{\pi}{2\,b_1} e^{-\frac{|d-\mu_1|}{b_1}} + \frac{1-\pi}{2\,b_2} e^{-\frac{|d-\mu_2|}{b_2}} \tag{13.4}$$

In summary, our model can be compactly expressed as:

$$p(d|\mathbf{x}, \mathbf{I}, \theta) = p(d|f_\theta\left(\psi(\mathbf{x}, \Psi_\theta(\mathbf{I}))\right)) \tag{13.5}$$

At inference time, we determine the final disparity $\hat{d}$ by choosing the mode with the highest density value:

$$\hat{d} = \underset{d\in\{\mu_1,\mu_2\}}{\text{argmax}}\ p(d) \tag{13.6}$$

Note that our formulation allows to query the disparity $\hat{d} \in \mathbb{R}$ at any continuous 2D pixel location, enabling ultra high-resolution predictions with sharply delineated object boundaries. This is illustrated in Fig.13.4.

Our model also allows for capturing the aleatoric uncertainty of the predicted disparity by evaluating the differential entropy of the continuous mixture distribution as:

$$h = -\int p(d) \log p(d) \, \mathrm{d}d \tag{13.7}$$

In practice, we use numerical quadrature to obtain an approximation of the integral.

13.2.3 Loss Function

We consider the supervised setting and train our model by minimizing the negative log-likelihood loss:

$$\mathcal{L}_{NLL}(\theta) = -\, \mathbb{E}_{d,\mathbf{x},\mathbf{I}} \, \log p(d|\mathbf{x}, \mathbf{I}, \theta) \tag{13.8}$$

where the input $\mathbf{I}$ is randomly sampled from the dataset, $\mathbf{x}$ is a random pixel location in the continuous image domain $\Omega = [0, W-1] \times [0, H-1]$, sampled as described in 13.2.4, and d is the ground truth disparity at location $\mathbf{x}$.

13.2.4 Training Protocol

Sampling Strategy. While a naïve strategy samples pixel locations $\mathbf{x}$ randomly and uniformly from the image domain Ω, our framework also allows for exploiting custom sampling strategies to focus on depth discontinuities during training. We adopt a *Depth Discontinuity Aware (DDA)* sampling approach during training that explicitly favors points located near object boundaries while at the same time maintaining a uniform coverage on the entire image space. More specifically, given a ground truth disparity map at training time, we first compute an object boundary mask in which a pixel is considered to be part of the boundary if its (4-connected) neighbors have a disparity that differs by more than 1 from its own disparity. This mask is then dilated using a $\rho \times \rho$ kernel to enlarge the boundary region. We report an analysis using different ρ values in the experimental

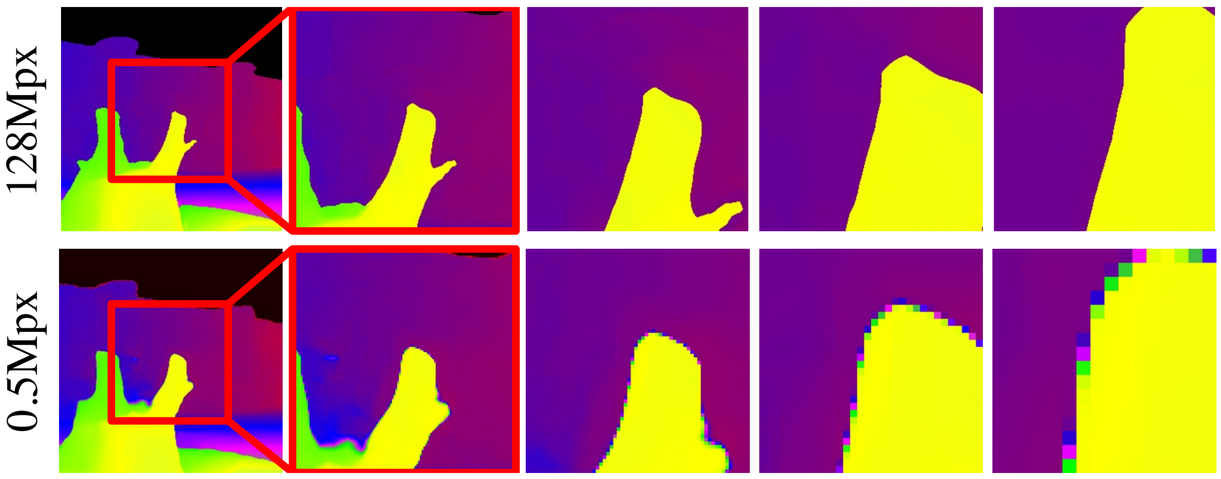


Figure 13.4: Ultra High-resolution Estimation. Comparison of our model using the PSM backbone at 128Mpx resolution (top) to the original PSM at 0.5Mpx resolution (bottom), both taking stereo pairs at 0.5Mpx resolution as input. Each column shows a different zoom-level. Note how our method leads to sharper boundaries and high resolution outputs.

section. Given the total number of training points N, we randomly and uniformly select $N/2$ points from the domain of all pixels belonging to depth discontinuity regions and $N/2$ points uniformly from the continuous domain of all remaining pixels. At inference time, we leverage our model to predict disparity values at each location of an (arbitrary resolution) grid.

Stereo Super-Resolution. Our continuous formulation allows us to exploit ground truth at higher resolution than the input $\mathbf{I}$, which we refer to as stereo super-resolution. In contrast, classical stereo methods cannot realize arbitrary super-resolution without changing their architecture.

13.3 Experimental Results

In this section, we first describe the datasets used for evaluation and implementation details. We then present an extensive evaluation that demonstrates the benefits of the proposed SMD Head in combination with different stereo backbones on several distinct tasks.

13.3.1 Datasets

UnrealStereo4K. Motivated by the lack of large-scale, realistic and high-resolution stereo datasets, we introduce a new photo-realistic *binocular stereo* dataset at 3840×2160 resolution with pixel-accurate ground truth. We create this synthetic dataset using the popular game engine Unreal Engine combined with the open-source plugin UnrealCV [196]. We additionally create a synthetic *active monocular* dataset (mimicking the Kinect setup) at 4112×3008 resolution by warping a gray-scale reference dot pattern to each image, following [205]. We split the dataset into 7720 training pairs, 80 validation pairs and 200 *in-domain* test pairs. To evaluate the generalization ability of our method, we also create an *out-of-domain* test set by rendering 200 stereo pairs from an unseen scene. Similarly, the active dataset contains 3856 training images, 40 validation images, 100 test images.

RealActive4K. We further collect a small real-world active dataset of an indoor room with a Kinect-like stereo sensor, including 2570 images at a resolution 4112×3008 pixels from which we use 2500 for training, 20 for validation and 50 for testing. We perform Block Matching with left-right consistency check to use as co-supervision for training models jointly on synthetic (UnrealStereo4K) and real data.

13.3.2 Implementation Details

Architecture. In principle, our SMD Head is compatible with any stereo backbone ψ_θ from the literature. In our implementation, we build on top of two state-of-the-art 3D stereo architectures: Pyramid Stereo Matching (PSM) network [31] and Hierarchical Stereo Matching (HSM) network [263]. PSM is a well-known and popular stereo network while HSM represents a method with good trade-off between accuracy and computation. Moreover, we also adopt a naïve U-Net structure [70] that takes as input concatenated images of a stereo pair in order to show the effectiveness of our model on 2D architectures. For the aforementioned networks, we follow the official code provided by the authors.

Our SMD Head f_θ is implemented as a multi-layer perceptron (MLP) following [208]. More specifically, the number of neurons is $(D, 1024, 512, 256, 128, 5)$. We use sine activations [223] except for the last layer that uses a sigmoid activation for regressing the five-parameter output. For the 3D backbone, we select the matching probabilities from the cost volume in combination with features of Ψ_θ at different resolutions as input to our SMD Head. For the 2D backbone case, instead, we select features from different layers of the decoder. We refer the reader to the supplementary material for details.

Training. We implement our approach in PyTorch [172] and use Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ as optimizer [118]. We train all models from scratch using a single NVIDIA V100 GPU. During training, we use random crops from $\mathbf{I}$ as input to the stereo backbone and sample $N = 50,000$ training points from each crop. We scale the ground truth disparity to $[0, 1]$ for each dataset for numerical stability. Moreover, for RGB inputs we perform chromatic augmentations on the fly, including random brightness, gamma and color shifts sampled from uniform distributions. We further apply horizontal and vertical random flipping while adapting the ground truth disparities accordingly. Please see the supplementary material for details regarding the training procedure for each dataset.

Evaluation Metrics. Following [35], we evaluate the *Soft Edge Error (SEE$_k$)* metric on pixels belonging to object boundaries, defined as the minimum absolute error between the predicted disparity and the corresponding local ground truth patch of size $k \times k$ ($k = \{3, 5\}$ in our experiments). Intuitively, *SEE* penalizes over-smoothing artifacts stronger compared to small misalignments in a local window, where the former is more harmful to subsequent applications.

While not our main focus, we also report the *End Point Error (EPE)* as the standard error metric obtained by averaging the absolute difference between predictions and ground truth disparity values to evaluate the overall performance. For both *SEE* and *EPE*, we compute the average (Avg) and $\sigma(\Delta)$ metrics, with the latter one representing the percentage of pixels having errors greater than Δ.

13.3.3 Ablation Study

We first examine the impact of different components and training choices of the proposed SMD-Nets on the *in-domain* UnrealStereo4K test set. Unless specified otherwise, we use 960×540 as resolution for the binocular input $\mathbf{I}$ and 3840×2160 for the corresponding ground truth, used for both supervision and testing purposes. The active input images consist of random dot patterns where the dots become indistinguishable at low resolution (e.g., 960×540). Therefore we use 2056×1504 as active input size while keeping the ground truth dimension at 4112×3008.

Output Representation. In Table 24.2, we evaluate the effectiveness of our mixture density output representation across both, 2D and 3D stereo backbones on multiple tasks including binocular stereo, monocular depth and active depth. We adopt U-Net and PSM on the binocular stereo dataset as representatives of 2D and 3D backbones and report results of HSM in the supplementary for the sake of space. We also use the same U-Net backbone for a monocular depth estimation task by replacing the input with only the reference image of a binocular stereo pair to show the advantage of our method on various tasks. For the active setup, we choose HSM as it represents a network designed specifically for high-resolution inputs which takes as input the monocular active image and the *fixed* reference dot pattern.

We compare our bimodal distribution to two other output representations, standard disparity regression and a unimodal Laplacian distribution [108]. For fairness, we implement these baselines by replacing the last layer of our SMD Head to predict the disparity d or the unimodal parameters (μ, b), respectively, where the former is trained with a standard L1 loss while the latter with a negative log-likelihood loss. For all cases we use the proposed bilinear feature interpolation and the naïve random sampling strategy.

Table 24.2 shows that the proposed method effectively addresses the over-smoothing problem at object boundaries, achieving the lowest *SEE* for all backbones on all tasks, compared to both the standard disparity regression and the unimodal representation.

	Ψ_θ	Dim.	*SEEk*3			*SEEk*5			*EPE*	
			Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(3)$
Binocular Stereo	U-Net	1	2.15	41.69	24.16	2.03	39.65	22.98	1.48	8.18
	(2D)	2	2.38	42.28	25.74	2.26	40.42	24.57	1.97	10.44
	[70]	5	**1.57**	**30.06**	**14.77**	**1.45**	**28.05**	**16.57**	**1.28**	**5.94**
	PSM	1	1.98	36.32	20.35	1.85	34.42	19.21	**1.10**	5.52
	(3D)	2	2.50	39.40	23.63	2.37	37.57	22.51	1.88	7.73
	[31]	5	**1.52**	**26.98**	**12.68**	**1.38**	**24.93**	**11.49**	**1.11**	**4.80**
Mono.	U-Net	1	3.29	60.18	41.37	3.25	58.49	40.08	4.21	35.92
	(2D)	2	4.01	61.06	43.19	3.86	59.40	41.90	5.49	41.88
	[70]	5	**2.92**	**51.32**	**32.33**	**2.78**	**49.54**	**31.06**	**4.06**	**30.59**
Active	HSM	1	3.40	47.87	24.80	3.18	46.14	23.76	**1.29**	5.84
	(3D)	2	4.93	57.05	33.44	4.69	55.47	32.41	2.83	10.70
	[263]	5	**2.69**	**41.84**	**17.35**	**2.43**	**39.83**	**16.17**	1.42	**5.48**

Table 13.1: Output Representation analysis on the UnrealStereo4K test set. "Dim." refers to the output dimension of the SMD Head where 1 indicates the point estimate d, 2 the unimodal output representation (μ, b) [108] and 5 our bimodal formulation $(\pi, \mu_1, b_1, \mu_2, b_2)$.

Sampling	ρ	*SEEk*3			*SEEk*5			*EPE*	
		Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(3)$
Random	-	1.52	26.98	12.68	1.38	24.93	11.49	1.11	4.80
DDA	0	1.34	21.62	9.77	1.19	19.58	8.59	1.08	4.44
DDA	10	**1.13**	**18.64**	**8.69**	**0.98**	**16.67**	**7.55**	**0.92**	**3.88**
DDA	20	1.30	20.42	9.88	1.15	18.40	8.71	1.11	4.44

Table 13.2: Sampling Strategy analysis on the UnrealStereo4K test set using the PSM backbone.

Moreover, we observe that the unimodal representation sacrifices *EPE* for capturing the uncertainty, while our method is on par with the standard L1 regression. On the stereo dataset, the 3D backbone (PSM) consistently outperforms the 2D backbone (U-Net), therefore we use PSM for the following ablation experiments.

Sampling Strategy. In Table 13.2, we show the impact of the sampling strategy adopted during training. More specifically, we compare the naïve uniform sampling strategy and the proposed *DDA* approach using different dilation kernel sizes $\rho \times \rho$. As can be observed, *DDA* enables SMD-Nets to focus on depth discontinuities, resulting in better *SEE* compared to random point selection. Moreover, we observe that sampling exactly at depth boundaries (i.e., $\rho = 0$) leads to slightly degraded *EPE* and is less effective on *SEE*

Eval. GT	Training GT	*SEEk3*			*SEEk5*			*EPE*	
		Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(3)$
960×540	960×540	1.19	20.36	9.16	0.93	16.55	7.08	1.02	4.30
960×540	3840×2160	**0.98**	**15.42**	**7.05**	**0.78**	**12.44**	**5.54**	**0.89**	**3.81**
3840×2160	960×540	1.33	23.35	10.82	1.19	21.34	9.63	1.03	4.30
3840×2160	3840×2160	**1.13**	**18.64**	**8.69**	**0.98**	**16.67**	**7.55**	**0.92**	**3.88**

Table 13.3: Ground Truth Resolution analysis on the UnrealStereo4K test set using the PSM backbone.

Method	*In-domain*										*Out-of-domain*									
	SEEk3			*SEEk5*			*EPE*				*SEEk3*			*SEEk5*			*EPE*			
	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(1)$	$\sigma(2)$	$\sigma(3)$	Avg	$\sigma(1)$	$\sigma(3)$	Avg	$\sigma(1)$	$\sigma(3)$	Avg	$\sigma(1)$	$\sigma(2)$	$\sigma(3)$
PSM [31]	1.73	33.06	16.57	1.61	31.11	15.44	1.09	11.88	6.94	5.19	2.19	36.94	20.07	1.99	34.09	18.25	1.53	16.92	10.25	7.83
PSM [31] + BF [275]	1.65	30.93	15.26	1.52	28.92	14.10	1.10	11.81	6.95	5.23	2.16	35.64	19.16	1.95	32.76	17.32	1.56	18.89	10.28	7.89
PSM [31] + SM [35]	1.50	29.22	12.71	1.37	27.16	11.54	1.10	11.65	6.69	4.97	2.03	33.91	16.74	1.82	30.92	14.82	1.54	16.43	9.73	7.36
PSM [31] + CE + SM [35]	1.33	27.31	10.14	1.19	25.25	8.99	**0.86**	10.40	**4.93**	**3.50**	1.84	29.87	13.30	1.62	26.84	11.46	1.37	13.29	7.84	**6.03**
PSM [31] + Ours	**1.13**	**18.64**	**8.69**	**0.98**	**16.67**	**7.55**	0.92	**8.24**	5.06	3.88	**1.59**	**24.58**	**12.54**	**1.38**	**21.63**	**10.73**	**1.27**	**12.11**	**7.69**	**6.06**
HSM [263]	2.01	41.63	23.81	1.89	39.69	22.62	1.16	14.81	8.20	5.84	2.43	44.49	26.17	2.24	41.74	24.33	1.75	22.03	12.73	9.23
HSM [263] + BF [275]	1.88	39.68	21.70	1.77	37.67	20.49	1.19	14.78	8.21	5.88	2.39	43.60	24.14	2.19	40.82	23.28	1.80	22.05	12.79	9.33
HSM [263] + SM [35]	1.83	40.52	22.30	1.70	38.53	21.07	1.17	14.73	8.11	5.74	2.31	43.76	25.16	2.11	40.97	23.29	1.76	21.88	12.54	9.03
HSM [263] + CE + SM [35]	2.00	45.71	25.99	1.87	43.72	24.71	1.17	16.17	8.12	5.46	2.61	48.27	28.84	2.41	45.56	26.98	1.91	26.12	14.40	10.14
HSM [263] + Ours	**1.31**	**24.31**	**10.81**	**1.17**	**22.30**	**9.67**	**1.00**	**11.40**	**6.09**	**4.34**	**2.03**	**34.82**	**17.75**	**1.82**	**31.88**	**15.83**	**1.66**	**19.16**	**10.72**	**7.77**

Table 13.4: Comparison on UnrealStereo4K. All methods evaluated on ground truth at 3840×2160 given input size 960×540.

which penalizes small misalignment in a local window. Instead, setting $\rho = 10$ allows the network to focus on larger regions near edges and results in the best performance, while increasing ρ does not improve performance further. Finally, it is worth to notice that this strategy also allows our model to improve the overall performance, achieving lower *EPE* metrics. In the following experiments, we thus adopt the *DDA* strategy using $\rho = 10$ for our SMD-Nets.

Ground Truth Resolution. Table 13.3 shows the results of our model trained and tested on the stereo data using ground truth maps at different resolutions, while maintain-

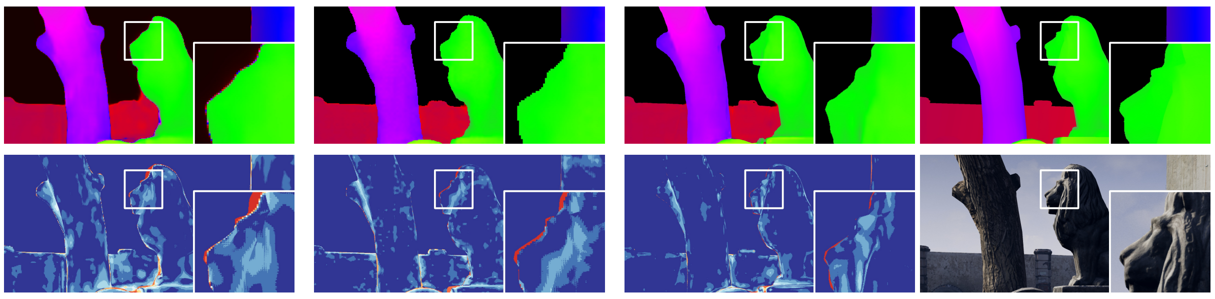

(a) PSM [31] (b) PSM [31]+CE+SM [35] (c) PSM+Ours (d) GT, Input

Figure 13.5: Qualitative Results on UnrealStereo4K. The first row shows the predicted disparity maps while the second row depicts the corresponding error maps. We zoom-in a patch in all images to better perceive details near depth boundaries.

ing the input size at 960×540. Towards this goal, we train our model adopting ground truth disparities 1) resized to the same resolution as the input using nearest interpolation and 2) at the original resolution (i.e.3840×2160). We notice that sampling points from higher resolution disparity maps always leads to better results compared to using low resolution ground truth. We remark that the proposed model effectively leverages high resolution ground truth thanks to its continuous formulation, without requiring additional memory compared to standard stereo networks based on CNNs.

13.3.4 Comparison to Existing Baselines

We now compare to several baselines [35, 275] which aim to address the over-smoothing problem. Bilateral median filtering (BF) is often adopted to sharpen disparity predictions [220, 275]. Chen et al. [35] address the over-smoothing problem of 3D stereo backbones using 1) a post-processing step to extract a single-modal (SM) distribution from the full discrete distribution; 2) a cross-entropy (CE) loss to enforce a unimodal distribution during training. We reimplement [35] as no official code is available. As [35] has been proposed for 3D backbones only, we use PSM [31] and HSM [263] as the stereo backbones in the following experiments.

UnrealStereo4K. Table 13.4 collects results obtained from different models on both *in-domain* and *out-of-domain* test splits of the binocular UnrealStereo4K dataset. We use the same input resolution of 960×540 for all methods. While our baseline methods can only use supervision with the same size as the input, we leverage our continuous formulation to supervise SMD-Nets using ground truth at 3840×2160, on which we also evaluate all methods. For our competitors, we upsample their outcome using nearest neighbor interpolation during testing. Both original PSM and HSM follow the same training setting of our SMD-Nets. Table 13.4 suggests that BF [275] and SM [35] slightly improve *SEE* on both backbones while leading to degraded performance on *EPE* metrics. Using the CE loss combined with SM [35] leads to effective improvement on both *SEE* and *EPE* on the PSM backbone. Interestingly, we notice that adopting the same CE

Method	*SEEk*3			*SEEk*5			*EPE*	
	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(3)$
PSM [31]	1.10	20.57	9.74	0.99	17.83	9.02	0.73	2.49
PSM [31] + CE + SM [35]	1.02	16.12	7.53	0.90	13.80	6.94	0.66	2.09
PSM [31] + Ours	**0.90**	**13.09**	**6.66**	**0.79**	**10.93**	**6.01**	**0.59**	**1.95**

Table 13.5: Comparison on KITTI 2015 Validation Set using boundaries extracted from instance segmentation masks to evaluate on depth discontinuity regions.

Method	All Areas			Non Occluded		
	Bg	Fg	All	Bg	Fg	All
GANet-deep [282]	1.48	3.46	1.81	1.34	3.11	1.63
HD^3-Stereo [271]	1.70	3.63	2.02	1.56	3.43	1.87
GwcNet-g [75]	1.74	3.93	2.11	1.61	3.49	1.92
PSM [31]	1.86	4.62	2.31	1.71	4.31	2.14
PSM [31] + CE + SM [35]	**1.54**	4.33	2.14	1.70	3.90	1.93
PSM [31] + Ours	1.69	**4.01**	**2.08**	**1.54**	**3.70**	**1.89**

Table 13.6: Comparison on KITTI 2015 Test Set, evaluated on the official online benchmark. All the reported numbers represent official submissions from the authors.

+ SM strategy leads to worse performance on HSM. A possible explanation is that the CE loss requires to trilinearly interpolate a matching cost probability distribution to the full resolution $W \times H \times D_{max}$ (with D_{max} denoting the maximum disparity), where HSM predicts a less fine-grained cost distribution compared to PSM, thus making the cross-entropy loss less effective. Moreover, we remark that the CE loss is more expensive to compute, compared to our simple continuous likelihood-based formulation and CE + SM can only be applied on 3D backbones. In contrast, our approach based on the bimodal output representation notably outperforms our competitors on *SEE* on both the *in-domain* and *out-of-domain* test sets, showing how our strategy predicts better disparities near boundaries. Moreover, we highlight that we achieve consistently better estimates on standard *EPE* metrics compared to the original backbone while performing comparably to the CE + SM baseline. Fig.23.4 shows our gains at object boundaries.

KITTI 2015. We fine-tune all methods trained using UnrealStereo4K on the KITTI 2015 training set. Since the provided ground truth disparities are sparse, we rely on the naïve random sampling strategy to train our model. On the validation set, we evaluate

Method	*SEEk*3			*SEEk*5			*EPE*	
	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(1)$	$\sigma(2)$	Avg	$\sigma(3)$
PSM [31]	3.35	46.50	29.40	2.61	41.04	24.87	4.12	17.43
PSM [31] + CE + SM [35]	**2.62**	34.80	**19.02**	**1.83**	28.92	**14.11**	**2.80**	**12.12**
PSM [31] + Ours	**2.61**	**34.26**	19.83	**1.88**	**28.71**	15.32	3.03	13.60

Table 13.7: Generalization on Middlebury v3. All models are trained on UnrealStereo4K and evaluated on the training set of Middlebury v3 dataset.

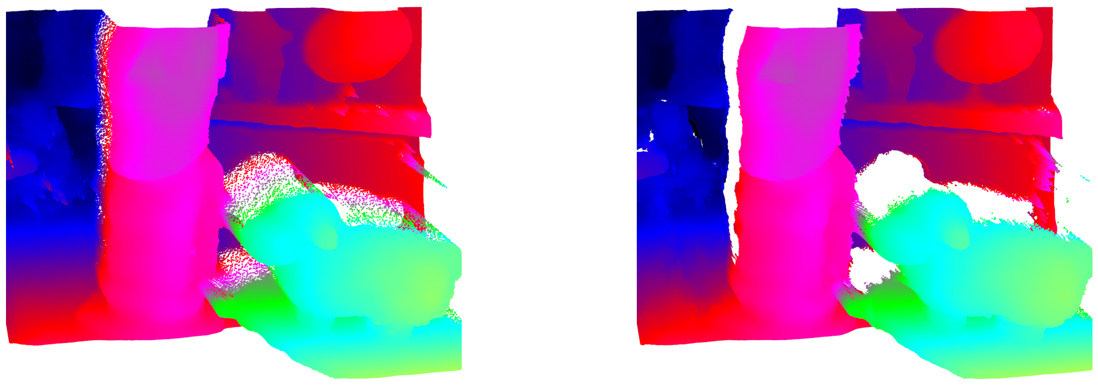
SEE on boundaries of instance segmentation maps from the KITTI dataset, following the evaluation procedure described in [35]. Furthermore, we predict disparities on the test set using the same fine-tuned model and submit to the online benchmark. Table 13.5 and 13.6 show our results using PSM as backbone (we provide additional results on the validation set adopting HSM in the supplement). Note that our SMD-Net not only achieves superior performance on both *SEE* and *EPE* metrics on the validation set compared to the original PSM and [35] (Table 13.5), but also outperforms both on the test set and is on par with state-of-the-art stereo networks on standard metrics of the KITTI benchmark (Table 13.6).

13.3.5 Synthetic-to-Real Generalization

Lastly, we demonstrate how models trained on the synthetic dataset generalize to the real-world domain for both binocular stereo and active depth estimation.

Middlebury v3. Table 13.7 reports the performance of supervised methods trained on the UnrealStereo4K and tested without fine-tuning on the training set of the Middlebury v3 dataset. We evaluate them using the high-resolution ground truth. Compared to the original PSM baseline, our SMD-Net achieves much better generalization on both *SEE* and *EPE* metrics while performing on par with [35].

RealActive4K. Moreover, we fine-tune our active depth models jointly on active UnrealStereo4K and RealActive4K with pseudo-ground truth from Block Matching. Fig. 13.6 shows that this allows for estimating sharp disparity edges for real captures even though Block Matching does not provide supervision in these areas. In contrast, standard



(a) Disparity Regression (L1) **(b)** SMD Head (Bimodal)

Figure 13.6: Generalization on RealActive4K using the HSM backbone. The point clouds of standard disparity regression using L1 loss (a) show bleeding artifacts whereas our bimodal distribution (b) leads to clean reconstructions.

disparity regression fails to predict clean object boundaries.

13.4 Conclusion

In this chapter, we propose SMD-Nets, a novel stereo matching framework aimed at improving depth accuracy near object boundaries and suited for disparity super-resolution. By exploiting bimodal mixture densities as output representation combined with a continuous function formulation, our method is capable of predicting sharp and precise disparity values at arbitrary spatial resolution, notably alleviating the common over-smoothing problem in learning-based stereo networks. Our model is compatible with a broad spectrum of 2D and 3D stereo backbones. Our extensive experiments demonstrate the advantages of our strategy on a new high-resolution synthetic stereo dataset and on real-world stereo pairs.

Chapter 14

Real-time self-adaptive deep stereo

The content of this chapter has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019) - “Real-time self-adaptive deep stereo” [238].

14.1 Introduction

As recently highlighted in [168, 235], learnable models suffer from loss in performance when tested on unseen scenarios due to the domain shift between training and testing data - often synthetic and real, respectively. Good performance can be regained by fine-tuning on *few* annotated samples from the target domain. Yet, obtaining ground-truth labels requires the use of costly active sensors (e.g., LIDAR) and noise removal by expensive manual intervention or post-processing [246]. Recent works [69, 168, 235, 284, 291] proposed to overcome the need for labels with unsupervised losses that require only stereo pairs from the target domain. Although effective, these techniques are inherently limited by the number of samples available at training time. Unfortunately, for many tasks, like autonomous driving, it is unfeasible to acquire, in advance, samples from all possible deployment domains (e.g., every possible road and/or weather condition).

We propose to address the domain shift issue by casting *adaptation* as a *continuous*

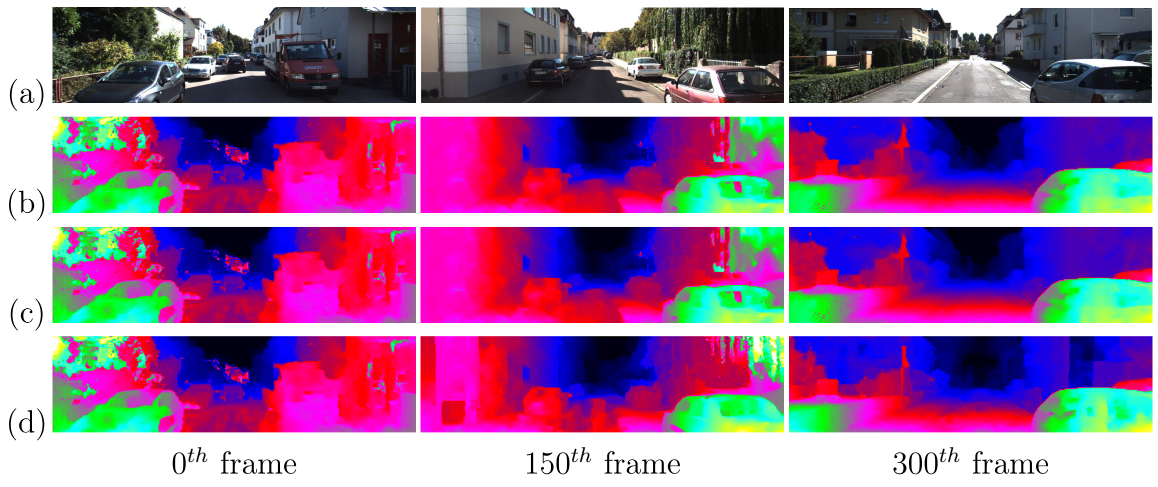


Figure 14.1: Disparity maps predicted by *MADNet* on a KITTI sequence [67]. Left images (a), no adaptation (b), online adaptation of the *whole* network (c), online adaptation by *MAD* (d). Green pixel values indicate larger disparities (i.e., closer objects).

learning process whereby a stereo network can evolve *online* based on the images gathered by the camera during its real deployment. We believe that the ability to continually adapt itself in real-time is key to any deep learning machinery intended to work in real scenarios. We achieve continuous online adaptation by: deploying one of the unsupervised losses proposed in literature (i.e., [64, 69, 235, 284]); computing error signals on the current frames; updating the whole network by back-propagation (from now on shortened as *back-prop*); and moving to the next pair of input frames. However, such adaptation reduces inference speed greatly. Therefore, to keep a high enough frame rate we propose a novel Modularly ADaptive Network (*MADNet*) architecture designed to be lightweight, fast and modular. This architecture exhibits accuracy comparable to DispNetC [151] using one-tenth parameters, runs at around 40 FPS for disparity inference and performs an online adaptation of the whole network at around 15 FPS. Moreover, to achieve an even higher frame rate during adaptation, at the cost of a slight loss in accuracy, we develop a Modular ADaptation (*MAD*) algorithm that leverages the modular architecture of *MADNet* in order to train sub-portions of the whole network independently. Using *MADNet* together with *MAD* we can adapt our network to unseen environments without supervision at approximately 25 FPS.

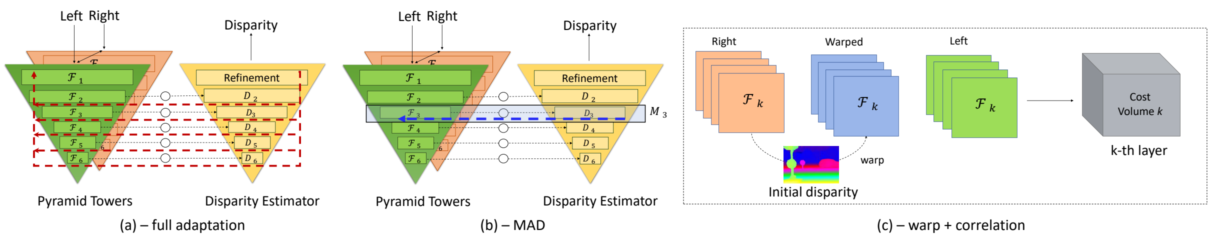


Figure 14.2: Sketch of *MADNet* architecture (a), each circle between an F_k and D_k represents a warp and correlation layer (c). Each pair ($\mathcal{F}_i$,$\mathcal{D}_i$) composes a module $\mathcal{M}_i$, adaptable independently by *MAD*,blue arrow in (b), faster than full back-prop, red arrow in (a).

Figure 14.1 shows the disparity maps predicted by *MADNet* on three successive frames of a video sequence from the KITTI dataset [67]: without undergoing any adaptation - row (b); by adapting online the *whole* network - row (c); and by our computationally efficient *MAD* approach - row (d). Rows (c) and (d) show how online adaptation can improve the quality of the predicted disparity maps significantly in as few as 150 frames (i.e., a latency of about 10 seconds for complete online adaptation and 6 seconds for *MAD*).

14.2 Online Domain Adaptation

Modern machine learning models reduce their accuracy when tested on data significantly different from the training set, an issue commonly referred to as *domain shift*. Despite all the research work to soften this issue, the most effective practice still relies on additional offline training on samples from the target environments. The domain shift curse is inherently present in deep stereo networks since most training iterations are performed on synthetic images quite different from real ones. Then, adaptation can be effectively achieved by fine-tuning the model offline on samples from the target domain by relying on expensive annotations or unsupervised loss functions [64, 69, 235, 284].

In this chapter we move one step further arguing that adaptation can be effectively performed online as soon as new frames are available, thereby obtaining a deep stereo system capable of adapting itself dynamically. For our online adaptation strategy we do not rely on the availability of ground truth annotations and, instead, use one of the proposed unsupervised losses. To adapt the model we perform on-the-fly a single train iteration (forward and backward pass) for each incoming stereo pair. Therefore, our model is always in training mode and continuously fine-tuning to the sensed environment.

14.2.1 MADNet - Modularly ADaptive Network

One of the main limitations that have prevented exploration of online adaptation is the computational cost of performing a full train iteration for each incoming frame. Indeed, we will show experimentally how it roughly corresponds to a reduction of the inference rate of the system to roughly one third, a price far too high to be paid with most modern architectures. To address this issue, we have developed Modularly ADaptive Network (*MADNet*), a novel lightweight model for depth estimation inspired by fast, yet accurate, architectures proposed for optical flow [200, 231].

We deploy a pyramidal strategy for dense disparity regression for two key purposes: i) maximizing speed and ii) obtaining a modular architecture as depicted in Figure 22.1. Two pyramidal towers extract features from the left and right frames through a cascade of independent modules sharing the same weights. Each module consists of convolutional blocks aimed at reducing the input resolution by two 3×3 convolutional layers, respectively with stride 2 and 1, followed by Leaky ReLU non-linearities. According to Figure 22.1, we count 6 blocks providing us with feature $\mathcal{F}$ from half resolution to 1/64, namely $\mathcal{F}_1$ to $\mathcal{F}_6$, respectively. These blocks extract 16, 32, 64, 96, 128 and 192 features.

At the lowest resolution (i.e., $\mathcal{F}_6$), we forward features from left and right images into a correlation layer [151] to get the raw matching costs. Then, we deploy a disparity decoder $\mathcal{D}_6$ consisting of 5 additional 3×3 convolutional layers, with 128, 128, 96, 64, and 1 output channels. Again, each layer is followed by Leaky ReLU, except the last one, which provides the disparity map at the lowest resolution.

Then, D_6 is up-sampled to level 5 by bilinear interpolation and used both for warping right features towards left ones before computing correlations and as input to $\mathcal{D}_5$. Thanks to our design, from $\mathcal{D}_5$ onward, the aim of the disparity decoders $\mathcal{D}_k$ is to refine and correct the up-scaled disparities coming from the lower resolution. In our design, the correlation scores computed between the original left and right features aligned according to the lower resolution disparity prediction guide the network in the refinement process. We compute all correlations inside our network along a [-2,2] range of possible shifts.

This process is repeated up to quarter resolution (i.e., $\mathcal{D}_2$), where we add a further refinement module consisting of 3×3 dilated convolutions [231], with, respectively 128, 128, 128, 96, 64, 32, 1 output channels and 1, 2, 4, 8, 16, 1, 1 dilation factors, before bilinearly upsampling to full resolution.

MADNet has a smaller memory footprint and delivers disparity maps much more rapidly than other more complex networks such as [31, 109, 129] with a small loss in accuracy. Concerning efficiency, working at decimated resolutions allows for computing correlations on a small horizontal window [231], while warping features and forwarding disparity predictions across the different resolutions enables to maintain a small search range and look for residual displacements only. With a 1080Ti GPU, *MADNet* runs at about 40 FPS at KITTI resolution and can perform online adaptation with full back-prop at 15 FPS.

14.2.2 MAD - Modular ADaptation

As we will show, *MADNet* is remarkably accurate with full online adaptation at 15 FPS. However, for some applications, it might be desirable to achieve a higher frame rate without losing the adaptation ability. Most of the time needed to perform online adaptation is spent executing back-prop and weights update across all the network layers. A naive way to speed up the process will be to *freeze* the initial part of the network and fine tune only a subset of k final layers, thus realizing a shorter back-prop that would yield a higher frame rate. However, there is no guarantee that these last k layers are indeed those that would benefit most from online fine-tuning. For example, the initial layers of the network should be probably adapted alike, as they directly interact with the images from a new, *unseen*, domain. In subsection 14.3.5 we will provide experimental results to show that training only the final layers is not enough for handling the drastic domain changes that typically occur in practical applications.

Following the key intuition that to keep up with fast inference we should pursue a partial, though effective, online adaptation, we developed $\underline{\text{M}}$odular $\underline{\text{AD}}$aptation (*MAD*) an online adaptation algorithm tailored to *MADNet*, though possibly extendable to any multi-scale inference network. Our method takes a network $\mathcal{N}$ and subdivides it into p non-overlapping portions, each referred to as module $\mathcal{M}_i$, $i \in [1, p]$, such that $\mathcal{N} = [\mathcal{M}_1, \mathcal{M}_2, ..\mathcal{M}_p]$. Each $\mathcal{M}_i$ ends with a final layer able to output a disparity estimation y_i. Thanks to its design, decomposing our network is straightforward by grouping layers working at the same resolution i from both $\mathcal{F}_i$ and $\mathcal{D}_i$ into a single module $\mathcal{M}_i$, e.g., $\mathcal{M}_3 = (\mathcal{F}_3, \mathcal{D}_3)$. At each training iteration, thus, we can optimize one of the modules independently from the others by using the prediction y_i to compute a loss function and then executing the shorter back-prop only across the layers of $\mathcal{M}_i$. For instance to optimize $\mathcal{M}_3$ we would use y_3 to compute a loss function and back-prop only through $\mathcal{D}_3$ and $\mathcal{F}_3$ following the blue path in Figure 22.1 (b). Conversely, full back-prop would follow the much longer red path in Figure 22.1 (a). This paradigm allows for

- Interleaved optimization of different $\mathcal{M}_i$, thereby approximating full back-prop over time while gaining considerable speed-up.
- Fast adaptation of single modules, which instantly provides benefits to the overall accuracy of the whole network thanks to its cascade architecture.

At deployment time, for each incoming stereo pair, we run a forward pass to obtain all estimates $[y_1, \ldots, y_p]$ at each resolution, then we choose a portion $\theta \in [1, \ldots, p]$ of the network to train according to some heuristic and finally update $\mathcal{M}_\theta$ according to a loss computed on y_θ. We consider a valid heuristic any function that outputs a probability distribution among the p modules of N from which we could perform sampling.

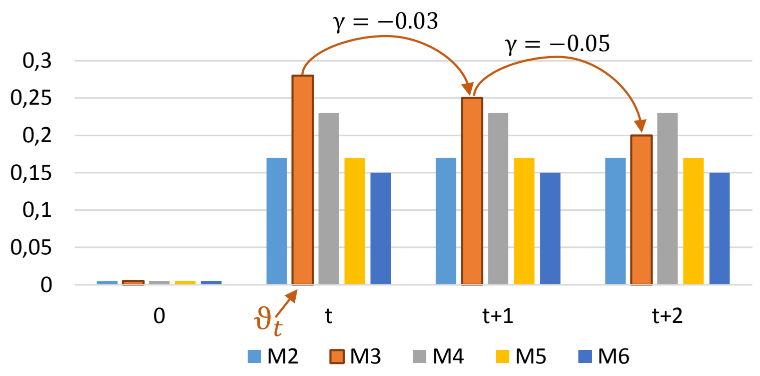


Figure 14.3: Example of reward/punishment mechanism. X axis shows time while Y histogram values. At time t, the most probable module selected for adaptation is $\mathcal{M}_3$. After two steps $(t+2)$, its probability gets demoted in favour of $\mathcal{M}_4$.

14.2.3 Reward/punishment selection

Among different functions, we obtained good results using a reward/punishment mechanism. We start by creating a histogram $\mathcal{H}$ with p bins (i.e., one per module) all initialized to 0. For each stereo pair we perform a forward pass to get the disparity predictions y_i and measure the performance of the model by computing a loss $\mathcal{L}_t$ using the full resolution disparity y and the input frames x (e.g., reprojection error between left and warped right frames as in [69]). Then, we sample the portion to train $\theta \in [1, \dots, p]$ from a probability distribution obtained applying the softmax function to the value of the bins in $(\mathcal{H})$:

$$\theta_t \sim softmax(\mathcal{H}). \tag{14.1}$$

We can compute one optimization step for layers of $\mathcal{M}_{\theta_t}$ with respect to the loss $\mathcal{L}_t^{\theta_t}$ computed on the lower scale prediction y_{θ_t}. We have now partially adapted the network to the current environment. At the following iteration, we update $\mathcal{H}$ before choosing the new θ_t, increasing the probability of being sampled for the $\mathcal{M}_{\theta_{t-1}}$ that have proven effective. To do so we compute a noisy expected value for $\mathcal{L}_t$ by linear extrapolation of the losses at the previous two-time steps

$$\mathcal{L}_{exp} = 2 \cdot \mathcal{L}_{t-1} - \mathcal{L}_{t-2}, \tag{14.2}$$

and quantify the effectiveness of the last module optimized as

$$\gamma = \mathcal{L}_{exp} - \mathcal{L}_t. \tag{14.3}$$

Finally, we can change the value of $\mathcal{H}[\theta_t]$ according to γ, i.e. effective adaptation will have $\mathcal{L}_{exp} > \mathcal{L}_t$, thus $\gamma > 0$. We found out that adding a temporal decay to $\mathcal{H}$ increases the stability of the system, leading to the following update rule

$$\begin{aligned} \mathcal{H} &= 0.99 \cdot \mathcal{H} \\ \mathcal{H}[\theta_{t-1}] &= \mathcal{H}[\theta_{t-1}] + 0.01 \cdot \gamma \end{aligned} \tag{14.4}$$

Figure 14.3 shows an example of histogram $\mathcal{H}$ at generic time frames t and $t+2$, highlighting the transition from $\mathcal{M}_3$ to $\mathcal{M}_4$ as most probable module thanks to the aforementioned mechanism.

14.3 Experimental results

14.3.1 Evaluation protocol and implementation

To properly address practical deployment scenarios in which there are no ground truth data available for fine-tuning in the actual testing environments, we train our stereo network using synthetic data only [151].

To test the online adaptation we use those weights as a common initialization and carry out an extensive evaluation on the large and heterogeneous KITTI raw dataset [67] with depth labels [246] converted into disparities by knowing the camera parameters. Overall, we assess the effectiveness of our proposal on 43k images. Specifically, according to the KITTI classification, we evaluate our framework in four heterogeneous environments, namely *Road*, *Residential*, *Campus* and *City*, obtained by concatenation of the available video sequences and resulting in 5674, 28067, 1149 and 8027 frames respectively. Although these sequences are all concerned with driving scenarios, each has peculiar traits that would lead deep stereo model to gross errors without suitable fine-tuning in the target domain. For example, *City* and *Residential* often depict road surrounded by buildings, while *Road* concerns mostly highways and country roads, where the most common objects are cars and vegetation.

By processing stereo pairs within sequences, we can measure how well the network adapts, by either full back-prop or *MAD*, to the target domain compared to a model trained offline. For all experiments, we analyze both average End Point Error (EPE) and the percentage of pixels with disparity error larger than 3 (D1-all). Due to the image format being different for each sequence, we extract a central crop of size 320×1216 from each frame, which suits to the downsampling factor of our architecture and allows for validating almost all pixels with available ground truth disparities.

Finally, we highlight that for both full back-prop and *MAD*, we compute the error rate on each frame *before* applying the model adaptation step. That is, we measure performances achieved by the current model on the stereo frame at time t and *then* adapt it according to the current prediction. Therefore, the model update carried out at time t will affect the prediction only from frame $t + 1$ and so on. As unsupervised loss for online adaptation, we rely on the photometric consistency between the left frame and the right one reprojected according to the predicted disparity. Following [69], to compute the reprojection error between the two images we combine the Structural Similarity Measure (SSIM) and the L1 distance, weighted by 0.85 and 0.15, respectively. We selected this unsupervised loss function as it is the fastest to compute among those proposed in literature [168, 235, 291] and does not require any additional information besides a pair of stereo images.

14.3.2 MADNet performance

Before assessing the performance obtainable through online adaptation, we test the effectiveness of *MADNet* by following the canonic two-phase training using synthetic [151] and

	GWCNet [75]	DispNetC [151]	StereoNet [113]	*MADNet*
D1-all	2.11	4.34	4.83	4.66
Time	0.32	0.06	0.02	0.02

Table 14.1: Comparison between stereo architectures on the KITTI 2015 test set without adaptation. Detailed results available in the KITTI online leader-board.

		City		Residential		Campus		Road		
Model	Adapt.	D1-all(%)	EPE	D1-all(%)	EPE	D1-all(%)	EPE	D1-all(%)	EPE	FPS
DispNetC	No	8.31	1.49	8.72	1.55	15.63	2.14	10.76	1.75	15.85
DispNetC	Full	4.34	1.16	3.60	1.04	8.66	1.53	3.83	1.08	5.22
DispNetC-GT	No	3.78	1.19	4.71	1.23	8.42	1.62	3.25	1.07	15.85
MADNet	No	37.42	9.96	37.41	11.34	51.98	11.94	47.45	15.71	39.48
MADNet	Full	2.63	1.03	2.44	0.96	8.91	1.76	2.33	1.03	14.26
MADNet	*MAD*	5.82	1.51	3.96	1.31	23.40	4.89	7.02	2.03	25.43
MADNet-GT	No	2.21	0.80	2.80	0.91	6.77	1.32	1.75	0.83	39.48

Table 14.2: Performance on the *City*, *Residential*, *Campus* and *Road* sequences from KITTI [67]. Experiments with DispNetC [151] (top) and *MADNet* (bottom) with and without online adaptations. *-GT* variants are fine-tuned on KITTI training set ground truth.

real data. Thus, after training on synthetic data, we perform fine-tuning on the training sets of KITTI 2012 and KITTI 2015 and submit to the KITTI 2015 online benchmark. On Table 14.1 we report our result compared to other (published) fast inference architectures on the leaderboard (runtime measured on NVIDIA 1080Ti) as well as with a slower and more accurate one, GWCNet [75]. At the time of writing, our method ranks 90th. Despite the mid-rank achieved in terms of absolute accuracy, *MADNet* compares favorably to StereoNet [113] ranked 92nd, the only other high frame rate proposal on the KITTI leaderboard. Moreover, we get close to the performance of the original DispNetC [151] while using $\frac{1}{10}$ of the parameters and running more than twice faster.

14.3.3 Online adaptation

We will now show how online adaptation is an effective paradigm, comparable, or better, to offline fine-tuning. Table 14.2 reports extensive experiments on the four different KITTI environments. We report results achieved by i) DispNetC [151] implemented in our framework and trained, from top to bottom, on synthetic data following authors' guidelines, using online adaptation or fine-tuned on ground truth and ii) *MADNet* trained with the same modalities and, also, using *MAD*. These experiments, together to subsection 14.3.2, support the three-fold claim of this work.

DispNetC: Full adaptation. On top of Table 14.2, focusing on the D1-all metric, we can notice how running full back-prop online to adapt DispNetC [151] decimates the number of outliers on all scenarios compared to the model trained on the synthetic dataset only. In particular, this approach can consistently halve D1-all on *Campus*, *Residential* and *City* and nearly reduce it to one third on *Road*. Alike, the average EPE drops significantly across the four considered environments, with improvement as high as a nearly 40% relative improvement on the Road sequences. These massive gains in accuracy, though, come at the price of slowing the network down significantly to about one-third of the original inference rate, i.e. from nearly 16 to 5.22 FPS. As mentioned above, the Table also reports the performance of the models fine-tuned offline on the 400 stereo pairs with ground truth disparities from the 2012 and 2015 training dataset [66, 155]. It is worth pointing out how online adaptation by full back-prop turns out competitive to fine-tuning offline by ground truth, and even more accurate in the Residential environment. This fact may hint at training usupervisedly by a more considerable amount of data possibly delivering better models than supervision by fewer data.

***MADNet*: Full adaptation.** On bottom of Table 14.2 we repeat the aforementioned experiments for *MADNet*. Due to the much higher errors yielded by the model trained on synthetic data only, full online adaptation turns out even more beneficial with *MADNet*, leading to a model which is more accurate than DispNetC with Full adaptation in all sequences but *Campus* and can run nearly three times faster (i.e. at 14.26 FPS compared to the 5.22 FPS of DispNetC-Full). These results also highlight the inherent effectiveness of the proposed *MADNet*. Indeed, as vouched by the rows dealing with *MADNet*-GT and DispNetC-GT, using for both our implementations and training them following the same standard procedure in the field (i.e., pretraining on synthetic data and fine-tuning on KITTI training sets), *MADNet* yields better accuracy than DispNetC while running about 2.5 times faster.

***MADNet*: *MAD*.** Once proved that online adaptation is feasible and beneficial, we show that *MADNet* employing *MAD* for adaptation (marked as *MAD* in column *Adapt.*) allows for effective and efficient adaptation. Since the proposed heuristic has a non-deterministic sampling step, we have run the tests regarding *MAD* five times each and reported here the average performance. We refer the reader to subsection 14.3.5 for analysis on the standard deviation across different runs. Indeed, *MAD* provides a significant improvement in all the performance figures reported in the table compared to the corresponding models trained by synthetic data only. Using *MAD*, *MADNet* can be adapted paying a relatively small computational overhead which results in a remarkably fast inference rate of about 25 FPS. Overall, these results highlight how, whenever one has no access to training data from the target domain beforehand, online adaptation is feasible and worth. Moreover, if speed is a concern *MADNet* combined with *MAD* provides a favourable trade-off between accuracy and efficiency.

Short-term Adaptation. Table 14.2 also shows how all adapted models perform significantly worse on *Campus* compared the other sequences. We ascribe this mainly to *Campus* featuring fewer frames (1149) compared the other sequences (5674, 28067, 8027), which implies a correspondingly lower number of adaptation steps executed online. Indeed, a key trait of online adaptation is the capability to improve performance as more and more frames are sensed from the environment. This favourable behaviour, not captured by the average error metrics reported in Table 14.2, is highlighted in Figure 14.4, which plots the D1-all error rate over time for *MADNet* models in the four modalities. While without adaptation the error keeps being always large, models adapted online clearly improve over time such that, after a certain delay, they become as accurate as the model that could have been obtained by offline fine-tuning had ground truth disparities been available. In particular, full online adaptation achieves performance comparable to fine-tuning by the ground truth after 900 frames (i.e., about 1 minute) while for *MAD* it takes about 1600 frames (i.e., 64 seconds) to reach an almost equivalent performance

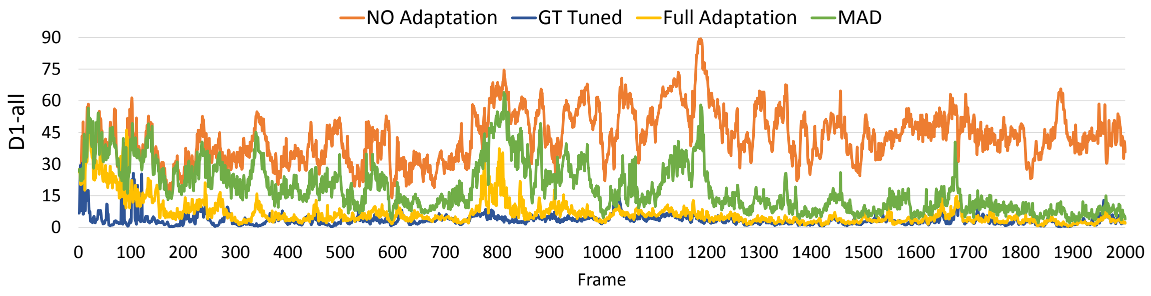


Figure 14.4: *MADNet*: error across frames in the *2011_ 09_ 30_ drive_ 0028_ sync* sequence (dataset, *Residential* environment).

level while providing a substantially higher inference rate (~ 25 vs ~ 15).

Long-term Adaptation. As Figure 14.4 hints, online adaptation delivers better performance processing a higher number of frames. In Table 14.3 we report additional results obtained by concatenating together the four environments without network resets to simulate a stereo camera traveling across different scenarios for ~ 43000 frames. Firstly, Table 14.3 shows how both DispNetC and *MADNet* models adapted online by full back-prop yield much smaller average errors than in Table 14.2, as small, indeed, as to outperform the corresponding models fine-tuned offline by ground truth labels. Hence, performing online adaptation through long enough sequences, even across different environments, can lead to more accurate models than offline fine-tuning on few samples with ground truth, which further highlights the great potential of our proposed *continuous learning* formulation. Moreover, when leveraging on *MAD* for the sake of run-time efficiency, *MADNet* attains larger accuracy gains through *continuous learning* than before (Table 14.3 vs. Table 14.2) shrinking the performance gap between *MAD* and Full back-prop. We believe that this observation confirms the results plotted in Figure 14.4: *MAD* needs more frame to adapt the network to a new environment, but given sequences long enough can successfully approximate full back propagation over time (i.e., 0.20 EPE difference and 1.2 D1-all between the two adaptation modalities on Table 14.3) while granting nearly twice FPS. On long term (e.g., beyond 1500 frames on Figure 14.4) running *MAD*, full adaptation or offline tuning on ground truth grants equivalent performance.

Model	Adapt.	D1-all(%)	EPE	FPS
DispNetC	No	9.09	1.58	15.85
DispNetC	Full	3.45	1.04	5.22
DispNetC-GT	No	4.40	1.21	15.85
MADNet	No	38.84	11.65	39.48
MADNet	Full	2.17	0.91	14.26
MADNet	*MAD*	3.37	1.11	25.43
MADNet-GT	No	2.67	0.89	39.48

Table 14.3: Results on the full KITTI raw dataset [67] (*Campus* → *City* → *Residential* → *Road*).

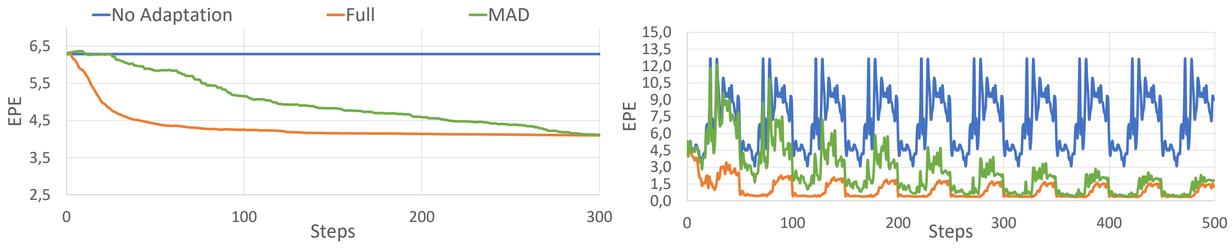


Figure 14.5: End-Point Error (EPE) on Middlebury *Motorcycle* pair (top) and Sintel *Alley-2* sequence (bottom) looped over 10 times.

14.3.4 Additional results

Here we show the generality of *MAD* on environments different from those depicted in the KITTI dataset. To this purpose, we run aimed experiments on the Sintel [27] and Middlebury [213] datasets and plot EPE trends for both *Full* and *Mad* adaptation on Figure 14.5. This evaluation allows for measuring the performance on a short sequence concatenated multiple times (i.e., Sintel) or when adapting on the same stereo pair (i.e., Middlebury) over and over.

On Middlebury (top) we perform 300 steps of adaptation on the *Motorcycle* stereo pair. The plots clearly show how *MAD* converges to the same accuracy of Full after around 300 steps while maintaining real-time processing (25.6 FPS on image scaled to a quarter of the original resolution). On Sintel (bottom), we adapt to the *Alley-2* sequence looped over 10 times. We can notice how the very few, i.e. 50, frames of the sequence are not enough to achieve good performance with *MAD*, since it performs the best on long-term adaptation as highlighted before. However, by looping over the same sequence,

Adaptation Mode	D1-all(%)	EPE	FPS
No	38.84	11.65	39.48
Last layer	38.33	11.45	38.25
Refinement	31.89	6.55	29.82
D_2+*Refinement*	18.84	2.87	25.85
MAD-SEQ	3.62	1.15	25.74
MAD-RAND	3.56 (±0.05)	1.13 (±0.01)	25.77
MAD-FULL	**3.37** (±0.1)	**1.11** (±0.01)	25.43

Table 14.4: Results on the raw dataset [67] using *MADNet* trained on synthetic data and different fast adaptation strategies

we can perceive how *MAD* gets closer to full adaptation, confirming the behavior already experimented on the KITTI environments.

14.3.5 Different online adaptation strategies

We carried out additional tests on the whole RAW dataset [67] and compared performance obtainable deploying different fast adaptation strategies for *MADNet*. Results are reported on Table 14.4 together with those concerning a network that does not perform any adaptation.

First, we compared *MAD* keeping the weights of the initial portions of the network frozen and training only: the last layer, the *Refinement* module or both D_2 and *Refinement* modules. Then, since *MAD* consists in splitting the network into independent portions and choosing which one to train, we compare our full proposal (MAD-*FULL*) to keeping the split and choosing the portion to train either randomly (MAD-*RAND*) or using a round-robin schedule (MAD-*SEQ*). Since MAD-*FULL* and MAD-*RAND* feature non-deterministic sampling steps, we report their average performance obtained across 5 independent runs on the whole dataset with the corresponding standard deviations between brackets.

By comparing the first four entries with the ones featuring *MAD* we can see how training only the final layers is not enough to successfully perform online adaptation. Even training as many as 13 last layers (i.e., $D_2 + Refinement$), at a computational cost comparable with *MAD*, we are at most able to halve the initial error rate, with performance still far from optimal. The three variants of *MAD* by training the whole network can successfully reduce the D1-all to $\frac{1}{10}$ of the original. Among the three options, our proposed layer selection heuristic provides the best overall performance even taking into account the slightly higher standard deviation caused by our sampling strategy. Moreover, the computational cost to pay to deploy our heuristic is negligible losing only 0.3 FPS compared to the other two options.

14.3.6 Deployment on embedded platforms

All the tests reported so far have been executed on a PC equipped with an NVIDIA 1080 Ti GPU. Unfortunately, for many application like robotics or autonomous vehicles, it is unrealistic to rely on such high end and power-hungry hardware. However, one of the key benefits of *MADNet* is its lightweight architecture conducive to easy deployment on low-power embedded platforms. Thus, we evaluated *MADNet* on an NVIDIA Jetson TX2 when processing stereo pairs at the full KITTI resolution and compared it to StereoNet [113] implemented using the same framework (i.e., the same level of optimization). We measured 0.26s for a single forward of *MADNet* versus 0.76-0.96s required by StereoNet, with 1 or 3 refinement modules respectively. Thus, for embedded applications *MADNet* is an appealing alternative to [113] since it is both faster and more accurate.

14.4 Conclusions and future work

The proposed online unsupervised fine-tuning approach can successfully tackle the domain adaptation issue for deep end-to-end disparity regression networks. We believe this to be key to practical deployment of these potentially ground-breaking deep learning systems in many relevant scenarios. For applications in which inference time is critical, we have proposed *MADNet*, a novel network architecture, and *MAD*, a strategy to effectively adapt it online very efficiently. We have shown how *MADNet* together with *MAD* can adapt to new environments by keeping a high prediction frame rate (i.e., 25FPS) and yielding better accuracy than popular alternatives like DispNetC. As main topic for future work, we plan to test and possibly extend *MAD* to any end-to-end stereo system. A first extension of this framework leverages proxy supervision obtained from traditional stereo algorithms in order to achieve better results [192]. We would also like to investigate alternative approaches to select the portion of the network to be updated online at each step.

Chapter 15

Guided stereo matching

The content of this chapter has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019) - “Guided Stereo Matching” [189].

15.1 Introduction

In this work, we propose to leverage a small set of sparse depth measurements to obtain, with deep stereo networks, dense and accurate estimations in any environment. It is worth pointing out that our proposal is different from depth fusion strategies (e.g., [6, 51, 145, 163]) aimed at combining the output of active sensors and stereo algorithms such as Semi-Global Matching [82]. Indeed, such methods mostly aim at selecting the most reliable depth measurements from the multiple available using appropriate frameworks whereas our proposal has an entirely different goal. In particular, given a deep network and a small set (e.g., less than 5% of the whole image points) of accurate depth measurements obtained by any means: can we improve the overall accuracy of the network without retraining? Can we reduce domain shift issues? Do we get better results training the network from scratch to exploit sparse measurements? To address these goals, we propose a novel technique acting at feature level and deployable with any state-of-the-art deep stereo network. Our strategy enhances the features corresponding to disparity hypotheses

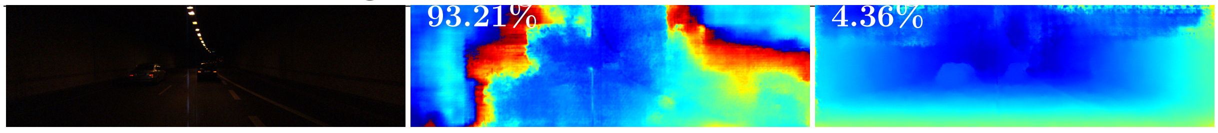


Figure 15.1: Guided stereo matching. (a) Challenging, reference image from KITTI 2015 [155] and disparity maps estimated by (b) iResNet [129] trained on synthetic data [151], or (c) guided by sparse depth measurements (5% density). Error rate (> 3) superimposed on each map.

provided by the sparse inputs maintaining the stereo reasoning capability of the original deep network. It is versatile, being suited to boost the accuracy of pre-trained models as well as to train a new instance from scratch to achieve even better results. Moreover, it can also be applied to improve the accuracy of conventional stereo algorithms like SGM. In all cases, our technique adds a negligible computational overhead to the original method. It is worth noting that active sensors, especially LiDAR-based, and standard cameras are both available as standard equipment in most autonomous driving setups. Moreover, since the cost of LiDARs is dropping and solid-state devices are already available [197], sparse and accurate depth measurement seems not to be restricted to a specific application domain. Thus, independently of the technology deployed to infer sparse depth data, to the best of our knowledge this work proposes the first successful attempt to leverage an external depth source to boost the accuracy of state-of-the-art deep stereo networks. We report extensive experiments conducted with two top-performing architectures with source code available, PSMNet by Chang et al. [31] and iResNet by Liang et al. [129], and standard datasets KITTI[66, 155], Middlebury 2014 [213] and ETH3D [215].

The outcome of such evaluation supports the three following main claims of this chapter:

- Given sparse ($< 5\%$ density) depth inputs, applying our method to pre-trained models always boosts its accuracy, either when the network is trained on synthetic data only or fine-tuned on the target environments.

- Training from scratch a network guided by sparse inputs dramatically increases its

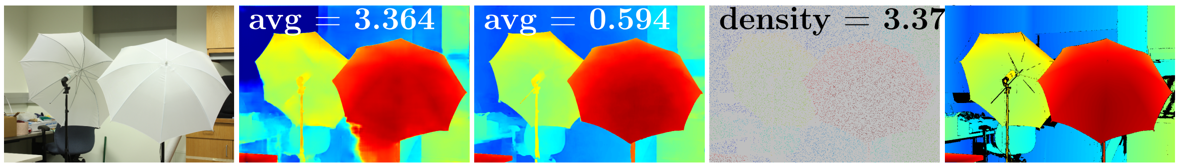


Figure 15.2: Example of improved generalization. (a) Reference image from Middlebury [213], disparity maps obtained by (b) iResNet [129] trained on SceneFlow synthetic dataset [151], (c) iResNet trained on SceneFlow for guided stereo, (d) visually enhanced sparse depth measurements taken from (e) ground truth depth. We stress the fact that (b) and (c) are obtained training on synthetic images only.

generalization capacity, significantly reducing the gap due to domain shifts (e.g., when moving from synthetic to real imagery).

- The proposed strategy can be applied seamlessly even to conventional stereo algorithms such as SGM.

In Figure 15.1 we can notice how on a very challenging stereo pair from KITTI 2015 [155] (a) a state-of-the-art deep stereo network trained on synthetic data produces inaccurate disparity maps (b), while guiding it with our method deploying only 5% of sparse depth data allows for much more accurate results (c) despite the domain shift.

15.2 Guided stereo matching

Given sparse, yet precise depth information collected from an external source, such as a LiDAR or any other means, our principal goal is to exploit such cues to assist state-of-the-art deep learning frameworks for stereo matching. For this purpose, we introduce a *feature enhancement* technique, acting directly on the intermediate features processed inside a CNN by peaking those directly related to the depth value suggested by the external measurement. This can be done by precisely acting where an equivalent representation of the cost volume is available. The primary goal of such an approach is to further increase the reliability of the already highly accurate disparity map produced by CNNs. Moreover, we also aim at reducing the issues introduced by domain shifts. By feeding sparse depth measurements into a deep network during training, we also expect that it can learn to exploit such information together with image content, compensating for domain shift if such measurements are fed to the network when moving to a completely different domain (e.g., from synthetic to real imagery). Our experiments will highlight how, following this strategy, given an extremely sparse distribution of values, we drastically improve the generalization capacity of a CNN. Figure 15.2 shows how deploying a 3.36% density of sparse depth inputs is enough to reduce the average error of iResNet from 3.364 to 0.594.

15.2.1 Feature enhancement

Traditional stereo algorithms collect into a cost volume the relationship between potentially corresponding pixels across the two images in a stereo pair, either encoding similarity or dissimilarity functions. The idea we propose consists in opportunely acting on such representation, still encoded within modern CNNs employing correlation or concatenation between features from the two images, favoring those disparities suggested by the sparse inputs. Networks following the first strategy [129, 151, 167, 225, 262] use a *correlation layer* to compute similarity scores that are higher for pixels that are most likely to match, while networks based on the second strategy rely upon a 3D volume of concatenated features. The cost volume of a conventional stereo algorithm has dimension $H \times W \times D$, with $H \times W$ being the resolution of the input stereo pair and D the maximum disparity displacement considered, while representative state-of-the-art deep stereo networks rely on data structures of dimension, respectively, $H \times W \times (2D+1)$ [151] and $H \times W \times D \times 2F$ [109], being F the number of features from a single image. Given sparse depth measurements z, we can easily convert them into disparities d by knowing the focal length f and baseline b of the setup used to acquire stereo pairs, as $d = b \cdot f \cdot \frac{1}{z}$.

With the availability of sparse external data in the disparity domain, we can exploit them to peak the correlation scores or the features activation related to the hypotheses suggested by these sparse hints and to dampen the remaining ones. For example, given

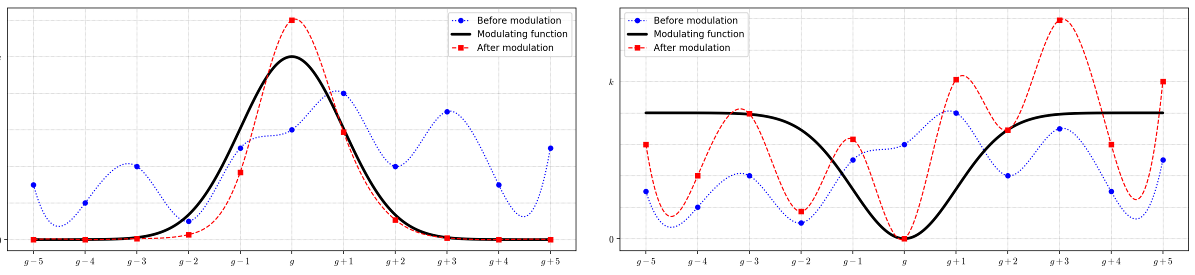


Figure 15.3: Application of the proposed feature enhancement. In blue, features $\mathcal{F}$ for pixel i, j in proximity of $d = g_{ij}$, in black the modulating function, in red enhanced features $\mathcal{G}$ for $v_{ij} = 1$, applied to correlation features (left) or dissimilarity functions (right).

a disparity value of k, we will enhance the k-*th* channel output of a correlation layer or the k-*th* slice of a 4D volume. For our purposes, we introduce two new inputs, both of size $H \times W$: a (sparse) matrix G, conveying the externally provided disparity values, and a binary mask V, specifying which elements of G are valid (i.e., if $v_{ij} = 1$). For each pixel with coordinates (i, j) in the reference image such that $v_{ij} = 1$ we alter features as discussed before, based on the known disparity value g_{ij}. On the other hand, every point with $v_{ij} = 0$ is left untouched. Thus, we rely on the ability of the deep network to reason about stereo and jointly leverage the additional information conveyed by sparse inputs.

In literature, some techniques were proposed to modify the cost volume of traditional stereo algorithms leveraging prior knowledge such as per-pixel confidence scores [184]. A simple, yet effective approach for this purpose consists in hard-replacing matching costs (features, in our case). In [228], matching costs of winning disparities were set to the minimum value and the remaining ones to the maximum, only for those pixels having high confidence score before optimization. The equivalent solution in our domain would consist in zeroing each element corresponding to a disparity d such that $g_{ij} \neq d$. However, this strategy has severe limitations: it is not well-suited for CNNs, either when injected into a pre-trained network – a large number of zero values would aggressively alter its behavior – or when plugged during the training from scratch of a new model – this would cause gradients to not be back-propagated on top of features where the zeros have been inserted. Moreover, no default behavior is defined in case of sub-pixel input disparities, unless they are rounded at the cost of a loss in precision.

Conversely, we suggest to modulate using a Gaussian function centred on g_{ij}, so that the single correlation score or $2F$ features corresponding to the disparity $d = g_{ij}$ are multiplied by the peak of the function, while any other element is progressively multiplied by lower factors, until being dampened the farther they are from g_{ij}. Specifically, our modulating function will be

$$k \cdot e^{-\frac{(d-g_{ij})^2}{2c^2}} \tag{15.1}$$

where c determines the width of the Gaussian, while k represents its maximum magnitude and should be greater than or equal to 1. Thus, to obtain a new feature volume $\mathcal{G}$ by multiplying the whole correlation or 3D volume $\mathcal{F}$ regardless of the value of v_{ij}, we apply

$$\mathcal{G} = \left(1 - v_{ij} + v_{ij} \cdot k \cdot e^{-\frac{(d-g_{ij})^2}{2c^2}}\right) \cdot \mathcal{F} \tag{15.2}$$

making the weight factor equal to 1 when $v_{ij} = 0$. An example of the effect of our modulation is given in Figure 15.3 (left).

15.2.2 Applications of guided stereo

We will now highlight some notable applications of our technique, that will be exhaustively discussed in the experimental result section.

Pre-trained deep stereo networks. The proposed Gaussian enhancement acts smoothly, yet effectively, on the features already learned by a deep network. Opportunely tuning the hyper-parameters k and c, we will prove that our method allows improving the accuracy of pre-trained state-of-the-art networks.

Training from scratch deep stereo networks. Compared to a brutal zero-product approach, the dampening mechanism introduced by the Gaussian function still allows gradient to flow, making this technique suited for deployment inside a CNN even at training time, so that it can learn from scratch how to better exploit the additional cues. Specifically, the gradients of $\mathcal{G}$ with respect to weights $\mathcal{W}$ will be computed as follows:

$$\frac{\partial \mathcal{G}}{\partial \mathcal{W}} = \left(1 - v_{ij} + v_{ij} \cdot k \cdot e^{-\frac{(d-g_{ij})^2}{2c^2}}\right) \cdot \frac{\partial \mathcal{F}}{\partial \mathcal{W}} \tag{15.3}$$

Thus, training from scratch a deep network leveraging the sparse input data is possible with our technique.

Conventional stereo matching algorithms. These methods, based on hand-crafted pipelines, can also take advantage of our proposal by leveraging sparse depth cues to improve their accuracy. Sometimes they do not use a similarity measure (e.g., zero mean normalized cross-correlation) to encode the matching cost, for which the same strategy described so far applies, but cost volumes are built using a dissimilarity measure between pixels (e.g., sum of absolute/squared differences or Hamming distance [276]). In both cases, the winning disparity is assigned employing a WTA strategy. When deploying a dissimilarity measure, costs corresponding to disparities close to g_{ij} should be reduced, while the others amplified. We can easily adapt Gaussian enhancement by choosing a modulating function that is the difference between a constant k and a Gaussian function with the same height, obtaining an enhanced volume $\mathcal{G}$ from initial costs $\mathcal{F}$ as

$$\mathcal{G} = \left[1 - v_{ij} + v_{ij} \cdot k \cdot \left(1 - e^{-\frac{(d-g_{ij})^2}{2c^2}}\right)\right] \cdot \mathcal{F} \tag{15.4}$$

Figure 15.3 (right) shows the effect of this formulation.

15.3 Experimental Results

In this section, we report exhaustive experiments proving the effectiveness of the Guided Stereo Matching paradigm showing that the proposed feature enhancement strategy always improves the accuracy of pre-trained or newly trained networks significantly. More-

	iResNet [129]			PSMNet [31]		
c	k=1	k=10	k=100	k=1	k=10	k=100
0.1	2.054	1.881	2.377	4.711	4.391	4.326
1	1.885	**1.338**	6.857	4.540	**3.900**	4.286
10	1.624	1.664	32.329	4.539	3.925	9.951

Table 15.1: Tuning of Gaussian hyper-parameters k and c. Experiments with iResNet (left) and PSMNet (right) trained on synthetic data and tested on KITTI 2015 (average errors without modulation: 1.863 and 4.716)

Model	Training Datasets		Guide		Error rate (%)				avg.
	SceneFlow	KITTI 12	Train	Test	>2	>3	>4	>5	(px)
iResNet [129]	✓				21.157	11.959	7.881	5.744	1.863
iResNet-*gd*	✓			✓	15.146	8.208	5.348	3.881	1.431
iResNet-*gd-tr*	✓		✓	✓	7.266	3.663	2.388	1.754	0.904
iResNet-*ft* [129]	✓	✓			9.795	4.452	2.730	1.938	1.049
iResNet-*ft-gd*	✓	✓		✓	7.695	3.812	2.524	1.891	0.994
iResNet-*ft-gd-tr*	✓	✓	✓	✓	**4.577**	**2.239**	**1.476**	**1.099**	**0.735**
PSMNet [31]	✓				39.505	27.435	20.844	16.725	4.716
PSMNet-*gd*	✓			✓	33.386	23.125	17.598	14.101	3.900
PSMNet-*gd-tr*	✓		✓	✓	12.310	3.896	2.239	1.608	1.395
PSMNet-*ft* [31]	✓	✓			6.341	3.122	2.181	1.752	1.200
PSMNet-*ft-gd*	✓	✓		✓	5.707	3.098	2.266	1.842	1.092
PSMNet-*ft-gd-tr*	✓	✓	✓	✓	**2.738**	**1.829**	**1.513**	**1.338**	**0.763**

Table 15.2: Experimental results on KITTI 2015 dataset [155]. Tag "*-gd*" refers to guiding the network only at test time, "*-tr*" to training the model to leverage guide, "*-ft*" refers to fine-tuning performed on KITTI 2012 [66].

over, when training the networks from scratch, our proposal increases the ability to generalize to new environments, thus enabling to better tackle domain shifts.

15.3.1 Training and validation protocols

We implemented our framework in PyTorch. For our experiments, we chose two state-of-the-art models representative of the two categories described so far and whose source code is available, respectively iResNet [129] for correlation-based architectures and PSMNet [31] for 3D CNNs. Both networks were pre-trained on synthetic data [151] following authors' instructions: 10 epochs for PSMNet [31] and 650k iterations for iResNet [129]. The only difference was the batch size of 3 used for PSMNet since it is the largest fitting in a single Titan Xp GPU used for this research. The proposed guided versions of these networks were trained accordingly following the same protocol. Fine-tuning on realistic datasets was carried out following the guidelines from the original works when available. In particular, both papers reported results and a detailed training protocol for KITTI datasets [31, 129], while training details are not provided for Middlebury [213] and ETH3D [215], despite results are present on both benchmarks. The following sections will report accurate details about each training protocol deployed in our experiments. To tune k and c, we ran a preliminary experiment applying our modulation on iResNet and PSMNet models trained on synthetic data and tested on KITTI 2015 training set [155]. Table 19.1 shows how the average error varies with different values of k and c. According to this outcome, for both networks we will fix k and c, respectively, to 10 and 1 for all the following experiments.

To simulate the availability of sparse depth cues, we randomly sample pixels from the ground truth disparity maps for both training and testing. For this reason, all the evaluation will be carried out on the training splits available from KITTI, Middlebury and ETH3D datasets. Finally, we point out that the KITTI benchmarks include a depth completion evaluation. However, it aims at assessing the performance of monocular camera systems coupled with an active sensor (i.e., LiDAR) and thus the benchmark does not provide the stereo pairs required for our purposes.

15.3.2 Evaluation on KITTI

At first, we assess the performance of our proposal on the KITTI 2015 dataset [155]. Table 17.2 collects results obtained with iResNet and PSMNet trained and tested in different configurations. For each experiment we highlight the imagery used during training, respectively the SceneFlow dataset alone [151] or the KITTI 2012 [66] used for fine-tuning ("*-ft*"). Moreover, we report results applying our feature enhancement to pre-trained networks (i.e., at test time only, "*-gd*") and training the networks from scratch ("*-gd-tr*"). For each experiment, we report the error rate as the percentage of pixels having a disparity error larger than a threshold, varying between 2 and 5, as well as the mean average error on all pixels with available ground truth. To obtain sparse measurements, we randomly

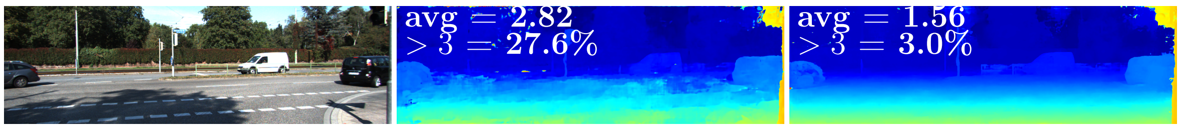


Figure 15.4: Comparison between variants of PSMNet [31]. From top to bottom, reference image from 000022 pair (KITTI 2015 [155]), disparity maps obtained by PSMNet [31] and PSMNet-*gd-tr*, both trained on synthetic images only.

sample pixels with a density, computed on the whole image, of 5% on SceneFlow [151]. On KITTI, we keep a density of 15%, then remove unlabelled pixels to obtain again 5% with respect to the lower portion of the images with available ground truth (i.e., a 220×1240 pixel grid).

From Table 17.2 we can notice how both baseline architectures (row 1 and 7) yield large errors when trained on SceneFlow dataset only. In particular, PSMNet seems to suffer the domain shift more than correlation-based technique iResNet. By applying the proposed feature enhancement to both networks, we can ameliorate all metrics sensibly, obtaining first improvements to network generalization capability. In particular, by looking at the > 3 error rate, usually taken as the reference metric in KITTI, we have an absolute reduction of about 3.8 and 4.3 % respectively for iResNet-*gd* and PSMNet-*gd* compared to the baseline networks. In this case, we point out once more that we are modifying only the features of a pre-trained network, by just altering what the following layers are used to process. Nonetheless, our proposal preserves the learned behavior of the baseline architecture increasing at the same time its overall accuracy.

When training the networks from scratch to process sparse inputs with our technique, iResNet-*gd-tr* and PSMNet-*gd-tr* achieve a notable drop regarding error rate and average error compared to the baseline models. Both reach degrees of accuracy comparable to those of the original network fine-tuned on KITTI 2012, iResNet-*ft* and PSMNet-*ft* without actually being trained on such realistic imagery, by simply exploiting a small amount of depth samples (about 5%) through our technique. Moreover, we can also apply the feature enhancement paradigm to fine-tuned models. From Table 17.2 we can notice

Model	Training Datasets		Guide		Error rate (%)				avg.
	SceneFlow	trainingQ	Train	Test	>0.5	>1	>2	>4	(px)
iResNet [129]	✓				69.967	50.893	30.742	16.019	2.816
iResNet-*gd*	✓			✓	62.581	40.831	22.154	10.889	2.211
iResNet-*gd-tr*	✓		✓	✓	44.385	25.555	12.505	5.776	1.470
iResNet-*ft* [129]	✓	✓			69.526	49.027	28.178	14.126	2.682
iResNet-*ft-gd*	✓	✓		✓	60.979	36.255	19.558	10.136	2.130
iResNet-*ft-gd-tr*	✓	✓	✓	✓	**31.526**	**17.045**	**8.316**	**4.307**	**0.930**
PSMNet [31]	✓				54.717	33.603	20.239	13.304	5.332
PSMNet-*gd*	✓			✓	53.090	31.416	18.619	12.588	4.921
PSMNet-*gd-tr*	✓		✓	✓	83.433	54.147	7.472	3.208	1.732
PSMNet-*ft* [31]	✓	✓			45.523	25.993	15.203	8.884	1.964
PSMNet-*ft-gd*	✓	✓		✓	44.004	25.151	14.337	8.676	1.894
PSMNet-*ft-gd-tr*	✓	✓	✓	✓	**32.715**	**15.724**	**6.937**	**3.756**	**1.348**

Table 15.3: Experimental results on Middlebury 2014 [213]. "*-gd*" refers to guiding the network only at test time, "*-tr*" to training the model to leverage guide, "*-ft*" refers to fine-tuning performed on *trainingQ* split.

Model	Training Datasets		Guide		Error rate (%)				avg.
	SceneFlow	ETH3D	Train	Test	>0.5	>1	>2	>4	(px)
iResNet [129]	✓				57.011	36.944	20.380	12.453	5.120
iResNet-*gd*	✓			✓	50.361	29.767	16.495	10.293	2.717
iResNet-*gd-tr*	✓		✓	✓	26.815	10.673	3.555	1.312	0.537
iResNet-*ft* [129]	✓	✓			48.360	26.212	11.865	4.678	0.997
iResNet-*ft-gd*	✓	✓		✓	47.539	22.639	8.153	2.445	0.820
iResNet-*ft-gd-tr*	✓	✓	✓	✓	**23.433**	**8.694**	**2.803**	**0.876**	**0.443**
PSMNet [31]	✓				45.522	23.936	12.550	7.811	5.078
PSMNet-*gd*	✓			✓	43.667	21.140	10.773	7.081	4.739
PSMNet-*gd-tr*	✓		✓	✓	96.976	71.970	2.730	0.512	1.266
PSMNet-*ft* [31]	✓	✓			28.560	11.895	4.272	1.560	0.560
PSMNet-*ft-gd*	✓	✓		✓	25.707	10.095	3.084	1.123	0.505
PSMNet-*ft-gd-tr*	✓	✓	✓	✓	**17.865**	**4.195**	**1.360**	**0.817**	**0.406**

Table 15.4: Experimental results on ETH3D dataset [215]. "*-gd*" refers to guiding the network only at test time, "*-tr*" to training the model to leverage guide, "*-ft*" refers to fine-tuning performed on the training split (see text).

again how our technique applied to the fine-tuned models still improves their accuracy. Nonetheless, fine-tuning the networks pre-trained to exploit feature enhancement leads to the best results across all configurations, with an absolute decrease of about 2.2 and 1.3% compared, respectively, to the already low error rate of iResNet-*ft* and PSMNet-*ft*. Finally, Figure 15.4 shows a comparison between the outputs of different PSMNet variants, highlighting the superior generalization capacity of PSMNet-*gd-tr* compared to baseline model.

15.3.3 Evaluation on Middlebury

We also evaluated our proposal on Middlebury 2014 [213], since this dataset is notoriously more challenging for end-to-end architectures because of the very few images available for fine-tuning and the more heterogeneous scenes framed compared to KITTI. Table 15.3 collects the outcome of these experiments. We use the same notation adopted for KITTI experiments. All numbers are obtained processing the *additional* split of images at quarter resolution, since higher-resolution stereo pairs do not fit into the memory of a single Titan Xp GPU. Fine-tuning is carried out on the *training* split. We compute error rates with thresholds 0.5, 1, 2 and 4 as usually reported on the online benchmark. Sparse inputs are randomly sampled with a density of 5% from ground truth data. We can notice how applying feature enhancement on both pre-trained models or training new instances from scratch gradually reduces the errors as observed on KITTI experiments. Interestingly, we point out that while this trend is consistent for iResNet-*gd* and iResNet-*gd-tr*, a different behavior occurs for PSMNet-*gd-tr*. In particular, we can notice a huge reduction of the error rate setting the threshold to > 2 and > 4. On the other hand, the percentage of pixels with lower disparity errors > 0.5 and > 1 gets much higher. Thus, with PSMNet, the architecture trained with guiding inputs seems to correct most erroneous patterns at the cost of increasing the number of small errors. Nonetheless, the average error always decreases significantly.

Regarding fine-tuning, we ran about 300 epochs for each baseline architecture to obtain the best results. After this phase, we can observe minor improvements for iResNet, while PSMNet improves its accuracy by a large margin. Minor, but consistent improvements are yielded by iResNet-*ft-gd* and PSMNet-*ft-gd*. Finally, we fine-tuned iResNet-*ft-gd-tr* and PSMNet-*ft-gd-tr* for about 30 epochs, sufficient to reach the best performance. Again, compared to all other configurations, major improvements are always yielded by guiding both networks.

15.3.4 Evaluation on ETH3D

Finally, we assess the performance of our method on the ETH3D dataset [215]. In this case we split the training dataset, using images from *delivery_ area_ 1l*, *delivery_ area_ 1s*, *electro_ 1l*, *electro_ 1s*, *facade_ 1s*, *forest_ 1s*, *playground_ 1l*, *playground_ 1s*, *terrace_ 1s*, *terrains_ 1l*, *terrains_ 1s* for fine-tuning and the remaining ones for testing. For *-ft* models, we perform the same number of training epochs as for Middlebury dataset, and Table 15.4 collects results for the same configurations considered before. We can notice behaviors similar to what reported in previous experiments. By guiding iResNet we achieve major improvements, nearly halving the average error, while the gain is less evident for PSMNet, although beneficial on all metrics. Training iResNet-*gd-tr* and PSMNet-*gd-tr* leads to the same outcome noticed during our experiments on Middlebury. In particular, PSMNet-*gd-tr* still decimates the average error and percentage of errors greater than 2 and 4 at the cost of a large amount of pixels with error greater than 0.5 and 1. With this dataset, fine-tuning the baseline models enables to significantly increase the accuracy of both, in particular decimating the average error from beyond 5 pixels to less than 1. Nevertheless, our technique is useful even in this case, enabling minor yet consistent improvements when used at test time only and significant boosts for iResNet-*ft-gd-tr* and PSMNet-*ft-gd-tr*, improving all metrics by a large margin.

15.3.5 Evaluation with SGM

To prove the effectiveness of our proposal even with conventional stereo matching algorithms, we evaluated it with SGM [82]. For this purpose, we used the code provided by [226], testing it on all datasets considered so far. As pointed out in Section 15.2.2, feature enhancement can be opportunely revised as described in Equation 21.6 to deal with dissimilarity measures. We apply this formulation before starting scanline optimizations. As for any previous experiments, we sample sparse inputs as described in Section 15.3, obtaining an average density below 5%. Table 15.5 reports the comparison between SGM

Alg.	Error rate (%)				avg.
	>2	>3	>4	>5	(px)
SGM [82]	11.845	8.553	7.109	6.261	2.740
SGM-*gd*	**5.657**	**4.601**	**4.162**	**3.892**	**2.153**
SGM [82]	15.049	8.843	6.725	5.645	2.226
SGM-*gd*	**6.753**	**4.294**	**3.625**	**3.282**	**1.680**

Table 15.5: Experimental results on KITTI. Comparison between raw [82] and guided SGM on KITTI 2012 (top) and 2015 (bottom).

Alg.	Error rate (%)				avg.
	>0.5	>1	>2	>4	(px)
SGM [82]	62.428	32.849	20.620	15.786	4.018
SGM-*gd*	**56.882**	**24.608**	**12.655**	**9.909**	**2.975**
SGM [82]	64.264	31.966	18.741	14.675	4.978
SGM-*gd*	**59.596**	**24.856**	**11.307**	**8.960**	**3.815**
SGM [82]	58.994	27.356	10.685	5.632	1.433
SGM-*gd*	**54.051**	**20.156**	**4.169**	**2.459**	**1.032**

Table 15.6: Experimental results on Middlebury 2014 and ETH3D datasets. Comparison between raw [82] and guided SGM on *training* (top), *additional* (middle) [213] and ETH3D [215] (bottom).

and its *cost enhanced* counterpart using sparse input cues (SGM-*gd*) on KITTI 2012 [66] (top) and KITTI 2015 [155] (bottom). With both datasets we can notice dramatic improvements in all metrics. In particular, the amount of outliers > 2 is more than halved and reduced in absolute by about 4, 3 and 2% for higher error bounds. Table 15.6 reports experiments on Middlebury *training* (top) and *additional* (bottom) splits [213], as well as on the entire ETH3D training set [215]. Experiments on Middlebury are carried out at quarter resolution for uniformity with previous experiments with deep networks reported in Section 15.3.3. The error rate is reduced by about 5.5% for > 0.5 on the three experiments, by about 7.5% for > 1, nearly halved on Middlebury and reduced by a factor 2.5 on ETH3D for > 2, and by 6, 6 and 3% for > 4. Finally, average errors are reduced by 1.1 on Middlebury and 1.4 on ETH3D.

The evaluation with SGM highlights how our technique can be regarded as a general purpose strategy enabling notable improvements in different contexts, ranging from state-of-the-art deep learning frameworks to traditional stereo algorithms.

Model / Alg.	<2%		avg.	
	All	NoG	All	NoG
iResNet [129]	18.42	18.37	1.28	1.28
iResNet+Martins et al.[146]	18.14	18.09	1.26	1.26
iResNet+Marin et al.(opt.)	15.20	18.37	1.07	1.28
iResNet-*gd*	11.12	10.99	1.04	1.03
iResNet-*gd-tr*	**5.38**	**5.27**	**0.77**	**0.77**
iResNet-*ft* [129]	5.29	5.30	0.81	0.81
iResNet-*ft*+Martins et al.[146]	5.26	5.28	0.80	0.80
iResNet-*ft*+Marin et al.[145] (opt.)	4.48	5.30	0.67	0.81
iResNet-*ft-gd*	3.14	3.13	0.64	0.64
iResNet-*ft-gd-tr*	**1.91**	**1.88**	**0.55**	**0.55**
PSMNet [31]	38.60	38.86	2.36	2.37
PSMNet+Martins et al.[146]	38.32	38.58	2.33	2.34
PSMNet+Marin et al.[145] (opt.)	34.85	38.86	1.99	2.17
PSMNet-*gd*	33.47	33.74	2.07	2.08
PSMNet-*gd-tr*	**21.57**	**21.30**	**1.60**	**1.59**
PSMNet-ft [31]	1.71	1.73	0.72	0.72
PSMNet-*ft*+Martins et al.[146]	1.82	1.83	0.72	0.72
PSMNet-*ft*+Marin et al.[145] (opt.)	1.52	1.73	0.66	0.72
PSMNet-*ft-gd*	1.13	1.15	0.60	0.61
PSMNet-*ft-gd-tr*	**0.67**	**0.67**	**0.47**	**0.47**
SGM [82]	9.42	9.54	1.24	1.24
SGM+Martins et al.[146]	9.41	9.53	1.24	1.24
SGM+Marin et al.[145] (opt.)	8.15	9.54	1.14	1.24
SGM-*gd*	**2.79**	**3.03**	**0.99**	**0.99**

Table 15.7: Experiments on KITTI Velodyne, seq. *2011_09_26_0011*.

15.3.6 Experiments with Lidar measurements

Finally, we evaluate the proposed paradigm using as guide the raw and noisy measurements from a Velodyne sensor [248], to underline the practical deployability of the proposed solution further. Table 15.7 reports experiment from sequence 2011_09_26_0011 of the KITTI raw dataset [67]. We compare our framework with fusion strategies proposed by Martins et al.[146] and Marin et al.[145], combining outputs by the stereo networks respectively with monocular estimates (using the network by Guo et al.[74]) and Lidar, reporting the ideal result as in [145]. Ground truth labels for evaluation are provided by [246]. Our proposal consistently outperforms fusion approaches by a large margin, evaluating on all pixels (All) as well as excluding those with Lidar (NoG) to stress that the improvement yielded by our method is not limited to pixels with associated Lidar measurement in contrast to fusion techniques [145].

15.4 Conclusions

In this chapter, we proposed Guided Stereo Matching, a novel paradigm to boost state-of-the-art deep architectures trained for dense disparity inference using as additional input cue a small set of sparse depth measurements provided by an external source. By enhancing the features that encode matching relationships between pixels across left and right images, we can improve the accuracy and robustness to domain shifts. Our feature enhancement strategy can be used seamlessly with pre-trained models, yielding significant accuracy improvements. More importantly, thanks to its fully-differentiable nature, it can even be used to train new instances of a CNN from scratch, in order to fully exploit the input guide and thus to remarkably improve overall accuracy and robustness to domain shifts of deep networks. Finally, our proposal can be deployed even with conventional stereo matching algorithms such as SGM, yielding significant improvements as well. The focus of future work will be on devising strategies to guide our method without relying on active sensors. For instance, selecting reliable depth labels leveraging confidence measures [184] – since this strategy proved to be successful for self-supervised adaptation [235, 238] and training learning-based confidence measures [239] – or from the output of a visual stereo odometry systems [252].

Chapter 16

Reversing the cycle: self-supervised deep stereo through enhanced monocular distillation

The content of this chapter has been presented at the European Conference on Computer Vision (ECCV 2020) - "Reversing the cycle: self-supervised deep stereo through enhanced monocular distillation" [11].

16.1 Introduction

In recent years, single-image depth estimation methods, in general up to a scale factor, gained ever-increasing attention. In this field, despite the *ill-posed* nature of the problem, deep learning architectures achieved outstanding results as reported in the literature. By construction, a monocular method does not infer depth by matching points between different views of the same scene. Therefore, compared to stereo approaches, monocular ones infer depth relying on different cues and thus potentially not affected by some inherent issues of stereo, such as occlusions. Even if supervision is sourced from stereo images [69], a set of practices suited for the specific monocular task allow networks to avoid undesired artifacts in correspondence of occlusions [70]. Starting from these observations, we argue that a single image method could potentially strengthen a stereo one, especially in occluded areas, but it would suffer the inherent scale factor issue. Purposely, in this chapter we prove that traditional stereo methods and monocular cues can be effectively deployed jointly in a *monocular completion network* able to alleviate both problems, and thus beneficial to obtain accurate and robust depth predictions.

Our contributions can be summarized as follows: i) A new general-purpose methodology to source accurate disparity annotations in a self-supervised manner given a stereo dataset without additional data from active sensors. To the best of our knowledge, our proposal is the first leveraging at training time a novel self-supervised monocular completion network aimed specifically at ameliorate annotations in critical regions such as occluded areas. ii) In order to reduce as much as possible inconsistent disparity annotations, we propose a novel consensus mechanism over multiple predictions exploiting input randomness of the monocular completion network. iii) The generated proxies are dense and accurate even if we do not rely on any active depth sensor (e.g. LiDAR). iv) Our proxies allow for training heterogeneous deep stereo networks outperforming self-supervised state-of-the-art strategies on KITTI. Moreover, the networks trained with our method show higher generalization to unseen environments.

16.2 Method

This section describes our strategy in detail, that allows us to distill highly accurate disparity annotations for a stereo dataset made up of raw rectified images only and then use them to supervise deep stereo networks. It is worth noting that, by abuse of notation, we use depth and disparity interchangeably although our proxy extraction method works entirely in the disparity domain. For our purposes, we rely on two main stages, as depicted in Fig. 24.3: 1) we train a monocular completion network (MCN) from sparse disparity points sourced by traditional stereo methods and 2) we train deep stereo networks using

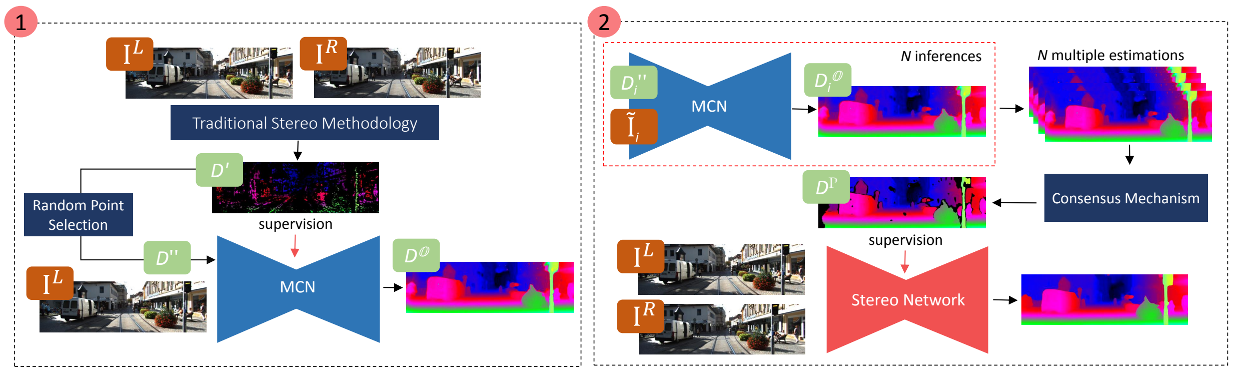


Figure 16.1: Overview of our methodology. ① Sparse disparity points from a traditional stereo method are given as input to a monocular completion network (MCN). Then, in ② we leverage MCN to distill accurate proxies through the proposed consensus mechanism. Such labels guide the training of a deep stereo network.

highly reliable points from MCN, selected by a novel consensus mechanism.

16.2.1 Monocular Completion Network (MCN)

Stereo algorithms struggle on occluded regions due to the difficulties to find correspondences between images. On the contrary, monocular methods do not rely on matching and thus, they are potentially not affected by this problem. In this stage, our goal is to obtain a strong guidance even on occluded areas relying on a monocular depth network. However, monocular estimates intrinsically suffer the scale factor ambiguity due to the lack of geometric constraints. Therefore, since stereo pairs are always available in our setup, we also leverage on reliable sparse disparity input points from traditional stereo algorithms in addition to the reference image. Thanks to this combination, MCN is able to predict dense depth maps preserving geometrical information.

Reliable disparity points extraction. At first, we rely on a traditional stereo matcher $\mathcal{S}$ (e.g. [276]) to obtain an initial disparity map $\mathcal{D}_L$ from a given stereo pair $(\mathcal{I}_L, \mathcal{I}_R)$ as

$$\mathcal{D}_L = \mathcal{S}(\mathcal{I}_L, \mathcal{I}_R) \tag{16.1}$$

However, since such raw disparity map contains several outliers, especially on ill-posed

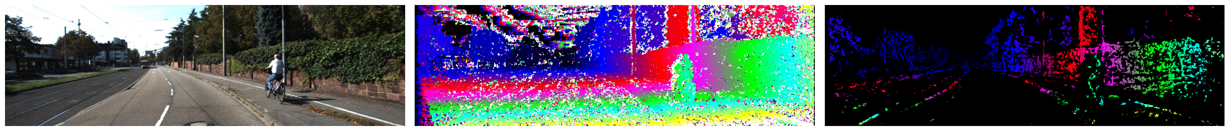


Figure 16.2: Disparity map filtering. From left to right, reference image from KITTI, the noisy disparity map computed by [276] and the outcome of filtering [239].

regions such as occlusions or texture-less areas as it can be noticed in Fig. 16.2, a filtering strategy $\mathcal{F}$ (e.g. [239]) is applied to discard spurious points

$$\mathcal{D}'_L = \mathcal{F}(\mathcal{S}(\mathcal{I}_L, \mathcal{I}_R)) \tag{16.2}$$

By doing so, only a subset $\mathcal{D}'_L$ of highly reliable points is preserved from $\mathcal{D}_L$ at the cost of a sparser disparity map. However, most of them do not belong to occluded regions thus not enabling supervision on such areas. This can be clearly perceived observing the outcome of a filtering strategy in Fig. 16.2.

Monocular Disparity Completion. Given $\mathcal{D}'_L$, we deploy a monocular completion network, namely MCN, in order to obtain a dense map $\mathcal{D}^{\mathbb{O}}$. We self-supervise MCN from stereo and, as in [70], to handle occlusions we horizontally flip $(\mathcal{I}_L, \mathcal{I}_R)$ at training time with a certain probability without switching them. Consequently, occluded regions (e.g. the left border of objects) are randomly swapped with not-occluded areas (e.g. the right borders), preventing to always expect high error on left and low error on right borders, thus forcing the network to handle both. This strategy turns out ineffective in case of self-supervised stereo, since after horizontal flip the stereo pair have to be switched in order to keep the same search direction along the epipolar line, thus making occlusions occur in the same regions. Even if this technique helps to alleviate errors in occluded regions, a pure monocular network struggles compared to a stereo method at determining the correct depth. This is well-known in the literature and shown in our experiments as well. Thus, we adopt a completion approach leveraging sparse reliable points provided by a traditional stereo method constraining the predictions to be properly scaled. Given the set of filtered points, only a small subset $\mathcal{D}''_L$, with $||\mathcal{D}''_L|| \ll ||\mathcal{D}'_L||$, is randomly selected and used as input, while $\mathcal{D}'_L$ itself is used for supervision purposes. The output of MCN is defined as

$$\mathcal{D}^{\mathbb{O}} = \text{MCN}(\mathcal{I}_L, \mathcal{D}''_L \xleftarrow{p \ll 1} (\mathcal{D}'_L)) \tag{16.3}$$

with $x \xleftarrow{p} (y)$ a random uniform sampling function extracting x values out of y per-pixel values with probability p. This sampling is crucial to both improve MCN accuracy, as shown in our experiments, as well as for the final distillation step discussed in the remainder. Once trained, MCN is able to infer scaled dense disparity maps $\mathcal{D}^{\mathbb{O}}$, as can be perceived in Fig. 16.3. Looking at the rightmost and central disparity maps, we can notice how the augmentation protocol enables to alleviate occlusion artifacts. Moreover, our overall completion strategy, compared to the output of the monocular network without disparity seeds (leftmost and center disparity maps), achieves much higher accuracy as well as correctly handles occlusions. Therefore, we effectively combine stereo from non-occluded regions and monocular prediction in occluded areas. Finally, we point out that we aim at specializing MCN on the training set to generate labels on it since its purpose is limited to distillation.

16.2.2 Proxy Distillation for Deep Stereo

Eventually, we leverage the trained MCN to distill offline proxy labels beneficial to supervise stereo networks. However, such data might still contain some inconsistent predictions, as can be perceived in the rightmost disparity map of Fig. 16.3. Therefore, our goal is to discard them, keeping trustworthy reliable depth estimates to train deep stereo networks.

Consensus mechanism and distillation. To this aim, given an RGB image $\mathcal{I}$ and the relative $\mathcal{D}'_L$, we perform N inferences of MCN by feeding it with $\mathcal{D}''_{Li}$ and $\tilde{\mathcal{I}}_i$, with $i \in [1, N]$. Respectively, $\mathcal{D}''_{Li}$ is sampled from $\mathcal{D}'_L$ according to the strategy introduced in Sec. 3.1 and $\tilde{\mathcal{I}}_i$ is obtained through random augmentation (explained later) applied to

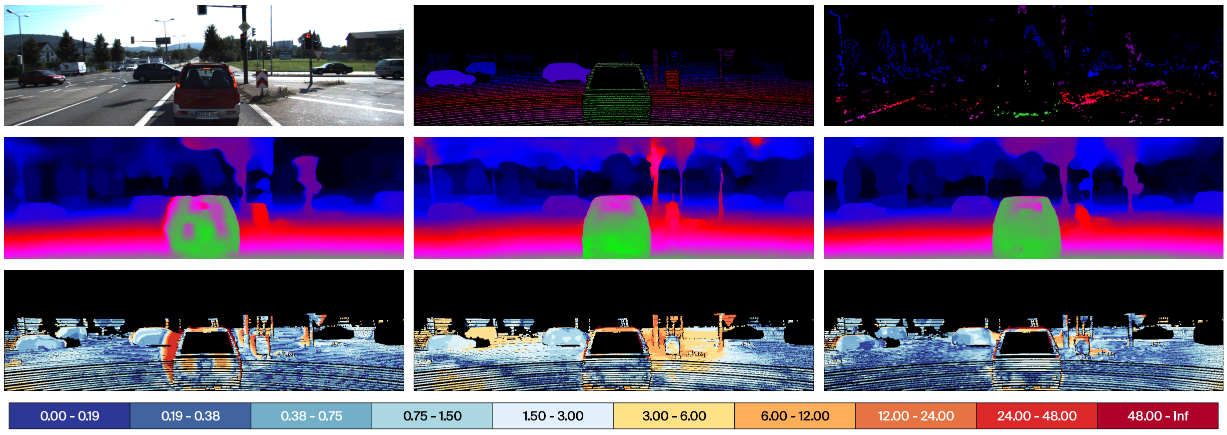


Figure 16.3: Occlusion handling and scale recovery. The first row depicts the reference image from KITTI, the ground truth and the disparity map by [276] filtered with [239]. In the middle, from left to right the output of monocular depth network [241] trained without occlusion augmentation, the same network using the occlusion augmentation and our MCN. Last row shows the corresponding error maps. Best viewed with colors.

$\mathcal{I}$. This way, we exploit consistencies and contradictions among multiple $\mathcal{D}_i^{\mathbb{O}}$ to obtain reliable proxy labels $\mathcal{D}^{\mathbb{P}}$, defined as

$$\mathcal{D}^{\mathbb{P}} \xleftarrow{\sigma^2(\{\mathcal{D}_i^{\mathbb{O}}\}_{i=1}^{N})<\gamma} \mu(\{\mathcal{D}_i^{\mathbb{O}}\}_{i=1}^{N}) \tag{16.4}$$

where $x \xleftarrow{\sigma^2(y)<\gamma} \mu(y)$ is a function that, given N values y for the same pixel, samples the mean value $\mu(y)$ only if the variance $\sigma^2(y)$ is smaller γ. Being distillation performed offline, this step does not need to be differentiable.

Fig. 16.4 shows that such a strategy allows us to largely regularize $\mathcal{D}^{\mathbb{P}}$ compared to $\mathcal{D}^{\mathbb{O}}$, preserving thin structures, e.g. the poles on the right side, yet achieving high density. It also infers significant portions of occluded regions compared to proxies sourced from traditional methods (e.g. SGM).

Deep Stereo Training. Once highly accurate proxy labels $\mathcal{D}^{\mathbb{P}}$ are available on the same training set, we exploit them to train deep stereo networks in a self-supervised manner. In particular, a regression loss is used to minimize the difference between stereo predictions and $\mathcal{D}^{\mathbb{P}}$.

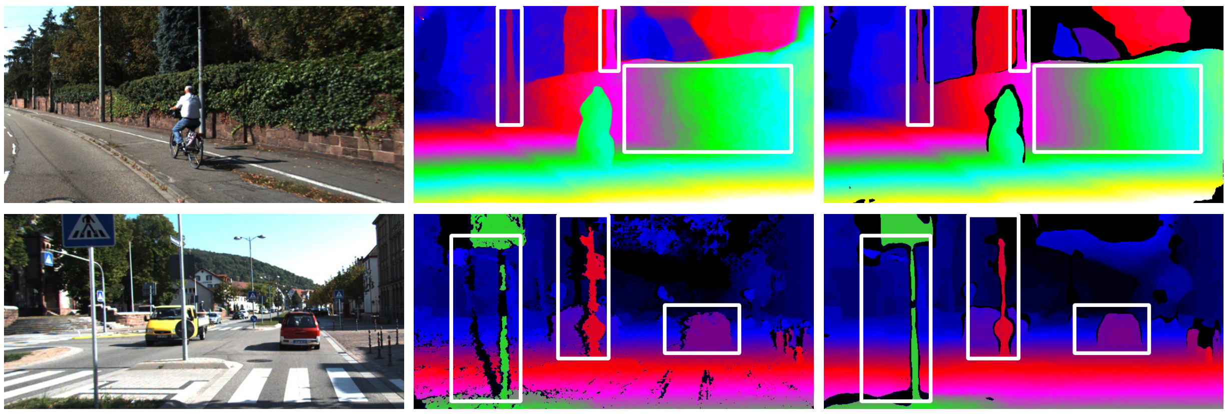


Figure 16.4: Proxy distillation. The first row depicts, from left to right, the reference image, the disparity map computed by a single inference of MCN and the one filtered and regularized using our consensus mechanism. The second row shows the reference image, the disparity map generated by SGM [83] filtered using the left-right consistency check strategy and our disparity map. Images from KITTI.

16.3 Experiments

In this section, we thoroughly evaluate our proposal, proving that sourcing labels with a monocular completion approach is beneficial to train deep stereo networks.

16.3.1 Implementation Details

Traditional stereo methods. We consider two main non-learning based solutions, characterized by different peculiarities, to generate accurate sparse disparity points from a rectified stereo pair. In particular, we use the popular semi-global matching algorithm SGM [83], exploiting the left-right consistency check (LRC) to remove wrong disparity assignments, and the WILD strategy proposed in [239] that selects highly reliable values from the maps computed by the local algorithm Block-Matching (BM) [276] exploiting traditional confidence measures. We refer to these methods (i.e. stereo method followed by a filtering strategy) as SGM/L and BM/W, respectively.

Monocular Completion Network. We adopt the publicly available self-supervised monocular architecture monoResMatch [241] trained with the supervision of disparity proxy labels specifically suited for our purposes. We modify the network to exploit accurate sparse annotations as input by concatenating them with the RGB image. We set the random sampling probability in Eq. 16.3 as $p = \frac{1}{1000}$. In our experiments, we train from scratch the MCN network following the same training protocol defined in [241] except for the augmentation procedure which includes the flipping strategy (with 0.25 probability) aimed at handling occlusion artifacts [70]. Instead, we empirically found out that generating $\mathcal{D}^{\mathbb{O}}$ using a larger set of points helps to achieve more accurate predictions at inference time. In particular, we fix $p = \frac{1}{20}$ and $p = \frac{1}{200}$ for BM/W and SGM/L respectively. Finally, for the consensus mechanism, we fix $N = 50$, the threshold $\gamma = 3$ and apply for each $\mathcal{I}_i$ color augmentation and random horizontal flip (with 0.5 probability).

Stereo networks. We considered both 2D and 3D deep stereo architectures, ensuring a comprehensive validation of our proposal. In particular, we designed a baseline architecture, namely Stereodepth, by extending [70] to process stacked left and right images, and iResNet [129] as examples of the former case, while PSMNet [31] and GWCNet [75] as 3D architectures. At training time, the models predict disparities $\mathcal{D}^S$ at multiple scales in which each intermediate prediction is upsampled at the input resolution. A weighted smooth L1 loss function (the lower the scale, the lower the weight) minimizes the difference between $\mathcal{D}^S$ and the disparity provided by the proxy $\mathcal{D}^{\mathbb{P}}$ considering only valid pixels, using Adam [118] as optimizer ($\beta_1 = 0.9$ and $\beta_2 = 0.999$). We adopt the original PyTorch [171] implementation of the networks if available. Moreover, all the models have been trained to fit a single Titan X GPU.

16.3.2 Evaluation of Proxy Label Generators

At first, we first evaluate the accuracy of proxies produced by our self-supervised approach with respect to traditional methods. We consider both D1 and EPE, computed on disparities, as error metrics on both non-occluded (*Noc*) and all regions (*All*). In particular, D1 represents the percentage of pixels for which the estimation error is ≥ 3 px and $\geq 5\%$ of its ground truth value, while EPE is obtained by averaging the absolute difference between predictions and ground truths. In addition, the density and the overlap with the

Method	Configuration					Statistics		Noc		All	
	Source	Filter	A	R	C	Density(%)	Overlap(%)	D1(%)	EPE	D1(%)	EPE
MONO	monoResMatch	-	-	-	-	100.0	100.0	26.63	2.96	27.00	2.99
BM	BM	-	-	-	-	100.0	100.0	34.48	16.14	35.46	16.41
SGM	SGM	-	-	-	-	100.0	100.0	6.65	1.67	8.12	2.16
BM/L	BM	LRC	-	-	-	57.89	62.09	16.09	6.42	16.22	6.46
SGM/L	SGM	LRC	-	-	-	86.47	92.28	3.99	1.00	4.01	1.00
SGM/L(*hole-filling*)	SGM	LRC	-	-	-	100.0	100.0	6.56	1.34	7.68	1.57
BM/W	BM	WILD	-	-	-	12.33	10.43	1.33	0.81	1.35	0.81
MCN-SGM/L	SGM	LRC	-	-	-	100.0	100.0	6.36	1.27	7.80	1.50
MCN-SGM/L-R	SGM	LRC	-	✓	-	100.0	100.0	5.28	1.13	5.73	1.21
MCN-SGM/L-AC	SGM	LRC	✓	-	✓	95.36	97.36	5.58	1.17	5.58	1.15
MCN-SGM/L-RC	SGM	LRC	-	✓	✓	93.50	96.32	2.95	0.86	3.14	0.89
MCN-SGM/L-ARC	SGM	LRC	✓	✓	✓	92.53	95.76	2.78	0.84	2.92	0.86
MCN-BM/W	BM	WILD	-	-	-	100.0	100.0	11.86	1.93	12.50	2.03
MCN-BM/W-R	BM	WILD	-	✓	-	100.0	100.0	6.79	1.40	7.11	1.45
MCN-BM/W-AC	BM	WILD	✓	-	✓	91.45	94.76	8.36	1.53	8.64	1.57
MCN-BM/W-RC	BM	WILD	-	✓	✓	91.12	95.28	3.79	0.95	4.03	1.0
MCN-BM/W-ARC	BM	WILD	✓	✓	✓	86.82	93.56	3.16	0.90	3.27	0.92

Table 16.1: Evaluation of proxy generators. We tested proxies generated by different strategies on the KITTI 2015 training set.

ground truth are reported to take into account filtering strategies. Table 16.1 reports a thorough evaluation of different methodologies and filtering techniques. It can be noticed how BM and SGM have different performances due to their complementarity (local vs semi-global), but containing several errors. Filtering strategies help to remove outliers, at the cost of sparser maps. Notice that restoring the full density through *hole-filling* [83] slightly improves the results of SGM/L, but it is not meaningful for BM/W since filtered maps are too sparse. Unsurprisingly, even if the depth maps produced by the vanilla monoResMatch are fully-dense, they are not accurate due to its inherent monocular nature. On the contrary, our monocular strategy MCN produces dense yet accurate maps thanks to the initial disparity guesses, regardless the sourcing stereo algorithm. Moreover, by applying Augmentation techniques (A) on the RGB image or selecting Random input points (R), allow to increase variance and to exploit our Consensus mechanism (C) to filter out unreliable values, thus achieving even better results. In fact, the consensus mechanism is able to discard wrong predictions preserving high density, reaching best performances when A and R are both applied. It is worth noting that if R is not performed, the network is fed with all the available guesses both at training and testing time, with remarkably worse results compared to configuration using random sampling.

Model	All			Obj		
	Valid	Correct	Accuracy (%)	Valid	Correct	Accuracy (%)
MCN-BM/W-ARC	11,551,461	11,247,966	97.37	1,718,267	1,642,872	95.61
MCN-SGM/L-ARC	12,201,763	11,860,923	97.20	1,788,154	1,672,222	93.52
MCN-LiDAR	11,773,897	11,636,787	98.83	1,507,222	1,459,726	96.84
GuideNet-LiDAR [264] †	2,973,882	2,915,110	98.02	221,828	210,912	95.07

Table 16.2: Model-guided comparison. Comparison between our self-supervised MCN model and the supervised GuideNet stereo architecture [264] using 142 ground truth images of the KITTI 2015 training set. † indicates that the network requires LiDAR points at training time. Accuracy is defined as 100-D1.

Disparity completion comparison. We validate our MCN combined with the consensus mechanism comparing it to GuideNet [264], a supervised architecture designed to generate high-quality disparity annotations exploiting multi-frame LiDAR points and stereo pairs as input. Following [264], we measure the valid pixels, correct pixels, and accuracy (i.e. 100.0 - D1) on 142 images of the KITTI 2015 training set. Table 16.2 clearly shows how MCN trained in a self-supervised manner achieves comparable accuracy with respect to GuideNet-LiDAR by exploiting sparse disparity estimates from both [239] and [83] but with a significantly higher number of points, even on foreground regions (Obj). Notice that LiDAR indicates that the network is fed with LiDAR points filtered according to [246]. To further demonstrate the generalization capability of MCN to produce highly accurate proxies relying on points from heterogeneous sources, we feed MCN-BM/W-R with raw LiDAR measurements. By doing so, our network notably outperforms GuideNet in this configuration, despite it leverages a single RGB image and has not been trained on LiDAR points.

16.3.3 Ablation study

In this subsection, we support the statement that a completion approach provides a better supervision compared to traditional stereo algorithms. We first run experiments on KITTI and then use our best configuration on DrivingStereo as well, showing that it is effective on multiple large stereo datasets.

KITTI. For the ablation study, reported in Table 24.2, we consider both 3D (PSM-

Backbone	Supervision	Noc		All	
		D1(%)	EPE	D1(%)	EPE
Stereodepth	PHOTO	**6.50**	**1.30**	**7.12**	**1.40**
PSMNet	PHOTO	6.62	**1.30**	7.67	1.50
Stereodepth	SGM-L	5.22	**1.13**	5.43	**1.15**
Stereodepth	SGM/L(*hole-filling*)	6.06	1.16	6.38	1.21
Stereodepth	BM/W	**5.19**	1.16	**5.37**	1.18
PSMNet	SGM/L	5.46	1.19	5.61	1.21
PSMNet	SGM/L(*hole-filling*)	6.06	1.23	6.32	1.26
PSMNet	BM/W	6.89	1.59	7.03	1.60
Stereodepth	MCN-SGM/L-R	5.11	1.11	5.37	1.14
Stereodepth	MCN-BM/W-R	4.75	1.05	4.96	1.07
Stereodepth	MCN-SGM/L-ARC	4.56	1.08	4.77	1.11
Stereodepth	MCN-BM/W-ARC	4.21	1.06	4.39	1.07
PSMNet	MCN-SGM/L-R	4.39	1.05	4.60	1.07
PSMNet	MCN-BM/W-R	4.30	1.06	4.49	1.08
PSMNet	MCN-SGM/L-ARC	4.02	1.05	4.20	1.07
PSMNet	MCN-BM/W-ARC	**3.68**	**0.99**	**3.85**	**1.01**
Stereodepth	LiDAR/SGM [246]	3.95	1.07	4.10	1.09
PSMNet	LiDAR/SGM [246]	**3.93**	**1.05**	**4.07**	**1.07**

Table 16.3: Ablation study. We trained Stereodepth and PSMNet on KITTI using supervision signals from different proxy generators and tested on KITTI 2015.

Net) and 2D (Stereodepth) networks featuring different computational complexity. First, we train the baseline configuration of the networks, i.e. relying image reconstruction loss functions (PHOTO) only as in [70]. Then, we leverage disparity values sourced by traditional stereo algorithms in which outliers have been removed by the filtering strategies adopted. Such labels provide a useful guidance for stereo networks and allow to obtain more accurate models w.r.t. the baselines. Nonetheless, proxies produced by MCN prove to be much more effective than traditional ones, improving both D1 and EPE by a notable margin regardless the stereo algorithm used to extract the input guesses. Moreover, it can be perceived that best results are obtained when the complete consensus mechanism is enabled.

Finally, we rely also on filtered LiDAR measurements from [246] in order to show differences with respect to supervision from active sensors. Noteworthy, models trained using proxies distilled by ARC configuration of MCN prove to be comparable or even better than using LiDAR with PSMNet and Stereodepth. This behaviour can be explained

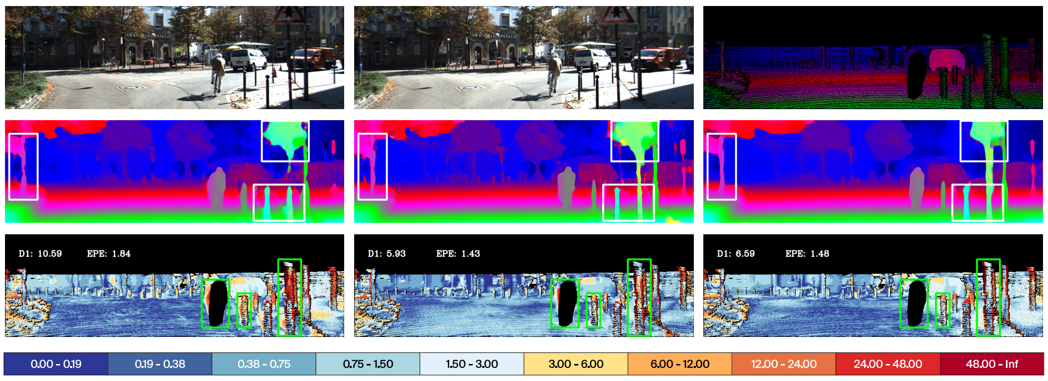


Figure 16.5: Impact of proxies. From top, input stereo pair and ground truth disparity map, predictions by Stereodepth trained with SGM/L (left), LiDAR (center) and our MCN-BM/W-ARC (right), error maps. Best viewed with colors.

due to a more representative and accurate supervision on occluded areas than traditional stereo and filtered LiDAR, thus making the deep networks more robust even there, as clearly shown in Fig. 16.5.

DrivingStereo. We validate the proposed strategy also on DrivingStereo, proving that our distillation approach is able to largely improve the performances of stereo networks also on different datasets. In particular, in Table 16.4 we compare Stereodepth and PSMNet errors when trained using MCN-BM/W-ARC method (i.e. the best configuration on KITTI) with LiDAR and BM/W. Again, our proposal outperforms BM/W, and reduces the gap with high quality LiDAR supervision. Moreover, to verify generalization capabilities, we test on KITTI also correspondent models trained on DrivingStereo, without performing any fine-tuning (DS $\rightarrow$ K), and vice versa (K $\rightarrow$ DS). It can be noticed that the gap between KITTI models (see Table 24.2) and those trained on DrivingStereo gets smaller, proving that the networks are able to perform matching correctly even in cross-validation scenario. We want to point out that this is due to our proxies, as can be clearly perceived by looking at rows 1-2 vs 3-4 in Table 16.4.

		Source → Target					
Backbone	Supervision	DS → DS		K → DS		DS → K	
		D1(%)	EPE	D1(%)	EPE	D1(%)	EPE
Stereodepth	BM/W	**4.46**	**1.20**	**4.67**	**1.10**	**6.35**	**1.36**
PSMNet	BM/W	8.81	1.94	5.06	1.30	7.07	1.65
Stereodepth	MCN-BM/W-ARC	2.47	0.94	2.97	0.96	5.64	1.22
PSMNet	MCN-BM/W-ARC	**1.87**	**0.86**	$\underline{\textbf{2.32}}$	$\underline{\textbf{0.88}}$	**5.16**	$\underline{\textbf{1.17}}$
Stereodepth	LiDAR [264]	1.20	0.69	3.60	1.23	4.57	$\underline{\textbf{1.17}}$
PSMNet	LiDAR [264]	$\underline{\textbf{0.59}}$	$\underline{\textbf{0.54}}$	**2.64**	**1.03**	$\underline{\textbf{4.52}}$	1.26

Table 16.4: Cross-validation analysis. We tested on the Target dataset models trained on the Source one, leveraging different proxies. Notice that no fine-tuning on the target dataset is performed in case of cross-validation.

Method	RMSE	RMSE log	D1 (%)	EPE	δ <1.25	$\delta < 1.25^2$	$\delta < 1.25^3$
Godard et al.[69] (stereo)	5.742	0.202	10.80	-	0.928	0.966	0.980
Lai et al.[122]	4.186	0.157	8.62	1.46	0.950	0.979	0.990
Wang et al.[253] (stereo only)	4.187	0.135	7.07	-	0.955	0.981	0.990
Zhong et al.[289]	4.857	0.165	6.42	-	0.956	0.976	0.985
Wang et al.[253] (stereo videos)	3.404	0.121	5.94	-	0.965	0.984	0.992
Zhong et al.[290]*	$\underline{\textbf{(3.176)}}$	(0.125)	(5.14)	-	(0.967)	-	-
Ours (Stereodepth)	3.882	0.117	4.39	1.07	0.971	0.988	$\underline{\textbf{0.993}}$
Ours (GWCNet)	3.614	0.111	3.93	1.04	0.974	$\underline{\textbf{0.989}}$	$\underline{\textbf{0.993}}$
Ours (iResNet)	3.464	$\underline{\textbf{0.108}}$	3.88	1.02	$\underline{\textbf{0.975}}$	0.988	$\underline{\textbf{0.993}}$
Ours (PSMNet)	3.764	0.115	$\underline{\textbf{3.85}}$	$\underline{\textbf{1.01}}$	0.974	0.988	$\underline{\textbf{0.993}}$

Table 16.5: Comparison with state-of-the-art. Results of different self-supervised stereo networks on the KITTI 2015 training set with max depth set to 80m. **Ours** indicates networks trained using MCN-BM/W-ARC labels. * indicates networks trained on the same KITTI 2015 data, therefore not directly comparable with other methods.

16.3.4 Comparison with state-of-the-art

We compare our models with state-of-the-art self-supervised stereo methods. Table 16.5 reports, in addition to D1 and EPE, also RMSE and RMSE log as depth error measurements and $\delta < 1.25, \delta < 1.25^2, \delta < 1.25^3$ accuracy metrics according to [253, 291]. Notice that some of these methods exploit additional information, such as stereo videos [253] or adaptation strategies [290]. Proxies distilled by MCN-BM/W-ARC can be successfully exploited using both 2D and 3D architectures, enabling even the simplest 2D network Stereodepth to outperform all the competitors. Our strategy is effective, allowing all the adopted backbones to improve depth estimation by a notable margin on 6 metrics out

Models	Dataset	E2E	D1-bg (%)	D1-fg (%)	D1-All (%)	D1-Noc (%)
Zbontar and LeCun (acrt) [279]	K	-	2.89	8.88	3.89	3.33
Tonioni et al. [238]	SF+K	✓	3.75	9.20	4.66	4.27
Mayer et al. [151]	SF+K	✓	4.32	4.41	4.34	4.05
Chang and Chen [31] (PSMNet)	SF+K	✓	1.86	4.62	2.32	2.14
Guo et al. [75] (GWCNet)	SF+K	✓	1.74	3.93	2.11	1.92
Zhang et al. [282]	SF+K	✓	**1.48**	**3.46**	**1.81**	**1.63**
Hirschmuller [82]	-	-	8.92	20.59	10.86	9.47
Zhou et al. [291]	K	✓	-	-	9.91	-
Li and Yuan [126]	K	✓	6.89	19.42	8.98	7.39
Tulyakov et al. [245]	K	-	3.78	10.93	4.97	4.11
Joung et al. [107]	K	-	-	-	4.47	-
Ours(PSMNet)	K	✓	**3.13**	**8.70**	**4.06**	**3.86**

Table 16.6: KITTI 2015 online benchmark. We submitted PSMNet, trained on MCN-BM/W-ARC labels, on the KITTI 2015 online stereo benchmark. In blue self-supervised methods, while in red supervised strategies. We indicate with E2E architectures trained in an end-to-end manner, while SF on the SceneFlow dataset [151].

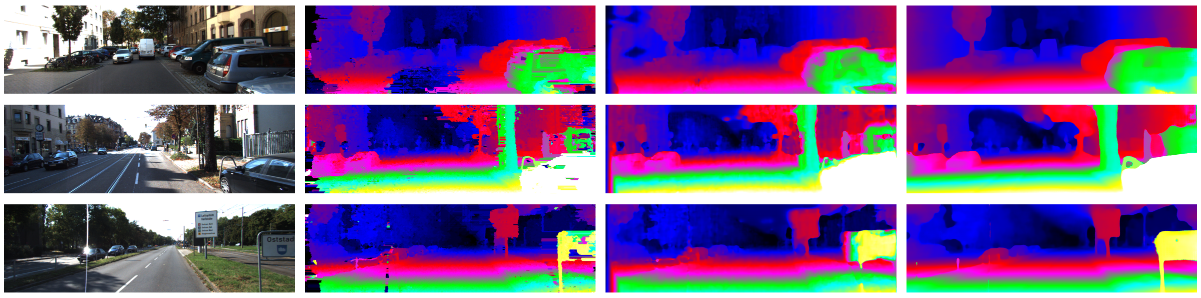


Figure 16.6: KITTI 2015 online benchmark qualitatives. From left to right, the reference images, and the disparity maps computed by [82], [126] and our PSMNet trained on MCN-BM/W-ARC labels.

of 7. Furthermore, we test our PSMNet trained using MCN-BM/W-ARC proxies on the KITTI 2015 online benchmark, reporting the results in Table 16.6. Our model not only outperforms [82] and self-supervised competitors, as can be also perceived in Fig. 16.6, but also supervised strategies [151, 238] on both non-occluded and all areas.

16.3.5 Generalization

Finally, we show experiments supporting that supervision from our MCN-BM/W-ARC labels achieves good generalization to different domains. To this aim, we run our KITTI networks on Middlebury 2014 and ETH3D, framing completely different environments.

Method	Training Dataset	Middlebury 2014 [213]		ETH3D [215]	
		BAD2 (%)	EPE	BAD2 (%)	EPE
Zhang et al.[282]	SF+K	**18.90**	**3.44**	**3.43**	**0.91**
Chang and Chen [31] (PSMNet)	SF+K	20.04	3.01	13.07	1.35
Guo et al.[75](GWCNet)	SF+K	21.36	3.29	19.96	1.88
Wang et al.[253](stereo only)	K	30.55	4.77	11.17	1.47
Wang et al.[253](stereo videos)	K	31.63	5.23	19.59	1.97
Lai et al.[122](stereo videos)	K	45.18	6.42	10.15	1.01
Ours(Stereodepth)	K	27.43	3.72	6.94	1.31
Ours(iResNet)	K	25.08	3.85	6.29	0.81
Ours(GWCNet)	K	20.75	3.17	**3.50**	**0.48**
Ours(PSMNet)	K	**19.56**	**2.99**	4.00	0.51

Table 16.7: Generalization test on Middlebury 2014 and ETH3D. We evaluate networks trained in self-supervised (blue) or supervised (red) fashion on KITTI (K) and SceneFlow dataset (SF) [151].

Table 16.7 shows the outcome of this evaluation. We report, on top, the performance of fully supervised methods trained on SceneFlow [151] and fine-tuned on KITTI for comparison. On bottom, we report self-supervised frameworks trained on the KITTI split from the previous experiments. All networks are transferred without fine-tuning. Compared to existing self-supervised strategies (rows 4-6), networks trained with our proxies achieve much better generalization on both the datasets, performing comparable (or even better) with ground truth supervised networks. Fig. 16.7 shows few examples from the two datasets, where the structure of the scene is much better recovered when trained on our proxies.

Table 16.7 shows the outcome of this evaluation. We report, on top, the performance of fully supervised methods trained on SceneFlow [151] and fine-tuned on KITTI for comparison. On bottom, we report self-supervised frameworks trained on the KITTI split from the previous experiments. All networks are transferred without fine-tuning. Compared to existing self-supervised strategies (rows 4-6), networks trained with our proxies achieve much better generalization on both the datasets, performing comparable (or even better) with ground truth supervised networks. Fig. 16.7 shows few examples from the two datasets, where the structure of the scene is much better recovered when trained on our proxies.

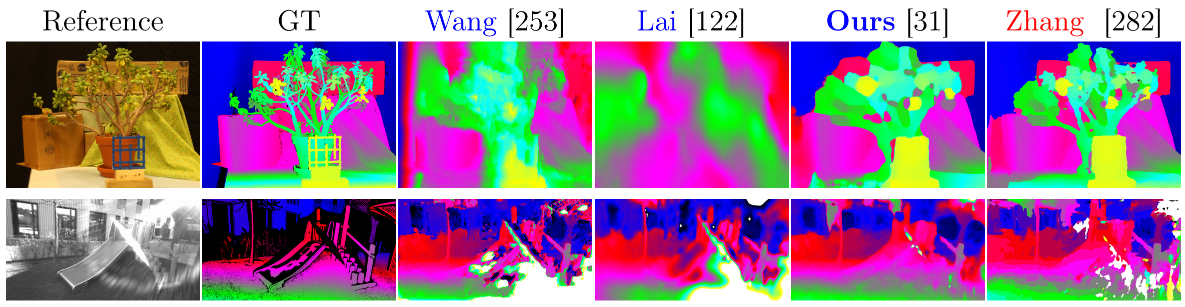


Figure 16.7: Examples of generalization. First row shows disparity maps obtained on a stereo pair from the Middlebury 2014 dataset, while second from ETH3D. Methods in **blue** are self-supervised, while in **red** are supervised with ground truth.

16.4 Conclusion

This chapter proposed a novel strategy to source reliable disparity proxy labels in order to train deep stereo networks in a self-supervised manner leveraging a monocular completion paradigm. Well-known stereo artefacts are soften by learning on such labels, that can be obtained from large RGB stereo datasets in which no additional depth information (e.g. LiDAR or active sensors) is available. Through an extensive ablation study on two popular stereo datasets, we proved that our approach is able to infer accurate yet dense maps starting from points sourced by (potentially) any traditional stereo algorithm, and that such labels provide a strong supervision for both 2D and 3D stereo networks with different complexity. We showed that these networks outperform state-of-the-art self-supervised methods on KITTI by a large margin and are, in terms of generalization on Middlebury 2014 and ETH3D, comparable or even better than ground truth supervised stereo networks.

Chapter 17

Learning end-to-end scene flow by distilling single tasks knowledge

The content of this chapter has been presented at the AAAI Conference on Artificial Intelligence (AAAI 2019) - “Learning End-To-End Scene Flow by Distilling Single Tasks Knowledge” [10].

17.1 Introduction

The term *Scene Flow* refers to the three-dimensional dense motion field of a scene [247] and enables to effectively model both 3D structure and movements of the sensed environment, crucial for a plethora of high-level tasks such as augmented reality, 3D mapping and autonomous driving. Dense scene flow inference requires the estimation of two crucial cues for each observed point: depth and motion across frames acquired over time. Such cues can be obtained deploying two well-known techniques in computer vision: stereo matching and optical flow estimation. The first one aims at inferring the disparity (i.e. depth) by matching pixels across two rectified images acquired by synchronized cameras, the second at determining the 2D motion between corresponding pixels across two consecutive frames, thus requiring at least four images for full scene flow estimation. For years, solutions to

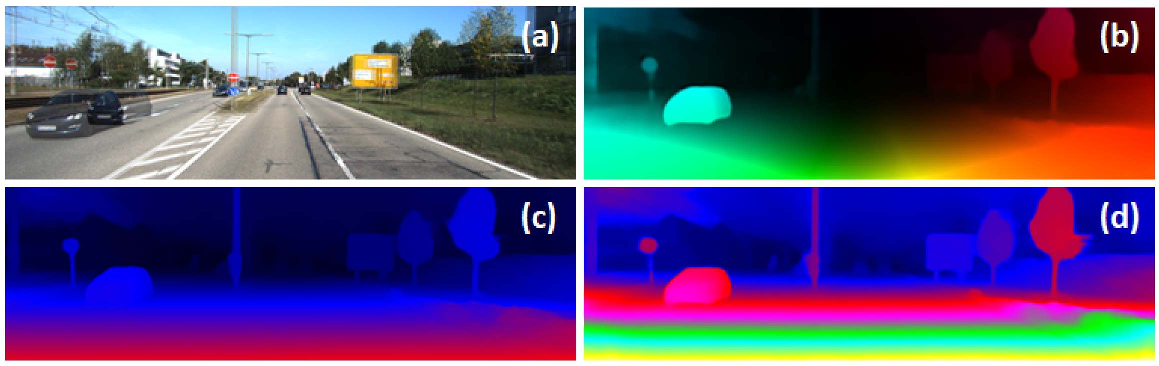


Figure 17.1: End-to-end scene flow with DWARF. (a) Superimposed reference images at time t_1 and t_2, (b) estimated optical flow, (c) disparity and (d) disparity change.

scene flow [22] have been rather accurate, yet demanding in terms of computational costs and runtime. Meanwhile, deep learning established as state-of-the-art for stereo matching [151] and optical flow [55]. Thus, more recent approaches to scene flow leveraged this novel paradigm stacking together single-task architectures [99, 138]. However, this strategy is demanding as well and requires separate and specific training for each network and does not fully exploit the inherent dependency between the tasks, e.g. the flow of 3D objects depends on their distance, their motion and camera ego-motion [234], as a single model could. On the other hand, either synthetic or real datasets annotated with full scene flow labels are rare compared to those disposable for stereo and flow alone. This constraint limits the *knowledge* available to a single network compared to the one exploitable by an ensemble of specialized ones.

To tackle previous issues, in this chapter, we propose a novel lightweight architecture for scene flow estimation jointly inferring disparity, optical flow and disparity change (i.e., the depth component of 3D motion). We design a custom layer, namely 3D correlation layer, by extending the formulation used to tackle the two tasks singularly [55, 151], in order to encode matching relationships across the four images. Moreover, to overcome the constraint on training data, we recover the missing knowledge leveraging standalone, state-of-the-art networks for stereo and flow to generate proxy annotations for our single scene flow architecture. Using this strategy on the KITTI dataset [155], we *distill* about

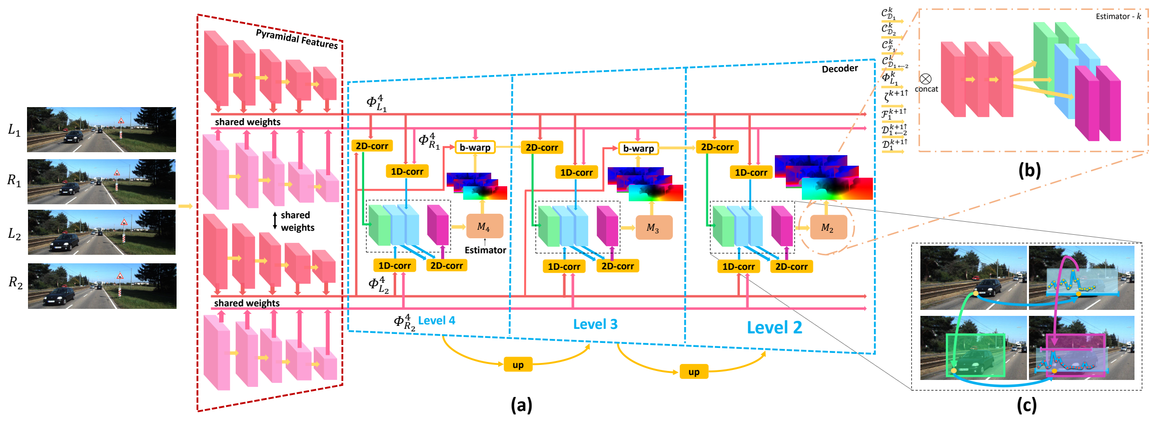


Figure 17.2: DWARF architecture. The full architecture (a) has shared encoders (pink) to extract pyramids of features. At each resolution k, correlation scores respectively in green, light blue, light blue and purple, are stacked and forwarded to the estimator to generate $\mathcal{F}_1^k$, $\mathcal{D}_1^k$ and $\mathcal{D}_{1\leftarrow 2}^k$. Such outputs are used to warp features at level $k-1$ until the final resolution is reached. Each estimator (b) is made of a common backbone followed by three task-specific heads. Correlation layers encode matching between pixels across the four images (c).

20× samples compared to the number of ground truth images available, enabling for more effective training and thus to more accurate estimations.

Our architecture for scene flow estimation through Disparity, Warping and Flow (dubbed **DWARF**) can be elegantly trained in an end-to-end manner from scratch and yields competitive results compared to state-of-the-art, although running about 10× faster thanks to efficient design strategies. Figure 17.1 shows a qualitative example of dense scene flow estimation achieved by our network, enabling 10 FPS on NVIDIA Titan 1080Ti and about 1 FPS on Jetson TX2 embedded system.

17.2 Proposed Architecture

In this section, we introduce the DWARF architecture built upon established principles from optical flow and stereo matching to obtain, in synergy, an end-to-end framework for full scene flow estimation. As already proved in different fields, coarse-to-fine designs enable for compact, yet accurate models.

Given a couple of stereo image pairs L_1, R_1, L_2 and R_2 referencing, respectively, the left and right images at time t_1 and t_2 we aim at estimating disparity $\mathcal{D}_1$ between L_1, R_1 to obtain its 3D position at time t_1, optical flow $\mathcal{F}_1$ between L_1, L_2 to get 2D motion vectors connecting pixels in L_1 to those in L_2 and disparity change $\mathcal{D}_{1\leftarrow 2}$, i.e. disparity $\mathcal{D}_2$ between L_2, R_2 mapped on corresponding pixels in image L_1 that allows to get z component of 3D motion vectors. To achieve this, our model performs a first extraction phase in order to retrieve a pyramid of features from each image, then in a coarse-to-fine manner it computes point-wise correlations across the four features representations and estimates the aforementioned disparity and motion vectors, going up to the last level of the pyramid to obtain the final output. Figure 22.1 sketches the structure of DWARF configured to process, for the sake of space, a pyramid down to $\frac{1}{32}$ of the original resolution. In the next section, we will describe in detail each module depicted in the figure.

17.2.1 Features Extraction

To extract meaningful representations from each input image, we design a compact encoder to obtain a pyramid of features ready to be processed in a coarse-to-fine manner. Purposely, DWARF has four encoders, one for each input image, with shared weights. Each one is built of a block of three 3×3 convolutional layers for each level in the pyramids of features, respectively with stride 2, 1 and 1. For the sake of space, Figure 22.1 (a) shows an example of 5 levels encoder. Actually DWARF deploys a 6 levels encoder down to $\frac{1}{64}$ resolution features (k=6), counting 18 convolutional layers, each followed by Leaky ReLU activations with $\alpha = 0.1$. By progressively decimating the spatial dimensions, we increase the amount of extracted features, respectively to 16, 32, 64, 96, 128 and 196. It generates features $\phi^k_{L_1}, \phi^k_{R_1}, \phi^k_{L_2}$ and $\phi^k_{R_2}$ with $k \in [1, 6]$, respectively for frames L_1, R_1, L_2 and R_2, deployed by the following module to extract matching relationships between pixels.

17.2.2 Warping

The main advantage introduced by a coarse-to-fine strategy consists of computing small disparity and flow vectors at each resolution and sum them while going up the pyramid. This strategy allows keeping a small range where to calculate correlation scores, as we will discuss in detail in the next section. Otherwise, a large search space would dramatically increase the complexity of the entire network.

Given features $\phi^k_{L_1}, \phi^k_{R_1}, \phi^k_{L_2}$ and $\phi^k_{R_2}$ extracted by the encoder at the k^{th} level, we have to bring all features closer to $\phi^k_{L_1}$ coordinates. To do so, estimates at previous pyramid level $(k+1)$ are upsampled, e.g. $\mathcal{D}^{k+1}_1$ to $\mathcal{D}^{k+1^\uparrow}_1$, and properly scaled to match stereo/flow at the next resolution k. Then, features are warped by means of backward warping, in particular $\phi^k_{L_2}$ according to optical flow $\mathcal{F}^{k+1^\uparrow}_1$ and $\phi^k_{R_1}$ according to $\mathcal{D}^{k+1^\uparrow}_1$. Finally, the motion that allows to warp $\phi^k_{R_2}$ towards $\phi^k_{L_1}$ is given by the sum of $\mathcal{F}^{k+1^\uparrow}_{L_1}$ and $\mathcal{D}^{k+1^\uparrow}_{1\leftarrow 2}$. This because the former encodes the mapping between present and future correspondences, while the latter the horizontal displacement occurring between L_2 and R_2, but on L_1 coordinate, thus the same as $\mathcal{F}^{k+1^\uparrow}_1$.

We will see how this translates into computing, at each resolution k, a refined scene flow field to ameliorate a prior, coarse estimation inferred at resolution $k+1$. At the lowest resolution in the pyramid, features are not warped since scene flow priors are not available.

17.2.3 Cost Volumes and 3D Correlation Layer

Since DWARF jointly reasons about stereo and optical flow, correlation layers fit very well in its design. At first we compute correlation scores encoding standalone tasks, i.e. estimation of disparity $\mathcal{D}_1$ between L_1, R_1, $\mathcal{D}_2$ between L_2, R_2 and flow $\mathcal{F}_1$ between L_1, L_2, obtaining $\mathcal{C}^k_{\mathcal{D}_1}(\phi^k_{L_1}, \phi^k_{R_1})$, $\mathcal{C}^k_{\mathcal{D}_2}(\phi^k_{L_2}, \phi^k_{R_2})$, $\mathcal{C}^k_{F_1}(\phi^k_{L_1}, \phi^k_{L_2})$ by means of two 1D and one 2D correlation layers depicted in light blue and green in Figure 22.1 (a). By defining the correlation between per-pixel features as $\langle\cdot\rangle$ and concatenation as $\otimes$, we obtain 1D and

2D correlations as

$$
\begin{aligned}
\mathcal{C}^k_{\mathcal{D}_t}(y,x) &= \bigotimes_{j\in[-r_x,r_x]} \langle \phi^k_{L_t}(y,x), \phi^k_{R_t}(y,x+j)_w \rangle \\
\mathcal{C}^k_{\mathcal{F}_1}(y,x) &= \bigotimes_{\substack{i\in[-r_y,r_y],\\ j\in[-r_x,r_x]}} \langle \phi^k_{L_1}(y,x), \phi^k_{L_2}(y+i,x+j)_w \rangle
\end{aligned}
\tag{17.1}
$$

with (y, x) pixel coordinates, r_y, r_x radius on y and x directions, $t \in [1, 2]$. Subscript w means warping via upsampled priors as described in Section 17.2.2. Although such features embody relationships about standalone tasks, they lack at encoding matching between the 3D motion of the scene. To overcome this limitation, we introduce a novel custom layer.

Figure 22.1 (c) depicts how correlation layers act in DWARF. While 2D correlation layer (green) encodes similarities between pixels aimed at estimating optical flow, 1D correlations (light blue) compute scores between left and right images independently from time. Each produces a correlation curve, superimposed on R_1 and R_2 in the figure. If a pixel does not change its disparity through time, the peaks in the two curves would ideally match. Otherwise, they will appear shifted by the magnitude of the disparity change. The rest of the curve will shrink/enlarge, with major differences in portions dealing with regions moving of different motions (e.g., background vs foreground objects). This pattern, if properly learned, acts as a bridge between depth and 2D flow, enabling to infer the full 3D motion. Unfortunately, this behaviour is not explicitly modelled by the layers mentioned above. Hence, we adopt a novel component, namely a 3D correlation layer, whose search volume is depicted in purple in Figure 22.1 (c). Since correlation curves are already available from 1D correlation layers, this translates into computing *correlations over correlations* volumes as

$$
\mathcal{C}^k_{\mathcal{D}_{1\leftarrow 2}} = \bigotimes_{\substack{i\in[-r_y,r_y],\\ j\in[-r_x,r_x],\\ h\in[-r_z,r_z]}} \langle \mathcal{C}^k_{\mathcal{D}_1}(y,x,d), \mathcal{C}^k_{\mathcal{D}_2}(y+i,x+j,d+h) \rangle \tag{17.2}
$$

with (y, x, d) pixel coordinates in the correlation volumes and r_z the search radius for displacement between 1D correlation curves. Specifically, the full search space of such operation is 3D, being it over pixel coordinates plus displacement between correlation curves.

17.2.4 Scene Flow Estimation

After the extraction of meaningful correlation features, we stack them into a features volume forwarded to a compact decoder network in charge of estimating the three components of the scene flow. As shown in Figure 22.1 (b), at each level the volume contains reference image features $\phi^k_{L_0}$, correlation scores, upsampled scene flow priors and latest features ζ extracted before estimation at level $k+1$. This input is forwarded to level k decoder. First, three convolutional layers with respectively 128, 128 and 96 channels rearrange the volume. Then, three independent *heads* are in charge of predicting $\mathcal{D}^k_1$, $\mathcal{F}^k_1$ and $\mathcal{D}^k_{1\leftarrow 2}$. Following this design, the network is forced to create a first holistic representation of the volume, then specialized by each sub-module. Each head has two task-specific 3×3 convolutional layers with 64 and 32 channels, producing ζ features from which a final 3×3 layer extracts the single component of scene flow at level k, e.g. $\mathcal{D}^k_1$. Such estimates, together with features ζ, are upsampled through a transposed convolution layer with stride 2, to provide coarse scene flow priors for warping at level $k-1$. Leaky ReLU units with $\alpha = 0.1$ follow all layers. Each estimator is optionally designed with dense connections [92] to boost accuracy. This design choice adds about 10 million parameters to DWARF.

17.2.5 Residual Refinement

Although the explicit reasoning about features matching across the four images is an effective way to guide the network towards scene flow estimation, it has limitations for pixels having missing correspondences. This fact occurs when, in one or multiple frames, they are occluded or no longer part of the observed scene. For instance, portions of the

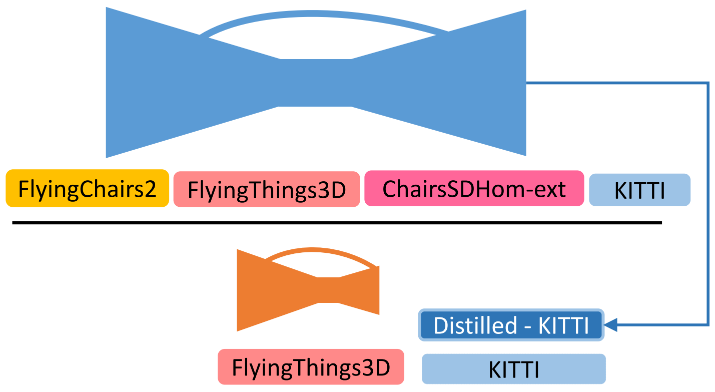


Figure 17.3: Knowledge distillation [81] scheme. From an ensemble of deep networks [99] (blue) trained on a variety of datasets we transfer knowledge to our compact model (orange).

sensed scene located near image borders at time t_1 are no longer framed at t_2 when the camera is moving. To soften this problem, three residual networks are deployed to refine each single component of the full scene flow estimates, taking as input ζ features from the top-level estimator and processing it with six 3×3 convolutional layers extracting respectively 128, 128, 128, 96, 64, 32 features, with a dilation factor of 1, 2, 4, 8, 16, and 1 respectively to increase the receptive field introducing moderate overhead. A Leaky ReLU with $\alpha = 0.1$ follows each layer. Then, a further 3×3 convolutional layer (without activation units) extracts residual scene flow, summed to previous final estimations in order to refine them.

17.2.6 Knowledge Distillation from Expert Models

As previously pointed out, although end-to-end training is elegant and easier to schedule, it prevents using task-specific datasets since ground truth labels are required for full scene flow. However, a proper training schedule across several datasets is needed to achieve the best accuracy on single tasks [97, 231]. To overcome this limitation, we leverage on knowledge distillation [81] employing expert models trained for the single tasks and used to teach to a student network, DWARF in this case.

Specifically, we choose the ensemble of networks proposed by Ilg et al. [99] to guide our simpler model, thanks to the availability of the source code and its excellent performance. Firstly a FlowNet-CSS and DispNet-CSS are in charge of optical flow and disparity estimation, then a third FlowNet-S architecture processes disparity $\mathcal{D}_2$ back warped according to computed optical flow and refines it to obtain $\mathcal{D}_{1\leftarrow 2}$. The three networks are trained in a multi-stage manner, starting from DispNet-CSS and FlowNet-CSS, and ending with the training of the final FlowNet-S. This allows for multi-dataset training, especially in the case of optical flow for which several sequential rounds of training on FlyingChairs2 and ChairsSDHom-ext [97] are performed to achieve the best accuracy. By teaching DWARF with the expert models, we are able to both i) bring the knowledge learned by the expert model on task-specific datasets (e.g., FlyingChairs2 and ChairsSDHom-ext) to our model and ii) distill an extended training set, counting a larger amount and more variegated samples. We will show in our experiments how the knowledge distillation scheme results more effective than training on the few ground truth images available from real datasets.

17.2.7 Training Loss

Given the set of learnable parameters of the network Θ, $\mathcal{D}_1^k(\Theta)$, $\mathcal{D}_{1\leftarrow 2}^k(\Theta)$ and $\mathcal{F}_1^k(\Theta)$ respectively the estimated disparity, disparity change and optical flow, $\mathcal{D}_1^k(GT)$, $\mathcal{D}_{1\leftarrow 2}^k(GT)$ and $\mathcal{F}_1^k(GT)$ the ground truth maps for specific scene flow components brought to each pyramid level k, we adopt the L1 norm to optimise DWARF:

$$\begin{aligned}\mathcal{L}(\Theta) = \gamma\|\Theta\|_1^2 + \epsilon_1 \sum_{k=l_1}^{L} \alpha_k \|\mathcal{D}_1(\Theta) - \mathcal{D}_1^k(GT)\|_1 \\ +\epsilon_2 \sum_{k=l_1}^{L} \alpha_k \|\mathcal{D}_{1\leftarrow 2}^k(\Theta) - \mathcal{D}_{1\leftarrow 2}^k(GT)\|_1 + \epsilon_3 \sum_{k=l_1}^{L} \alpha_k \|\mathcal{F}_1^k(\Theta) - \mathcal{F}_1^k(GT)\|_1\end{aligned} \tag{17.3}$$

We deploy a 6 levels pyramidal structure, extracting features up to level 6, halving the spatial resolution down to $\frac{1}{64}$. We set $l_0 = 2$, thus estimating scene flow up to quarter resolution and then bilinearly upsampling to the original input resolution. This strategy allows us to keep low memory requirements and fast inference time. The search spaces are set to 9, 9×9 and $9 \times 9 \times 1$ respectively for 1D, 2D and 3D correlations. A search range of 1 on disparity change keeps low the overall complexity of the network, yet significantly improving the accuracy on all metrics.

17.3 Experimental Results

We report extensive experiments aimed at assessing the accuracy and performance of DWARF. First, we describe in detail the training schedules. Then, we conduct an ablation study to measure the contribution of each component and compare DWARF to state-of-the-art deep learning approaches. Finally, we focus on DWARF run-time performance, extensively studying its behaviour on a variety of hardware platform, including a popular embedded device equipped with a low-power GPU.

17.3.1 Training Datasets and Protocol

It is a common practice to initialize end-to-end networks on large synthetic datasets before fine-tuning on real data [55, 97, 151]. Despite the large availability of synthetic datasets for flow and stereo [28, 55, 151], only the one proposed in [151] provides ground truth for full scene flow estimation. In this field, KITTI 2015 [155] represents the unique example of a realistic benchmark for scene flow. Therefore, we scheduled training on these two datasets and, optionally, we leverage knowledge distillation [81] from an expert network [99] to augment the variety of realistic samples and consequently to better train DWARF.

Flying Things 3D. We set $\alpha_6 = 0.32$, $\alpha_5 = 0.08$, $\alpha_4 = 0.02$, $\alpha_3 = 0.01$ and $\alpha_2 = 0.005$, $\gamma = 0.0004$ and cross-task weights to $\epsilon_1 = 1$, $\epsilon_2 = 1$ and $\epsilon_3 = 0.5$. Ground truth values are down-scaled to match the resolution of the level and scaled by a factor of 20, as done by [55, 231]. The network has been trained for 1.2M steps with a batch size of 4 randomly selecting crops with size 768×384, using Adam optimiser [118], with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and initial learning rate of 10^{-4}, which has been halved after 400K, 600K, 800K and 1M steps.

KITTI 2015. We fine-tuned the network using the 200 training images from the KITTI Scene Flow [156] dataset with a batch size of 4 for 50K steps. Again, Adam optimizer [118] has been adopted with the same parameters as before. The initial learning rate is set to 3×10^{-5}, halved after 25K, 35K and 45K steps. We minimise loss only at level $k = 2$. Specifically, we upsample through bilinear interpolation the predictions at the quarter resolution and apply the fine-tuning loss described in 17.2.7 at full resolution. Predictions at lower levels have not been optimized explicitly. We set $\epsilon_1 = 1$, $\epsilon_2 = 1$ and $\epsilon_3 = 0.5$, all the α_k set to 0 with the exception of α_2 set to 0.001 while γ is left untouched. Images are firstly padded to 1280×384 pixels, then random crops of size 896×320 are extracted at each iteration.

Distilled-KITTI. Finally, we perform knowledge distillation to produce an extended set of images for fine-tuning DWARF. Specifically, we use the 4000 total images available from the multiview extension of the KITTI 2015 training set and we produce proxy annotations leveraging FlowNet-CSS, DispNet-CSS and FlowNet-S. We use the trained models made available by the authors, trained on multiple task-specific datasets and fine-tuned on the aforementioned KITTI 2015 split (i.e., 200 images). We point out that, excluding the task-specific synthetic datasets, the expert models are trained with the same real ground truth (i.e., no additional annotations) and are used only to distill more *proxy* labels. Moreover, despite the extremely accurate estimates produced by the expert models on the KITTI training split (below 2% error rate on full scene flow), the labels sourced through distillation are yet *noisy*.

Data augmentation. We perform data augmentation by applying random gamma correction in [0.8,1.2], additive brightness in [0.5,2.0], and colour shifts in [0.8,1.2] for each channel separately. To increase robustness against brightness changes, we applied augmentation independently to every single image. Instead, random zooming, with probability 0.5 of re-scaling the image by a random factor in [1,1.8], has been applied in the same way to L_1, R_1, L_2 and R_2 and the relative ground truths.

Configuration			Params	Flow	Disparity	Change
Dense	3Dcorr	Refine	M	EPE	EPE	EPE
			5.06	7.435	1.959	2.283
✓			13.50	6.758	1.837	2.092
✓	✓		15.87	6.738	1.827	2.149
✓	✓	✓	19.62	**6.440**	**1.784**	**2.039**

Table 17.1: Ablation study on the FlyingThings3D test set. For each variant of DWARF, we report End Point Error (EPE) for flow, disparity and change respectively.

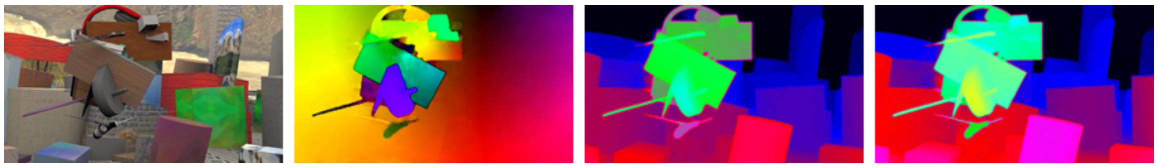


Figure 17.4: Qualitative results on FlyingThings 3D [151] test split. From left to right, reference image at t_1, $\mathcal{F}_1$, $\mathcal{D}_1$ and $\mathcal{D}_{1\leftarrow 2}$.

17.3.2 Ablation Studies

In this section, we study the effectiveness of each architectural choice. Tables 17.1 and 17.2 report experimental results on FlyingThings3D [151] test set and KITTI 2015 training set [156] by i) increasing the complexity of the network and ii) introducing the knowledge distillation process.

FlyingThings3D. This dataset provides 4248 frames for validation. In Table 17.1 we report average End-Point-Error (EPE) for the disparity, flow and change (respectively D1, F1 and D2) on 3822 images, obtained by filtering the validation set according to the guidelines. We trained four DWARF variants, starting from the simple *Standalone tasks* version (i.e., without Dense, 3D correlation and Refinement module) up to the full DWARF architecture. We can notice how the addition of each module always yields better accuracy on most metrics. At first, adding dense connections improves over the baseline model at the cost of nearly triple the number of parameters. By introducing the 3D correlation module, we still improve the capability of the proposed solution to estimate the 3D motion of the scene, this time adding about 2M parameters to the previous 13.5. Finally, adding the Refinement module yields a consistent error reduction on all metrics.

Configuration							
Dense	3Dcorr	Refine	Sup.	F1-All	D1-All	D2-All	SF-All
✓	✓	✓	Gt	18.53	4.58	9.32	20.85
✓	✓	✓	Px	20.71	3.94	9.14	23.07
✓	✓	✓	Px + Gt	20.47	**3.91**	9.43	23.01
✓	✓	✓	Px → Gt	**16.75**	4.22	**8.26**	**19.00**

Table 17.2: Impact of knowledge distillation and its scheduling on 40 images from KITTI training set. We report the percentage of pixels with error higher than 3 and 5% respectively for flow, disparity, change and full scene flow.

Figure 17.4 shows qualitative results on FlyingThings3D validation set obtained by the full model.

KITTI 2015. For this experiment, we split the KITTI 2015 training set into 160 images for fine-tuning and reserve the last portion of 40 images for validation purposes only. We report F1, D1 and D2, respectively the percentages of pixels having absolute error larger than 3 and relative error larger than 5% for the three tasks, considering only the non occluded regions (*Noc*) and the whole image (*All*). In this ablation experiments, we aim at assessing the impact of the knowledge distillation protocol on the final accuracy of DWARF. Purposely, we fine-tuned DWARF on two different datasets: i) the 160 images mentioned above from KITTI 2015 and ii) 3200 images from Distilled-KITTI, corresponding to the 160 sequences belonging to KITTI 2015 multiview extension, respectively reported in the first and the second rows in Table 17.2. We can notice that proxy labels (Px) yield worse performance for optical flow and consequently for full scene flow while allow improving the accuracy for disparity estimation. Combining the two approaches (i.e., replacing 160 images of the Distilled-KITTI dataset with the real available ground truth, third row) produces result close to using only distilled labels. Finally, running a multi-stage fine-tuning made of 40k steps with proxy labels and further 10k with ground truth (ie, first learning from many yet noisy annotations and then focusing on few perfect labels) dramatically improves the results on optical flow and thus full scene flow, as shown in the fourth row of the table.

Configuration			Jetson TX2			1080 Ti
Dense	3DCorr	Refine	Max-Q	Max-P	Max-N	(≈250W)
			0.79s	0.65s	0.57s	0.09s
✓			1.26s	1.05s	0.91s	0.10s
✓	✓		1.47s	1.22s	1.06s	0.11s
✓	✓	✓	2.21s	1.83s	1.59s	0.14s

Table 17.3: Runtime analysis for different variants of DWARF on NVIDIA Jetson TX2 (using Max-Q, Max-P, Max-N configurations) and NVIDIA GTX 1080Ti. Time in seconds.

17.3.3 Run-time Analysis

In Table 17.3, we report the time required to process a couple of stereo images for all variants of DWARF using two different devices. For this purpose, we considered NVIDIA 1080Ti GPU and NVIDIA Jetson TX2, an embedded system equipped with a low-power GPU. The latter device can work with three increasing energy-consumption configurations: *Max-Q* (<7.5W), *Max-P* (≈10W) and *Max-N* (<15W). Even in its more complex configuration, our network can estimate in the *Max-P* configuration the scene flow on KITTI (4 images padded at 1280×384) in less than 2s, draining about $\frac{1}{25}$ of the energy required by the 1080Ti.

17.3.4 KITTI 2015 Online Benchmark

Finally, Table 17.4 reports results for DWARF and state-of-the-art solutions for scene flow, both traditional and based on deep learning. For the final submission, we included all the training data (4000 proxies, 200 ground truths). We followed a 50k (proxy) plus 5k (ground truth) schedule, halving the learning rate at 25K and 35K while reducing it by one quarter at 50K. Despite yielding lower accuracy compared to much more complex state-of-the-art architectures [99, 138], our network allows us to achieve competitive results using $\sim$ 100M fewer parameters and running more than $10\times$ faster. Compared to approaches closer to ours [209], we can notice that our architecture is much more accurate on all metrics, with margins of about 2.58, 1.8 and 1.73% respectively on F1-All, D1-All and

Method	D1-All	D2-All	F1-All	SF-All	Params (M)	Runtime (s)
Behl et al. [22]	4.46	5.95	6.22	8.08	-	600
(Vogel et al. 2015)	4.27	6.79	6.68	8.97	-	300
Ilg et al. [99]	2.16	6.45	8.60	11.34	116.68	1.72
Ma et al. [138]	2.55	4.04	4.73	6.31	136.38	1.03
Saxena et al. [209]	5.13	8.46	12.96	15.69	8.05	0.13
DWARF (ours)	3.33	6.73	10.38	12.78	19.62	0.14

Table 17.4: Results on the KITTI 2015 online benchmark. Results for [99] from the original paper, since no longer available online. Runtime on NVIDIA 1080 Ti.

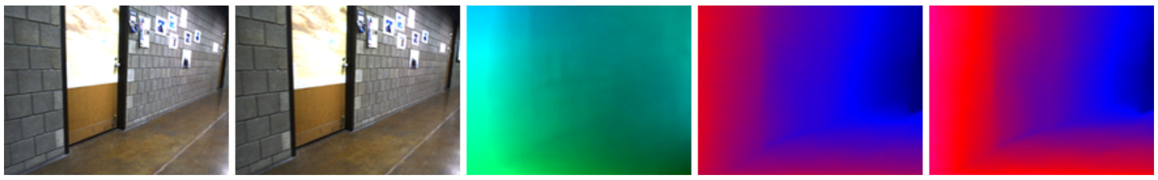


Figure 17.5: Qualitative results on the WeanHall dataset [14]. From left to right, reference frames L_1 and L_2, $\mathcal{F}_1$, $\mathcal{D}_1$ and $\mathcal{D}_{1\leftarrow 2}$.

D2-All, leading to a 2.91% improved scene flow estimation, thanks to both 3D correlation layer and knowledge distillation introduced in this work. Despite counting more than double parameters, DWARF runs almost at the same speed. For a complete comparison with state-of-the-art algorithms, please refer to the KITTI 2015 online benchmark. At the time of writings, DWARF ranks 15^{th}.

17.3.5 Additional Qualitative Results

We also carried out additional experiments on the WeanHall dataset [14], a collection of indoor stereo images. Since no ground truth is provided, we report qualitative results only. Figure 17.5 depicts some examples extracted from this dataset processed by the same DWARF model used to submit results to the online KITTI benchmark, proving effective generalization to unseen indoor environments.

17.4 Conclusion

In this chapter, we proposed DWARF, a novel and lightweight architecture for accurate scene flow estimation. Instead of combining a stack of task-specialized networks as done by other approaches, our proposal is easily and elegantly trained in an end-to-end fashion to tackle all the tasks at once. Exhaustive experimental results prove that DWARF is competitive with state-of-the-art approaches [99, 138], counting $6\times$ fewer parameters and running significantly faster. Future work aims at self-adapting DWARF in an online manner [238].

Chapter 18

Learning monocular depth estimation with unsupervised trinocular assumptions

The content of this chapter has been presented at the International Conference on 3D Vision (3DV 2018) - "Learning monocular depth estimation with unsupervised trinocular assumptions" [187].

18.1 Introduction

In the monocular depth estimation task, the work of Godard et al. [69] represents one of the first works to tackle unsupervised monocular depth estimation task. Deploying stereo imagery for training, a CNN learns to infer disparity from a single reference image and warps the target image accordingly to minimize the appearance error between the warped and the reference image. This strategy yields state-of-the-art performance [69, 281]. The CNN is trained to infer disparity from a single reference image and the target image is warped accordingly minimizing the appearance error between warped and reference image. This way, the depth representation learned by the network is affected by artifacts

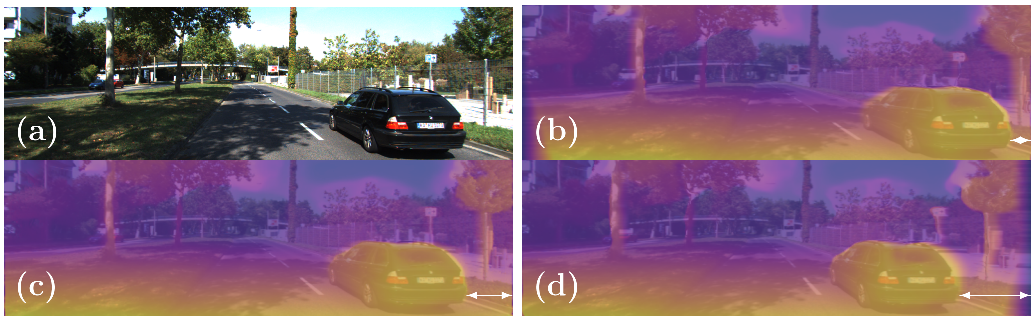


Figure 18.1: Overview of 3Net. a) Given a single reference image from KITTI 2015 training set [155], our network learns depth representations according to two additional points of view on left (b) and right (d) of the input (a), enabling to infer a more accurate depth map (c). White arrows highlight the different points of view.

in specific image regions inherited from the stereo setup (e.g., the left border using the left image as the reference) and in occluded areas. The post-processing step proposed in [69] partially compensates for these artifacts. However, it requires a double forward of the input image and its horizontally flipped version thus obtaining two predictions with artifacts, respectively, on the left and right side of depth discontinuities. Such issues are softened in the final map at the cost of doubling processing time and memory footprint.

In this chapter, we propose to explicitly take into account these artifacts training our network on imagery acquired by a trinocular setup. By assuming the availability of three horizontally aligned images at training time, our network learns to process the frame in the middle and produce inverse depth (i.e., disparity) maps according to all the available viewpoints. By doing so, we can attenuate the aforementioned occlusion artifacts because they occur in different regions of the estimated outputs. However, since trinocular setups are generally uncommon and hence datasets seldom available, we will show how to rely on popular stereo datasets such as CityScapes [47] and KITTI [67] to enforce our trinocular training assumption. Experimental results clearly prove that, deploying stereo pairs with a smart strategy aimed at emulating a trinocular setup, our Three-view Network (3Net) is able anyway to learn a three-view representation of the scene as shown intuitively in Figure 18.1 and how it leads to more robust monocular depth estimation compared to state-of-the-art methods trained on the same binocular stereo pair with a conventional

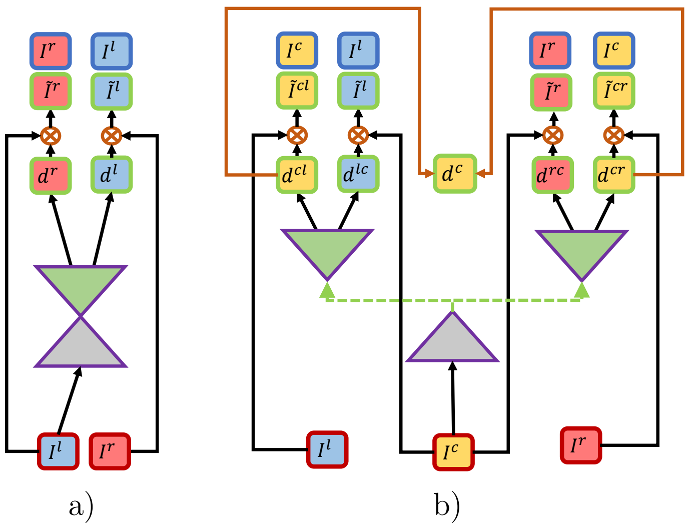


Figure 18.2: Training frameworks enforcing a) binocular [69] and b) trinocular assumptions.

paradigm. Figure 18.1 highlights the behavior of 3Net: we can see how disparity maps (b) and (d), from the point of view of two frames respectively on the left and right side of the reference image, show mirrored artifacts in occluded regions. Combining the two opposite views enables to compensate for these issues and produces a more accurate map (c) centered on the reference frame. Please note that KITTI does not explicitly contain trinocular views as those shown in Figure 18.1 and that this behavior is learned by 3Net trained only on standard binocular data. Indeed, images and depth maps in (b) and (d) are inferred by our network. Exhaustive experimental results on the KITTI 2015 stereo dataset [155] and the Eigen split [60] of the KITTI dataset [67] clearly show that 3Net, trained on standard binocular stereo pairs, improves state-of-the-art methods for unsupervised monocular depth estimation, regardless of the cues deployed for training.

18.2 Method overview

In this section, we propose a framework aimed at enforcing a trinocular assumption for training in an unsupervised manner a network for monocular depth estimation. We will outline the rationale behind this choice and the differences with known techniques in the

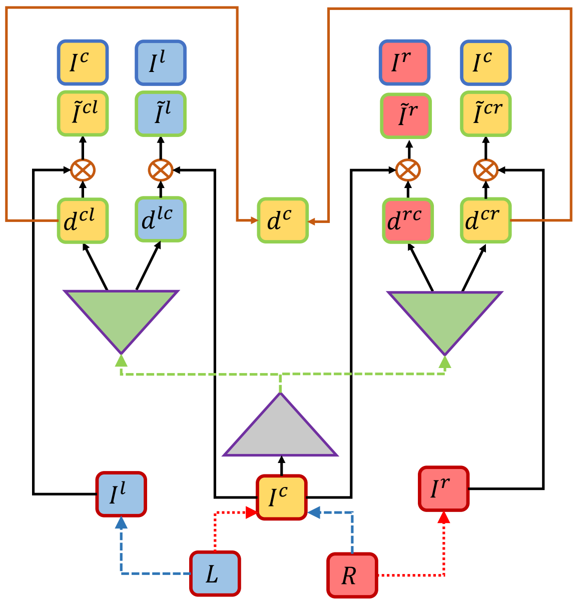


Figure 18.3: Scheme of interleaved training. A binocular stereo pair is used to train the network enforcing a trinocular assumption by first setting $L \rightarrow I^l$, $R \rightarrow I^c$ (blue arrows) and optimizing the model according to losses on $\tilde{I}^l$, $\tilde{I}^{cl}$, then setting $L \rightarrow I^c$, $R \rightarrow I^r$ (red arrows) and optimizing the model according to losses on $\tilde{I}^{cr}$, $\tilde{I}^r$.

literature. Then, deploying a conventional binocular stereo dataset, we will show how our strategy allows advancing state-of-the-art.

18.2.1 Trinocular assumption and network design

While traditional depth-from-mono frameworks learn to estimate $d(I)$ from an input image I by minimizing the prediction error with respect to a ground truth map $\hat{d}(I)$ whose pixels are labelled with real depth measurements, the introduction of image-reconstruction based losses moved this task to an unsupervised learning paradigm. In particular, estimated depth is used to project across different points of view exploiting 3D geometry and camera pose thus obtaining supervision signals through the minimization of the re-projection error. According to the literature, the training methodology based on images acquired with a stereo camera, as in [64, 69], removes the need to infer pose estimation required when gathering data with a single unconstrained camera.

Coaching a CNN to infer depth emulating a stereo system for training introduces artifacts in the learned representation (i.e., disparity) intrinsically because of well-known issues arising when dealing with pixels having no direct matches across the two images, such as on left border or occlusions. Godard et al. [69] deal with this problem using a simple, yet effective, trick. By processing a horizontally flipped input image and then back-flipping the result, artifacts will appear on the opposite side w.r.t. the result obtained on the un-flipped frame (e.g., on the right border rather than on the left). Thus, combining the two predictions allows removing artifacts partially. Nevertheless, this strategy requires two forwards, hence doubling memory footprint and runtime, which would not be necessary if the CNN could learn to estimate disparity concerning a frame acquired on the left w.r.t reference image. Guided by this intuition, we rethink the training protocol of [69] to exploit a trinocular configuration, on which the image we want to learn the depth of is the central frame of three horizontally aligned points of view. Figure 18.2 gives an overview of our framework b) and the one by Godard et al. [69] leveraging binocular stereo a). While a) trains the network to estimate a depth representation for I^l by means of disparity map d^l, used to warp I^r to $\tilde{I}^l$ and measure the appearance difference with I^l, we process I^c to obtain d^{lc} and d^{cr}, disparity maps assuming as target I^l and I^r, then we warp these latter two images to obtain $\tilde{I}^{cl}$ and $\tilde{I}^{cr}$ to finally compute supervision signals as re-projection error w.r.t. I^c. Finally, in our framework, d^{lc} and d^{cl} are combined to attenuate occlusions and obtain the final d^c from a single forward pass, conversely to [69] which requires two forwards. Eventually, as [69] estimates d^r to enforce losses between $\tilde{I}^r$,I^r and the LRC consistency, our network generates d^{lc} and d^{rc} to exploit losses between $\tilde{I}^l$,I^l and $\tilde{I}^r$,I^r.

Figure 18.2 also highlights a further main difference between the two frameworks. While a traditional UNet architecture is used by previous works in literature [69, 293], we build two separate decoders respectively in charge of estimating d^{cl} and d^{cr} separately. This strategy adds a negligible overhead regarding memory and runtime requirements, being the encoder the most computationally expensive module of the framework (i.e., the decoder mostly applies upsampling operations). According to our experiments, training a single decoder to infer a disparity representation for both points of view yields slightly worse results.

18.2.2 Interleaved training for binocular images

To effectively learn mirrored representation and compensate for occlusions/borders, the framework outlined so far relies on a set of three horizontally aligned images at training time. Although sensors designed to acquire such imagery are currently available, for instance the aforementioned Bumblebee XB3, it is still quite uncommon to find publicly available images obtained in such configuration. Indeed, in this sense, the Oxford RobotCar dataset [141] represents an exception providing a large amount of street scenes captured with the trinocular XB3 sensor. Unfortunately, the provided calibration parameters only allow obtaining aligned views between left-right and center-right cameras, hence not permitting to align the three views as we desire. Nonetheless, we describe in this section how to train our framework leveraging the proposed trinocular assumption with a much more common binocular setup (e.g., KITTI dataset). Given a stereo pair made of images L and R, Figure 18.3 depicts how to enforce the trinocular assumption by scheduling an *interleaved training* of the network. We update the parameters of the network by optimizing its four outputs d^{cl}, d^{lc}, d^{rc} and d^{cr} in two steps:

1. Firstly, we assign L to I^l and R to I^c as shown by the blue arrows in Figure 18.3. In other words, we assume that the stereo pair represents the left and center images of a *virtual* trinocular system in which the right frame is missing. In this case, we use as supervision signal the reconstruction error between $\tilde{I}^{cl}, I^c$ and $\tilde{I}^l, I^l$, producing gradients that flow to the left decoder and the encoder.

2. Then, as shown by the red arrows in Figure 18.3 we change the role of L and R assuming them, respectively, as I^c and I^r. In this phase, we suppose to have the center and right images available hence implicitly assuming that in our virtual trinocular system the left image is missing. Thus, using the supervision given by

re-projection errors on pairs $\tilde{I}^r, I^r$ and $\tilde{I}^{cr}, I^c$, we optimize the parameters of the right decoder and the (shared) encoder.

It is worth to note that, following this protocol, every time we run a training iteration on a stereo pair the network learns all the depth representations output of our framework. Moreover, the two learned disparity pairs from the two views are obtained according to the same baseline (i.e., the same of the training stereo pairs), making them consistent and hence easy to combine in d^c. Therefore, the network learns a trinocular representation even if it actually never sees the scene with such setup. Indeed, this strategy is very effective as supported by experimental evidence in Section 18.4.

18.3 Implementation details

In this section, we provide a detailed description of our framework, designed with the TensorFlow APIs.

18.3.1 Network architecture

For our 3Net we follow a quite established design strategy adopted by other methods in this field [69, 123, 131, 293], based on an encoder-decoder architecture. The peculiarity of our approach consists in two different decoders, as depicted in Figure 18.3, in charge of learning disparity representations w.r.t. two points of view located respectively on the left and right side of the input image. In our network, depicted in Figure 18.3, each decoder generates outputs at four different scales, respectively: full, half, quarter and $\frac{1}{8}$ resolution. As encoder, we tested VGG [222] and ResNet50 [79] to obtain the most fair and complete comparison w.r.t. [69], being it our baseline and state-of-the-art. To obtain the final map d_c, we merge the contribution of d_{cl} and d_{cr} using the same post-processing procedure applied in [69], thus keeping 5% left-most pixels from d_{cl}, 5% right-most from d_{cr} and averaging the remaining ones.

18.3.2 Training losses

We train 3Net to minimize a multi-component loss made of appearance, smoothness and consistency-check terms similarly to [69], namely $\mathcal{L}_{ap}, \mathcal{L}_{ds}$ and $\mathcal{L}_{lcr}$.

$$\mathcal{L}_{total} = \beta_{ap}(\mathcal{L}_{ap}) + \beta_{ds}(\mathcal{L}_{ds}) + \beta_{lcr}(\mathcal{L}_{lcr}) \tag{18.1}$$

The first term uses a weighted sum of SSIM [254] and L1 between all four warped pairs and real images as shown on top of Figure 18.3. The second applies an edge aware smoothness constraint to estimated disparities d^{cl}, d^{lc}, d^{rc} and d^{cr} as described in [69]. Finally, the consistency-check term includes left-right losses between pairs d^{cl}, d^{lc}.

$$\mathcal{L}_{lcr} = \mathcal{L}_{lr}(d^{cl}, d^{lc}) + \mathcal{L}_{lr}(d^{cr}, d^{rc}) \tag{18.2}$$

For a detailed description of $\mathcal{L}_{ap}, \mathcal{L}_{ds}$ and $\mathcal{L}_{lr}$ please refer to [69].

Thus, according to the interleaved training schedule described in Section 18.2.2, we optimize 3Net splitting the function 18.1 into two sub-losses $\mathcal{L}_{p_1}, \mathcal{L}_{p_2}$ deployed in the two different phases:

$$\begin{aligned}\mathcal{L}_{p_1} =& \beta_{ap}(\mathcal{L}_{ap}(\tilde{I}^{cl}, I^c) + \mathcal{L}_{ap}(\tilde{I}^{l}, I^l)) \\ &+ \beta_{ds}(\mathcal{L}_{ds}(d^{cl}, I^c) + \mathcal{L}_{ds}(d^{lc}, I^l)) \\ &+ \beta_{lcr}(\mathcal{L}_{lr}(d^{cl}, d^{lc})\end{aligned} \tag{18.3}$$

$$\begin{aligned}\mathcal{L}_{p_2} =& \beta_{ap}(\mathcal{L}_{ap}(\tilde{I}^{cr}, I^c) + \mathcal{L}_{ap}(\tilde{I}^{r}, I^r)) \\ &+ \beta_{ds}(\mathcal{L}_{ds}(d^{cr}, I^c) + \mathcal{L}_{ds}(d^{rc}, I^l)) \\ &+ \beta_{lcr}(\mathcal{L}_{lr}(d^{cr}, d^{lr})\end{aligned} \tag{18.4}$$

We also evaluated an additional loss term $\mathcal{L}_{cc} = |d^{cl} - d^{cr}|$ to enforce consistency

			Proposed method		Lower is better		Higher is better			
Method	Train set	Abs Rel	Sq Rel	RMSE	RMSE log	D1-all	δ <1.25	$\delta < 1.25^2$	$\delta < 1.25^3$	Forwards
Godard et al. [69]	K	0.124	1.388	6.125	0.217	30.272	0.841	0.936	0.975	×1
3Net	K	0.119	1.201	5.888	0.208	31.851	0.844	0.941	0.978	×1
Godard et al. [69] + pp	K	0.117	1.177	5.804	0.206	29.945	0.848	0.943	0.977	×2
3Net + pp	K	0.114	1.088	5.756	0.203	31.141	0.848	0.944	0.979	×2
Godard et al. [69]	CS+K	0.104	1.070	5.417	0.188	25.523	0.875	0.956	0.983	×1
3Net	CS+K	0.101	0.954	5.211	0.181	24.632	0.875	0.958	0.985	×1
Godard et al. [69] + pp	CS+K	0.100	0.934	5.141	0.178	25.077	0.878	**0.961**	**0.986**	×2
3Net + pp	CS+K	**0.097**	**0.893**	**5.079**	**0.176**	**23.867**	**0.881**	**0.961**	**0.986**	×2

Table 18.1: Comparison between 3Net and [69], both using VGG as encoder, on KITTI 2015 training dataset [155].

between depth representation centered on I^c, being the baseline equal on both directions. However, this term propagates occlusions artifacts between the two depth maps leading to worse results. We point out that despite the interleaved training protocol outlined, in any phase the outcome of 3Net always consists of four depth maps d^{cl}, d^{lc}, d^{rc} and d^{cr}. Of course, this happens at testing/inference time as well, when 3Net is fed with a single image. Considering that decoders outputs depth maps at four scales, all losses are computed for each of them as in [69].

18.3.3 Training protocol

We assume as baseline the framework proposed by Godard et al. [69] using a binocular setup for unsupervised training. For a fair comparison, we train our models following the same guidelines reported in [69]. In particular, we use CityScapes [47] (CS) and KITTI raw sequences [67] datasets for training, this latter sub-sampled according to two training splits of data [60, 69] to be able to compare our results with any recent works in this field using unsupervised learning. We refer to these two subsets as KITTI split (K) and Eigen split (E) [60]. The three training sets count respectively about 23k, 29k and 22.6k stereo pairs. As pointed out by first works using image reconstruction losses [69, 293], training on different datasets helps the network to achieve higher-quality results. Therefore, to better assess the performance of each method, we report experimental results training the networks on K or E. Moreover, we also report results training on CityScapes and then fine tuning on K or E (respectively, referred to as CS+K and CS+E in the tables). Consistently

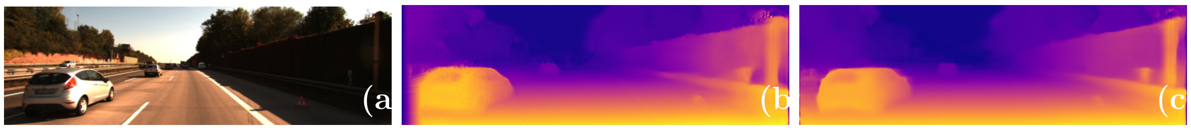


Figure 18.4: Depth maps predicted from input image (a) by Godard et al. [69] (b) and 3Net (c) running a single forward pass.

with [69], we run 50 epochs of training on each single dataset using a batch size of 8 and input resized to 256×512. We use Adam optimizer [118] with $\beta_1 = 0.9, \beta_2 = 0.999$ and $\varepsilon = 10^{-8}$, setting an initial learning rate of 10^{-4} halved after 30 and 40 epochs. We maintain the same hyperparameters configuration for β_{ap}, β_{ds} and β_{lrc} defined in [69] and the same data augmentation procedure as well.

18.4 Experimental results

In this section, we assess the performance of our 3Net framework with respect to state-of-the-art. In all our tests, we report 7 main metrics measuring the average depth error (Abs Rel, Sq Rel, RMSE and RMSE log, the lower the better) and three accuracy scores ($\delta < 1.25, \delta < 1.25^2$ and $\delta < 1.25^3$, the higher the better). First, we report experiments on the K split assuming Godard et al. [69] as baseline. Then, we exhaustively compare 3Net with top performing unsupervised frameworks for depth-from-mono estimation, highlighting how our proposal is state-of-the-art. It is worth stressing that the proposed interleaved training procedure of 3Net, described in Section 18.2.2, allows for a fair comparison with any other method included in our evaluation being all trained exactly on the same (binocular) datasets. Finally, we report qualitative results concerning the trinocular representation learned by 3Net.

18.4.1 KITTI split

Table 18.1 reports experimental results on the KITTI 2015 stereo dataset [155]. The evaluation was carried out on 200 stereo pairs with available high quality ground truth

			Proposed method		Lower is better		Higher is better		
Method	Supervision	Train set	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Kumar et al. [48] (photo. + adv.)	Temporal	E	0.211	1.980	6.154	0.264	0.732	0.898	0.959
Zhou et al. [293]	Temporal	E	0.208	1.768	6.856	0.283	0.678	0.885	0.957
Zhou et al. [293] updated [270]	Temporal	E	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Mahjourian et al. [142]	Temporal	E	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Yin et al. [270] GeoNet	Temporal	E	0.164	1.303	6.090	0.247	0.765	0.919	0.968
Yin et al. [270] GeoNet ResNet50	Temporal	E	0.155	1.296	5.857	0.233	0.793	0.931	0.973
Wang et al. [249]	Temporal	E	0.151	1.257	5.583	0.228	0.810	0.936	**0.974**
Poggi et al. [186] PyD-Net (200)	Stereo	E	0.153	1.363	6.030	0.252	0.789	0.918	0.963
Godard et al. [69]	Stereo	E	0.148	1.344	5.927	0.247	0.803	0.922	0.964
Zhan et al. [281]	Stereo+Temp.	E	0.144	1.391	5.869	0.241	0.803	0.928	0.969
3Net	Stereo	E	0.142	1.207	5.702	0.240	0.809	0.928	0.967
Godard et al. [69] ResNet50	Stereo	E	0.133	1.142	5.533	0.230	0.830	0.936	0.970
3Net ResNet50	Stereo	E	0.129	0.996	5.281	0.223	0.831	0.939	**0.974**
Godard et al. [69] ResNet50 + pp	Stereo	E	0.128	1.038	5.355	0.223	0.833	0.939	0.972
3Net ResNet50 + pp	Stereo	E	**0.126**	**0.961**	**5.205**	**0.220**	**0.835**	**0.941**	**0.974**
Zhou et al. [293]	Temporal	CS+E	0.198	1.836	6.565	0.275	0.718	0.901	0.960
Mahjourian et al. [142]	Temporal	CS+E	0.159	1.231	5.912	0.243	0.784	0.923	0.970
Yin et al. [270] GeoNet ResNet50	Temporal	CS+E	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Wang et al. [249]	Temporal	CS+E	0.148	1.187	5.496	0.226	0.812	0.938	0.975
Poggi et al. [186] PyD-Net (200)	Stereo	CS+E	0.146	1.291	5.907	0.245	0.801	0.926	0.967
Godard et al. [69]	Stereo	CS+E	0.124	1.076	5.311	0.219	0.847	0.942	0.973
3Net	Stereo	CS+E	0.117	0.905	4.982	0.210	0.856	0.948	0.976
Godard et al. [69] ResNet50	Stereo	CS+E	0.121	1.037	5.212	0.216	0.854	0.944	0.973
3Net ResNet50	Stereo	CS+E	0.113	0.885	4.898	0.204	0.862	0.950	0.977
Godard et al. [69] ResNet50 + pp	Stereo	CS+E	0.114	0.898	4.935	0.206	0.861	0.949	0.976
3Net ResNet50 + pp	Stereo	CS+E	**0.111**	**0.849**	**4.822**	**0.202**	**0.865**	**0.952**	**0.978**

Table 18.2: Evaluation on the KITTI dataset [67] using the split of Eigen et al. [60], with maximum depth set to 80m. Results concerned with state-of-the-art techniques for unsupervised monocular depth estimation leveraging video sequences (Temporal), binocular stereo pairs (Stereo) and both cues (Stereo+Temp.).

disparity annotations. Additionally, being the outputs of [69] and 3Net disparity maps, in our evaluation we include the D1-all score representing the percentage of pixels having a disparity error larger than 3.

We compare the raw output d_c of our network with the map predicted by Godard et al. with and without post-processing [69] (namely "+pp" in the table) running, respectively, a single or two forwards of the network. Moreover, since 3Net can benefit from the same refinement technique by running two predictions on I^c and its horizontally flipped version, we also provide results for our network applying the same post-processing. Therefore, we estimate post-processed d^{cl} and d^{cr} before combining them into d^c. Anyway, we report for clarity in the last column of the table, the number of forwards required by each entry.

As reported on the first two rows of Table 18.1, training the networks on KITTI data only, our method outperforms the competitor on all metrics except D1-all when running a single forward and it performs very similar to the post-processed version of [69] reported in the third row of the table. Rows 3 and 4 highlight that, performing two forwards and post-processing, 3Net + pp outperforms Godard et al. + pp again on all metrics except D1-all.

Previous works in literature [69, 142, 249, 270, 281, 293] proved that transfer learning from CityScape dataset [47] to KITTI is beneficial and leads to more accurate depth estimation, thus we follow this guideline training on CS+K as well. The last four rows of Table 18.1 compare both frameworks with and without post-processing. Without post-processing, we can notice how pre-training on CityScapes dataset allows 3Net to outperform [69] on all metrics including D1-all. In the last two rows, applying post-processing to the output of both models, 3Net outperforms the competitor on all metrics tying on $\delta < 1.25^2$ and $\delta < 1.25^3$. Figure 18.4 qualitatively shows depth maps predicted by [69] (b) and 3Net (c) without applying any post-processing to better perceive the improvements lead by our framework.

Summarizing, experiments on the KITTI split highlighted how enforcing the trinocular assumption is more effective than leveraging a conventional stereo paradigm for training. Moreover, these results prove that stereo pairs can be used in a smarter way following our interleaving strategy.

18.4.2 Eigen split

Table 24.1 reports evaluation with the split of data of Eigen et al. [60], made of 697 images and relative depth measurements acquired with a Velodyne sensor. The table collects results concerning most recent works addressing unsupervised monocular depth estimation. For each method, we indicate the kind of supervision it leverages on: monocular sequences (*Temporal*), stereo pairs (*Stereo*) or stereo sequences (*Stereo+Temp.*). We report results either training on E only or on CS+E, allowing to compare our scores with state-of-the-art approaches. On top, we report results for models trained on the Eigen split of data. For GeoNet [270], Godard et al. [69] and our method we report results for both VGG and ResNet50 encoders. We can notice that, in general, methods trained

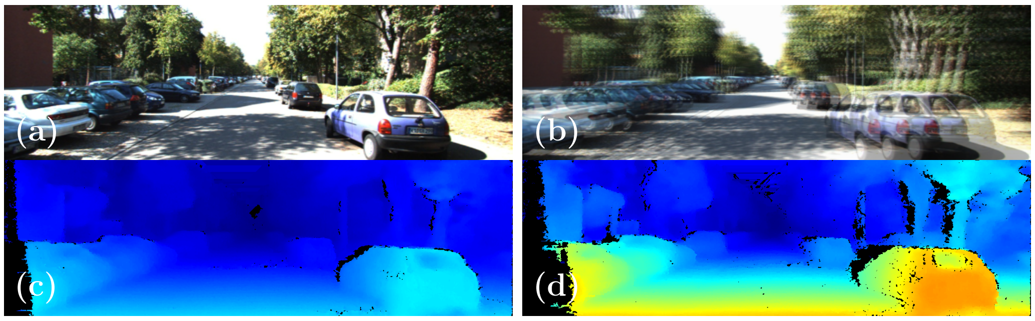


Figure 18.5: Qualitative example of learned trinocular setup. Given a single, input image (a), 3Net can generate two additional points of view, shown superimposed to the real frame in (b). Running traditional stereo algorithms [82], assuming the left-most generated frame as reference, allows to obtain disparity maps with narrow (c) and wide (d) baseline (encoded with colormap jet). We point out that the center frame (a) is the only real image.

using stereo data usually outperform those trained on monocular video sequences, as evident from recent literature [69, 142, 249, 270, 281, 293]. Zhan et al. [281] leveraging temporally adjacent stereo frames outperform, on most metrics, [69]. Nevertheless, 3Net achieves better scores except for $\delta < 1.25^3$ still without exploiting temporal supervision. This proves that a smarter deployment of binocular training samples, i.e. by applying our interleaved training to fulfill trinocular hypothesis, is an effective alternative to sequence supervision. It is worth to note that Wang et al. [249] obtain better scores on most metrics (RMSE, RMSE log and δ metrics) w.r.t. [69] and 3Net with the VGG encoder. However, by switching to the ResNet50 encoder, Godard et al. [69] overtakes most recent works that use *Time* supervision [249, 270] with and without post-processing. Systematically, 3Net always outmatches [69] and consequently all its competitors. In particular, we point out that 3Net ResNet50 without post-processing already achieves some better scores compared to Godard et al. ResNet50 + pp performing, respectively, a single and a double forward.

On the bottom of Table 24.1, we resume results achieved by models trained on CS+E. We observe the same trend highlighted in the previous experiments, being [69] and our proposal the most effective solutions for this task thanks to stereo supervision. In equal conditions, i.e. same encoder and number of forwards, 3Net always outperforms the framework of Godard et al. exploiting the trinocular assumption. Moreover, the proposed technique leads to major improvements such that 3Net VGG outperforms ResNet50 model by Godard et al. [69] (rows 20^{th} and 21^{st}), 3Net ResNet50 without post-processing achieves more accurate results than the best configuration ResNet50 + pp [69] (rows 22^{nd} and 23^{rd}) and finally 3Net ResNet50 + pp outmatches all known frameworks for unsupervised depth-from-mono estimation. These facts clearly highlight that our proposal is state-of-the-art.

It is important to underline that the availability of a *real* trinocular rig would most probably allow training a more accurate model, given the larger amount of images w.r.t. a binocular stereo rig. The interleaved training proposed in this work allows to overcome the lack of trinocular training samples using binocular pairs and also allows for a more fair comparison with other techniques leveraging this latter configuration only. This fact proves that the effectiveness of our strategy is due to the rationale behind it and not driven by a more extensive availability of data.

18.5 View synthesis

Finally, we show through qualitative images some outcomes of 3Net obtainable exploiting the embodied trinocular assumption. A peculiar experiment allowed by our framework consists of generating three horizontally aligned views from a single input image. This feature is possible thanks to estimated d^{lc} and d^{rc}, used to warp the input image towards two new viewpoints, respectively, on the left and the right. In other words, given I^c at test time we compute $\tilde{I}^l$ and $\tilde{I}^r$, producing a trinocular setup of horizontally aligned images. Figure 18.5 shows an example of a single frame (a) taken from the KITTI dataset and how our network generates the three views superimposed in (b). These views effectively enable to realize a multi-baseline setup. Thus we can run any stereo algorithm between the possible pairs. For instance, we run the Semi-Global Matching algorithm (SGM) [82] between $\tilde{I}^l$ and I^c, showing the results in Figure 18.5 (c), then we run SGM between $\tilde{I}^l$ and $\tilde{I}^r$ obtaining the disparity map shown in (d). The two disparity maps assume the same frame as the reference image ($\tilde{I}^l$) and two different targets, according to two different *narrow* and *wide* virtual baseline. The shorter baseline is learned from the KITTI acquisition rig while the longer one is inherited by our trinocular assumption although actually, it does not exist at all in the training set. This fact can be perceived by looking at the different disparity ranges encoded, with colormap jet, on (c) and (d). This feature of our network paves the way to exciting novel developments. For instance, a conceivable application would consist in the synthesis of *augmented* stereo pairs to train CNNs for disparity inference [69, 151] or to improve recent techniques such as single view stereo [137].

18.6 Conclusions

In this chapter, we proposed a novel methodology for unsupervised training of a depth-from-mono CNN. By enforcing a trinocular assumption, we overcome some limitations due to binocular stereo images used as supervision and obtain a more accurate depth estimation with our 3Net architecture. Although three horizontally aligned views are seldom available, we proposed an interleaved training protocol allowing to leverage on traditional binocular datasets. This latter technique also ensures for a fair comparison w.r.t. all previous works and allows us to prove that 3Net outperforms all unsupervised techniques known in the literature, establishing itself as state-of-the-art. Moreover, 3Net learns a trinocular representation of the world, making it suitable for image synthesis purposes and other interesting future developments.

Chapter 19

Geometry meets semantics for semi-supervised monocular depth estimation

The content of this chapter has been presented at the Asian Conference on Computer Vision (ACCV 2018) - "Geometry meets semantics for semi-supervised monocular depth estimation" [277].

19.1 Introduction

In this chapter, we propose to train a CNN architecture to perform both semantic segmentation and depth estimation from a single image. By optimizing our model jointly on the two tasks, we enable it to learn a more effective feature representation which yields improved depth estimation accuracy. We rely on unsupervised image re-projection loss [69] to pursue depth prediction whilst we let the network learn semantic information from the observed scene by supervision signals from pixel-level ground truth semantic maps. Thus, with respect to recent work [69], our proposal requires semantically annotated imagery, thereby departing from a totally unsupervised towards a semi-supervised learning

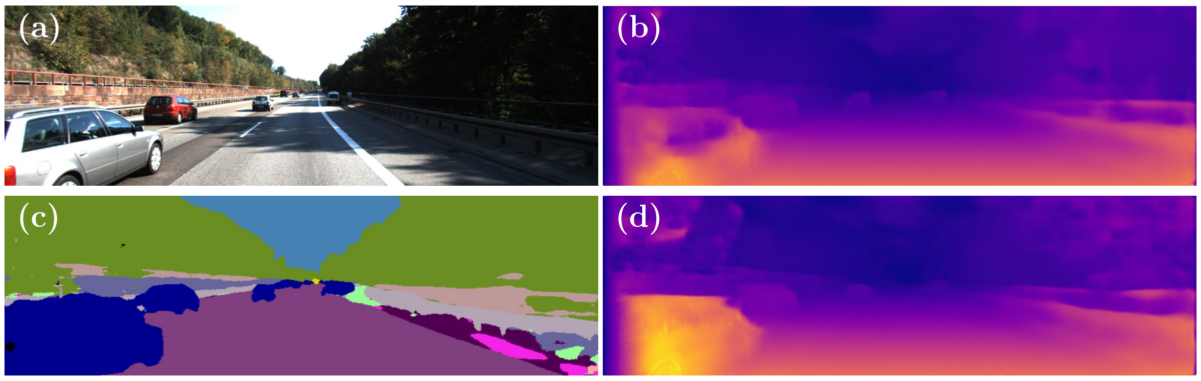


Figure 19.1: Joint depth from mono and semantic segmentation. (a) Input image, (b) depth map by state-of-the-art method [69], (c) semantic and (d) depth maps obtained by our network.

paradigm (*i.e.* unsupervised for depth and supervised for semantics). Yet, though manual annotation of per-pixel semantic labels is tedious, it is much less prohibitive than collecting ground truth depths. Besides, while the former task may be performed off-line after acquisition, as recently proposed for some images of the KITTI dataset [13], one may very unlikely obtain depth labels out of already collected frames.

To the best of our knowledge, this work is the first to propose integration of unsupervised monocular depth estimation with supervised semantic segmentation. By applying this novel paradigm, we improve a state-of-the-art encoder-decoder depth estimation architecture [69] according to two main contributions:

- we propose to introduce an additional decoder stream based on the same features as those deployed for depth estimation and trained for semantic segmentation; thereby, the overall architecture is trained to optimize both tasks jointly.

- we propose a novel loss term, the *cross-domain discontinuity* loss $\mathcal{L}_{cdd}$, aimed at enforcing spatial proximity between depth discontinuities and semantic contours.

Experimental results on the KITTI dataset prove that tackling the two tasks jointly does improve monocular depth estimation. For example, Fig. 19.1 suggests how recognizing objects like cars (c) can significantly ameliorate depth estimation (d) with respect to a depth-from-mono approach lacking any awareness about scene semantics (b). It is also worth highlighting that, unlike all previous unsupervised frameworks in this field, our proposal delivers not only the depth map (Fig.19.1 (d)) but also the semantic segmentation of the input image (Fig.19.1, (c)) by an end-to-end training process.

19.2 Proposed method

In this section, we present our proposal for joint semantic segmentation and depth estimation from a single image. We first explain the main intuitions behind our work, then we describe the network architecture and the loss functions deployed in our deep learning framework.

Estimating the distance of objects from a camera through a single acquisition is an ill-posed problem. Despite this lack of information, modern deep learning monocular frameworks achieved astounding results by learning effective feature representations from the observed environment. Common to latest work in this field [60, 69, 123, 293] is the design of deep encoder-decoder architectures, with a first contractive portion progressively decimating image dimensions to reduce the computational load and increase the receptive field, followed by an expanding portion which restores the original input resolution. In particular, the encoding layers learn a high level feature representation crucial to infer depth. Although it is hard to tell what kind of information the network is actually learning at training time, we argue semantic to play an important role. Recent works like [69, 293] somehow support this intuition. Indeed, although the authors trained and evaluated their depth estimators on the KITTI dataset [155], a preliminary training on CityScapes [47] turned out beneficial to achieve the best accuracy with both frameworks, despite the very different camera setup between the two datasets. Common to the datasets is, in fact, the kind of sensed environment and, thus, the overall semantics of the scenes under perception. This observation represents the main rationale underpinning our proposal. By explicitly training the network to learn the semantic context of the sensed environment

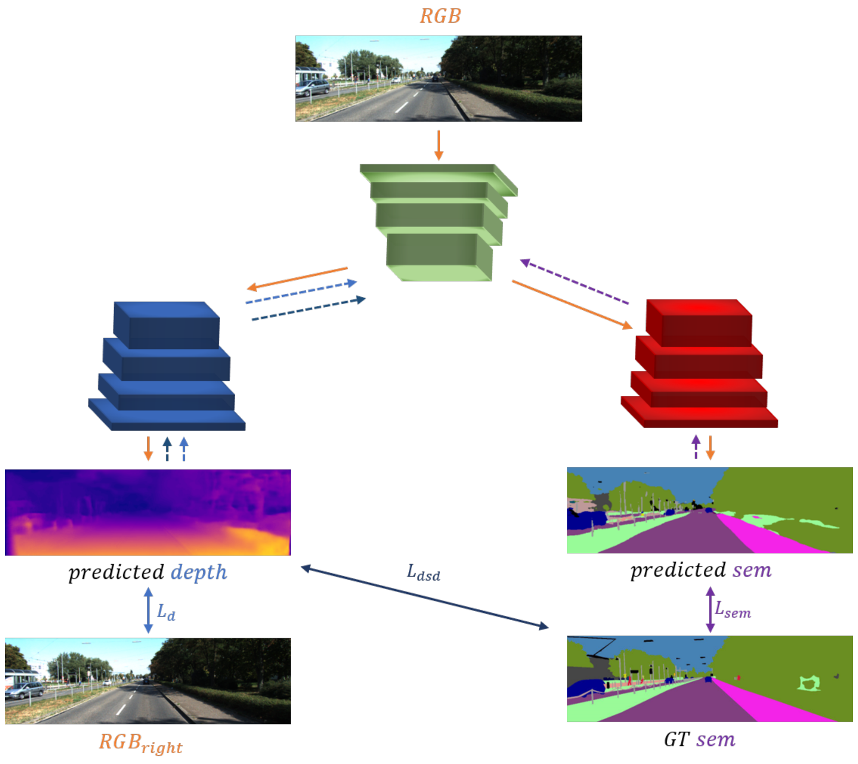


Figure 19.2: Schematic representations of the proposed network architecture and semi-supervised learning framework. A single encoder (green) is shared between a depth (blue) and a semantic (red) decoder. The depth decoder is optimized to minimize $\mathcal{L}_d$ and $\mathcal{L}_{cdd}$, the semantic decoder to minimize $\mathcal{L}_s$.

we shall expect to enrich the feature representation resulting from the encoding module and thus obtain a more accurate depth estimation. This may be realized by a deep model in which a single encoder is shared between two decoders in charge of providing, respectively, a depth map and a semantic segmentation map. Accordingly, minimization of the errors with respect to pixel-level semantic labels provides gradients that flow back into the encoder at training time, thereby learning a shared feature representation aware of both depth prediction as well as scene semantics. According to our claim, this should turn out conducive to better depth prediction.

Inspired by successful attempts to predict depth from a single image, we design a suitable encoder-decoder architecture for joint depth estimation and semantic segmentation. The encoder is in charge of learning a rich feature representation by increasing the receptive field of the network while reducing the input dimension and computational overhead. Popular encoders for this task are VGG [222] and ResNet50 [79] . The decoder restores the original input resolution by means of up-sampling operators followed by 3×3 convolutions linked by means of skip connections with the encoder at the corresponding resolution. As illustrated in Fig. 19.2, to infer both depth and semantics we keep relying on a single encoder (green) and replicate the decoder to realize a second estimator. The two decoders (blue, red) do not share weights and are trained to minimize different losses, which deal with the depth prediction (blue) and semantic segmentation (red) tasks. While the two decoders are updated by different gradients flows, the shared encoder (green) is updated according to both flows, thereby learning a representation optimized jointly for the two tasks. We validate our approach by extending the architecture proposed by Godard et al.[69] for monocular depth estimation: the encoder produces two inverse depth (i.e., disparity) maps by processing the left image of a stereo pair. Then, the right image is used to obtain supervision signals by warping the left image according to the estimated disparities, as explained in the following section.

Figure 19.3 shows how the shared representation used to jointly tackle both tasks enables to reconstruct better shapes when estimating depth (e) thanks to the semantic context (d) learned by the network compared to standalone learning of depth (c) as in [69].

19.2.1 Loss functions

To train the proposed architecture, we rely on the following multi-task loss function

$$\mathcal{L}_{tot} = \alpha_d \mathcal{L}_d + \alpha_s \mathcal{L}_s + \alpha_{cdd} \mathcal{L}_{cdd} \tag{19.1}$$

which consists in the weighted sum of three terms, namely the *depth* ($\mathcal{L}_d$), *semantic* ($\mathcal{L}_s$) and *cross-domain discontinuity* ($\mathcal{L}_{cdd}$) terms. As shown in in Fig. 19.2, each term back-propagates gradients through a different decoder: in particular, $\mathcal{L}_d$ and $\mathcal{L}_{cdd}$ through

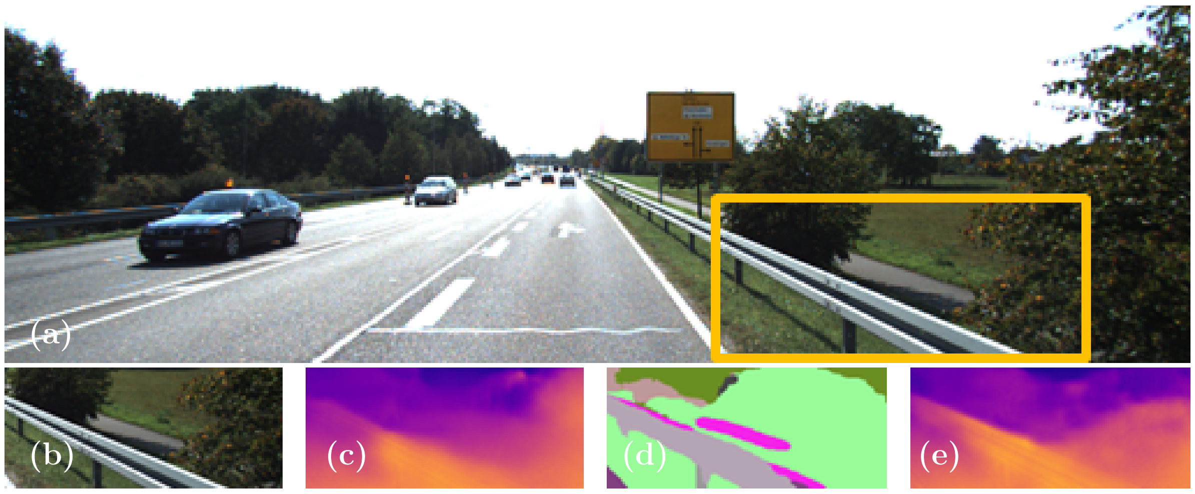


Figure 19.3: Example of improved depth estimation enabled by semantic knowledge. (a) input image, (b) region extracted from the scene, (c) depth map predicted by [69], depth (d) and semantic (e) maps predicted by our framework. We can clearly notice how the the structure of the guard rail is better preserved by our method (e) compared to [69] in (c).

the depth (blue) decoder whilst $\mathcal{L}_s$ through the semantic (red) decoder. All gradients then converge so to flow back into the shared (green) encoder.

Depth term

The depth term, $\mathcal{L}_d$, in our multi-task loss is computed according to the unsupervised training paradigm proposed by Godard et al.[69]:

$$\mathcal{L}_d = \beta_{ap}(\mathcal{L}^l_{ap} + \mathcal{L}^r_{ap}) + \beta_{ds}(\mathcal{L}^l_{ds} + \mathcal{L}^r_{ds}) + \beta_{lr}(\mathcal{L}^l_{lr} + \mathcal{L}^r_{lr}) \tag{19.2}$$

where the loss consists in the weighted sum of three terms, namely the *appearance*, *disparity smoothness* and *left-right consistency* terms. The first term measures the image re-projection error by means of the SSIM [254] and L1 difference between the original and warped images, I and $\tilde{I}$. The smoothness term penalizes large disparity differences between neighboring pixels along the x and y directions unless these occur in presence of strong intensity gradients in the reference image I. Finally, the left-right consistency enforces coherence between the predicted disparity maps, d^l and d^r, for left and right images. As proposed in [69], in our learning framework $\mathcal{L}_d$ is computed at four different scales.

Semantic term

The semantic term $\mathcal{L}_s$ within our total loss is given by the standard cross-entropy between the predicted and ground truth pixel-wise semantic labels:

$$\mathcal{L}_s = \mathcal{C}(p_t, \overline{p}_t) = H(p_t, \overline{p}_t) + KL(p_t, \overline{p}_t) \tag{19.3}$$

where H denotes the entropy and KL the $KL-$divergence. The semantic term, $\mathcal{L}_s$, is computed at full resolution only.

Cross-domain discontinuity term

We also introduce a novel cross-task loss term aimed at enforcing an explicit link between the two learning tasks by leveraging on the ground truth pixel-wise semantic labels to improve depth prediction. We found that the most effective manner to realize this consists in deploying the observation that depth discontinuities are likely to co-occur with semantic boundaries. Accordingly, we have designed the following *cross-domain discontinuity*, $\mathcal{L}_{cdd}$, term:

$$\mathcal{L}_{cdd} = \frac{1}{N}\sum_{i,j} sgn(|\delta_x sem^l_{i,j}|)e^{-||\frac{\delta_x d^l_{i,j}}{d^l_{i,j}}||} + sgn(|\delta_y sem^l_{i,j}|)e^{-||\frac{\delta_y d^l_{i,j}}{d^l_{i,j}}||} \tag{19.4}$$

where *sem* denotes the ground truth semantic map and d the predicted disparity map. Differently from the smoothness term $\mathcal{L}^l_{ds}$ in the disparity domain, the novel $\mathcal{L}_{cdd}$ term detects discontinuities between semantic labels encoded by the sign of the absolute value of the gradients in the semantic map. The idea behind this loss is that there should be a gradient peak between adjacent pixels belonging to different classes. Nevertheless, we do not care about its magnitude since the numeric labels do not have any mathematical meaning.

19.3 Experimental results

In this section, we compare the performance of our semi-supervised joint depth estimation and semantic segmentation paradigm with respect to the proposal by Godard et al.[69], which represents nowadays the undisputed state-of-the-art for unsupervised monocular depth estimation. As discussed in Sec. 19.2, our method as well as the baseline used in our experiments, *i.e.* [69], require rectified stereo pairs at training time. Suitable datasets for this purpose are thus CityScapes [47] and KITTI [67], which provide a large number of training samples, *i.e.* about 23k and 29k rectified stereo pairs respectively. However, our method requires also pixel-wise ground truth semantic labels at training time, which limits the actual amount of training samples available for our experiments. In particular, CityScapes includes about 3k finely annotated images, while the KITTI 2015 benchmark recently made available pixel-wise semantic ground truths for about 200 images [13]. Therefore, to carry out a fair evaluation of the actual contribution provided by semantic information in the depth-from-mono task to the baseline fully unsupervised approach, we trained both methods based on the reduced datasets featuring stereo pairs alongside with semantically annotated left frames.

19.3.1 Implementation details

We adhere to the original training protocol by Godard et al., scheduling 50 epochs on the CityScapes dataset and 50 further on the KITTI 2015 images. For quantitative evaluation, we split the KITTI 2015 dataset into train and test sets, providing more details in the next section. We train on 256×512 images using a batch dimension of 2, we set the previously introduced hyper-parameters as follows: $\alpha_d = 1$, $\alpha_s = 0.1$, $\alpha_{cdd} = 0.1$, $\beta_{ap} = 1$, $\beta_{lr} = 1$, $\beta_{ds} = \frac{1}{r}$ (being r the down-sampling factor at that resolution) and $\gamma = 0.85$. Models are trained using Adam optimizer [118], with $\beta_1 = 0.9$, $\beta_1 = 0.999$ and $\epsilon = 10^{-8}$. The initial learning rate is set to 10^{-4}, halved after 30 and 40 epochs. We perform data augmentation on input RGB images, in particular random gamma, brightness and color shifts sampled within the ranges [0.8,1.2] for gamma, [0.5,2.0] for brightness, and [0.8,1.2] for each color channel separately. Moreover we flip images horizontally with a probability of 50%. If the flip occurs, the right image in the stereo pair becomes the new reference image and we do not provide supervision signals from semantics (as right semantic maps are not available in the datasets). We implemented our network with both VGG and ResNet50 encoders, as in [69]. The semantic decoder adds about 20.5M parameters, resulting in nearly 50 and 79 million parameters for the two models (31 and 59, respectively, for [69]).

19.3.2 Monocular depth estimation: evaluation on KITTI 2015

We quantitatively assess the effectiveness of our proposal on the KITTI 2015 training dataset for stereo [155]. It provides 200 synchronized pairs of images together with ground truth disparity and semantic maps [13]. As already mentioned, to carry out a fair comparison between our approach and [69], we can use only these samples and thus the numerical results reported in our work cannot be compared directly with those in [69]. Then, we randomly split the 200 pairs from KITTI into 160 training samples and 40 samples used only for evaluation[1]. We measure the accuracy of the predicted depth maps after training for 50 epochs on CityScapes and then fine-tuning for 50 more epochs on the samples selected from KITTI.

Table 19.1 reports quantitative results using VGG or ResNet50 as backbone encoder. Each model, one per row in the table, is trained with four different strategies:

- $\mathcal{L}_d$ uses only the depth term as loss (*i.e.*, equivalently to the baseline approach by Godard et al.[69]).
- $\mathcal{L}_d$+$\mathcal{L}_s$ adds the semantic term to the depth term.
- $\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$ minimizes our proposed total loss function (Equation 19.1).

[1]The testing samples, belonging to the KITTI 2015 dataset, are: 000001, 000003, 000004, 000019, 000032, 000033, 000035, 000038, 000039, 000042, 000048, 000064, 000067, 000072, 000087, 000089, 000093, 000095, 000105, 000106, 000111, 000116, 000119, 000123, 000125, 000127, 000128, 000129, 000134, 000138, 000150, 000160, 000161, 000167, 000174, 000175, 000178, 000184, 000185 and 000193.

					Lower is better		Higher is better		
	Encoder	pp	Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
$\mathcal{L}_d$ [69]	VGG		0.160	2.707	7.220	0.239	0.837	0.928	0.966
$\mathcal{L}_d$+$\mathcal{L}_s$	VGG		0.155	2.511	6.968	**0.234**	0.841	**0.931**	**0.968**
$\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$	VGG		**0.154**	**2.453**	**6.949**	0.235	**0.844**	**0.931**	0.967
$\mathcal{L}_d$+$\mathcal{L}_{cdd}$	VGG		0.161	2.758	7.128	0.240	0.841	0.928	0.964
$\mathcal{L}_d$ [69]	VGG	✓	0.149	2.203	6.582	0.223	0.844	0.936	0.972
$\mathcal{L}_d$+$\mathcal{L}_s$	VGG	✓	0.147	2.229	6.583	0.223	0.847	**0.938**	**0.972**
$\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$	VGG	✓	**0.145**	**2.040**	**6.362**	**0.221**	**0.849**	**0.938**	0.971
$\mathcal{L}_d$+$\mathcal{L}_{cdd}$	VGG	✓	0.150	2.278	6.539	0.225	0.843	0.934	0.970
$\mathcal{L}_d$ [69]	ResNet		0.159	2.411	6.822	0.239	0.830	0.930	0.967
$\mathcal{L}_d$+$\mathcal{L}_s$	ResNet		0.152	2.385	6.775	0.231	0.843	0.934	0.970
$\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$	ResNet		**0.143**	**2.161**	**6.526**	**0.222**	**0.850**	**0.939**	**0.972**
$\mathcal{L}_d$+$\mathcal{L}_{cdd}$	ResNet		0.155	2.282	6.658	0.232	0.840	0.932	0.968
$\mathcal{L}_d$ [69]	ResNet	✓	0.148	2.104	6.439	0.224	0.839	0.936	0.972
$\mathcal{L}_d$+$\mathcal{L}_s$	ResNet	✓	0.144	2.050	6.351	0.220	0.849	0.938	0.972
$\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$	ResNet	✓	**0.136**	**1.872**	**6.127**	**0.210**	**0.854**	**0.945**	**0.976**
$\mathcal{L}_d$+$\mathcal{L}_{cdd}$	ResNet	✓	0.144	1.973	6.199	0.217	0.849	0.940	0.975

Table 19.1: Ablation experiments on KITTI 2015 evaluation split, using different configurations of losses, encoders and post-processing (pp). Best setup highlighted in bold for each configuration.

- $\mathcal{L}_d$+$\mathcal{L}_{cdd}$ minimizes only the losses dealing with the depth decoder.

The table provides results yielded by the four considered networks according to standard performance evaluation metrics [69].

This ablation highlights how introducing the second decoder trained to infer semantic segmentation maps, significantly improves depth prediction according to all performance metrics for both type of encoder. Moreover, adding the cross-domain discontinuity term, $\mathcal{L}_{cdd}$, leads in most cases to further improvements. On the other hand, minimizing $\mathcal{L}_d$ and $\mathcal{L}_{cdd}$ alone leads to inferior performance compared to the baseline method. We obtain the best configuration according to all metrics using ResNet50 when both $\mathcal{L}_s$ and $\mathcal{L}_{cdd}$ are minimized alongside with the depth term $\mathcal{L}_d$.

Moreover, we also evaluated the output obtained by all models after performing the post-processing step proposed by [69], that consists in forwarding both the input image I and its horizontally flipped counterpart $\hat{I}$. This produces two depth maps d_I and $d_{\hat{I}}$, the latter is flipped back obtaining $\hat{d}_{\hat{I}}$ and averaged with the former, in order to reduce artifacts near occlusions. We can notice that the previous trend is confirmed. In particular, the

	Lower is better				Higher is better		
	Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Zhou et al. [293]	0.286	7.009	8.377	0.320	0.691	0.854	0.929
Mahjourian et al. [142]	0.235	2.857	7.202	0.302	0.710	0.866	0.935
Yin et al. [270]	0.236	3.345	7.132	0.279	0.714	0.903	0.950
Godard et al. [69]	0.159	2.411	6.822	0.239	0.830	0.930	0.967
Ours	**0.143**	**2.161**	**6.526**	**0.222**	**0.850**	**0.939**	**0.972**

Table 19.2: Comparison with other self supervised method on KITTI 2015 evaluation split. Both [69] and our method use ResNet50 encoder.

full loss $\mathcal{L}_d + \mathcal{L}_s + \mathcal{L}_{cdd}$ leads to the best result on most scores. Furthermore, including the post-processing step allows the VGG model trained with our full loss to outperform the baseline ResNet50 architecture supervised by traditional depth losses only. This fact can be noticed in Table 19.1 comparing row 7 with row 13, observing that the former leads to better results except for δ_3 metric.

To further prove the effectiveness of our proposed method we compare it with other self-supervised approach as [270],[293],[142]. Thus, we have ran experiments with the source code available from [270],[293],[142] using the same testing data as for [69] and our method. Table 19.2 shows the outcome of this evaluation. We point out that we used the weights made available by the authors of [270],[293],[142], trained on a much larger amount of data (i.e., the entire Cityscapes and KITTI sequences, some of them overlapping with the testing split as well) w.r.t. the much lower supervision provided to our network. Despite this fact, monocular supervised works [270],[293],[142] perform poorly compared to both [1] and our approach, confirming our semi-supervised framework to outperform them as well. We also point out that our test split relies on high-quality ground truth labels for evaluation, available from KITTI 2015 stereo dataset, while the Eigen split used to validate [270],[293],[142] provides much worse quality depth measurements, as also argued by the authors of [69].

As our final test we also compare our method with the recent multi-task learning approach by Kendall et al. [110]. Differently from our approach, they jointly learn depth, semantic and instance segmentation in fully supervised manner. They run experiments

Tiny Cityscapes, a split obtained by resizing the validation set of Cityscapes to 128 256 resolution. To compare our results to theirs we have taken our ResNet50 model trained on Cityscapes and validated it following the same protocol. Their depth-only model (trained supervised) achieves 0.640 inverse mean depth error, dropping to 0.522 when trained to tackle semantic and instance segmentation as well. Our ResNet50 network (trained unsupervised) starts with 1.705 error for depth-only, dropping to 1.488. Thus, the two approaches achieve 22% and 15 % improvement respectively. We point out that, besides relying on supervised learning for depth, [110] exploits both semantic and instance segmentation, requiring additional manually annotated labels, while we only enforce our cross-domain discontinuity loss.

19.3.3 Semantic segmentation: evaluation on KITTI 2015

Although our proposal is aimed at ameliorating depth prediction by learning richer features exploiting semantics, our network also delivers a semantic segmentation of the input image. To gather hints about the accuracy of this additional outcome of our network, we evaluated the semantic maps generated on the same KITTI evaluation split defined before. Differently from the monocular depth estimation task, results concerning semantic segmentation are quite far from the state-of-the-art. In particular, we obtain 88.51% and 88.19% per-pixel accuracy, respectively, with models based on VGG and ResNet50. We ascribe this to our architecture - inspired by [69] - being optimized for unsupervised depth prediction, whereas different design choices are often found in networks pursuing semantic segmentation (i.e., atrous convolutions, SPP layers ...). We also found that training the basic encoder-decoder for semantic segmentation only yields to 86.72% and 88.18% per-pixel accuracy with VGG and ResNet50, respectively. Thus, while semantics helps depth prediction inasmuch as to outperform the state-of-the-art within the proposed framework, the converse requires further studies as the observed improvements are indeed quite minor. Therefore, we plan to investigate on how to design a network architecture and associated semi-supervised learning framework whereby the synergy between monocular depth prediction and semantic segmentation may be exploited in order to significantly improve accuracy in both tasks.

19.4 Conclusion

We have proposed a deep learning architecture to improve unsupervised monocular depth estimation by leveraging on semantic information. We have shown how training our architecture end-end to infer semantics and depth jointly enables us to outperform the state-of-the-art approach for unsupervised monocular depth estimation [69]. Our single-encoder/dual-decoder architecture is trained in a semi-supervised manner, *i.e.* using ground truth labels only for the semantic segmentation task. Despite obtaining ground truth labels for semantic is tedious and requires accurate and time-consuming manual annotation, it is still more feasible than depth labeling. In fact, this latter task requires expensive active sensors to be used at acquisition time and becomes almost unfeasible offline, on already captured frames. Thus, our method represents an attractive alternative to improve self-supervised training without adding more image samples.

Chapter 20

Generative Adversarial Networks for unsupervised monocular depth prediction

The content of this chapter has been presented at 3D Reconstruction in the Wild 2018 (3DRW2018) - "Generative Adversarial Networks for unsupervised monocular depth prediction" [9].

20.1 Introduction

Recently, Generative Adversarial Networks (GANs) [71] proved to be very effective when dealing with high-level tasks such as image synthesis, style transfer and more. In this framework, two architectures are trained to solve competitive tasks. The first one, referred to as *generator*, produces a new image from a given input (e.g., a synthetic frame from noise, an image with a transferred style, etc.) while the second one called *discriminator* is trained to distinguish between real images and those generated by the first network. The two models play a *min-max* game, with the generator trained to produce outputs good enough to fool the discriminator and this latter network trained to not being fooled

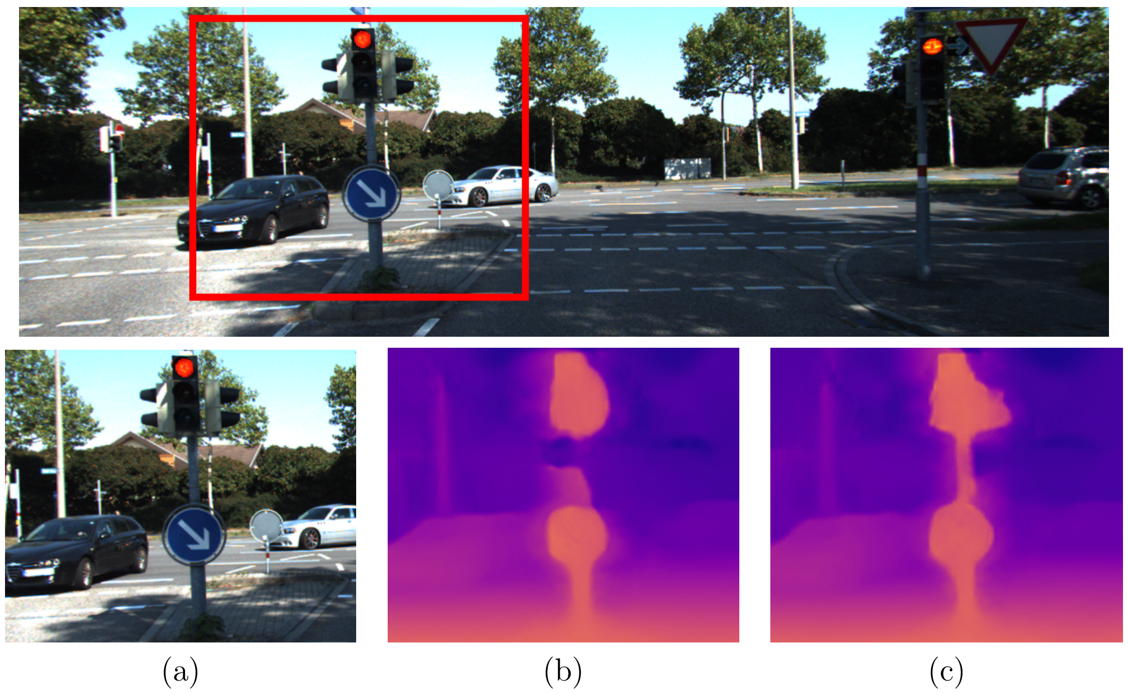


Figure 20.1: Estimated depth maps from single image. On top, frame from KITTI 2015 dataset, on bottom (a) detail from reference image (red rectangle), (b) depth predicted by Godard et al. [69] and (c) by our GAN architecture.

by the generator.

Considering the methodology adopted by state-of-the-art methods for unsupervised monocular depth estimation and the intrinsic ability of GANs to detect inconsistencies in images, in this chapter we propose to infer depth from monocular images by means of a GAN architecture. Given a stereo pair, at training time, our generator learns to produce meaningful depth representations, with respect to left and right image, by exploiting the epipolar constraint to align the two images. The warped images and the real ones are then forwarded to the discriminator, trained to distinguish between the two cases. The rationale behind our idea is that a generator producing accurate depth maps will also lead to better reconstructed images, harder to be distinguished from original unwarped inputs. At the same time, for the discriminator will be harder to be fooled, pushing the generator to build more realistic warped images and thus more accurate depth predictions.

In this work, we report extensive experimental results on the KITTI 2015 dataset, which provides a large amount of unlabeled stereo images and thus it is ideal for unsupervised training. Moreover, we highlight and fix inconsistencies in the commonly adapted split of Eigen [60], replacing Velodyne measurements with more accurate labels recently made available on KITTI [246]. Therefore, our contribution is threefold:

- Our framework represents, to the best of our knowledge, the first method to tackle monocular depth estimation within a GAN paradigm
- It outperforms traditional methods
- We propose a more reliable evaluation protocol for the split of Eigen et al. [60]

20.2 Method overview

In this section we describe our adversarial framework for unsupervised monocular depth estimation. State-of-the-art approaches rely on single network to accomplish this task. In contrast, at the core of our strategy there is a novel loss function based on a two players min-max game between two adversarial networks, as shown in Figure 22.1. This is done by using both a generative and a discriminative model competing on two different tasks, each one aimed at prevailing the other. This section discusses the geometry of the problem and how it is used to take advantages of 2D photometric constraints with a generative adversarial approach in a totally unsupervised manner. We refer to our framework as *MonoGAN*.

20.2.1 Generator model for monocular depth estimation

The main goal of our framework is to estimate an accurate depth map from a single image without relying on hard to find ground truth depth labels for training. For this purpose, we can model this problem as a domain transfer task: given an input image x, we want to obtain a new representation $y = G(x)$ in the depth domain. In other contexts, GAN models have been successfully deployed for image-to-image translation [295]. For our purpose a generator network, depicted in blue in Figure 22.1, is trained to learn a

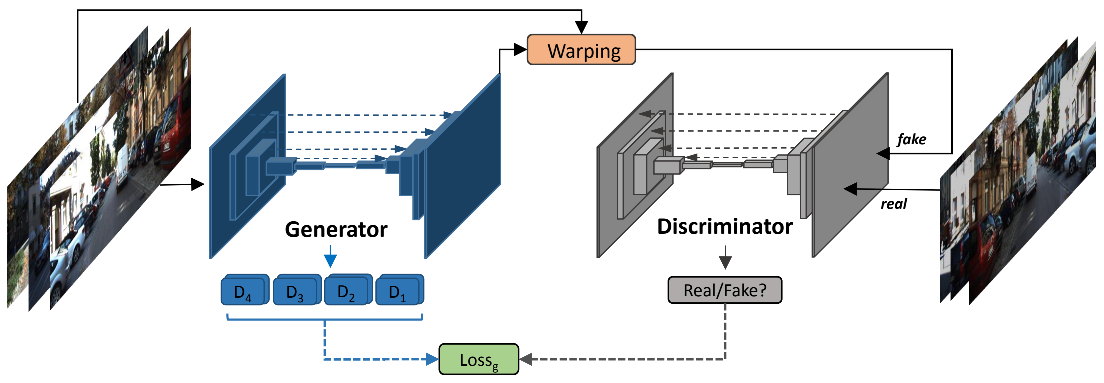


Figure 20.2: Proposed adversarial model. Given a single input frame, depth maps are produced by a Generator (blue) and used to warp images. Discriminator (gray) process both raw and warped images, trying to classify the former as real and the latter as fake. The generator is pushed to improve depth prediction to provide a more realistic warping to fool the discriminator. At the same time the discriminator learns to improve its ability to perform this task.

transfer function $G : \mathcal{I} \rightarrow \mathcal{D}$ mapping an input image from $\mathcal{I}$ to $\mathcal{D}$, respectively, the RGB and the depth domain. To do so, it is common practice to train the generator with loss signals enforcing structure consistency across the two domains to preserve object shapes, spatial consistency, etc. Similarly, this can be done for our specific goal by exploiting view synthesis. That is, projecting RGB images into 3D domain according to estimated depth and then back-projecting to new synthesized view for which we need a real image to compare with. To make it possible, for each training sample at least two images from different points of view are required to enable the image reconstruction process described so far. In literature, this strategy is used by other unsupervised techniques for monocular depth estimation, exploiting both unconstrained sequences [293] or stereo imagery [69]. In this latter case, given two images i^l and i^r acquired by a stereo setup, the generator estimates inverse depth (i.e., disparity) d^l used to obtain a synthesized image $\tilde{i}^l$ by warping i^r with bilinear sampler function [101] being it fully differentiable and thus enabling end-to-end training. If d^l is accurate, shapes and structures are preserved after warping, while an inaccurate estimation would lead to distortion artifacts as shown on the right of Figure 20.3. This process is totally unsupervised with respect to the $\mathcal{D}$ domain and thus it does not require at all ground truth labels at training time. Moreover, by estimating a second output d^r, representing the inverse mapping from i^l to i^r, allows to use additional supervisory signals by enforcing consistency in the $\mathcal{D}$ domain (i.e., Left-Right consistency constraint).

20.2.2 Discriminator model

To successfully accomplish domain transfer, GANs rely on a second network trained to distinguish images produced by the generator from those belonging to the target domain, respectively *fake* and *real* samples. We follow the same principle using the gray model in Figure 22.1, but acting differently from other approaches. In particular, to discriminate synthesized images from real ones we need a large amount of samples in the target domain. While for traditional domain transfer applications this does not represent an issue (requiring images without annotation), this becomes a limitation when depth is the target domain being ground truth label difficult to source in this circumstance. To overcome this limitation, we train a discriminator to work on the RGB domain to tell original input images from synthesized ones. Indeed, if estimated disparity by the generator is not accurate, the warping process would reproduce distortion artifacts easily detectable by the discriminator. On the other hand, an accurate depth prediction would lead to a reprojected image harder to be recognized from a real one. Figure 20.3 shows, on the left, an example of real image and, on the right, a warped one synthesized according to an inaccurate depth estimation. For instance, by looking at the tree, we can easily tell the real image from the warped one. By training the discriminator on this task, the generator is constantly forced to produce more accurate depth maps thus leading to a more realistic reconstructed image in order to fool it. At the same time the discriminator is constantly pushed to improve its ability to tell real images from synthesized ones. Our proposal aims at such *virtuous behavior* by properly modeling the adversarial contribution of the two networks as described in detail in the next section.

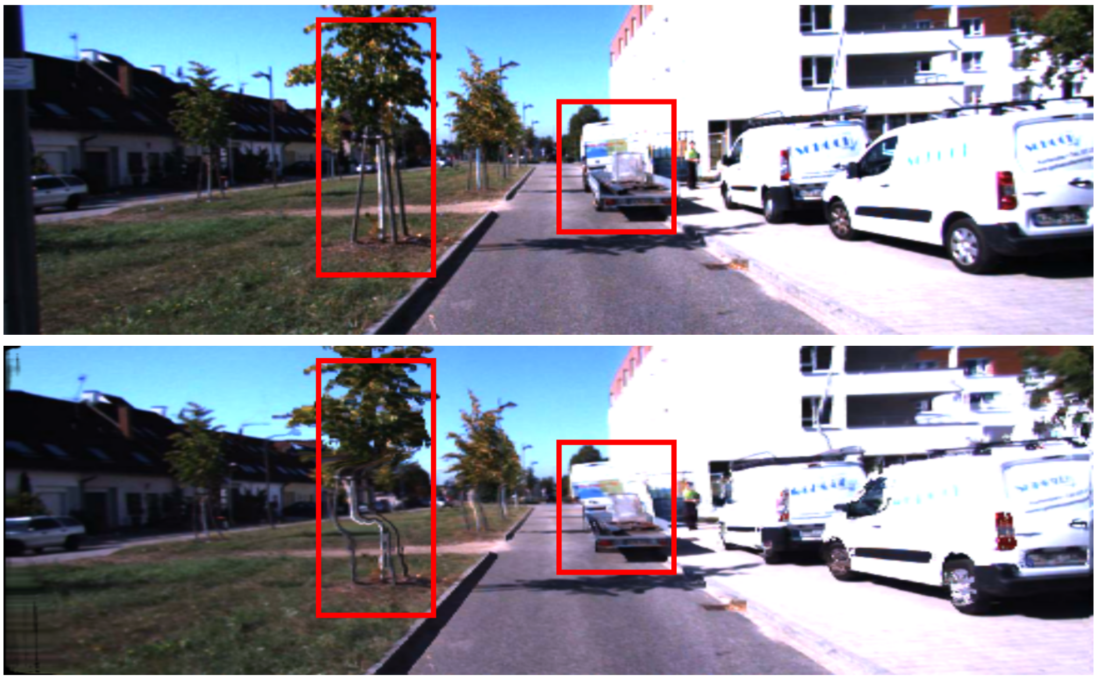


Figure 20.3: Example of real (top) and warped (bottom) image according to an estimated depth. We can clearly notice how inaccurate predictions lead to warping artifacts on the reprojected frame (e.g., distorted trees) not perceivable elsewhere.

20.3 Adversarial formulation

To train the framework outlined so far in end-to-end manner we define an objective function $\mathcal{L}(G, D)$ sum of two terms, a $\mathcal{L}_{GAN}$ expressing the min-max game between generator G and discriminator D:

$$\begin{aligned}\mathcal{L}_{GAN} = \min_G \max_D V(G, D) =& \mathbb{E}_{i_0 \sim \mathcal{I}}[\log(D(i_0))] \\ &+ \mathbb{E}_{i_1 \sim \tilde{\mathbb{I}}}[\log(1 - D(i_1))]\end{aligned} \tag{20.1}$$

with i_0 and i_1 belonging, respectively, to real images $\mathcal{I}$ and fake images $\tilde{\mathcal{I}}$ domains being the latter obtained by bilinear warping according to depth estimated by G and a data term $\mathcal{L}_{data}$ resulting in:

$$\mathcal{L}(G, D) = \mathcal{L}_{GAN} + \mathcal{L}_{data} \tag{20.2}$$

According to this formulation, generator G and discriminator D are trained to minimize loss functions $\mathcal{L}_G$ and $\mathcal{L}_D$:

$$\mathcal{L}_G = \mathcal{L}_{data} + \alpha_{adv}\mathbb{E}_{i_0,i_1\sim\tilde{\mathbb{I}}}[\log(D(i_1))] \tag{20.3}$$

$$\mathcal{L}_D = -\frac{1}{2}\mathbb{E}_{i_0\sim\mathcal{I}}\log(D(i_0)) - \frac{1}{2}\mathbb{E}_{i_1\sim\tilde{\mathcal{I}}}\log(1 - D(i_1)) \tag{20.4}$$

To give an intuition, G is trained to minimize the loss from data term and the probability that D will classify a warped image $i_1 \sim \tilde{\mathcal{I}}$ as fake. This second contribution is weighted according to α_{adv} factor, hyper-parameter of our framework. Consistently, D is trained to classify a raw image $i_0 \sim \mathcal{I}$ as real and a warped one as fake. Despite our framework processes a transfer from $\mathcal{I}$ to depth domain $\mathcal{D}$, we highlight how in the proposed adversarial formulation the discriminator does not process any sample from domain $\mathcal{D}$, neither fake nor real. Thus it does not require any ground truth depth map and perfectly fits with an unsupervised monocular depth estimation paradigm.

20.3.1 Data term loss

We define the data term $\mathcal{L}_{data}$ part of the generator loss function $\mathcal{L}_G$ as follows:

$$\mathcal{L}_{data} = \beta_{ap}(\mathcal{L}_{ap}) + \beta_{ds}(\mathcal{L}_{ds}) + \beta_{lr}(\mathcal{L}_{lr}) \tag{20.5}$$

where the loss consists in the weighted sum of three terms. The first one measures the reconstruction error between warped image $\tilde{I}$ and real one I by means of SSIM [254] and L1 difference of the two. The second term is a *smoothness* constraint that penalizes large disparity differences between neighboring pixels along the x and y directions unless a strong intensity gradients in the reference image I occurs. Finally, by building the generator to output a second disparity map d^r, we can add the term proposed in [69] as third supervision signal, enforcing left-right consistency between the predicted disparity

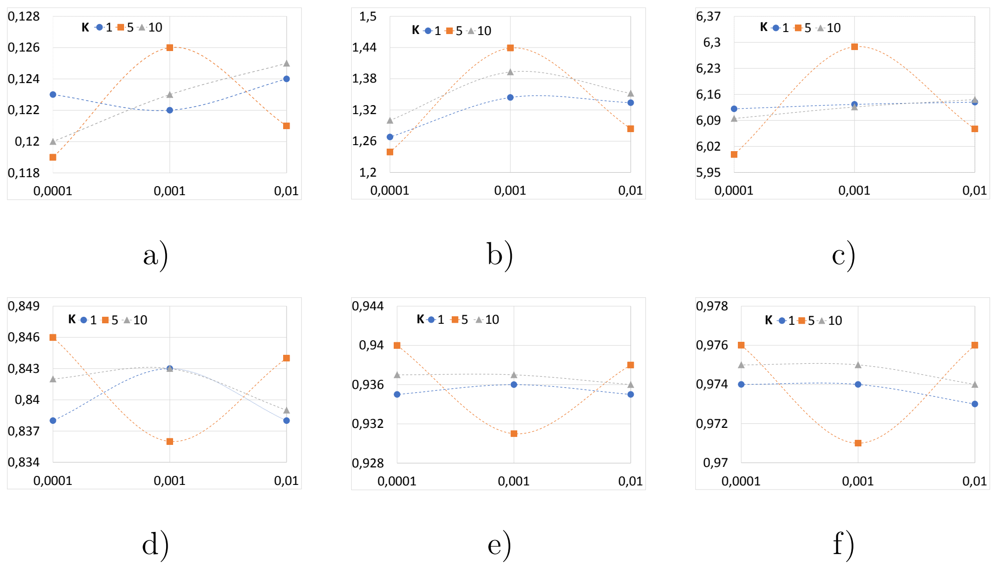


Figure 20.4: Analysis of hyper-parameters α_{adv} and k of our GAN model, on x axis α_{adv}, on y axis an evaluation metric. a) Abs Rel, b) Sq Rel and c) RMSE metrics (lower is better). d) $\delta < 1.25$, e) $\delta < 1.25^2$, f) $\delta < 1.25^3$ metrics (higher is better). Interpolation is used for visualization purpose only. We can notice how our proposal using a weight α_{adv} of 0.0001 and a step k of 5achieves the best performance with all metrics.

maps, d^l and d^r, for left and right images. Moreover, estimating d^r also enables to compute the three terms for both images in a training stereo pair.

20.4 Experimental results

In this section we assess the performance of our proposal with respect to literature. Firstly, we describe implementation details of our model outlining the architecture of generator and discriminator networks. Then, we describe the training protocols followed during our experiments reporting an exhaustive comparison on KITTI 2015 stereo dataset [155] with state-of-the-art method [69]. This evaluation clearly highlights how the adversarial formulation proposed is beneficial when tackling this unsupervised monocular depth estimation. Moreover, we compare our proposal with other frameworks known in literature, both supervised and unsupervised, on the split of data used by Eigen et al. [60]. In this latter case we provide experimental results on the standard Eigen split as well as

				Proposed method		Lower is better		Higher is better		
Exp.	Method	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	D1-all	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
i)	Godard et al. [69]	K	0.124	1.388	6.125	0.217	30.272	0.841	0.936	0.975
	MonoGAN	K	**0.119**	**1.239**	**5.998**	**0.212**	**29.864**	**0.846**	**0.940**	**0.976**
ii)	Godard et al. [69]	CS	0.699	10.060	14.445	0.542	94.757	0.053	0.326	0.862
	MonoGAN	CS	**0.668**	**9.488**	**14.051**	**0.526**	**94.092**	**0.063**	**0.394**	**0.876**
iii)	Godard et al. [69]	CS+K	0.104	1.070	5.417	0.188	25.523	0.875	0.956	0.983
	MonoGAN	CS+K	**0.102**	**1.023**	**5.390**	**0.185**	**25.081**	**0.878**	**0.958**	**0.984**
iv)	Godard et al. [69] + pp	CS+K	0.100	0.934	**5.141**	0.178	25.077	0.878	**0.961**	**0.986**
	MonoGAN + pp	CS+K	**0.098**	**0.908**	5.164	**0.177**	**23.999**	**0.879**	**0.961**	**0.986**

Table 20.1: Results on KITTI stereo 2015 [155]. We compare MonoGAN with [69] using different training schedules, respectively only KITTI sequences (K), only CityScapes (CS) and both sequentially (CS+K). Adversarial contribution always improves the results. We indicate with pp results obtained after applying the final post-processing step proposed in [69].

on a similar one made of more reliable data. This evaluation highlights once again the effectiveness of our proposal.

20.4.1 Implementation Details

For our GAN model, we deploy a VGG-based generator as in [69] counting 31 million parameters. We designed the discriminator in a similar way but, since the task of the discriminator is easier compared to the one tackled by the generator, we reduced the amount of feature maps extracted by each layer by a factor of two to obtain a less complex architecture. In fact, it counts about 8 million parameters, bringing the total number of variables of the overall framework to 39 million at training time. At test time, the discriminator is no longer required, restoring the same network configuration of [69] and thus the same computational efficiency.

For a fair comparison, we tune hyper-parameters such as learning rate or weights applied to loss terms to match those in [69], trained with a multi-scale data term while the adversarial contribution is computed at full resolution only. Being the task of D easier compared to depth estimation performed by G, we interleave the updates applied to the two. To this aim we introduce a further hyper-parameter k as the ratio between the number of training iterations performed on G and those on D, in addition to α_{adv}. In other words, discriminator weights are updated only every k updates of the generator. We

			Proposed method		Lower is better		Higher is better		
Method	cap	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Zhou et al. [293]	80 m	CS+K	0.198	1.836	6.565	0.275	0.718	0.901	0.960
Mahjourian et al. [142]	80 m	CS+K	0.159	1.231	5.912	0.243	0.784	0.923	0.970
Yin et al. [270]	80 m	CS+K	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Wang et al. [249]	80 m	CS+K	0.148	1.187	5.496	0.226	0.812	0.938	0.975
Poggi et al. [186] (200)	80 m	CS+K	0.146	1.291	5.907	0.245	0.801	0.926	0.967
Godard et al. [69]	80 m	CS+K	0.124	1.076	5.311	0.219	0.847	0.942	0.973
MonoGAN	80 m	CS+K	0.124	1.055	5.289	0.220	0.847	0.942	0.973
Godard et al. [69] + pp	80 m	CS+K	**0.118**	0.923	5.015	**0.210**	0.854	0.947	**0.976**
MonoGAN + pp	80 m	CS+K	**0.118**	**0.908**	**4.978**	**0.210**	**0.855**	**0.948**	**0.976**
Garg et al. [64]	50 m	K	0.169	1.080	5.104	0.273	0.740	0.904	0.962
Zhou et al. [293]	50 m	CS+K	0.190	1.436	4.975	0.258	0.735	0.915	0.968
Mahjourian et al. [142]	50 m	CS+K	0.151	0.949	4.383	0.227	0.802	0.935	0.974
Poggi et al. [186] (200)	50 m	CS+K	0.138	0.937	4.488	0.230	0.815	0.934	0.972
Godard et al. [69]	50 m	CS+K	0.117	0.762	3.972	0.206	0.860	0.948	0.976
MonoGAN	50 m	CS+K	0.118	0.761	3.995	0.208	0.860	0.949	0.976
Godard et al. [69] + pp	50 m	CS+K	**0.112**	0.680	3.810	**0.198**	0.866	**0.953**	**0.979**
MonoGAN + pp	50 m	CS+K	**0.112**	**0.673**	**3.804**	**0.198**	**0.868**	**0.953**	**0.979**

Table 20.2: Results for unsupervised techniques on the original Eigen et al. [60] split based on raw Velodyne data.

will report evaluations for different values of parameter k. To jointly train both generator and discriminator we use two instances of Adam optimizer [118], with $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$. The learning rate is the same for both instances: it is set at $\lambda = 10^{-4}$ for the first 30 epochs and then halved each 10 epochs. Number of epochs is set to 50 as for [69]. Training data are extracted from both KITTI raw sequences [155] and CityScapes dataset [47] providing respectively about 29000 and 23000 stereo pairs, these latter samples are cropped to remove lower part of the image frames (depicting a portion of the car used for acquisition) as in [69]. Moreover, as in [69] we perform data augmentation by randomly flipping input images horizontally and applying the following transformations: random gamma correction in [0.8,1.2], additive brightness in [0.5,2.0], and color shifts in [0.8,1.2] for each channel separately. The same procedure is applied before forwarding images to both generator and discriminator.

20.4.2 Hyper-parameters analysis

As mentioned before, our GAN model introduces two additional hyper-parameters: the weight α_{adv} applied to the adversarial loss acting on the generator and the iteration interval k between subsequent updating applied to the discriminator. Figure 20.4 reports

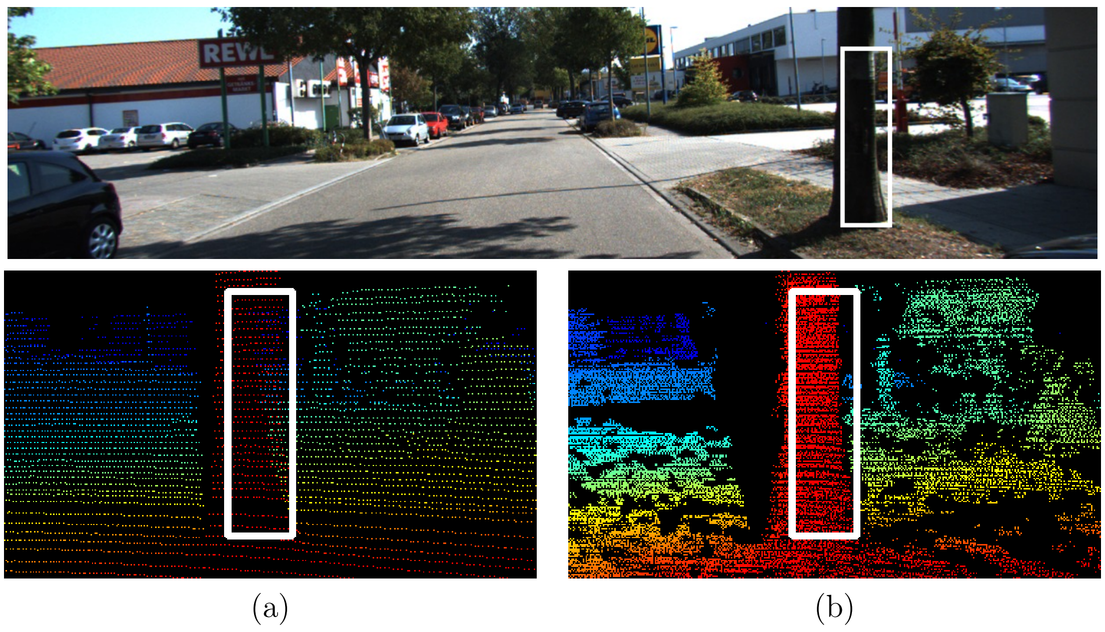


Figure 20.5: Qualitative comparison between (a) reprojected raw Velodyne points as done in the original Eigen split for results reported in Table 20.2 and (b) reprojected ground truth labels filtered according to [246], available on the KITTI website, deployed for our additional experiments reported in Table 20.3. Warmer colors encode closer points.

an analysis aimed at finding the best configuration (α_{adv}, k). On each plot, we report an evaluation metric used to measure accuracy in the field of monocular depth estimation (e.g., in [69]) as a function of both α_{adv} and k. Respectively, on top we report from left to right Abs Rel, Sq Rel and RMSE (lower scores are better), on bottom $\delta < 125$, $\delta < 125^2$ and $\delta < 125^3$ (higher scores are better). These results were obtained training MonoGAN on the 29000 KITTI stereo images [155], with α_{adv} set to 0.01, 0.001 and 0.0001 and k to 1, 5 and 10, for a total of 9 models trained and evaluated in Figure 20.4. We can notice how the configuration α_{adv}=0.0001 and k=5 achieves the best performance with all evaluated metrics. According to this analysis we use these hyper-parameters in the next experiments, unless otherwise stated. It is worth to note that despite the much smaller magnitude of α_{adv} compared to weights α_{ap}, α_{ds} and α_{lr} in data term (20.5), its contribution will affect significantly depth estimation accuracy as reported in the remainder.

20.4.3 Evaluation on KITTI dataset

Table 20.1 reports experimental results on the KITTI 2015 stereo dataset. For this evaluation, 200 images with provided ground truth disparity from KITTI 2015 stereo dataset are used for validation, as proposed in [69]. We report results for different training schedules: running 50 epochs on data from KITTI only (K), from CityScapes only (CS) and 50 epochs on CityScapes followed by 50 on KITTI (CS+K). We compare our proposal to state-of-the-art method for unsupervised monocular depth estimation proposed by Godard et al. [69] reporting for this method the outcome of the evaluation available in the original paper. Table 20.1 is divided into four main sections, representing four different experiments. In particular, i) compares MonoGAN with [69] when both trained on K. We can observe how our framework significantly outperforms the competitor on all metrics. Experiment ii) concerns the two models trained on CityScapes data [47] and evaluated on KITTI stereo images, thus measuring the generalization capability across different environments. In particular, CityScapes and KITTI images differ not only in terms of scene contents but also for the camera setup. We can notice that MonoGAN better generalizes when dealing with different data. In iii), we train both models on CityScapes first and then on KITTI, showing that MonoGAN better benefits from using different datasets at training time compared to [69] thus confirming the positive trend outlined in the previous experiments. Finally, in iv) we test the network trained in iii) refining the results with the same post-processing step described in [69]. It consists in predicting depth for both original and horizontally flipped input image, then taking 5% right-most pixels from the first and 5% left-most from the second, while averaging the two predictions for remaining pixels. With such post-processing, excluding one case out of 6 (i.e., with the RMSE metric) MonoGAN has better or equivalent performance compared to [69]. Overall, the evaluation on KITTI 2015 dataset highlights the effectiveness of the proposed GAN paradigm. In experiments iii) and iv), we exploited adversarial loss only during the second part of the training (i.e., on K) thus starting from the same model of

			Proposed method		Lower is better		Higher is better		
Method	cap	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Godard et al. [69]	80 m	CS+K	0.097	0.728	4.279	0.151	0.898	0.973	0.991
MonoGAN	80 m	CS+K	**0.096**	**0.699**	**4.236**	**0.150**	**0.899**	**0.974**	**0.992**
Godard et al. [69] + pp	80 m	CS+K	0.092	0.596	3.977	0.145	0.902	0.975	0.992
MonoGAN + pp	80 m	CS+K	**0.090**	**0.566**	**3.911**	**0.143**	**0.906**	**0.977**	**0.993**
Godard et al. [69]	50 m	CS+K	0.095	0.607	**4.100**	0.149	0.896	0.975	0.992
MonoGAN	50 m	CS+K	**0.094**	**0.600**	4.110	**0.148**	**0.897**	**0.976**	**0.993**
Godard et al. [69] + pp	50 m	CS+K	0.091	0.544	3.996	0.145	0.899	0.976	0.993
MonoGAN + pp	50 m	CS+K	**0.089**	**0.522**	**3.958**	**0.143**	**0.902**	**0.978**	**0.994**

Table 20.3: Results for MonoGAN and Godard et al. [69] on 93.5% of Eigen et al. [60] split using accurate ground truth labels [246] recently made available by KITTI evaluation benchmark.

[69] trained as in experiment ii), with the aim to assess how the discriminator improves the performance of a pre-trained model. Moreover, when fine-tuning we find beneficial to change the α_{adv} weight, similarly to traditional learning rate decimation techniques. In particular, we increased the adversarial weight α_{adv} from 0.0001 to 0.001 after 150k iterations (out of 181k total).

20.4.4 Evaluation on Eigen split

We report additional experiments conducted on the split of data proposed by Eigen et al. in [60]. We compare to other monocular depth estimation framework following the same protocol proposed in [69] using the same crop dimensions and parameters.

Table 20.2 reports a detailed comparison of unsupervised methods. On top, we evaluated depth maps up to a maximum distance of 80 meters. We can observe how MonoGAN performs on par or better than Godard et al. [69] outperforming it in terms of Sq Rel and RMSE errors and $\delta < 1.25$, $\delta < 1.25^2$ metrics. On the bottom of the table, we evaluate up to 50 meters maximum distance to compare with Garg et al. [64]. This evaluation substantially confirms the previous trend. As for experiments on KITTI 2015 stereo dataset, we find out that increasing by a factor 10 the adversarial weight α_{adv} from 0.0001 to 0.001 after 150k iterations out of 181k total increases the accuracy of MonoGAN. Apparently, the margin between MonoGAN and [69] is much lower on this evaluation data. However, as already pointed out in [69] and [246], depth data obtained through Velodyne projection are affected by errors introduced by the rotation of the sensor, the motion of the vehicle and surrounding objects and also incorrect depth readings due to occlusion at object boundaries. Therefore, to better assess the performance of our proposal with respect to state-of-the-art we also considered the same split of images with more accurate ground truth labels made available by Uhrig et al. [246] and now officially distributed as depth ground truth maps by KITTI benchmark. These maps are obtained by filtering Velodyne data with disparity obtained by the Semi Global Matching algorithm [82] so as to remove outliers from the original measurements. Figure 20.5 shows a qualitative comparison between depth labels from raw Velodyne data reprojected into the left image, deployed in the original Eigen split, and labels provided by [246], deployed for our additional evaluation. Comparing (a) and (b) to the reference image at the top we can easily notice in (a) several outliers close to the tree trunk border not detectable in (b). Unfortunately, accurate ground truth maps provided by [246] are not available for 45 images of the original Eigen split. Therefore, the number of testing images is reduced from from 697 to 652. However, at the expense of a very small reduction of validation samples (i.e., 6.5%) we get much more reliable ground truth data according to [246]. With such accurate data, Table 20.3 reports a comparison between [69] and MonoGAN with and without post-processing, thresholding at 80 and 50 meters as for previous experiment on standard Eigen split. From Table 20.3 we can notice how with all metrics, excluding one case, MonoGAN on this more reliable dataset outperforms [69] confirming the trend already reported in Table 20.1 on the accurate KITTI 2015 benchmark.

20.5 Conclusions

In this chapter, we proposed to tackle monocular depth estimation as an image generation task by means of a Generative Adversarial Networks paradigm. Exploiting at training time stereo images, the generator learns to infer depth from the reference image and from this data to generate a warped target image. The discriminator is trained to distinguish between real images and fake ones generated by the generator. Extensive experimental results confirm that our proposal outperforms known techniques for unsupervised monocular depth estimation.

Chapter 21

Learning monocular depth estimation infusing traditional stereo knowledge

The content of this chapter has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019) - "Learning monocular depth estimation infusing traditional stereo knowledge" [241].

21.1 Introduction

Self-supervised learning paradigms for monocular depth estimation [69, 142, 186, 187, 265, 293] became very popular to overcome the need for costly ground truth annotations, usually obtained employing expensive active sensors and human post-processing [66, 155, 246]. Following this strategy, Convolutional Neural Networks (CNNs) can be trained to tackle depth estimation as an image synthesis task from stereo pairs or monocular sequences [69, 293]. For this purpose, using stereo pairs rather than monocular sequences as supervision turned out to be more effective according to the literature. Although the former strategy is more constrained since a stereo setup is necessary for training, it does neither require to infer relative pose between adjacent frames in a sequence nor to segment moving objects in the scene. Moreover, a stereo setup does not require camera motion, conversely to a monocular setup, to provide meaningful supervision. Other means for self-supervision consist into distilling *proxy* labels in place of more expensive annotations for various tasks [74, 119, 132, 143, 235, 239].

In this chapter, we propose monocular Residual Matching (shorten, monoResMatch), a novel end-to-end architecture trained to estimate depth from a monocular image leveraging a virtual stereo setup. In the first stage, we map input image into a features space, then we use such representation to estimate a first depth outcome and consequently synthesize features aligned with a *virtual* right image. Finally, the last refinement module performs stereo matching between the real and synthesized representations. Differently from other frameworks following a similar rationale [137] that combines heterogeneous networks for synthesis [260] and stereo [151], we use a single architecture trained in end-to-end fashion yielding a notable accuracy improvement compared to the existing solutions. Moreover, we leverage traditional knowledge from stereo to obtain accurate proxy labels in order to improve monocular depth estimation supervised by stereo pairs. We will show that, despite the presence of outliers in the produced labels, training according to this paradigm results in superior accuracy compared to image warping approaches for self-supervision. Experimental results on the KITTI raw dataset [67] will show that the synergy between the two aforementioned key components of our pipeline enables to achieve state-of-the-art results compared to other self-supervised frameworks for monocular depth estimation not requiring any ground truth annotation.

21.2 Monocular Residual Matching

In this section, we describe in detail the proposed *monocular Residual Matching* (monoResMatch) architecture designed to infer accurate and dense depth estimation in a self-supervised manner from a single image. Figure 21.1 recaps the three key components of our network. First, a multi-scale feature extractor takes as input a single raw image and computes deep learnable representations at different scales from quarter resolution F_L^2 to

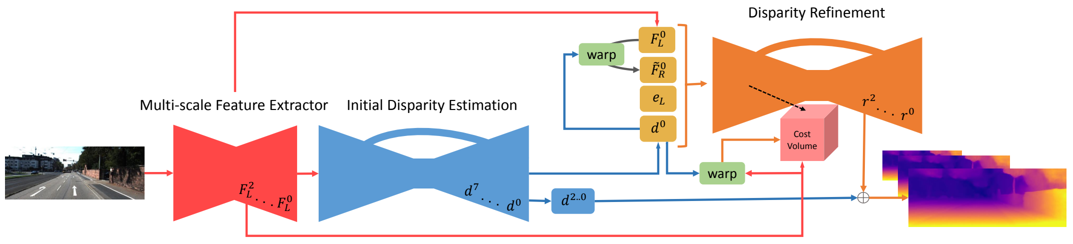


Figure 21.1: Illustration of our *monoResMatch* architecture. Given one input image, the multi-scale feature extractor (in red) generates high-level representations in the first stage. The initial disparity estimator (in blue) yields multi-scale disparity maps aligned with the left and right frames of a stereo pair. The disparity refinement module (in orange) is in charge of refining the initial left disparity relying on features computed in the first stage, disparities generated in the second stage, matching costs between high-dimensional features F_L^0 extracted from input and synthetic $\tilde{F}_R^0$ from a *virtual* right viewpoint, together with absolute error e_L between F_L^0 and back-warped $\tilde{F}_R^0$.

full-resolution F_L^0 in order to toughen the network to ambiguities in photometric appearance. Second, deep high-dimensional features at input image resolution are processed to estimate, through an hourglass structure with skip-connections, multi-scale inverse depth (i.e., disparity) maps aligned with the input and a *virtual* right view learned during training. By doing so, our network learns to emulate a binocular setup, thus allowing further processing in the stereo domain [137]. Third, a disparity refinement stage estimates residual corrections to the initial disparity. In particular, we use deep features from the first stage and back-warped features of the *virtual* right image to construct a cost volume that stores the stereo matching costs using a correlation layer [151].

Our entire architecture is trained from scratch in an end-to-end manner, while SVS [137] by training its two main components, Deep3D [260] and DispNetC [151], on image synthesis and disparity estimation tasks separately (with the latter requiring additional, supervised depth labels from synthetic imagery [151]).

Extensive experimental results will prove that monoResMatch enables much more accurate estimations compared to SVS and other state-of-the-art approaches.

21.2.1 Multi-scale feature extractor

Inspired by [129], given one input image I we generate deep representations using layers of convolutional filters. In particular, the first 2-stride layer convolves I with 64 learnable filters of size 7×7 followed by a second 2-stride convolutional layer composed of 128 filters with kernel size 4×4. Two deconvolutional blocks, with stride 2 and 4, are deployed to upsample features from lower-spatial resolution to full input resolution producing 32 features maps each. A 1×1 convolutional layer with stride 1 further processes upsampled representations.

21.2.2 Initial Disparity Estimation

Given the features extracted by the first module, this component is in charge of estimating an initial disparity map. In particular, an encoder-decoder architecture inspired by DispNet processes deep features at quarter resolution from the multi-scale feature extractor (i.e., *conv2*) and outputs disparity maps at different scales, specifically from $\frac{1}{128}$ to full-resolution. Each down-sampling module, composed of two convolutional blocks with stride 2 and 1 each, produces a growing number of extracted features, respectively 64, 128, 256, 512, 1024, and each convolutional layer uses 3×3 kernels followed by ReLU non-linearities. Differently from DispNet, which computes matching costs in the early part of this stage using features from the left and right images of a stereo pair, our architecture lacks such necessary information required to compute a cost volume since it processes a single input image. Thus, no 1-D correlation layer can be imposed to encode geometrical constraints in this stage of our network. Then, upsampling modules are deployed to enrich feature representations through skip-connections and to extract two disparity maps, aligned respectively with the input frame and a *virtual* viewpoint on its right as in [69]. This process is carried out at each scale using 1-stride convolutional layers with kernel size 3×3.

21.2.3 Disparity Refinement

Given an initial estimate of the disparity at each scale obtained in the second part of the network, often characterized by errors at depth discontinuities and occluded regions, this stage predicts corresponding multi-scale residual signals [80] by a few stacked non-linear layers that are then used to compute the final left-view aligned disparity map. This strategy allows us to simplify the end-to-end learning process of the entire network. Moreover, motivated by [137], we believe that geometrical constraints can play a central role in boosting the final depth accuracy. For this reason, we embed matching costs in feature space computed employing a horizontal correlation layer, typically deployed in deep stereo algorithms. To this end, we rely on the right-view disparity map computed previously to generate right-view features $\tilde{F}_R^0$ from the left ones F_L^0 using a differentiable bilinear sampler [101]. The network is also fed with error e_L, i.e.the absolute difference between left and *virtual* right features at input resolution, with the latter back-warped at the same coordinates of the former, as in [129].

We point out once more that, differently from [137], our architecture produces both a synthetic right view, i.e.its features representation, and computes the final disparity map following stereo rationale. This makes monoResMatch a single end-to-end architecture, effectively performing stereo out of a single input view rather than the combination of two models (i.e., Deep3D [260] and DispNetC [151] for the two tasks outlined) trained independently as in [137]. Moreover, exhaustive experiments will highlight the superior accuracy achieved by our fully self-supervised, end-to-end approach.

21.2.4 Training Loss

In order to train our multi-stage architecture, we define the total loss as a sum of two main contributions, a $\mathcal{L}_{init}$ term from the initial disparity estimation module and a $\mathcal{L}_{ref}$ term from the disparity refinement stage. Following [70], we embrace the idea to up-sample the predicted low-resolution disparity maps to the full input resolution and then compute

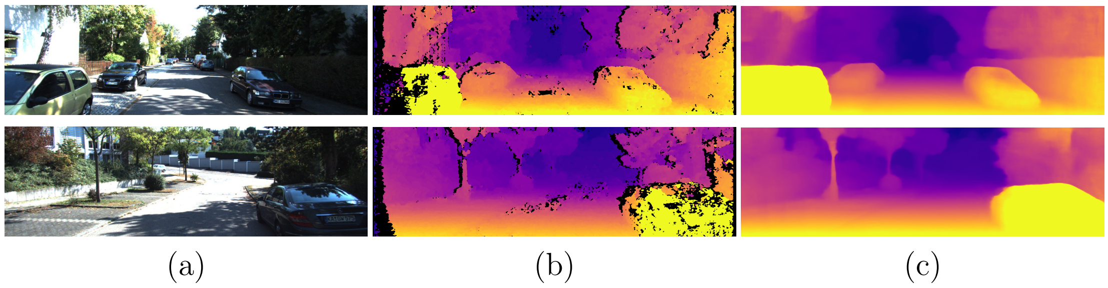


Figure 21.2: Examples of proxy labels computed by SGM. Given the source image (a), the network exploits the SGM supervision filtered with left-right consistency check (b) in order to train monoResMatch to estimate the final disparity map (c). No post-processing from [69] is performed on (c) in this example.

the corresponding signals. This simple strategy is designed to force the inverse depth estimation to reproduce the same objective at each scale, thus leading to much better outcomes. In particular, we obtain the final training loss as:

$$\mathcal{L}_{total} = \sum_{s=1}^{n_i} \mathcal{L}_{init} + \sum_{s=1}^{n_r} \mathcal{L}_{ref} \tag{21.1}$$

where s indicates the output resolution, n_i and n_r the numbers of considered scales during loss computation, while $\mathcal{L}_{init}$ and $\mathcal{L}_{ref}$ are formalised as:

$$\begin{aligned}\mathcal{L}_{init} =& \alpha_{ap}(\mathcal{L}^l_{ap} + \mathcal{L}^r_{ap}) + \alpha_{ds}(\mathcal{L}^l_{ds} + \mathcal{L}^r_{ds}) \\ &+ \alpha_{ps}(\mathcal{L}^l_{ps} + \mathcal{L}^r_{ps})\end{aligned} \tag{21.2}$$

$$\mathcal{L}_{ref} = \alpha_{ap}\mathcal{L}^l_{ap} + \alpha_{ds}\mathcal{L}^l_{ds} + \alpha_{ps}\mathcal{L}^l_{ps} \tag{21.3}$$

where $\mathcal{L}_{ap}$ is an image reconstruction loss, $\mathcal{L}_{ds}$ is a smoothness term and $\mathcal{L}_{ps}$ is our proxy-supervised loss. Each term contains both the left and right components for the initial disparity estimator, and the left components only for the refinement stage.

Proxy-supervised loss. Given the proxy disparity maps obtained by a conventional stereo algorithm we coach the network using reverse Huber (berHu) loss [166]:

$$\mathcal{L}_{ps} = \frac{1}{N} \sum_{i,j} berHu(d_{ij}, d^{st}_{ij}, c) \tag{21.4}$$

$$berHu(d_{ij}, d^{st}_{ij}, c) = \begin{cases} |d_{ij} - d^{st}_{ij}| & \text{if } |d_{ij} - d^{st}_{ij}| \leq c \\ \frac{|d_{ij} - d^{st}_{ij}|^2 - c^2}{2c} & \text{otherwise} \end{cases} \tag{21.5}$$

where d_{ij} and d^{st}_{ij} are, respectively, the predicted disparity and the proxy annotation for pixel at the coordinates i, j of the image, while c is adaptively set as $\alpha \max_{i,j} |d_{ij} - d^{st}_{ij}|$, with $\alpha = 0.2$.

21.2.5 Proxy labels distillation

To generate accurate proxy labels, we use the popular SGM algorithm [82], a fast yet effective solution to infer depth from a rectified stereo pair without training. In our implementation, initial matching costs are computed for each pixel p and disparity hypothesis d applying a 9×7 census transform and computing Hamming distance on pixel strings. Then, scanline optimization along eight different paths refines the initial cost volume as follows:

$$\begin{aligned} E(p,d) =& C(p,d) + \min_{j>1}[C(p',d), C(p',d \pm 1) + P1, \\ & C(p', d \pm q) + P2] - \min_{k<D_{max}} (C(p',k)) \end{aligned} \tag{21.6}$$

being $C(p,d)$ the matching cost for pixel p and hypothesis d, P_1 and P_2 two smoothness penalties, discouraging disparity gaps between p and previous pixel p' along the scanline path. The final disparity map D is obtained applying a winner-takes-all strategy to each pixel of the reference image. Although SGM generates quite accurate disparity labels, outliers may affect the training of a depth model negatively, as noticed by Tonioni et al.[235].

They applied a learned confidence measure [179] to filter out erroneous labels when computing the loss. Differently, we run a non-learning based left-right consistency check to detect outliers. Purposely, by extracting both disparity maps D^L and D^R with SGM, respectively for the left and right images, we apply the following criteria to invalidate (i.e., set to -1) pixels having different disparities across the two maps:

$$D(p) = \begin{cases} D(p) & \text{if } |D^L(p) - D^R(p - D^L(p))| \leq \varepsilon \\ -1 & \text{otherwise} \end{cases} \tag{21.7}$$

The left-right consistency check is a simple strategy that removes many wrong disparity assignments, mostly near depth discontinuities, without needing any training that would be required by [235]. Therefore, our proxy labels generation process does not rely at all on ground truth depth labels. Figure 21.2 shows an example of distilled labels (b), where black pixels correspond to outliers filtered out by left-right consistency. Although some of them persist, we can notice how they do not affect the final prediction by the trained network and how our proposal can recover accurate disparity values in occluded regions on the left side of the image (c).

21.2.6 Experimental results

In this section, we describe implementation details and then present exhaustive evaluations of monoResMatch on various training/testing configurations, showing that our proposal consistently outperforms self-supervised state-of-the-art approaches. As standard in this field, we assess the performance of monocular depth estimation techniques following the protocol by Eigen et al.[60], extracting data from the KITTI [67] dataset, using sparse LiDAR measurements as ground truth for evaluation. Additionally, we also perform an exhaustive ablation study proving that proxy supervision from SGM algorithm and effective architectural choices enable our strategy to improve predicted depth map accuracy by a large margin.

				Lower is better				Higher is better		
Method	Supervision		Train set	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
	Image	SGM								
Godard et al.[69] ResNet50	✓		K	0.128	1.038	5.355	0.223	0.833	0.939	0.972
Poggi et al.[187] ResNet50	✓		K	0.126	0.961	5.205	0.220	0.835	0.941	0.974
monoResMatch	✓		K	0.116	0.986	5.098	0.214	0.847	0.939	0.972
monoResMatch	✓	✓	K	**0.111**	**0.867**	**4.714**	**0.199**	**0.864**	**0.954**	**0.979**
Godard et al.[69] ResNet50	✓		CS,K	0.114	0.898	4.935	0.206	0.861	0.949	0.976
Poggi et al.[187] ResNet50	✓		CS,K	0.111	0.849	4.822	0.202	0.865	0.952	0.978
Godard et al.[69] ResNet50	✓	✓	CS,K	0.110	0.822	4.675	0.199	0.862	0.953	0.980
monoResMatch (no-refinement)	✓	✓	CS,K	0.107	0.781	4.588	0.195	0.869	0.957	0.980
monoResMatch (no-corr)	✓	✓	CS,K	0.104	0.766	4.553	0.192	0.875	0.958	0.980
monoResMatch (no-pp)	✓	✓	CS,K	0.098	0.711	4.433	0.189	0.888	0.960	0.980
monoResMatch	✓	✓	CS,K	**0.096**	**0.673**	**4.351**	**0.184**	**0.890**	**0.961**	**0.981**

Table 21.1: Ablation studies on the Eigen split [60], with maximum depth set to 80m. All networks run post-processing as in [69] unless otherwise specified.

Implementation details

Following the standard protocol in this field, we used CityScapes followed by KITTI for training. We implemented our architecture using the TensorFlow framework, counting approximately 42.5 millions of parameters, summing variables from the multi-scale feature extractor (0.51 M), the initial disparity stage (41.4 M) and the refinement module (0.6 M). In the experiments, we pre-trained monoResMatch on CS running about 150k iteration using a batch size of 6 and random crops of size 512×256 on 1024×512 resized images from the original resolution. We used Adam optimizer [118] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. We set the initial learning rate to 10^{-4}, manually halved after 100k and 120k steps, then continuing until convergence. After the first pre-initialisation procedure, we perform fine-tuning of the overall architecture on 22600 KITTI raw images from K. Specifically, we run 300k steps using a batch size of 6 and extracting random crops of size 640×192 from resized images at 1280×384 resolution. At this stage, we employed a learning rate of 10^{-4}, halved after 180k and 240k iterations. We fixed the hyper-parameters of the different loss components to $\alpha_{ap} = 1, \alpha_{ds} = 0.1$ and $\alpha_{ps} = 1$, while $n_i = 4$ and $n_r = 3$. As in [69], data augmentation procedure has been applied to both images from CS and K at training, in order to increase the robustness of the network. At test time, we post-process disparity as in [69, 187, 265]. Nevertheless, we preliminary highlight that, differently from the strategies mentioned above, effects such as disparity

			Lower is better				Higher is better		
Method	Supervision	Train set	Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	$\delta < 1.25^2$	$\delta < 1.25^3$
Zou et al.[296]	Seq	CS,K	0.146	1.182	5.215	0.213	0.818	0.943	0.978
Mahjourian et al.[142]	Seq	CS,K	0.159	1.231	5.912	0.243	0.784	0.923	0.970
Yin et al.[270] GeoNet ResNet50	Seq	CS,K	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Wang et al.[249]	Seq	CS,K	0.148	1.187	5.496	0.226	0.812	0.938	0.975
Poggi et al.[186] PyD-Net (200)	Stereo	CS,K	0.146	1.291	5.907	0.245	0.801	0.926	0.967
Godard et al.[69] ResNet50	Stereo	CS,K	0.114	0.898	4.935	0.206	0.861	0.949	0.976
Poggi et al.[187] 3Net ResNet50	Stereo	CS,K	0.111	0.849	4.822	0.202	0.865	0.952	0.978
Pilzer et al.[177] (Teacher)	Stereo	CS,K	0.098	0.831	4.656	0.202	0.882	0.948	0.973
Yang et al.[265]	Seq+Stereo	K$_o$, K$_r$, K$_o$	0.097	0.734	4.442	0.187	0.888	0.958	0.980
monoResMatch	Stereo	CS,K	**0.096**	**0.673**	**4.351**	**0.184**	**0.890**	**0.961**	**0.981**

Table 21.2: Quantitative evaluation on the test set of KITTI dataset [67] using the split of Eigen et al.[60], maximum depth: 80m. Last four entries include post-processing [69]. K$_o$, K$_r$, K$_o$ are splits from K, defined in [265]. Best results are shown in bold.

ramps on the left border are effectively solved by simply picking random crops on proxy disparity maps generated by SGM, as clearly visible in Figure 21.2 (c).

Proxy supervision is obtained through SGM implementation from [227], which allows us to quickly generate disparity maps aligned with the left and right images for both CS and K. We process such outputs using left-right consistency check in order to reduce the numbers of outliers using an ϵ of 1. We assess the accuracy of our proxy generator on 200 high-quality disparity maps from KITTI 2015 training dataset [67], measuring 96.1% of pixels having disparity error smaller than 3. Compared to Tonioni et al.[235], we register a negligible drop in accuracy from 99.6% reported in their paper. However, we do not rely on any learning-based confidence estimator as they do [179], so we maintain label distillation detached from the need for ground truth as well. Since SGM runs over images at full resolution while monoResMatch inputs are resized to 1280×384 before extracting crops, we enforce a scaling factor to SGM disparities given by $\frac{1280}{W}$, where W is the original image *width*. Consequently, the depth map estimated by monoResMatch must be properly multiplied by $\frac{W}{1280}$ at test time. The architecture is trained end-to-end on a single Titan XP GPU without any stage-wise procedure and infers depth maps in 0.16s per frame at test time, processing images at KITTI resolution (i.e., about 1280×384 to be compatible with monoResMatch downsampling factors).

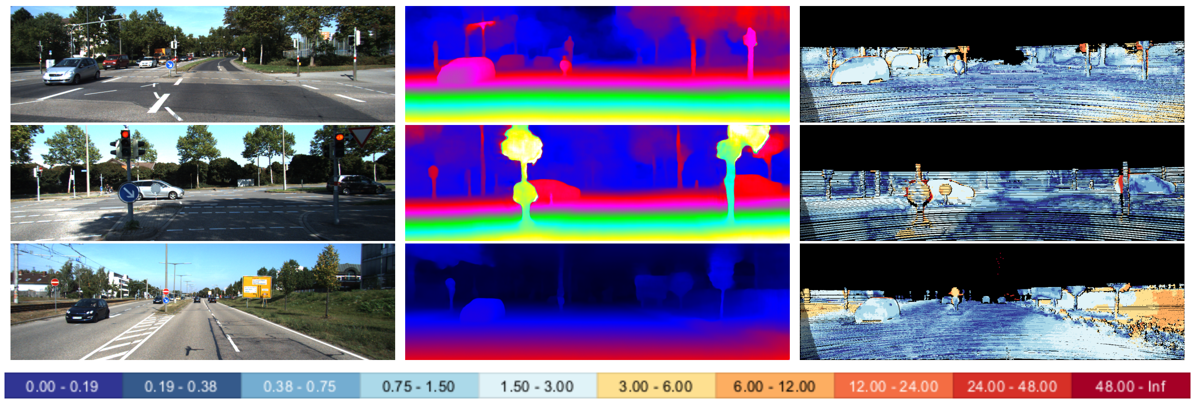


Figure 21.3: Stereo evaluation of our depth-from-mono framework. From left to right the input image, the predicted depth and the errors with respect to ground truth. The last line reports the color code used to display the seriousness of the shortcomings (same of [155])

In this section we examine the impact of i) proxy-supervision from SGM and ii) the different components of monoResMatch. The outcomes of these experiments, conducted on the Eigen split, are collected in Table 21.1.

Proxy-supervised loss analysis. We train *monodepth* framework by Godard et al.[69] from scratch adding our proxy-loss, then we compare the obtained model with the original one, as well as with the more effective strategy used by 3Net [187]. We can observe that proxy-loss enables a more accurate *monodepth* model (row 3) compared to [69], moreover it also outperforms virtual trinocular supervision proposed in [187], attaining better metrics with respect of both, but $\delta < 1.25$ for 3Net. Specifically, by recalling Figure 21.2, the proxy distillation couples well with a cropping strategy, solving well-known issues for stereo supervision such as disparity ramps on the left border.

Component analysis. Still referring to Table 21.1, we evaluate different configurations of our framework by ablating the key modules peculiar to our architecture. First, we train monoResMatch on K without proxy supervision (row 3) to highlight that our architecture already outperforms [69] (row 1). Training on CS+K with proxy labels, we can notice how without any refinement module (*no-refinement*), our framework already

						Lower is better				Higher is better		
Method	Supervision					Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
	200-acrt	100	200	500	700							
Luo et al.[137]	✓					0.101	0.673	4.425	**0.176**	-	-	-
monoResMatch	✓					**0.089**	**0.575**	**4.186**	0.181	0.897	0.964	0.982
Luo et al.[137]	✓		✓			0.100	0.670	4.437	0.192	0.882	0.958	0.979
monoResMatch	✓		✓			**0.096**	**0.573**	**3.950**	**0.168**	**0.897**	**0.968**	**0.987**
Luo et al.[137]	✓			✓		0.094	0.635	4.275	0.179	0.889	0.964	0.984
monoResMatch	✓			✓		**0.093**	**0.567**	**3.914**	**0.165**	**0.901**	**0.969**	**0.987**
Luo et al.[137]	✓				✓	**0.094**	0.626	4.252	0.177	0.891	0.965	0.984
monoResMatch	✓				✓	0.095	**0.567**	**3.942**	**0.166**	**0.899**	**0.969**	**0.987**
Guo et al.[74]		✓				**0.096**	0.641	4.095	**0.168**	0.892	0.967	0.986
monoResMatch		✓				0.098	**0.597**	**3.973**	0.169	**0.895**	**0.968**	**0.987**

Table 21.3: Experimental results on the Eigen split [60], maximum depth: 80m. Comparison between methods supervised by few annotated samples. Best results in direct comparisons are shown in bold, best overall scores are in red, consistently attained by monoResMatch.

Method	D1-bg	D1-fg	D1-all
monodepth [69]	27.00	28.24	27.21
OCV-BM	24.29	30.13	25.27
SVS [137]	25.18	20.77	24.44
monoResMatch	**22.10**	**19.81**	**21.72**

Table 21.4: Quantitative results on the test set of the KITTI 2015 Stereo Benchmark [155]. Percentage of pixels having error larger than 3 or 5% of the ground truth. Best results are shown in bold.

outperforms the proxy-supervised ResNet50 model of Godard et al.[69]. Adding the disparity refinement component without encoding any matching relationship (*no-corr*) enables small improvements, becoming much larger on most metrics when a correlation layer is introduced (*no-pp*) to process real and synthesized features as to resemble stereo matching. Finally, post-processing as in [69] (row 11) still ameliorates all scores, although the larger contribution is given by the correlation-based refinement module, as perceived by comparing *no-refinement* and *no-pp* entries. Finally, by comparing rows 4 and 11 we can also perceive the impact given by CS pretraining on our full model.

21.3 Conclusion

In this chapter, we proposed monoResMatch, a novel framework for monocular depth estimation. It combines i) pondered design choices to tackle depth-from-mono in analogy to stereo matching, thanks to a correlation-based refinement module and ii) a more robust self-supervised training leveraging on proxy ground truth labels generated through a traditional (i.e.non-learning based) algorithm such as SGM. In contrast to state-of-the-art models [74, 137, 265], our architecture is elegantly trained in an end-to-end manner. Through exhaustive experiments, we prove that plugging proxy-supervision at training time leads to more accurate networks and, coupling this strategy with monoResMatch architecture, is state-of-the-art for self-supervised monocular depth estimation.

Chapter 22

Towards real-time unsupervised monocular depth estimation on CPU

The content of this chapter has been presented at International Conference on Intelligent Robots and Systems (IROS 2018) - "Towards real-time unsupervised monocular depth estimation on CPU" [186].

22.1 Introduction

Current architectures for monocular depth estimation are very *deep* and complex; for these reasons they require dedicated hardware such as high-end and power-hungry GPUs. This fact precludes to infer depth from a single image in many interesting applications fields characterized by low-power constraints (e.g. UAVs, wearable devices, ...) and thus in this work we propose a novel architecture for accurate and unsupervised monocular depth estimation aimed at overcoming this issue. By building our deep network inspired by the success of pyramidal architectures in other fields [78, 200] we are able to decimate the amount of parameters w.r.t. state-of-the-art solutions thus dramatically reducing both memory footprint and runtime required to infer depth. We call our model Pyramidal Depth Network (PyD-Net) and we train it in unsupervised manner as proposed in [69], representing the top-performing method in this field. Compared to such work, our model is about 94% smaller enabling on CPUs a notable speed-up at the cost of a slightly reduced depth accuracy. Moreover, our proposal outperforms other state-of-the-art methods. Our design strategy enables the deployment of PyD-Net even on embedded devices, such as the Raspberry Pi 3, thus allowing to infer a full depth map at about 2 Hz using less than 150 MB out of 1 GB memory available in such inexpensive device. To the best of our knowledge, our proposal is the first approach enabling fast and accurate unsupervised monocular depth estimation on standard and embedded CPUs.

22.2 Method

In this chapter we propose a novel framework for accurate and unsupervised monocular depth estimation with very limited resource requirements enabling such task even on CPUs of low power devices. State-of-the-art architectures proposed for this purpose [69] run in real time on high-end GPUs (e.g., Titan X), increasing the running time to nearly a second when running on standard CPUs and more than 10 s on embedded CPUs. Moreover, they count a huge number of parameters and thus require a large amount of memory at forward time. For these reasons, real-time performance with such models are feasible only with high-end and power hungry GPUs.

To overcome this issue, we propose a compact CNN, enabling accuracy comparable to state-of-the-art, with very limited memory footprint at test time (i.e., $<$ 150 MB) and capable to infer depth at about 2 fps on embedded devices such as the Raspberry Pi 3 and tens of fps on standard CPUs whereas other methods are far behind.

To this aim some recent works in other fields have shown how classical computer vision principles, such as image pyramid, can be effectively adopted to design more compact networks. SpyNet [200] and PWC-Net [231] are examples in the field of optical flow estimation with the latter representing state-of-the-art on MPI Sintel and KITTI flow benchmarks. The main difference with U-Net like networks is the presence of multiple small decoders working at different resolutions, directly on a pyramid of images [200] or features [231] extracted by a very simple encoder compared to popular ones such as VGG [222] or ResNet [79]. Results at each resolution are up-sampled to the next level to refine flow estimation. This method allows for a large reduction in the number of parameters together with a faster computation in optical flow and we follow a similar strategy for our monocular depth estimation network depicted in Figure 22.1. To train PyD-Net we adopt the unsupervised protocol proposed by Godard et al. [69] by casting depth prediction as an image reconstruction problem. For training unlabeled stereo pairs are required: for each sample, the left frame is processed through the network to obtain inverse depth maps (i.e., disparity maps) with respect to left and right images. These maps are used to warp the two input images towards each other and the reconstruction error is used as supervisory signal for back-propagation.

22.2.1 PyD-Net architecture

In this section we describe the proposed PyD-Net architecture depicted in Figure 22.1, a network enabling results comparable to state-of-the-art methods but with much less parameters, memory footprint and execution time.

Pyramidal features extractor

Input features are extracted by a small encoder architecture inspired by [231], made of 12 convolutional layers. At full resolution, the first layer produces the first level of the pyramid by applying convolutions with stride 2 followed by a second convolutional layer. Adopting this scheme at each level the resolution is decimated down to the lowest resolution (highest level of the pyramid) producing a total of 6 levels, from L1 to L6, corresponding respectively to image resolution from half to $\frac{1}{64}$ of the original input size. Each down-sampling module produces a larger number of extracted features, respectively 16, 32, 64, 96, 128, and 192, and each convolutional layers deploys 3×3 kernels and is followed by a leaky ReLU with $\alpha = 0.2$. Despite the small receptive field, this *coarse-*

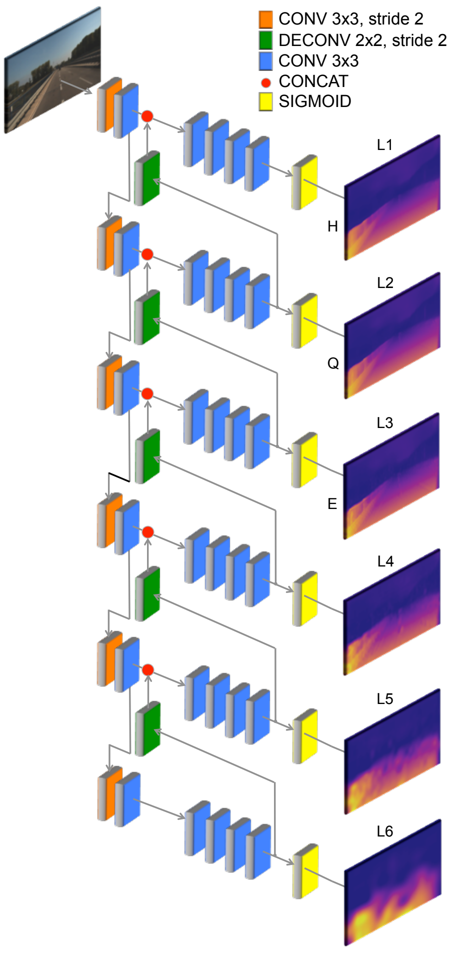


Figure 22.1: PyD-Net architecture. A pyramid of features is extracted from the input image and at each level a shallow network infers depth at that resolution. Processed features are then up-sampled to the above level to refine estimation, up to the highest one.

to-fine strategy allows us to include global context at the higher levels (i.e., lower image resolution) of the pyramid as well as to refine details at the lower levels (i.e., higher image resolution) and at the same time to significantly reduce the amount of parameters and memory footprint.

Depth decoders and upsampling

At the highest level of the pyramid, extracted features are processed by a depth decoder made of 4 convolutional layers, producing respectively 96, 64, 32 and 8 feature maps. The output of this decoder is used for two purposes: i) to extract a depth map at the current resolution, by means of a sigmoid operator and ii) to pass the processed features at the next level in the pyramid, by means of a 2×2 deconvolution with stride 2 which increases by a factor 2 the spatial resolution. The next level concatenates the features extracted from the input frame with those up-sampled and process them with a new decoder, repeating this procedure up to the highest resolution level. Each convolutional layer uses 3×3 kernels and is followed, as for deconvolutional layers, by leaky ReLU activations, except the last one which is followed by a Sigmoid activation to normalize the outputs. With such design, at each scale PyD-Net learns to predict depth at full resolution. We will show in the experimental results how this design strategy, up-sampling depth maps from lower resolution decoders, allows to quickly infer depth maps with accuracy comparable to state-of-the-art. Indeed, it requires only a subset of decoders at test time reducing memory requirements and runtime thus making our proposal suited for CPU deployment.

22.2.2 Training loss

We train PyD-Net to estimate depth at each resolution deploying a multi-scale loss function as sum of different contributions computed at scales $s \in [1..6]$

$$\mathcal{L}_s = \alpha_{ap}(\mathcal{L}^l_{ap} + \mathcal{L}^r_{ap}) + \alpha_{ds}(\mathcal{L}^l_{ds} + \mathcal{L}^r_{ds}) + \alpha_{lr}(\mathcal{L}^l_{lr} + \mathcal{L}^r_{lr}) \tag{22.1}$$

		Lower is better				Higher is better			
Method	Training dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params.
Eigen et al. [60]	K	0.203^5	1.548^4	6.307^4	0.282^5	0.702^4	0.890^5	0.958^5	54.2M
Liu et al. [131]	K	0.201^4	1.584^5	6.471^5	0.273^4	0.680^5	0.898^4	0.967^1	40.0M
Zhou et al. [293]	K	0.208^6	1.768^6	6.856^6	0.283^6	0.678^6	0.885^6	0.957^6	34.2M
Godard et al. [69]	K	0.148^1	1.344^1	5.927^1	0.247^1	0.803^1	0.922^1	0.964^2	31.6M
PyD-Net (50)	K	0.163^3	1.399^3	6.253^3	0.262^3	0.759^3	0.911^3	0.961^4	1.9M
PyD-Net (200)	K	0.153^2	1.363^2	6.030^2	0.252^2	0.789^2	0.918^2	0.963^3	1.9M
Garg et al. [64] cap 50m	K	0.169^4	1.080^4	5.104^4	0.273^4	0.740^4	0.904^4	0.962^4	16.8M
Godard et al. [69] cap 50m	K	0.140^1	0.976^1	4.471^1	0.232^1	0.818^1	0.931^2	0.969^2	
PyD-Net (50) cap 50m	K	0.155^3	1.045^3	4.776^3	0.247^3	0.774^3	0.921^3	0.967^3	
PyD-Net (200) cap 50m	K	0.145^2	1.014^2	4.608^2	0.227^2	0.813^2	0.934^1	0.972^1	
Zhou et al. [293]	CS+K	0.198^4	1.836^4	6.565^4	0.275^4	0.718^4	0.901^4	0.960^4	
Godard et al. [69]	CS+K	0.124^1	1.076^1	5.311^1	0.219^1	0.847^1	0.942^1	0.973^1	
PyD-Net (50)	CS+K	0.148^3	1.316^3	5.929^3	0.244^2	0.800^3	0.925^3	0.967^2	
PyD-Net (200)	CS+K	0.146^2	1.291^2	5.907^2	0.245^3	0.801^2	0.926^2	0.967^2	

Table 22.1: Evaluation on KITTI [66] using the split of Eigen et al. [60]. For training, K refers to KITTI dataset, CS+K means training on CityScapes [47] followed by fine-tuning on KITTI as outlined in [69]. On top and middle of the table evaluation of all existing methods trained on K, at the bottom evaluation of unsupervised methods trained on CS+K. We report results for PyD-Net with two training configurations.

The loss signal computed at each level of the pyramid is a weighted sum of three contributions computed on left and right images and predictions as in [69]. The first term represents the reconstruction error $\mathcal{L}_{ap}$, measuring the difference between the original image I^l and the warped one $\tilde{I}^l$ by means of SSIM [254] and L1 difference.

The disparity smoothness term, $\mathcal{L}_{ds}$, discourages depth discontinuities according to L1 penalty unless a gradient δI occurs on the image.

The third and last term includes the left-right consistency check, a well-known cue from traditional stereo algorithms [211], enforcing coherence between predicted left d^l and right d^r depth maps.

The three terms are also computed for right image predictions, as shown in Equation 22.1. As in [69], the right input image and predicted output are used only at training time, while at testing time our framework works as a monocular depth estimator.

22.3 Implementation details and training protocol

We implemented PyD-Net in TensorFlow [3] and for experiments we deployed a pyramid with 6 levels (i.e., from 1 to 6) producing depth maps at a maximum resolution of half the original input size, up-sampled at full resolution by means of bilinear interpolation. We adopt this strategy since in our experiments deploying levels up to full resolution with PyD-Net does not improve the accuracy significantly and increases the complexity of the network. With this setting, PyD-Net counts 1.9 million parameters and runs in about 15ms on a Titan X Maxwell GPU while [69], with the VGG model, counts 31 million parameters and requires 35ms. More importantly, our simpler model enables its effective deployment even on low-end CPUs aimed at embedded systems or smartphones.

We assess the effectiveness of our proposal with respect to the result reported in [69]. For a fair comparison with [69] we train our network with the same protocol for 50 epochs on batches of 8 images resized to 512×256, using 30 thousand images from KITTI raw data [66]. Moreover, we also provide results training PyD-Net for 200 epochs, showing how the final accuracy increases. It is worth noting that a longer schedule does not improve the performance of [69], already reaching top performance after 50 epochs. On a Titan X GPU training takes about, respectively, 10 and 40 hours. Note that [69] requires 20 hours for 50 epochs. The weights for our loss terms are always set to $\alpha_{ap} = 1$ and $\alpha_{lr} = 1$, while left-right consistency weight is set to $\alpha_{ds} = 0.1/r$, being r the down-sampling factor at each resolution layer as suggested in [69]. The inferred maps are multiplied by $0.3\times$ image width, producing an inverse depth map proportional to maximum disparity between the training pairs. We use Adam optimizer [118] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$. We used a learning rate of 10^{-4} for the first 60% epochs, halved every 20% epochs until the end. We perform data augmentation by randomly flipping input images horizontally and applying the following transformations: random gamma correction in [0.8,1.2], additive brightness in [0.5,2.0], and color shifts in [0.8,1.2] for each channel separately.

In [69] an additional post-processing step was proposed to filter out and replace artifacts near depth discontinuities and image borders induced by training on stereo pairs. However, it requires to forward the input image twice thus doubling the processing time and memory, for this reason we do not include it in our evaluation.

22.4 Experimental results

We evaluate PyD-Net with respect to state-of-the-art on the KITTI dataset [66]. In particular, we first test the accuracy of our model on a portion of the full KITTI dataset commonly used in this field [60], then we focus on performance analysis of PyD-Net on different hardware devices, highlighting how our model can run even on low-powered CPU at about 2 Hz still enabling satisfying results, even more accurate than most techniques known in literature.

22.4.1 Accuracy evaluation on Eigen split

We compare the performance of PyD-Net with respect to known techniques for monocular depth estimation using the same protocol of [69]. To do so, we use a test split of 697 images as proposed in [60]. The remaining 32 scenes are used to extract 22600 frames for training as in [64, 69]. As in [69], all methods use the same crop as [60] to be directly comparable. Table 22.1 reports extensive comparison with both supervised [60, 131] and unsupervised methods [69, 293]. To compare the performance of the considered methods we use metrics commonly adopted in this field [60] and we split our experiments into four main comparisons that we are going to discuss in detail.

In the first part of Table 22.1, we compare PyD-Net trained on 50 and 200 epochs to supervised works by Eigen et al. [60] and Liu et al. [131], as well as with other unsupervised techniques by Zhou et al. [293] and Godard et al. [69]. We report for each method the amount of parameters and, for each metric, the rank with respect to all the considered models. Excluding [69] we can notice how PyD-Net, with a very low number of parameters and with both training configurations, significantly outperforms all considered methods on all metrics with the exception of $\delta < 0.125^3$ on which Liu et al. [131] is even better than [69]. Compared to [69], our network is less accurate but training PyD-Net for 200 epochs yields almost equivalent results.

To compare with the results reported by Garg et al. [64], in the middle part of Table 22.1 we evaluate predicted maps up to a maximum depth of 50 meters as in [69]. Despite the smaller amount of parameters, reduced by a factor 8+, our network outperforms [64] with both training configurations and has performance very close, and even better with metrics $\delta < 0.125^2$ and $\delta < 0.125^3$ training for 200 epochs, to Godard et al. [69] a network counting more than 16$\times$ parameters. As for previous experiment we can notice that training PyD-Net for 200 epochs always yields better accuracy.

In the third part of Table 22.1, we compare the performance of PyD-Net with respect to Zhou et al. [293] and Godard et al. [69] unsupervised frameworks when trained on additional data. In particular, we first train the network for 50 epochs on CityScapes dataset and then we perform a fine-tuning on KITTI raw data according to the learning rate schedule described before. We can notice how training on additional data is beneficial for all the networks substantially confirming the previous trend. Godard et al. method outperforms all other approaches while training PyD-Net for 200 epochs yields overall best performance for this method. However, even training PyD-Net for only 50 epochs always enables to achieve a better accuracy compared to the much complex network by Zhou et al. [293].

To summarize, our lightweight PyD-Net architecture outperforms more complex state-of-the-art methods [60, 64, 131, 293] and has results in most cases comparable to top-performing approach [69]. Therefore, in the next section we evaluate in detail the impact of our design with respect to this latter method in terms of accuracy and execution time, on three heterogeneous hardware architectures, with different setting of the two networks.

	Power	250+ [W]	91+ [W]	3.5 [W]
Model	Res.	Titan X	i7-6700K	Raspberry Pi 3
Godard et al. [69]	F	0.035 s	0.67 s	10.21 s
Godard et al. [69]	H	0.030 s	0.59 s	8.14 s
PyD-Net	H	0.020 s	0.12 s	1.72 s
Godard et al. [69]	Q	0.028 s	0.54 s	6.72 s
PyD-Net	Q	0.011 s	0.05 s	0.82 s
Godard et al. [69]	E	0.027 s	0.47 s	5.23 s
PyD-Net	E	0.008 s	0.03 s	0.45 s

Table 22.2: Runtime analysis. We report for PyD-Net and [69] the average runtime required to process the same KITTI image with 3 heterogeneous architectures at Full, Half, Quarter and Eight resolution. The measured power consumption for the Raspberry Pi 3 concerns the whole system plus a Logitech HD C310 USB camera while for CPU and GPU it concerns only such devices.

		Lower is better				Higher is better		
Method	Res.	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 0.125$	$\delta < 0.125^2$	$\delta < 0.125^3$
Godard et al. [69]	F	0.124	1.076	5.311	0.219	0.847	0.942	0.973
Godard et al. [69]	H	0.126	1.051	5.347	0.222	0.843	0.940	0.972
PyD-Net (50)	H	0.148	1.316	5.929	0.244	0.800	0.925	0.967
PyD-Net (200)	H	0.146	1.291	5.907	0.245	0.801	0.926	0.967
Godard et al. [69]	Q	0.132	1.091	5.632	0.231	0.830	0.935	0.970
PyD-Net (50)	Q	0.152	1.342	6.185	0.252	0.789	0.920	0.964
PyD-Net (200)	Q	0.148	1.285	6.146	0.252	0.787	0.919	0.965
Godard et al. [69]	E	0.160	1.601	7.121	0.270	0.773	0.909	0.958
PyD-Net (50)	E	0.169	1.659	7.161	0.280	0.751	0.901	0.954
PyD-Net (200)	E	0.167	1.643	7.222	0.282	0.747	0.898	0.953

Table 22.3: Comparison between [69] and PyD-Net at different resolutions. All models were trained on CS+K datasets and results are not post-processed to achieve maximum speed. As for Table 22.1, we report results for PyD-Net with two training configurations.

22.4.2 Runtime analysis on different architectures

Having assessed the accuracy of PyD-Net with respect to state-of-the-art, we compare on different hardware platforms the performance of our network with the top-performing one by Godard et al. [69] . The reduced amount of parameters makes our model much less memory demanding and much faster thus allowing for real-time processing even on CPUs. This fact is highly desirable since GPU acceleration is not always feasible in applications scenarios characterized by low power constraints. Moreover, the pyramidal structure depicted in Figure 22.1 infers depth at different levels, getting more accurate at the higher resolution. This also happens for other models producing multi-scale outputs [69, 293]. Thus, given a trained instance of any of these models, we can process outputs up to a lower resolution (e.g., half or quarter) to reduce the amount of computations, memory requirements and runtime. Therefore, we'll also investigate the impact of such strategy in terms of accuracy and execution time for our method and [69].

Table 22.2 reports running time analysis for PyD-Net and Godard et al. [69] models estimating depth maps at different resolutions and with different devices. More precisely, the target systems are a Titan X Maxwell GPU, an i7-6700K CPU with 4 cores (4.2 Ghz) and a Raspberry Pi 3 board (ARM v8 processor Cortex-A53 1.2 Ghz). We report single forward time at full (F), half (H), quarter (Q) and eight (E) resolution. Full image resolution is set to 256×512 as in [69]. For PyD-Net we report results up to half resolution for the reason previously outlined. Moreover, results do not include the post-processing step proposed in [69] since it would duplicate the execution time and memory with small improvements in terms of accuracy. First of all, we can notice how the model by Godard et al. [69] is very fast on the high-end Titan X GPU while its performance drops dramatically when running on the Intel i7 CPU falling below 2 Hz. Moreover, it becomes unsuited for practical deployment on embedded CPUs such as the ARM processor of the Raspberry Pi 3. In this latter case it requires more than 10 seconds to process a single depth map at full resolution. We can also notice how early stopping of the network to infer depth at reduced resolution leads to a very small decrease of running time for this method hardly bringing the framerate above 2 Hz on the Intel i7 and requiring more than 5 seconds on a Raspberry Pi 3 even stopping at $\frac{1}{8}$ resolution. Looking at PyD-Net, even at the highest resolution H it takes 120 ms on the Intel i7 and less than 2 s on the ARM processor leading to $5\times$ speed up with respect to [69] at the same resolution. Moving to lower resolutions PyD-Net runs at 20 and 40 Hz, respectively, at Q and E resolutions yielding a speed up of $11\times$ and $18\times$ with respect to [69]. Moreover, PyD-Net breaks the 1 Hz barrier even on the Raspberry Pi 3, with 1.2 and 2.2 Hz and a speed up w.r.t. [69] of $8\times$ and $11\times$, respectively, at Q and E resolutions. On the same platform, equipped with 1 GB of RAM, our model requires 200, 150 and 120 MB, respectively, at H, Q and E resolution while the Godard et al. model about 275 MB at any resolution thus leaving a significantly smaller amount of memory available for other purposes.

These experiments highlight how PyD-Net enables, at the cost of small loss in accuracy, real-time performance on a standard CPU and it is also suited for practical deployment on devices with embedded CPUs. To better assess the trade-off between accuracy and execution time we report in Table 23.1 detailed experimental results concerning PyD-Net and [69] with different configurations/resolution. Results in the table were obtained from models trained on CS+K and evaluated on Eigen split [60]. We can observe how at E resolution PyD-Net performs similarly to the model proposed by Godard et al. [69] providing output of the same dimensions. However, the gain in terms of runtime is quite high for PyD-Net as highlighted in the previous evaluation. In particular our competitor barely breaks the 1 Hz barrier on the i7 CPU and it is far behind on the Raspberry Pi, while PyD-Net runs, respectively, at 40 fps and about 2 fps on the same platforms. As expected, from Table 23.1, stopping at lower resolution we can observe a loss in accuracy for both methods. However, it is worth to note that such reduction is more gradual for our network. Moreover, at E resolution the accuracy of Godard et al. network is substantially equivalent to PyD-Net with the advantages in terms of execution time previously discussed and reported in Table 22.2. Finally, from the table we can also notice that even at the lowest resolution E, PyD-Net outperforms all remaining methods [60, 64, 131, 293] working at full resolution reported in Table 22.1. Figure 22.2 reports a qualitative comparison between PyD-Net and Godard et al. [69] outputs at different resolutions.

The detailed evaluation reported proves that the proposed method can be effectively deployed on CPUs and actually it represents, to the best of our knowledge, the first architecture suited for CPU-based embedded systems enabling, for instance, its effective deployment with a Raspberry Pi 3 and a USB camera using a standard power bank for smartphones. Moreover, despite its reduced complexity it enables unsupervised training

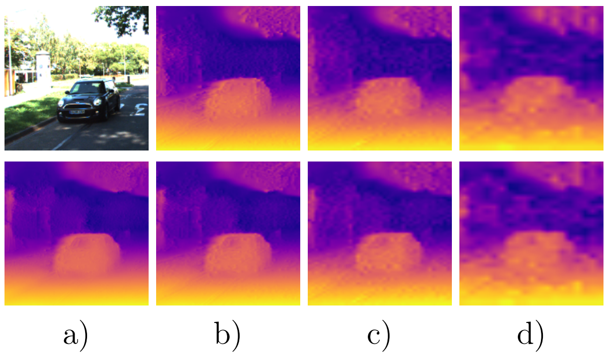


Figure 22.2: Qualitative comparison on a portion of a KITTI image between PyD-net (top) and Godard et al. [69] (bottom) respectively at F, H, Q and E resolution. Detailed timing analysis at each scale is reported in Table 22.2.

and outperforms almost all methodologies proposed in literature for monocular depth estimation including supervised ones.

22.5 Conclusions and future work

In this chapter we proposed PyD-Net, a novel and efficient architecture for unsupervised monocular depth estimation. As state-of-the-art method [69], it can be trained in unsupervised manner on rectified stereo pairs enabling comparable accuracy. However, the peculiar design of our network makes it suited for real-time applications on standard CPUs and also enables its effective deployment on embedded systems. Moreover, simplified configurations of our network allow to infer depth map at about 2 Hz on a Raspberry Pi 3 with accuracy higher than most state-of-the-art methods. Future work is aimed at mapping PyD-Net on embedded devices specifically tailored for computer vision applications, such as the Intel Movidius NCS, thus paving the way for real-time monocular depth estimation in applications with hard low-power constraints (e.g., UAVs, wearable and assistive systems, etc). A notable example can be found in [12], in which a strategy to achieve robust monocular depth estimation on lightweight handheld devices characterized by severe constraints concerning power consumption and computational resources has been proposed.

Chapter 23

Enabling energy-efficient unsupervised monocular depth estimation on ARMv7-based platforms

The content of this chapter has been presented at Design, Automation and Test in Europe Conference (DATE 2019) - “Enabling Energy-Efficient Unsupervised Monocular Depth Estimation on ARMv7-Based Platforms” [173].

23.1 Introduction

Monocular techniques, in particular, are an attractive solution for all those low-cost or portable applications where the use of multiple cameras would be too costly or cumbersome. The drawback is the computational power they need to achieve low latency. State-of-the-art methods rely on power-hungry GPUs indeed, a too costly option for many embedded systems where the primary constraints are power and energy consumption, form-factor, and assembling costs. Here’s come the challenge: bring depth estimation onto CPU-based COTS platforms, yet preserving accuracy and performance. That’s also the ultimate goal of this chapter, which describes design and optimization practices for deploying energy efficient unsupervised monocular depth estimation on the ARMv7-A core.

Two key factors are paramount to make the GPU-to-CPU shift success: (*i*) the design of a network topology able to reach high accuracy with an amount of resources compliant to that of an ordinary CPU, (*ii*) the availability of a vertical optimization stack for fast code optimization and effective deploying to real hardware. Concerning the first point, we borrowed the *Pyramidal Depth Network* (PyD-Net) recently introduced in [186] for unsupervised monocular depth estimation. It consists of a lightweight Convolutional Neural Network (CNN) yielding close to state-of-the-art and almost real-time performance on commercial high-end CPUs. The PyD-Net also offers multi-resolution estimation, a key feature to enable adaptive energy-accuracy scaling. Starting from this model, we propose smart strategies for network refinement, redundancy removal and software porting. The optimization stages have been integrated into an inference optimizer which delivers a low-latency/high-throughput code of the PyD-Net. The front-end runs a high-level optimization through quantization to 16-bit and 8-bit fixed-point; the method is fast, highly accurate and hardware-friendly. The back-end is in charge of the code compilation and the optimal mapping onto the target ARMv7 architecture; it leverages a new optimized version of integer neural kernels designed to exploit the computing resources of the Single Instruction Multiple Data (SIMD) datapath available in the Cortex-A architecture.

The savings achieved with the proposed design and optimization strategies bring embedded depth estimation beyond the state-of-the-art in terms of accuracy/energy-efficiency. The results obtained with the ARMv7-A core show the quantized versions of the PyD-Net (16- and 8-bit) have marginal accuracy losses and substantial speed-up w.r.t. the floating-point (32-bit). That leads to high energy efficiency, demonstrating both portability and scalability.

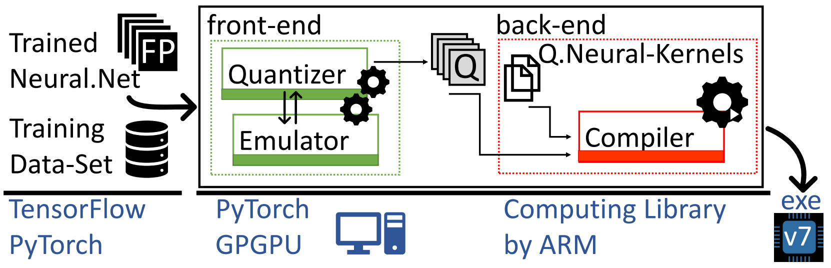


Figure 23.1: Integrated flow for PyD-Net optimization and deployment on ARMv7.

23.2 Optimization Framework

The framework depicted in Fig. 23.1 is designed for a fast deploying of fixed-point PyD-Nets with ARMv7 cores. Written in Python (ver. 3.6.5), it consists of two main parts. The *front-end* takes the PyD-Net trained with 32-bit floating-point (FP in Fig. 23.1) and returns the fixed-point model Q. The quantization scheme is accuracy-driven and implements a hybrid strategy: the bit-width (16-bit or 8-bit, user defined) is static for all the layers, while the radix-point is assigned on a per-layer basis. This design choice allows a proper organization of the low-level code and an efficient management of the hardware resources. Incremental re-training (i.e. fine-tuning) is operated in order to recover the loss introduced by quantization.

The quantizer leverages a fixed-point emulator for fast, yet accurate loss assessment during optimization. It makes use of PyTorch libraries (ver. 0.4.1) and implements a customized version of fake-quantization [100] that works as follows. During the inference stage, a software wrapper converts data (stored in fixed-point) to floating-point. This strategy enables operations with GPU cores. Once processed, data are converted back in fixed-point and adjusted with auxiliary transformations (e.g. saturation, truncation, binary-shift) that replicate the behavior of the fixed-point units of the ARMv7 core (e.g. saturation of the accumulator register, set-up of the radix-point position). A comparison against the results produced with the ARMv7 core revealed the emulator is highly accurate: the maximum absolute error is 8e-3 at the output of the network, with no impact on the final accuracy of the depth estimation. If compared to a standard training run, the execution time increases by 20% (from 10 to 12 min. per epoch).

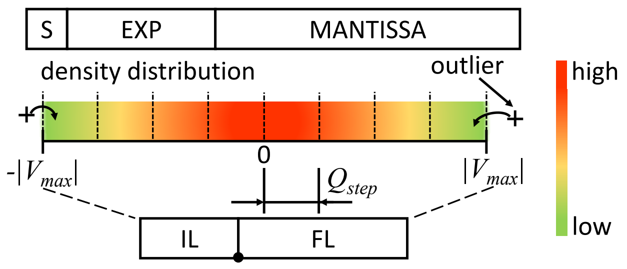


Figure 23.2: Floating-Point to Fixed-point quantization.

The *back-end* flow, powered by the GNU Arm Embedded Toolchain [2] is fed with the fixed-point model Q and returns a binary file. A set of new integer kernels optimized for 16- and 8-bit (*Q.Neural-Kernel*) enrich the ARM Computing Library repository. Such kernels are designed to implement the proposed per-layer fixed-point scheme and to improve the utilization of the SIMD media accelerator of the ARMv7.

23.2.1 Fixed-Point Scaling

A real value V can be represented with a binary string Q of length BW (bit-width) using the following mapping function:

$$V = Q \cdot 2^{-FL} \tag{23.1}$$

where FL the fraction length, i.e. the position of the radix-point in Q. Given a set of real values, e.g. the weights and activations of the PyD-Net model, the choice of BW and FL affects the information loss due to quantization. This scenario is pictorially illustrated in Fig. 23.2, where the gradient bar reflects the density distribution of the floating-point values, $|V_{max}|$ is the largest modulus in the set, Q_{step} represents the quantization step.

Since the bit-width BW is defined by the available hardware (16 or 8 for the ARMv7), the problem reduces to searching the optimal FL (the integer length IL is then given by BW-IL). FL is constrained by $|V_{max}|$ as described in the following equation:

$$FL = \left\lfloor \log_2 \left(\frac{2^{BW-1} - 1}{|V_{max}|} \right) \right\rfloor \tag{23.2}$$

However, a trade-off with the quantization step Q_{step} does also exist: the smaller the

FL, the larger the Q_{step}. The decision of which constraint to guard more ($|V_{max}|$ or Q_{step}) mainly depends on the distribution of the original values and their importance in the neural model.

It is worth mentioning that our problem formulation considers a symmetric distribution $[-|V_{max}|, +|V_{max}|]$; outliers are clamped to smaller values to reduce $|V_{max}|$ and improve accuracy. The adopted quantization scheme is linear (i.e. uniform intervals) with binary radix-point scaling.

Even though other quantization strategies may achieve higher accuracy, their hardware implementation results less efficient. For instance, an asymmetric method, e.g. [100], requires additional output pipeline stages that affect performance as demonstrated in [230], while floating-point scaling makes use of data-type conversions that reduce to simple shift operation with binary scaling. For the specific case of PyD-Net, our quantization strategy achieves almost the same accuracy of floating-point, making other complex schemes irrelevant.

We adopted a dynamic fixed-point scheme where the fraction length is defined layer-by-layer. More specifically, the optimization procedure aims at finding the FL_{opt} that minimizes the L2 distance between the original 32-bit floating point values X and the quantized values Q. Applied to both activations and weights independently, the procedure encompasses the following stages. First, a range analysis of weights and activations distribution; for the latter case, a subset of the training set is used (referred as *calibration set*).

Second, extraction of lower-bound and upper-bound of the fraction length: FL_{lb}, FL_{ub}. Both are calculated with Eq. 23.2.1 using the different V_{max} as follows:

$$V_{max} = \begin{cases} \max(|X_{min}|), |X_{max}|) & \text{for } FL_{lb} \\ \max(|X_{min}|), |X_{max}|)/K & \text{for } FL_{ub} \end{cases} \tag{23.3}$$

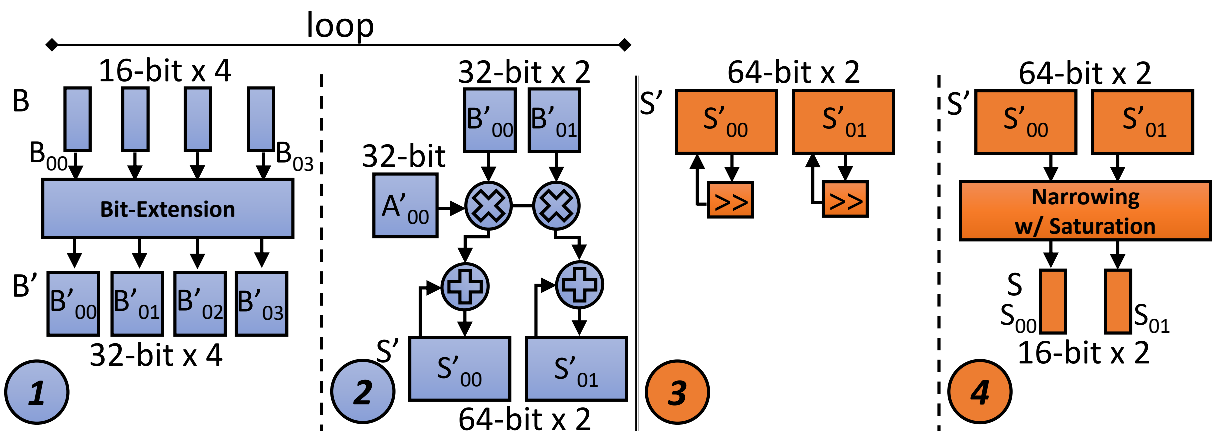


Figure 23.3: Q.Neural-Kernel: execution flow for 16-bit fixed-point.

with K an arbitrary large integer[1]. Third, repeated test using the available $FL \in [FL_{min}, FL_{max}]$ and selection of FL_{opt}, i.e. the one that minimizes the L2 error. Half-even rounding is used for the quantization of trainable parameters.

23.2.2 Fixed-Point Convolution Kernels for the ARMv7

The belief that fixed-point representations reduce energy consumption due to less complex arithmetic is not exactly true. The actual benefit lies in the ability to store the same amount of information with fewer bits which in turn enables a more efficient use of the memory bandwidth [233]. However, quantization alone isn't much use. It requires a smart orchestration of the hardware processing units in order to keep pace with the higher throughput brought with bit-width scaling.

The NEON Media Processing Engine of the ARMv7-A architecture has an advanced arithmetic SIMD unit with parallel floating-point and integer units. The register file can be configured to host 8-, 16-, 32-, 64-, 128-bit data, while the integer data-path supports 8-, 16-, 32- or 64-bit operations. To explain the implemented kernel routine we resort to the example of Fig. 23.3, which gives a sketch of the convolution S of two matrices A and B, with B as the $N \times N$ kernel of a generic layer and A the input feature. More precisely, it illustrates the parallel calculation of two outputs S_{00} and S_{01}. In general, $S_{ij} = \sum_{x}^{N\times N} A_{ix} \cdot B_{xj}$. The example is for 16-bit fixed-point; the same holds for 8-bit, yet

[1]We empirically verified K=100 is an optimal choice for PyD-Net

with doubled parallelism. The flow is as follows:

(1) the 16-bit (8-bit) input operands, $A_{i,x}$ and $B_{x,j}$ are extended to 32-bit (16-bit) obtaining $A'_{i,x}$ and $B'_{x,j}$ (Fig. 23.3 refers to $B_{x,j}$ only).

(2) two (four) fused multiply& accumulate (MAC) operations are executed in parallel; the result is stored into a 64-bit (32-bit) register $S'_{i,j}$

(3) after N×N loops, the two (four) results are ready to be packed and then stored in the main memory; an output processing stage (highlighted in orange) is in charge of the radix-point shifting according to the desired radix-point position.

(4) the result is shrunk to the original bit-width, i.e. 16-bit (8-bit), and eventually saturated.

The bit-extension of step **(1)** guarantees 32- (16-) guard-bits for the accumulation. This operation is paramount as it avoids overflow/underflow during accumulation. Bypassing this stage may achieve twice the parallelism, but results are highly inaccurate. Overall, the number of parallel MAC is 2 for 16-bit and 4 for 8-bit. Further improvements are limited by the maximum bit-width of the register file: 128-bit. To notice that the parallelism of FP32 is 4. Contrary to what is thought, floating-point is intrinsically more efficient as it makes better use of the local registers. Also, it does not require the additional output stage steps (**(3)** and **(4)** in Fig. 23.3). Despite that, the performance of fixed-point convolutional networks improve over FP32. This is due to the following factors: (*i*) enhanced utilization of memory bandwidth, as the cost of accessing 16-bit (8-bit) data is half (quarter) the cost of floating-point; (*ii*) smaller memory footprint for storing weights and partial results, hence less RAM usage, (*iii*) higher hit-rate in cache. This analysis is confirmed by experimental evidence.

23.3 Results

23.3.1 Experimental set-up

PyD-Net and dataset. PyD-Net infers disparity maps at different resolutions thus enabling accuracy-effort scaling. The three options available, i.e. H, Q and E, have been explored for a parametric analysis of functional properties, i.e. depth accuracy, and non-functional properties, i.e. binary and RAM space, throughput and energy efficiency, under different arithmetic precision, i.e. 32-bit floating-point (**FP32**), 16-bit fixed-point (**FX16**) and 8-bit fixed-point (**FX8**). The baseline is resolution H at FP32 (H@FP32).

The baseline and the starting point of our work is the pre-trained PyD-Net model. It was trained, as described in [186], for 200 epochs on batches of 8 images randomly picked from the training-set and resized to 512×256.

Front-end. The quantization stage encompasses a range analysis of the activations in order to define the lower-/upper-bound of the fraction-length. For this stage, we used a calibration set filled with 5000 images randomly picked from the training-set. To notice that the intersection between testing-set and the calibration-set is void.

The fine-tuning stage (applied at post-quantization) consists of 25 training epochs made run over the full training-set using the Adam optimizer. The hyper-parameters are as follows: learning rate 1.0e-7, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, weight decay = 0. The loss function and its parameters are those described in [186]. The flow is run on the NVIDIA Titan XP GPU with CUDA 8.0.

Back-end and hardware. The proposed GEMM-based Q.Neural-Kernels are written in C++ and inline assembly code. They are integrated into the ARM Compute Library version 18.05 built with `scons` ver. 2.4.1 and the *gcc-linaro* toolchain ver. 6.4.0-2018.05. To notice that other existing libraries do not support the dynamic fixed-point scheme adopted in this work.

The embedded platform used as test-bench is the Raspberry PI 3B loaded with a 32-bit Ubuntu Mate 16.04. The board hosts a quad-core BCM2837 chip-set powered to 1.2V.

	Lower is better				Higher is better		
Config.	Abs Rel	Sq Rel	RMSE	RMSE log	$a1$	$a2$	$a3$
H@FP32	0.146	1.298	5.859	0.241	0.802	0.927	0.968
H@FX16	0.147	1.331	5.925	0.243	0.801	0.926	0.967
H@FX16-ft	0.147	1.302	5.945	0.244	0.798	0.925	0.967
H@FX8	0.177	1.893	6.621	0.272	0.768	0.911	0.958
H@FX8-ft	0.148	1.337	6.018	0.246	0.795	0.924	0.967
Q@FP32	0.149	1.350	6.128	0.246	0.795	0.923	0.966
Q@FX16	0.149	1.365	6.155	0.246	0.794	0.923	0.966
Q@FX16-ft	0.149	1.342	6.176	0.248	0.790	0.921	0.966
Q@FX8	0.183	2.364	7.457	0.270	0.766	0.908	0.957
Q@FX8-ft	0.158	1.427	6.290	0.257	0.778	0.917	0.964
E@FP32	0.162	1.699	7.141	0.266	0.768	0.907	0.959
E@FX16	0.165	1.770	7.327	0.271	0.762	0.904	0.957
E@FX16-ft	0.162	1.712	7.163	0.266	0.768	0.907	0.958
E@FX8	0.193	2.758	8.507	0.288	0.745	0.893	0.950
E@FX8-ft	0.171	1.829	7.430	0.276	0.751	0.901	0.956

Table 23.1: Experimental results concerning depth estimation accuracy. Comparison between original PyDNet [186] (FP32) and optimized architectures at different resolutions. The four CPU cores, which belong to the ARMv8-A family, support the ARMv7-A 32-bit instruction-set in backward compatibility mode. The reason behind the adoption of the ARMv7 architecture lies in the possibility of using the depth estimation on a wider range of systems, e.g. the older Raspberry PI 2, which show lower power consumption. All the power-management features were disabled during the experimental campaign; the board drains 3.5 under full utilization (4 cores ON and maximum workload); moving from FP32 to FX16 or FX8 has no effect on the total power consumption, dominated by memory accesses.

23.3.2 Accuracy vs Quantization

Table 23.1 reports an evaluation of several variants of the original PyD-Net architecture. We point out once more that the model by Godard et al. [69], counting 15× parameters with respect to PyD-Net (30 vs 1.9 million), is not suited for embedded devices and not used as benchmark for non-functional properties in the next section[2]. We discuss in detail the effects introduced at each resolution (H, Q, E) by different data types (floating point,

[2]For the sake of space, please refer to [186] for a comparison with Godard et al. [69]. For the same reason, parameters a1, a2 and a3 correspond, respectively, to $\delta < 1.25$, $\delta < 1.25^2$ and $\delta < 1.25^3$ in [186].

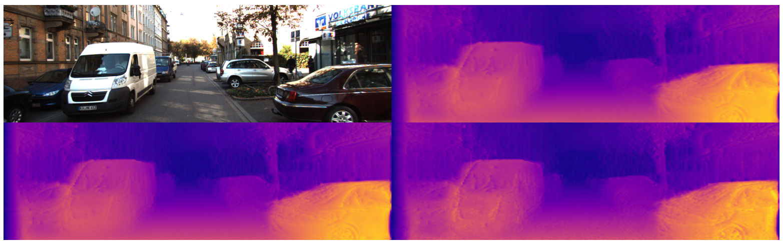


Figure 23.4: Top row: Input image from KITTI dataset (left) and depth map H@FP32 computed by the original PyD-Net network [186] (right). Bottom row: depth maps H@FX16-ft (left) and H@FX8-ft (right).

16 or 8 bit fixed point, referred to FP32, FX16 and FX8) and optional fine-tuning (-ft) carried out after quantization, reporting error and accuracy metrics commonly adopted for evaluating depth-from-mono performance [60], assuming a 80m cap distance [69].

Looking at the table, one can notice a similar trend for the three resolution H, Q, and E, for all the seven metrics. The 16-bit fixed-point quantization FX16 introduces a negligible accuracy drop if compared to the original PyD-Net FP32. For each of the three resolutions, the additional fine-tuning (-ft) has a marginal impact on the performance; that's due to the fact that FX-16 is already very close to FP32. The 8-bit fixed-point quantization FX8 is more prone to accuracy drop, with substantial loss for Q and E. However, fine-tuning the network dramatically improves performance, thus closing the gap with FX16 quantization and, most notably, with the original FP32 strategy. These facts can also be perceived by Fig. 23.4.

23.3.3 Performance, Memory Space and Energy Efficiency

Table 23.2 collects the hardware-related metrics measured during the test-run: memory space (for weights storage), RAM usage (during PyD-Net processing) and energy efficiency (average frames per). The throughput (frames/second) can be derived by multiplying energy efficiency (frames/J) by the total power (3.5).

As expected, energy efficiency improves when working at lower resolutions: the upper depth estimators of the net are disabled alleviating the workload. For instance, using

Resolution	Precision	Space ()	RAM ()	Frame/J
H	FP32	7.6	206	0.141 (×**1.00**)
	FX16	3.8	118	0.156 (×**1.11**)
	FX8	1.9	62	0.210 (×**1.49**)
Q	FP32	7.2	60	0.386 (×**2.74**)
	FX16	3.6	34	0.420 (×**2.99**)
	FX8	1.8	19	0.635 (×**4.51**)
E	FP32	6.8	53	0.794 (×**5.64**)
	FX16	3.4	28	0.922 (×**6.55**)
	FX8	1.7	14	1.299 (×**9.23**)

Table 23.2: Non-functional metrics of PyD-net at different resolutions and precisions on ARMv7-A

FP32 the improvement from high (H@FP32) to low resolution is 5.64×. That's the savings brought by the reconfigurable topology of the PyD-Net. Considering the same resolution, energy efficiency gets larger with smaller data representations. For instance, at high resolution the gain from H@FP32 to H@FX8 is 49%; at low resolution, the gain grows up to 63.7%. To notice that, as previously outlined, moving from floating-point to fixed-point affects accuracy only marginally. The combined action of resolution scaling and precision scaling enables even larger optimization: from H@FP32 (0.141Frames/) to E@FX8 (1.299Frames/) the energy efficiency increases by 9.23×.

This first analysis gives clear evidence of the scaling properties of the quantized PyD-Net model, the benefits of multiple precision arithmetic, and the effectiveness of the porting flow on the target architecture. A sensing technology with such ability to implement accuracy-energy scaling represents a practical option for adaptive embedded systems: contexts or applications which tolerate lower accuracy might pursue higher energy efficiency by tuning resolution (coarse-knob) and precision (fine-knob).

Concerning memory, both the binary space and the RAM usage are important metrics that reflect the efficiency of the proposed implementation. As expected, the space for storing network parameters reduces linearly with the precision, e.g. from 7.6 with H@FP32 to 1.9 with H@FX8. At low resolution and low precision the memory space is just 1.7 (E@FX8), which brings the overall savings w.r.t. H@FP32 up to ×4.5. Even more

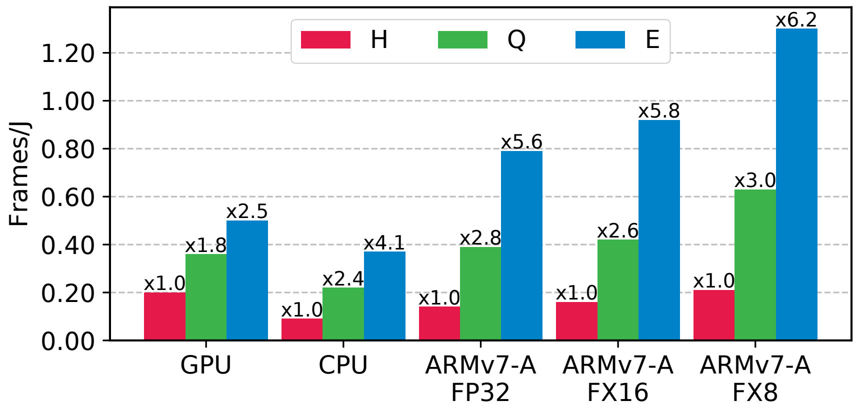


Figure 23.5: Energy efficiency vs. Resolution using GPU, CPU and the ARMv7-A.

interesting is the analysis of the RAM. Its utilization is dramatically reduced with an overall scaling factor of 14.7×: from H@FP32 (206) to E@FX8 (14). Indeed, the same PyD-Net compiled using prior kernels available in the ACL library showed fixed-point implementations are more resource hungry than floating-point.

As a final remark, Fig. 23.5 provides a technology comparison among different hardware options: a GPU (Titan X Maxwell), a high-end CPU (Intel i7-6700K CPU), and the ARMv7 at FP32, FX8 and FX16. The bar chart shows the energy efficiency (Frames/) for all the possible permutation of resolution, precision and hardware; the labels refer to the normalization w.r.t. high resolution (H) for each hardware option separately. Moving from H to E with the ARMv7@FX8 improves energy by 6.2×. To notice that ARMv7@FX16 and ARMv7@FX8 outperform both the GPU and CPU by far. A more interesting aspect the lower the arithmetic precision, the larger the gain brought by resolution scaling. While in GPU and CPU the gain from H to E is limited to ×2.5 and ×4 respectively, it grows to 5.6× with the ARMv7@FP32 and 6.2× with the ARMv7@FX8. This feature might open interesting optimization problems for resource management at run-time.

23.4 Conclusions

This chapter introduces a comprehensive design&optimization framework aimed at improving the energy efficiency of depth perception on low power embedded devices. The target is the ARMv7, a RISC architecture widely adopted for low-cost systems. Compared to a high-end CPU (Intel i7) and a GPU (NVIDIA Titan X), the proposed implementation reaches higher energy efficiency with a negligible accuracy degradation that is tolerable by many embedded applications. More interestingly, the joint co-operation between (*i*) the design of the tiny, yet reconfigurable PyD-Net and (*ii*) the optimization enabled by the hardware-friendly fixed-point quantization allows achieving a scalability that goes beyond the state-of-the-art. These features pave the way to a widespread deployment of adaptive energy-accuracy monocular 3D sensing in additional application domains constrained by stringent energy requirements.

Chapter 24

Distilled semantics for comprehensive scene understanding from videos

The content of this chapter has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2020) - “Distilled Semantics for Comprehensive Scene Understanding from Videos” [243].

24.1 Introduction

What information would an autonomous agent be keen to gather from its sensory subsystem to tackle tasks like navigation and interaction with the explored environment? It would need to be informed about the geometry of the surroundings and the type of objects therein, and likely better know which of the latter are actually moving and how they do so. What if all such cues may be provided by as simple a sensor as a single RGB camera?

Nowadays, deep learning is advancing the state-of-the-art in classical computer vision problems at such a quick pace that single-view holistic scene understanding seems to be no longer out-of-reach. Indeed, highly challenging problems such as monocular depth estimation and optical flow can nowadays be addressed successfully by deep neural networks, often through unified architectures [16, 270, 296]. Self-supervised learning techniques have yielded further major achievements [154, 293] by enabling effective training of deep networks without annotated images. In fact, labels are hard to source for depth estimation due to the need of active sensors and manual filtering, and are even more cumbersome in the case of optical flow. Concurrently, semi-supervised approaches [41, 277] proved how a few semantically labelled images can improve monocular depth estimation significantly. These works have also highlighted how, while producing per-pixel class labels is tedious yet feasible for a human annotator, manually endowing images with depth and optical flow ground truths is prohibitive.

In this chapter, we propose the first-ever framework for comprehensive scene understanding from monocular videos. As highlighted in Figure 24.1, our multi-stage network architecture, named ΩNet, can predict depth, semantics, optical flow, per-pixel motion probabilities and motion masks. This comes alongside with estimating the pose between adjacent frames for an uncalibrated camera, whose intrinsic parameters are also estimated. Our training methodology leverages on self-supervision, knowledge distillation and multi-task learning. In particular, peculiar to our proposal and key to performance is distillation of proxy semantic labels gathered from a state-of-the-art pre-trained model [130] within a self-supervised and multi-task learning procedure addressing depth, optical flow and motion segmentation. Our training procedure also features a novel and effective self-distillation schedule for optical flow mostly aimed at handling occlusions and relying on tight integration of rigid flow, motion probabilities and semantics. Moreover, ΩNet is lightweight, counting less than 8.5M parameters, and fast, as it can run at nearly 60 FPS and 5 FPS on an NVIDIA Titan Xp and a Jetson TX2, respectively. As vouched by thorough experiments, the main contributions of our work can be summarized as follows:

• The first real-time network for joint prediction of depth, optical flow, semantics and motion segmentation from monocular videos

• A novel training protocol relying on proxy semantics and self-distillation to effectively address the self-supervised multi-task learning problem

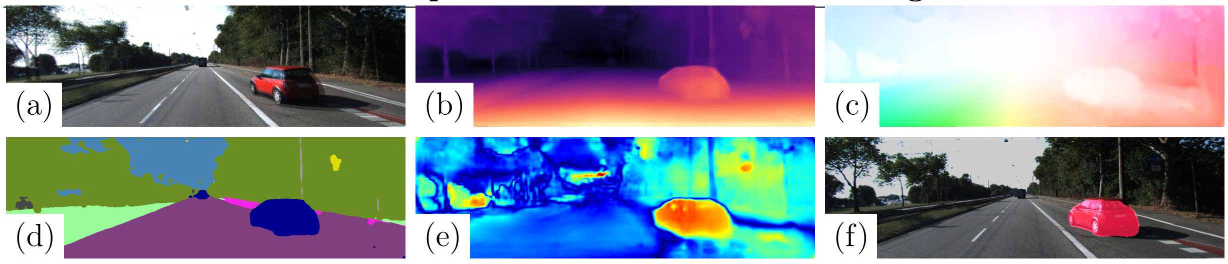


Figure 24.1: Given an input monocular video (a), our network can provide the following outputs in real-time: depth (b), optical flow (c), semantic labels (d), per-pixel motion probabilities (e), motion mask (f).

- State-of-the-art self-supervised monocular depth estimation, largely improving accuracy at long distances
- State-of-the-art optical flow estimation among monocular multi-task frameworks, thanks to our novel occlusion-aware and semantically guided training paradigm
- State-of-the-art motion segmentation by joint reasoning about optical-flow and semantics

24.2 Overall Learning Framework

Our goal is to develop a real-time comprehensive scene understanding framework capable of learning strictly related tasks from monocular videos. Purposely, we propose a multi-stage approach to learn first geometry and semantics, then elicit motion information, as depicted in Figure 24.2.

24.2.1 Geometry and Semantics

Self-supervised depth and pose estimation. We propose to solve a self-supervised single-image depth and pose estimation problem by exploiting geometrical constraints in a sequence of N images, in which one of the frames is used as the target view I_t and the other ones in turn as the source image I_s. Assuming a moving camera in a stationary scene, given a depth map D_t aligned with I_t, the camera intrinsic parameters K and the relative pose $T_{t\rightarrow s}$ between I_t and I_s, it is possible to sample pixels from I_s in order

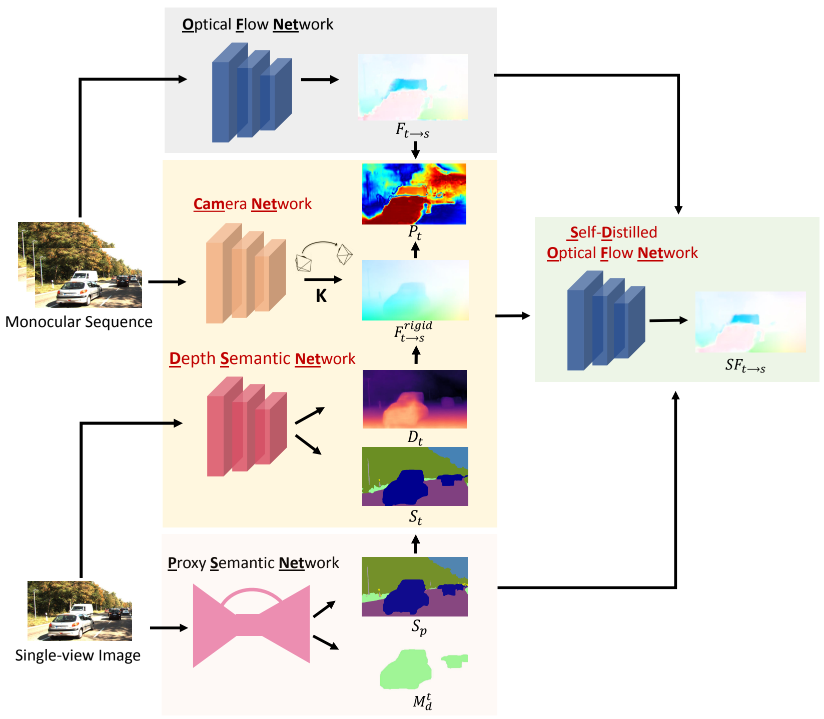


Figure 24.2: Overall framework for training ΩNet to predict depth, camera pose, camera intrinsics, semantic labels and optical flow. In red architectures composing ΩNet.

to synthesise a warped image $\widetilde{I}_t$ aligned with I_t. The mapping between corresponding homogeneous pixels coordinates $p_t \in I_t$ and $p_s \in I_s$ is given by:

$$p_s \sim K T_{t\to s} D_{p_t} K^{-1} p_t \tag{24.1}$$

Following [293], we use the sub-differentiable bilinear sampler mechanism proposed in [101] to obtain $\widetilde{I}_t$. Thus, in order to learn depth, pose and camera intrinsics we train two separate CNNs to minimize the photometric reconstruction error between $\widetilde{I}_t$ and I_t, defined as:

$$\mathcal{L}_{ap}^{D} = \sum_{p} \psi(I_t(p), \widetilde{I}_t(p)) \tag{24.2}$$

where ψ is a photometric error function between the two images. However, as pointed out in [70], such a formulation is prone to errors at occlusion/disocclusion regions or in static camera scenarios. To soften these issues, we follow the same principles as suggested in [70], where a minimum per-pixel reprojection loss is used to compute the photometric error, an automask method allows for filtering-out spurious gradients when the static camera assumption is violated, and an edge-aware smoothness loss term is used as in [69]. Moreover, we use the depth normalization strategy proposed in [249].

We compute the rigid flow between I_t and I_s as the difference between the projected and original pixel coordinates in the target image:

$$F_{t\rightarrow s}^{rigid}(p_t) = p_s - p_t \tag{24.3}$$

Distilling semantic knowledge. The proposed distillation scheme is motivated by how time-consuming and cumbersome obtaining accurate pixel-wise semantic annotations is. Thus, we train our framework to estimate semantic segmentation masks S_t by means of supervision from cheap proxy labels S_p distilled by a semantic segmentation network, pre-trained on few annotated samples and capable to generalize well to diverse datasets. Availability of proxy semantic labels for the frames of a monocular video enables us to train a single network to predict jointly depth and semantic labels. Accordingly, the joint loss is obtained by adding a standard cross-entropy term $\mathcal{L}_{sem}$ to the previously defined self-supervised image reconstruction loss $\mathcal{L}_{ap}^{D}$. Moreover, similarly to [277], we deploy a cross-task loss term, $\mathcal{L}_{edge}^{D}$, aimed at favouring spatial coherence between depth edges and semantic boundaries. However, unlike [277], we do not exploit stereo pairs at training time.

24.2.2 Optical Flow and Motion Segmentation

Self-supervised optical flow. As the 3D structure of a scene includes stationary as well as non-stationary objects, to handle the latter we rely on a classical optical flow formulation. Formally, given two images I_t and I_s, the goal is to estimate the 2D motion vectors $F_{t\rightarrow s}(p_t)$ that map each pixel in I_t into its corresponding one in I_s. To learn such a mapping without supervision, previous approaches [133, 154, 270] employ an image reconstruction loss $\mathcal{L}_{ap}^{F}$ that minimizes the photometric differences between I_t and the back-warped image $\widetilde{I}_t$ obtained by sampling pixels from I_s using the estimated 2D optical flow $F_{t\to s}(p_t)$. This approach performs well for non-occluded pixels but provides misleading information within occluded regions.

Pixel-wise motion probability. Non-stationary objects produce systematic errors when optimizing $\mathcal{L}_{ap}^{D}$ due to the assumption that the camera is the only moving body in an otherwise stationary scene. However, such systematic errors can be exploited to identify non-stationary objects: at pixels belonging to such objects the rigid flow $F_{t\to s}^{rigid}$ and the optical flow $F_{t\to s}$ should exhibit different directions and/or norms. Therefore, a pixel-wise probability of belonging to an object independently moving between frames s and t, P_t, can be obtained by normalizing the differences between the two vectors. Formally, denoting with $\theta(p_t)$ the angle between the two vectors at location p_t, we define the per-pixel motion probabilities as:

$$P_t(p_t) = \max\{\frac{1-\cos\theta(p_t)}{2}, 1-\rho(p_t)\} \tag{24.4}$$

where $\cos\theta(p_t)$ can be computed as the normalized dot product between the vectors and evaluates the similarity in direction between them, while $\rho(p_t)$ is defined as

$$\rho(p_t) = \frac{\min\{\|F_{t\to s}(p_t)\|_2, \|F_{t\to s}^{rigid}(p_t)\|_2\}}{\max\{\|F_{t\to s}(p_t)\|_2, \|F_{t\to s}^{rigid}(p_t)\|_2\}}, \tag{24.5}$$

i.e.a normalized score of the similarity between the two norms. By taking the maximum of the two normalized differences, we can detect moving objects even when either the directions or the norms of the vectors are similar. A visualization of $P_t(p_t)$ is depicted in Fig. 24.3(d).

Semantic-aware Self-Distillation Paradigm. Finally, we combine semantic information, estimated optical flow, rigid flow and pixel-wise motion probabilities within a final training stage to obtain a more robust self-distilled optical flow network. In other words, we train a new instance of the model to infer a self-distilled flow $\mathrm{SF}_{t\to s}$ given the estimates $\mathrm{F}_{t\to s}$ from a first self-supervised network and the aforementioned cues. As previously discussed and highlighted in Figure 24.3(c), standard self-supervised optical flow is prone to errors in occluded regions due to the lack of photometric information but can provide good estimates for the dynamic objects in the scene. On the contrary, the estimated rigid flow can properly handle occluded areas thanks to the minimum-reprojection mechanism [70]. Starting from these considerations, our key idea is to split the scene into stationary and potentially dynamics objects, and apply on them the proper supervision. Purposely, we can leverage several observations:

1. **Semantic priors.** Given a semantic map S_t for image I_t, we can binarize pixels into static M_t^s and potentially dynamic M_t^d, with $M_t^s \cap M_t^d = \emptyset$. For example, we expect that points labeled as *road* are static in the 3D world, while pixels belonging to the semantic class *car* may move. In M_t^d, we assign 1 for each potentially dynamic pixel, 0 otherwise, as shown in Figure 24.3(e).

2. **Camera Motion Boundary Mask.** Instead of using a backward-forward strategy [296] to detect boundaries occluded due to the ego-motion, we analytically compute a binary boundary mask M_t^b from depth and ego-motion estimates as proposed in [142]. We assign a 0 value for out-of-camera pixels, 1 otherwise as shown in Figure 24.3(f).

3. **Consistency Mask.** Because the inconsistencies between the rigid flow and $\mathrm{F}_{t\to s}$ are not only due to dynamic objects but also to occluded/inconsistent areas, we can leverage Equation (24.4) to detect such critical regions. Indeed, we define the consistency mask as:

$$M_t^c = P_t < \xi, \xi \in [0, 1] \tag{24.6}$$

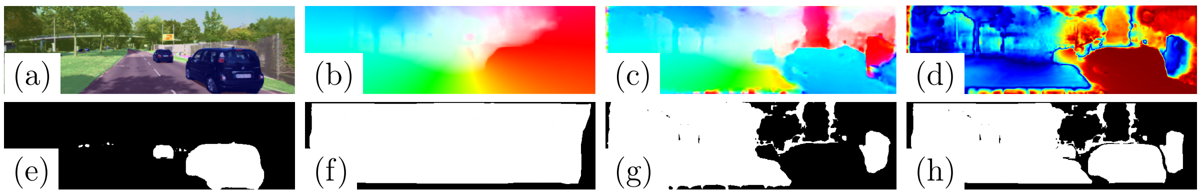


Figure 24.3: Overview of our semantic-aware and self-distilled optical flow estimation approach. We leverage semantic segmentation S_t (a) together with rigid flow $F^{rigid}_{t\rightarrow s}$ (b), teacher flow $F_{t\rightarrow s}$ (c) and motion probabilities P_t (d), the warmer the higher. From a) we obtain semantic priors M^d_t (e), combined with boundary mask M^b_t (f) and consistency mask M^c_t (g) derived from (d) as in Eq. 24.6, in order to obtain the final mask M (h) as in Eq. 24.7.

This mask assigns 1 where the condition is satisfied, 0 otherwise (i.e. inconsistent regions) as in Figure 24.3(g).

Finally, we compute the final mask M, in Figure 24.3(h), as:

$$M = \min\{\max\{M^d_t, M^c_t\}, M^b_t\} \tag{24.7}$$

As a consequence, M will effectively distinguish regions in the image for which we can not trust the supervision sourced by $F_{t\rightarrow s}$, i.e.inconsistent or occluded areas. On such regions, we can leverage our proposed self-distillation mechanism. Then, we define the final total loss for the self-distilled optical flow network as:

$$\mathcal{L} = \sum \alpha_r \phi(SF_{t\rightarrow s}, F^{rigid}_{t\rightarrow s}) \cdot (1 - M) + \alpha_d \phi(SF_{t\rightarrow s}, F_{t\rightarrow s}) \cdot M + \psi(I_t, \widetilde{I}^{SF}_t) \cdot M \tag{24.8}$$

where ϕ is a distance function between two motion vectors, while α_r and α_d are two hyper-parameters.

24.3 Experimental results

Using standard benchmark datasets, we present here the experimental validation on the main tasks tackled by ΩNet.

24.3.1 Monocular Depth Estimation

In this section, we compare our results to other state-of-the-art proposals and assess the contribution of each component to the quality of our monocular depth predictions.

Comparison with state-of-the-art. We compare with state-of-the-art self-supervised networks trained on monocular videos according to the protocol described in [60]. We follow the same pre-processing procedure as [293] to remove static images from the training split while using all the 697 images for testing. Since the predicted depth is defined up to a scale factor, we align the scale of our estimates by multiplying them by a scalar that matches the median of the ground truth, as introduced in [293]. We adopt the standard performance metrics defined in [60]. Table 24.1 reports extensive comparison with respect to several monocular depth estimation methods. We outperform our main competitors such as [16, 43, 270, 296] that solve multi-task learning or other strategies that exploit additional information during the training/testing phase [29, 261]. Moreover, our best configuration, i.e.pre-training on CS and using 1024×320 resolution, achieves better results in 5 out of 7 metrics with respect to the single-task, state-of-the-art proposal [70] (and is the second best and very close to it on the remaining 2) which, however, leverages on a larger ImageNet pre-trained model based on ResNet-18. It is also interesting to note how our proposal without pretraining obtains the best performance in 6 out of 7 measures on 640×192 images (row 1 vs 15). These results validate our intuition about how the use of semantic information can guide geometric reasoning and make a compact network provide state-of-the-art performance even with respect to larger and highly specialized depth-from-mono methods.

Ablation study. Table 24.2 highlights how progressively adding the key innova-

							Lower is better		Higher is better		
Method	M	A	I	CS	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Godard et al.[70]					0.132	1.044	5.142	0.210	0.845	0.948	0.977
Godard et al.[70] (1024 × 320)			✓		**0.115**	0.882	4.701	0.190	**0.879**	**0.961**	0.982
Zhou et al.[292]			✓		0.121	0.837	4.945	0.197	0.853	0.955	0.982
Mahjourian et al.[142]				✓	0.159	1.231	5.912	0.243	0.784	0.923	0.970
Wang et al.[249]				✓	0.151	1.257	5.583	0.228	0.810	0.936	0.974
Bian et al.[23]				✓	0.128	1.047	5.234	0.208	0.846	0.947	0.970
Yin et al.[270]	✓			✓	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Zou et al.[296]	✓			✓	0.146	1.182	5.215	0.213	0.818	0.943	0.978
Chen et al.[43]	✓		✓		0.135	1.070	5.230	0.210	0.841	0.948	0.980
Luo et al.[135]	✓				0.141	1.029	5.350	0.216	0.816	0.941	0.976
Ranjan et al.[16]	✓				0.139	1.032	5.199	0.213	0.827	0.943	0.977
Xu et al.[261]		✓	✓		0.138	1.016	5.352	0.217	0.823	0.943	0.976
Casser et al.[29]		✓			0.141	1.026	5.290	0.215	0.816	0.945	0.979
Gordon et al.[72]	✓	✓			0.128	0.959	5.230	-	-	-	-
Ω**Net**(640 × 192)	✓	✓			0.126	0.835	4.937	0.199	0.844	0.953	0.982
Ω**Net**(1024 × 320)	✓	✓			0.125	0.805	4.795	0.195	0.849	0.955	0.983
Ω**Net**(640 × 192)	✓	✓		✓	0.120	0.792	4.750	0.191	0.856	0.958	0.984
Ω**Net**(1024 × 320)	✓	✓		✓	0.118	**0.748**	**4.608**	**0.186**	0.865	**0.961**	**0.985**

Table 24.1: Depth evaluation on the Eigen split [60] of KITTI [67]. We indicate additional features of each method. M: multi-task learning, A: additional information (e.g.object knowledge, semantic information), I: feature extractors pre-trained on ImageNet [53], CS: network pre-trained on Cityscapes [47].

tions proposed in [70, 72, 249] contributes to strengthen ΩNet, already comparable to other methodologies even in its baseline configuration (first row). Interestingly, a large improvement is achieved by deploying joint depth and semantic learning (rows 5 vs 7), which forces the network to simultaneously reason about geometry and content within the same shared features. By replacing DSNet within ΩNet with a larger backbone [270] (rows 5 vs 6) we obtain worse performance, validating the design decisions behind our compact model. Finally, by pre-training on CS we achieve the best accuracy, which increases alongside with the input resolution (rows 8 to 10).

									Lower is better		Higher is better		
Resolution	Lear. Intr. [72]	Norm. [249]	Min. Repr. [70]	Autom. [70]	Sem. [39]	Pre-train	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
640 × 192	-	-	-	-	-	-	0.139	1.056	5.288	0.215	0.826	0.942	0.976
640 × 192	✓	-	-	-	-	-	0.138	1.014	5.213	0.213	0.829	0.943	0.977
640 × 192	✓	✓	-	-	-	-	0.136	1.008	5.204	0.212	0.832	0.944	0.976
640 × 192	✓	✓	✓	-	-	-	0.132	0.960	5.104	0.206	0.840	0.949	0.979
640 × 192	✓	✓	✓	✓	-	-	0.130	0.909	5.022	0.207	0.842	0.948	0.979
640 × 192 †	✓	✓	✓	✓	-	-	0.134	1.074	5.451	0.213	0.834	0.946	0.977
640 × 192	✓	✓	✓	✓	✓	-	0.126	0.835	4.937	0.199	0.844	0.953	0.980
416 × 128	✓	✓	✓	✓	✓	✓	0.126	0.862	4.963	0.199	0.846	0.952	0.981
640 × 192	✓	✓	✓	✓	✓	✓	0.120	0.792	4.750	0.191	0.856	0.958	0.984
1024 × 320	✓	✓	✓	✓	✓	✓	0.118	0.748	4.608	0.186	0.865	0.961	0.985

Table 24.2: Ablation study of our depth network on the Eigen split [60] of KITTI. †: our network is replaced by a ResNet50 backbone [270].

Method	Cap (m)	Abs Rel	Sq Rel	RMSE	RMSE log
Godard et al.[70]	0-8	**0.059**	**0.062**	0.503	**0.082**
ΩNet†	0-8	0.060	0.063	**0.502**	**0.082**
ΩNet	0-8	0.062	0.065	0.517	0.085
Godard et al.[70]	0-50	0.125	0.788	3.946	0.198
ΩNet†	0-50	0.127	0.762	4.020	0.199
ΩNet	0-50	**0.124**	**0.702**	**3.836**	**0.195**
Godard et al.[70]	0-80	0.132	1.044	5.142	0.210
ΩNet†	0-80	0.134	1.074	5.451	0.213
ΩNet	0-80	**0.126**	**0.835**	**4.937**	**0.199**

Table 24.3: Depth errors by varying the range. †: our network is replaced by a ResNet50 backbone [270].

Depth Range Error Analysis. We dig into our depth evaluation to explain the effectiveness of ΩNet with respect to much larger networks. Table 24.3 compares, at different depth ranges, our model with more complex ones [70, 270]. This experiment shows how ΩNet superior performance comes from better estimation of large depths: ΩNet outperforms both competitors when we include distances larger than 8 m in the evaluation, while it turns out less effective in the close range.

24.3.2 Semantic Segmentation

In Table 24.4, we report the performance of ΩNet on semantic segmentation for the 19 evaluation classes of CS according to the metrics defined in [17, 47]. We compare ΩNet against state-of-the art networks for real-time semantic segmentation [34, 127] when training on CS and testing either on the validation set of CS (rows 1-3) or the 200 semantically annotated images of K (rows 4-6). Even though our network is not as effective as the considered methods when training and testing on the same dataset, it shows greater generalization capabilities to unseen domains: it significantly outperforms other methods when testing on K for $\text{mIoU}_{\text{category}}$ and pixel accuracy, and provides similar results to [34] for $\text{mIoU}_{\text{class}}$. We relate this ability to our training protocol based on proxy labels (P) instead of ground truths (S). Moreover, as we have already effectively distilled the knowledge from DPC [39] during pre-training on CS, there is only a slight benefit in training on both CS and K (with proxy labels only) and testing on K (row 7). Finally,

Method	Train	Test	mIoU Class	mIoU Cat.	Pix.Acc.
DABNet [127]	CS(S)	CS	69.62	87.56	94.62
FCHardNet [34]	CS(S)	CS	**76.37**	**89.22**	**95.35**
Ω**Net**	CS(P)	CS	54.80	82.92	92.50
DABNet [127]	CS(S)	K	35.40	61.49	80.50
FCHardNet [34]	CS(S)	K	**44.74**	68.20	72.07
Ω**Net**	CS(P)	K	43.80	**74.31**	**88.31**
Ω**Net**	CS(P) + K(P)	K	46.68	75.84	88.12

Table 24.4: Semantic segmentation on Cityscapes (CS) and KITTI (K). S: training on ground truth, P: training on proxy labels.

although achieving 46.68 mIoU on fine segmentation, we obtain 89.64 mIoU for the task of segmenting static from potentially dynamic classes, an important result to obtain accurate motion masks.

24.3.3 Optical Flow

In Table 24.5, we compare the performance of our optical flow network with competing methods using the KITTI 2015 stereo/flow training set [67] as testing set, which contains 200 ground truth optical flow measurements for evaluation. We exploit all the raw K images for training, but we exclude the images used at testing time as done in [296] , to be consistent with experimental results of previous self-supervised optical flow strategies [16, 43, 270, 296]. From the table, we can observe how our self-distillation strategy allows SD-OFNet to outperform by a large margin competitors trained on K only (rows 5-11), and it even performs better than models pre-initialized by training on synthetic datasets [207]. Moreover, we submitted our flow predictions to the online KITTI flow benchmark after retraining the network including images from the whole official training set. In this configuration, we can observe how our model achieves state-of-the-art $F1$ performances with respect to other monocular multi-task architectures.

			train		test
Method	Dataset	Noc	All	F1	F1
Meisteret al.[154] - C	SYN + K	-	8.80	28.94%	29.46%
Meister et al.[154] - CSS	SYN + K	-	8.10	23.27%	23.30%
Zou et al.[296]	SYN + K	-	8.98	26.0%	25.70%
Ranjan et al.[16]	SYN + K	-	5.66	20.93%	25.27%
Wang et al.[253] **	K	-	5.58	-	18.00%
Yin et al.[270]	K	8.05	10.81	-	-
Chen et al.[43] †	K	5.40	8.95	-	-
Chen et al.[43] (online) †	K	4.86	8.35	-	-
Ranjan et al.[16]	K	-	6.21	26.41%	-
Luo et al.[135]	K	-	5.84	-	21.56%
Luo et al.[135] *	K	-	5.43	-	20.61%
ΩNet (Ego-motion)	K	11.72	13.50	51.22%	-
OFNet	K	3.48	11.61	25.78%	-
SD-OFNet	K	**3.29**	**5.39**	**20.0**%	**19.47%**

Table 24.5: Optical flow evaluation on the KITTI 2015 dataset. †: pre-trained on ImageNet, SYN: pre-trained on SYNTHIA [207], *: trained on stereo pairs, **: using stereo at testing time.

24.3.4 Motion Segmentation

In Table 24.6 we report experimental results for the motion segmentation task on the KITTI 2015 dataset. We compare our methodology with respect to other state-of-the-art strategies that performs multi-task learning and motion segmentation [16, 135, 253] using the metrics and evaluation protocol proposed in [135]. It can be noticed how our segmentation strategy outperforms all the other existing methodologies by a large margin. This demonstrates the effectiveness of our proposal to jointly combine semantic reasoning and motion probability to obtain much better results. We also report, as upper bound, the accuracy enabled by injecting semantic proxies [39] in place of ΩNet semantic predictions to highlight the low margin between the two.

24.3.5 Runtime analysis

Finally, we measure the runtime of ΩNet on different hardware devices, i.e.a Titan Xp GPU, an embedded NVIDIA Jetson TX2 board and an Intel i7-7700K@4.2 GHz CPU. Timings averaged over 200 frames at 640×192 resolution. Moreover, as each component of ΩNet may be used on its own, we report the runtime for each independent task. As

Method	Pixel Acc.	Mean Acc.	Mean IoU	f.w. IoU
Yang et al.[268] *	0.89	0.75	0.52	0.87
Luo et al.[135]	0.88	0.63	0.50	0.86
Luo et al.[135] *	0.91	0.76	0.53	0.87
Wang et al.[253] (Full) **	0.90	0.82	0.56	0.88
Ranjan et al.[16]	0.87	0.79	0.53	0.85
Ω**Net**	**0.98**	**0.86**	**0.75**	**0.97**
Ω**Net** (Proxy [39])	0.98	0.87	0.77	0.97

Table 24.6: Motion segmentation evaluation on the KITTI 2015 dataset. *: trained on stereo pairs, **: using stereo at testing time.

Device	Watt	D	DS	OF	Cam	Ω
Jetson TX2	15	12.5	10.3	6.5	49.2	4.5
i7-7700K	91	5.0	4.2	4.9	31.4	2.4
Titan XP	250	170.2	134.1	94.1	446.7	57.4

Table 24.7: Runtime analysis on different devices. We report the power consumption in Watt and the FPS. D: Depth, S: Semantic, OF: Optical Flow, Cam: camera pose, Ω: Overall architecture.

summarized in Table 24.7, our network runs in real-time on the Titan Xp GPU and at about 2.5 FPS on a standard CPU. It also fits the low-power NIVIDA Jetson TX2, achieving 4.5 FPS to compute all the outputs.

24.4 Conclusions

In this chapter, we have proposed the first real-time network for comprehensive scene understanding from monocular videos. Our framework reasons jointly about geometry, motion and semantics in order to estimate accurately depth, optical flow, semantic segmentation and motion masks at about 60 FPS on high-end GPU and 5FPS on embedded systems. To address the problem we have proposed a novel learning procedure based on distillation of proxy semantic labels and semantic-aware self-distillation of optical-flow information. Thanks to this paradigm, we have demonstrated state-of-the-art performance on standard datasets for depth and optical flow estimation as well as for motion segmentation.

Chapter 25

On the uncertainty of self-supervised monocular depth estimation

The content of this chapter has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2020) - "On the uncertainty of self-supervised monocular depth estimation" [190].

25.1 Introduction

As for other perception strategies, it is essential to find out failure cases, when occurring, in monocular depth estimation networks. For instance, in an autonomous driving scenario, the erroneous perception of the distance to pedestrians or other vehicles might have dramatic consequences. Moreover, the ill-posed nature of depth-from-mono perception task makes this eventuality much more likely to occur compared to techniques leveraging scene geometry [211, 216]. In these latter cases, estimating the *uncertainty* (or, complementary, the *confidence*) proved to be effective for depth-from-stereo, by means of both model-based [89] and learning-based [111, 184] methods, optical flow [98], and semantic segmentation [93, 111]. Despite the steady progress in other related fields, uncertainty estimation for self-supervised paradigms remains almost unexplored or, when faced, not quantitatively

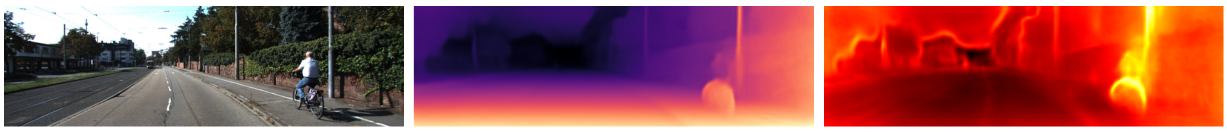


Figure 25.1: How much can we trust self-supervised monocular depth estimation? From a single input image (top) we estimate depth (middle) and uncertainty (bottom) maps. Best with colors.

evaluated [119].

Whereas concurrent works in this field [70, 241, 255] targeted uniquely depth accuracy, we take a breath on this rush and focus for the first time, to the best of our knowledge, on uncertainty estimation for self-supervised monocular depth estimation networks, showing how this practise enables to improve depth accuracy as well.

Our main contributions can be summarized as follows:

- A comprehensive evaluation of uncertainty estimation approaches tailored for the considered task.
- An in-depth investigation of how the self-supervised training paradigm deployed impacts uncertainty and depth estimation.
- A new and peculiar *Self-Teaching* paradigm to model uncertainty, particularly useful when the pose is unknown during the training process, always enabling to improve depth accuracy.

Deploying standard metrics in this field, we provide exhaustive experimental results on the KITTI dataset [67]. Figure 25.1 shows the output of a state-of-the-art monocular depth estimator network enriched to model uncertainty. We can notice how our proposal effectively allows to detect wrong predictions (e.g., in the proximity of the person riding the bike).

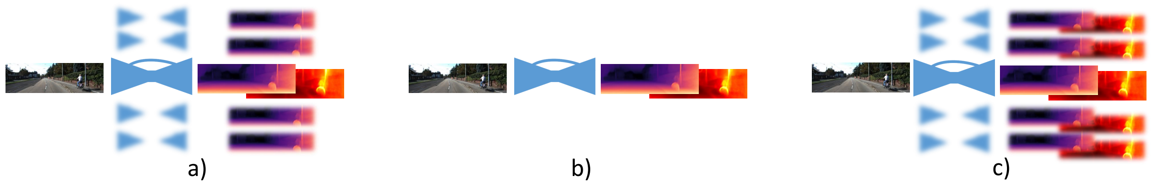


Figure 25.2: Overview of uncertainty estimation implementations. Respectively a) empirical methods model uncertainty as the variance of predictions from a subset of all the possible instances of the same network, b) predictive are trained to estimate depth and uncertainty as mean and variance of a distribution and c) Bayesian methods are approximated [164] by sampling multiple predictive models and summing single uncertainties with the variance of the depth predictions.

25.2 Depth-from-mono and uncertainty

In this section, we introduce how to tackle uncertainty modelling with self-supervised depth estimation frameworks. Given a still image $\mathcal{I}$ any depth-from-mono framework produces an output map d encoding the depth of the observed scene. When full supervision is available, to train such a network we aim at minimizing a loss signal $\mathcal{L}_{fs}$ obtained through a generic function $\mathcal{F}$ of inputs estimated d and ground truth d^* depth maps.

$$\mathcal{L}_{fs} = \mathcal{F}(d, d^*) \tag{25.1}$$

When traditional supervision is not available, it can be replaced by self-supervision obtained through image reconstruction. In this case, the ground truth map d^* is replaced by a second image $\mathcal{I}^\dagger$. Then, by knowing camera intrinsics K, $K^\dagger$ and the relative camera pose $(R|t)$ between the two images, a reconstructed image $\tilde{\mathcal{I}}$ is obtained as a function π of intrinsics, pose, image $\mathcal{I}^\dagger$ and depth d, enabling to compute a loss signal $\mathcal{L}_{ss}$ as a generic $\mathcal{F}$ of inputs $\tilde{\mathcal{I}}$ and $\mathcal{I}$.

$$\mathcal{L}_{ss} = \mathcal{F}(\tilde{\mathcal{I}}, \mathcal{I}) = \mathcal{F}(\pi(\mathcal{I}^\dagger, K^\dagger, R|t, K, d), \mathcal{I}) \tag{25.2}$$

$\mathcal{I}$ and $\mathcal{I}^\dagger$ can be acquired either by means of a single moving camera or with a stereo rig. In this latter case, $(R|t)$ is known beforehand thanks to the stereo calibration parameters, while for images acquired by a single camera it is usually learned jointly to depth, both up to a scale factor. A popular choice for $\mathcal{F}$ is a weighted sum between L1 and Structured Similarity Index Measure (SSIM) [254]

$$\mathcal{F}(\tilde{\mathcal{I}},\mathcal{I}) = \alpha \cdot \frac{1 - \text{SSIM}(\tilde{\mathcal{I}},\mathcal{I})}{2} + (1-\alpha) \cdot |\tilde{\mathcal{I}} - \mathcal{I}| \tag{25.3}$$

with α commonly set to 0.85 [70]. In case of K frames used for supervision, coming for example by joint monocular and stereo supervision, for each pixel q the minimum among computed losses allows for robust reprojection [70]

$$\mathcal{L}_{ss}(q) = \min_{i \in [0..K]} \mathcal{F}(\tilde{\mathcal{I}}_i(q), \mathcal{I}(q)) \tag{25.4}$$

Traditional networks are deterministic, producing a single output typically corresponding to the mean value of the distribution of all possible outputs $p(d^*|\mathcal{I},\mathcal{D})$, $\mathcal{D}$ being a dataset of images and corresponding depth maps. Estimating the variance of such distribution allows for modelling uncertainty on the network outputs, as shown in [97, 108] and depicted in Figure 25.2, a) in empirical way, b) by learning a predictive model or c) combining the two approaches.

First and foremost, we point out that the self-supervision provided to the network is *indirect* with respect to its main task. This means that the network estimates are not optimized with respect to the desired statistical distribution, i.e.depth d^*, but they are an input parameter of a function (π) optimized over a different statistical model, i.e.image $\mathcal{I}$. While this does not represent an issue for empirical methods, predictive methods like negative log-likelihood minimization can be adapted to this paradigm as done by Klodt and Vedaldi [119]. Nevertheless, we will show how this solution is sub-optimal when the pose is unknown, i.e.when π is function of two unknown parameters.

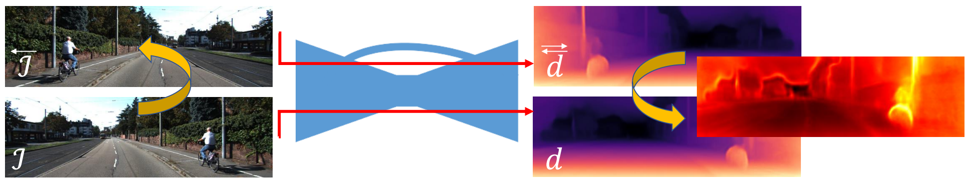

$\overleftarrow{\mathcal{I}}$
$\mathcal{I}$
$\overset{\leftrightarrows}{d}$
d

Figure 25.3: Uncertainty by image flipping. The difference between the depth d, inferred from image $\mathcal{I}$, and the depth $\overset{\leftrightarrows}{d}$, from the flipped image $\overleftarrow{\mathcal{I}}$, provides a basic form of uncertainty.

25.2.1 Uncertainty by image flipping

A simple strategy to estimate uncertainty is inspired by the post-processing (*Post*) step proposed by Godard et al.[69]. Such a refinement consists of estimating two depth maps d and $\overleftarrow{d}$ for image $\mathcal{I}$ and its horizontally flipped counterpart $\overleftarrow{\mathcal{I}}$. The refined depth map d^r is obtained by averaging d and $\overset{\leftrightarrows}{d}$, i.e.back-flipped $\overleftarrow{d}$. We encode the uncertainty for d^r as the difference between the two

$$u_{Post} = |d - \overset{\leftrightarrows}{d}| \tag{25.5}$$

i.e., the variance over a small distribution of outputs (i.e., two), as typically done for empirical methods outlined in the next section. Although this method requires $2\times$ forwards at test time compared to the raw depth-from-mono model, as shown in Figure 25.3, it can be applied seamlessly to any pre-trained framework without any modification.

25.2.2 Empirical estimation

This class of methods aims at encoding uncertainty empirically, for instance, by measuring the variance between a set of all the possible network configurations. It allows to explain the model uncertainty, namely *epistemic* [108]. Strategies belonging to this category [98] can be applied to self-supervised frameworks straightforwardly.

Dropout Sampling (*Drop*). Early works estimated uncertainty in neural networks [140] by sampling multiple networks from the distribution of weights of a single architecture. Monte Carlo Dropout [229] represents a popular method to sample N independent models without requiring multiple and independent trainings. At training time, connections between layers are randomly dropped with a probability p to avoid overfitting. At test time, all connections are kept. By keeping dropout enabled at test time, we can perform multiple forwards sampling a different network every time. Empirical mean $\mu(d)$ and variance $\sigma^2(d)$ are computed, as follows, performing multiple (N) inferences:

$$\mu(d) = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{25.6}$$

$$u_{Drop} = \sigma^2(d) = \frac{1}{N}\sum_{i=1}^{N}(d_i - \mu(d))^2 \tag{25.7}$$

At test time, using the same number of network parameters, N× forwards are required.

Bootstrapped Ensemble (*Boot*). A simple, yet effective alternative to weights sampling is represented by training an ensemble of N neural networks [124] randomly initializing N instances of the same architecture and training them with bootstrapping, i.e.on random subsets of the entire training set. This strategy produces N specialized models. Then, similarly to dropout sampling, we can obtain empirical mean $\mu(d)$ and variance $\sigma^2(d)$ in order to approximate the mean and variance of the distribution of depth values. It requires N× parameters to be stored, results on N× independent trainings, and a single forward pass for each stored configuration at test time.

Snapshot Ensemble (*Snap*). Although the previous method is compelling, obtaining ensembles of neural networks is expensive since it requires carrying out N independent training. An alternative solution [90] consists of obtaining N snapshots out of a single training by leveraging on cyclic learning rate schedules to obtain C pre-converged models. Assuming an initial learning rate λ_0, we obtain λ_t at any training iteration t as a function of the total number of steps T and cycles C as in [90]

$$\lambda_t = \frac{\lambda_0}{2} \cdot \left(\cos \left(\frac{\pi \cdot \mod (t-1, \lceil \frac{T}{C} \rceil)}{\lceil \frac{T}{C} \rceil} \right) + 1 \right) \tag{25.8}$$

Similarly to *Boot* and *Drop*, we obtain empirical mean $\mu(d)$ and variance $\sigma^2(d)$ by choosing N out of the C models obtained from a single training procedure.

25.2.3 Predictive estimation

This category aims at encoding uncertainty by learning a predictive model. This means that at test time these methods produce estimates that are function of network parameters and the input image and thus reason about the current observations, modelling *aleatoric heteroscedastic* uncertainty [108]. Since often learned from real data distribution, for instance as a function of the distance between the predictions and the ground truth or by maximizing log-likelihood, these approaches need to be rethought to deal with self-supervised paradigms.

Learned Reprojection (*Repr*). To learn a function over the prediction error employing a classifier is a popular technique used for both stereo [184, 219] and optical flow [139]. However, given the absence of ground truth labels, we cannot apply this approach to self-supervised frameworks seamlessly. Nevertheless, we can drive one output of our network to mimic the behavior of the self-supervised loss function used to train it, thus learning ambiguities affecting the paradigm itself (e.g., occlusions, low texture and more). Indeed, the per-pixel loss signal is supposed to be high when the estimated depth is wrong. Thus, uncertainty u_{Repr} is trained adding the following term to $\mathcal{L}_{ss}$

$$\mathcal{L}_{Repr} = \beta \cdot |u_{Repr} - \mathcal{F}(\tilde{\mathcal{I}}, \mathcal{I})| \tag{25.9}$$

Since multiple images $\mathcal{I}^\dagger$ may be used for supervision, i.e.when combining monocular and stereo, usually for each pixel q the minimum reprojection signal is considered to train the network, thus u_{Repr} is trained accordingly

$$\mathcal{L}_{Repr}(q) = \beta \cdot |u_{Repr}(q) - \min_{i \in [0..K]} \mathcal{F}(\tilde{\mathcal{I}}_i(q), \mathcal{I}(q))| \tag{25.10}$$

In our experiments, we set β to 0.1 and stop $\mathcal{F}$ gradients inside $\mathcal{L}_{Repr}$ for numerical stability. A similar technique appeared in [36], although not evaluated quantitatively.

Log-Likelihood Maximization (*Log*). Another popular strategy [165] consists of training the network to infer mean and variance of the distribution $p(d^*|\mathcal{I}, \mathcal{D})$ of parameters Θ. The network is trained by log-likelihood maximization (i.e., negative log-likelihood minimization)

$$\log p(d^*|w) = \frac{1}{N} \sum_q \log p(d^*(q)|\Theta(\mathcal{I}, w)) \tag{25.11}$$

w being the network weights. As shown in [98], the predictive distribution can be modelled as Laplacian or Gaussian respectively in case of L1 or L2 loss computation with respect to d^*. In the former case, this means minimizing the following loss function

$$\mathcal{L}_{Log} = \frac{|\mu(d) - d^*|}{\sigma(d)} + \log \sigma(d) \tag{25.12}$$

with $\mu(d)$ and $\sigma(d)$ outputs of the network encoding mean and variance of the distribution. The additional logarithmic term discourages infinite predictions for any pixel. Regarding numerical stability [108], the network is trained to estimate the log-variance in order to avoid zero values of the variance. As shown by Klodt and Vedaldi [119], in absence of ground truth d^* one can model the uncertainty u_{Log} according to photometric matching

$$\mathcal{L}_{Log} = \frac{\min_{i \in [0..K]} \mathcal{F}(\tilde{\mathcal{I}}_i(q), \mathcal{I}(q))}{u_{Log}} + \log u_{Log} \tag{25.13}$$

Recall that $\mathcal{F}$ is computed over π according to Equation 25.2. Although for stereo supervision this formulation is equivalent to traditional supervision, i.e.π is function of a single unknown parameter d, in case of monocular supervision this formulation jointly explain

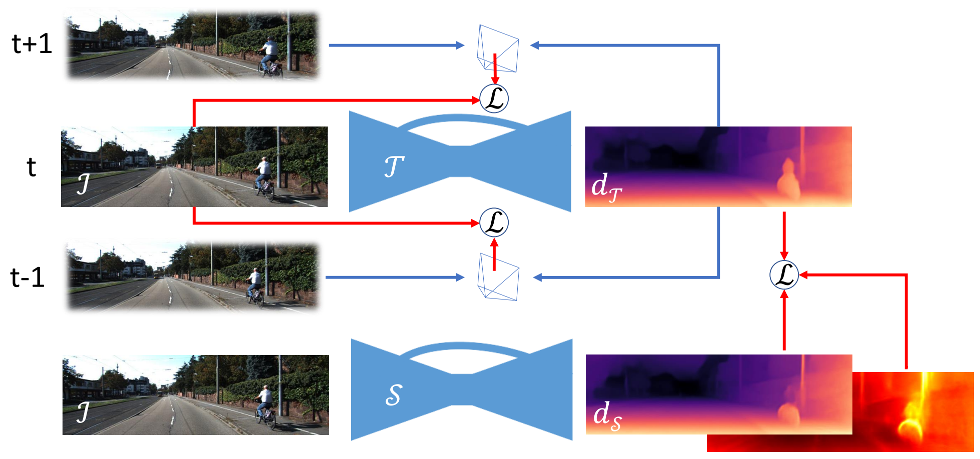


Figure 25.4: Self-Teaching scheme. A network $\mathcal{T}$ is trained in self-supervised fashion, e.g.on monocular sequences $[t-1, t, t+1]$. A new instance $\mathcal{S}$ of the same is trained on $d_{\mathcal{T}}$ output of $\mathcal{T}$.

uncertainty for depth and pose, both unknown variables in π. We will show how this approach leads to sub-optimal modelling and how to overcome this limitation with the next approach.

Self-Teaching (*Self*). In order to decouple depth and pose when modelling uncertainty, we propose to source a direct form of supervision from the learned model itself. By training a first network in a self-supervised manner, we obtain a network instance $\mathcal{T}$ producing a noisy distribution $d_{\mathcal{T}}$. Then, we train a second instance of the same model, namely $\mathcal{S}$, to mimic the distribution sourced from $\mathcal{T}$. Typically, teacher-student frameworks [288] applied to monocular depth estimation [177] deploy a complex architecture to supervise a more compact one. In contrast, in our approach the teacher $\mathcal{T}$ and the student $\mathcal{S}$ share the same architecture and for this reason we refer to it as Self-Teaching (*Self*). By assuming an L1 loss, we can model for instance negative log-likelihood minimization as

$$\mathcal{L}_{Self} = \frac{|\mu(d_{\mathcal{S}}) - d_{\mathcal{T}}|}{\sigma(d_{\mathcal{S}})} + \log \sigma(d_{\mathcal{S}}) \tag{25.14}$$

We will show how with this strategy i) we obtain a network $\mathcal{S}$ more accurate than $\mathcal{T}$ and ii) in case of monocular supervision, we can decouple depth from pose and achieve a much more effective uncertainty estimation. Figure 25.4 summarizes our proposal.

25.2.4 Bayesian estimation

Finally, in Bayesian deep learning [108], the model uncertainty can be explained by marginalizing over all possible w rather than choosing a point estimate. According to Neal [164], an approximate solution can be obtained by sampling N models and by modelling mean and variance as

$$p(d^*|\mathcal{I},\mathcal{D}) \approx \sum_{i=1}^{N} p(d^*|\Theta(\mathcal{I},w_i)) \tag{25.15}$$

If mean and variance are modelled for each w_i sampling, we can obtain overall mean and variance as reported in [98, 108]

$$\mu(d) = \frac{1}{N}\sum_{i=1}^{N} \mu_i(d_i) \tag{25.16}$$

$$\sigma^2(d) = \frac{1}{N}\sum_{i=1}^{N} (\mu_i(d_i) - \mu(d))^2 + \sigma_i^2(d_i) \tag{25.17}$$

The implementation of this approximation is straightforward by combining empirical and predictive methods [98, 108]. Purposely, in our experiments we will pick the best empirical and predictive methods, e.g.combining *Boot* and *Self* (*Boot+Self*).

25.3 Experimental results

In this section, we exhaustively evaluate self-supervised strategies for joint depth and uncertainty estimation.

25.3.1 Evaluation protocol, dataset and metrics

At first, we describe all details concerning training and evaluation to ensure full reproducibility.

Architecture and training schedule. We choose as baseline model Monodepth2 [70], thanks to the code made available and to its possibility to be trained seamlessly according to monocular, stereo, or both self-supervision paradigms. In our experiments, we train any variant of this method following the protocol defined in [70], on batches of 12 images resized to 192×640 for 20 epochs starting from pre-trained encoders on ImageNet [53]. Moreover, we always follow the augmentation and training practices described in [70]. Finally, to evaluate *Post* we use the same weights made publicly available by the authors. Regarding empirical methods, we set N to 8 and the number of cycles C for *Snap* to 20. We randomly extract 25% of the training set for each independent network in *Boot*. Dropout is applied after convolutions in the decoder only. About predictive models, a single output channel is added in parallel to depth prediction channel.

Dataset. We compare all the models on the KITTI dataset [67]. Following standards in the field, we deploy the Eigen split [60] and set 80 meters as the maximum depth. For this purpose, we use the improved ground truth introduced in [246], much more accurate than the raw LiDAR data, since our aim is a strict evaluation rather than a comparison with existing monocular methods.

Depth metrics. To assess depth accuracy, we report for the sake of page limit three out of seven standard metrics defined in [60]. Moreover, we also report the number of training iterations (#Trn), parameters (#Par), and forwards (#Fwd) required at testing time to estimate depth. In the case of monocular supervision, we scale depth as in [293].

Uncertainty metrics. To evaluate how significant the modelled uncertainties are, we use sparsification plots as in [98]. Given an error metric ϵ, we sort all pixels in each depth map in order of descending uncertainty. Then, we iteratively extract a subset of pixels (i.e., 2% in our experiments) and compute ϵ on the remaining to plot a curve, that

Method	Sup	#Trn	#Par	#Fwd	Abs Rel	RMSE	δ <1.25
Monodepth2 [70]	M	1×	1×	1×	0.090	3.942	0.914
Monodepth2-*Post* [70]	M	1×	1×	2×	0.088	3.841	0.917
Monodepth2-*Drop*	M	1×	1×	N×	0.101	4.146	0.892
Monodepth2-*Boot*	M	N×	N×	1×	0.092	3.821	0.911
Monodepth2-*Snap*	M	1×	N×	1×	0.091	3.921	0.912
Monodepth2-*Repr*	M	1×	1×	1×	0.092	3.936	0.912
Monodepth2-*Log*	M	1×	1×	1×	0.091	4.052	0.910
Monodepth2-*Self*	M	(1+1)×	1×	1×	**0.087**	3.826	**0.920**
Monodepth2-*Boot+Log*	M	N×	N×	1×	0.092	3.850	0.910
Monodepth2-*Boot+Self*	M	(1+N)×	N×	1×	0.088	**3.799**	0.918
Monodepth2-*Snap+Log*	M	1×	1×	1×	0.092	3.961	0.911
Monodepth2-*Snap+Self*	M	(1+1)×	1×	1×	0.088	3.832	0.919

a) Depth evaluation

	Abs Rel		RMSE		$\delta \geq 1.25$	
Method	AUSE	AURG	AUSE	AURG	AUSE	AURG
Monodepth2-*Post*	0.044	0.012	2.864	0.412	0.056	0.022
Monodepth2-*Drop*	0.065	0.000	2.568	0.944	0.097	0.002
Monodepth2-*Boot*	0.058	0.001	3.982	-0.743	0.084	-0.001
Monodepth2-*Snap*	0.059	-0.001	3.979	-0.639	0.083	-0.002
Monodepth2-*Repr*	0.051	0.008	2.972	0.381	0.069	0.013
Monodepth2-*Log*	0.039	0.020	2.562	0.916	0.044	0.038
Monodepth2-*Self*	0.030	0.026	2.009	1.266	0.030	0.045
Monodepth2-*Boot+Log*	0.038	0.021	2.449	0.820	0.046	0.037
Monodepth2-*Boot+Self*	**0.029**	**0.028**	**1.924**	**1.316**	**0.028**	**0.049**
Monodepth2-*Snap+Log*	0.038	0.022	2.385	1.001	0.043	0.039
Monodepth2-*Snap+Self*	0.031	0.026	2.043	1.230	0.030	0.045

b) Uncertainty evaluation

Table 25.1: Quantitative results for monocular (M) supervision. Evaluation on Eigen split [60] with improved ground truth [246].

is supposed to shrink if the uncertainty properly encodes the errors in the depth map. An ideal sparsification (*oracle*) is obtained by sorting pixels in descending order of the ϵ magnitude. In contrast, a random uncertainty can be modelled as a constant, giving no information about how to remove erroneous measurements and, thus, a flat curve. By plotting the difference between estimated and oracle sparsification, we can measure the Area Under the Sparsification Error (AUSE, the lower the better). Subtracting estimated sparsification from random one enables computing the Area Under the Random Gain (AURG, the higher the better). The former quantifies how close the estimate is to the oracle uncertainty, the latter how better (or worse, as we will see in some cases) it is compared to no modelling at all. We assume Abs Rel, RMSE or $\delta \geq 1.25$ (since $\delta < 1.25$ defines an accuracy score) as ϵ.

25.3.2 Monocular (M) supervision

Depth. Table 25.1a reports depth accuracy for Monodepth2 variants implementing the different uncertainty estimation strategies when trained with monocular supervision. We can notice how, in general, empirical methods fail at improving depth prediction on most metrics, with *Drop* having a large gap from the baseline. On the other hand, *Boot* and *Snap* slightly reduce RMSE. Predictive methods as well produce worse depth estimates, except the proposed *Self* method, which improves all the metrics compared to the baseline, even when post-processed. Regarding the Bayesian solutions, both *Boot* and *Snap* performs worse when combined with *Log*, while they are always improved by the proposed *Self* method.

Uncertainty. Table 25.1b resumes performance of modelled uncertainties at reducing errors on the estimated depth maps. Surprisingly, empirical methods rarely perform better than the *Post* solution. In particular, empirical methods alone fail at performing better than a random chance, except for *Drop* that, on the other hand, produces much worse depth maps. Predictive methods perform better, with *Log* and *Self* yielding the best results. Among them, our method outperforms *Log* by a notable margin. Combining empirical and predictive methods is beneficial, often improving over single choices. In particular, *Boot+Self* achieves the best overall results.

Summary. In general *Self*, combined with empirical methods, performs better for both depth accuracy and uncertainty modelling when dealing with M supervision, thanks to disentanglement between depth and pose. We believe that empirical methods performance can be ascribed to depth scale, being unknown during training.

25.3.3 Stereo (S) supervision

Depth. On Table 25.2a we show the results of the same approaches when trained with stereo supervision. Again, *Drop* fails to improve depth accuracy, together with *Repr* among predictive methods. *Boot* produces the best improvement, in particular in terms

Method	Sup	#Trn	#Par	#Fwd	Abs Rel	RMSE	δ <1.25
Monodepth2 [70]	S	1×	1×	1×	0.085	3.942	0.912
Monodepth2-*Post* [70]	S	1×	1×	2×	0.084	3.777	**0.915**
Monodepth2-*Drop*	S	1×	1×	N×	0.129	4.908	0.819
Monodepth2-*Boot*	S	N×	N×	1×	0.085	**3.772**	0.914
Monodepth2-*Snap*	S	1×	N×	1×	0.085	3.849	0.912
Monodepth2-*Repr*	S	1×	1×	1×	0.085	3.873	0.913
Monodepth2-*Log*	S	1×	1×	1×	0.085	3.860	**0.915**
Monodepth2-*Self*	S	(1+1)×	1×	1×	0.084	3.835	**0.915**
Monodepth2-*Boot+Log*	S	N×	N×	1×	0.085	3.777	0.913
Monodepth2-*Boot+Self*	S	(1+N)×	N×	1×	0.085	3.793	0.914
Monodepth2-*Snap+Log*	S	1×	1×	1×	**0.083**	3.833	0.914
Monodepth2-*Snap+Self*	S	(1+1)×	1×	1×	0.086	3.859	0.912

a) Depth evaluation

	Abs Rel		RMSE		$\delta \geq 1.25$	
Method	AUSE	AURG	AUSE	AURG	AUSE	AURG
Monodepth2-*Post*	0.036	0.020	2.523	0.736	0.044	0.034
Monodepth2-*Drop*	0.103	-0.029	6.163	-2.169	0.231	-0.080
Monodepth2-*Boot*	0.028	0.029	2.291	0.964	0.031	0.048
Monodepth2-*Snap*	0.028	0.029	2.252	1.077	0.030	0.051
Monodepth2-*Repr*	0.040	0.017	2.275	1.074	0.050	0.030
Monodepth2-*Log*	0.022	0.036	0.938	2.402	**0.018**	0.061
Monodepth2-*Self*	0.022	0.035	1.679	1.642	0.022	0.056
Monodepth2-*Boot+Log*	**0.020**	**0.038**	**0.807**	**2.455**	**0.018**	**0.063**
Monodepth2-*Boot+Self*	0.023	0.035	1.646	1.628	0.021	0.058
Monodepth2-*Snap+Log*	0.021	0.037	0.891	2.426	**0.018**	0.061
Monodepth2-*Snap+Self*	0.023	0.035	1.710	1.623	0.023	0.058

b) Uncertainty evaluation

Table 25.2: Quantitative results for stereo (S) supervision. Evaluation on Eigen split [60] with improved ground truth [246].

of RMSE. Traditional *Log* improves this time over the baseline, according to RMSE and $\delta < 1.25$ metrics while, *Self* consistently improves the baseline on all metrics, although it does not outperform *Post*, which requires two forward passes.

Uncertainty. Table 25.2b summarizes the effectiveness of modelled uncertainties. This time, only *Drop* performs worse than *Post* achieving negative AURG, thus being detrimental at sparsification, while other empirical methods achieve much better results. In these experiments, thanks to the known pose of the stereo setup, *Log* deals only with depth uncertainty and thus performs extremely well. *Self*, although allowing for more accurate depth as reported in Table 25.2a, ranks second this time. Considering Bayesian implementations, again, both *Boot* and *Snap* are always improved. Conversely, compared to the M case, *Log* this time consistently outperforms *Self* in any Bayesian formulation.

Summary. When the pose is known, the gap between *Log* and *Self* concerning depth accuracy is minor, with *Self* performing better when modelling only predictive uncertainty and *Log* slightly better with Bayesian formulations. For uncertainty estimation, *Log* consistently performs better. The behavior of empirical methods alone confirms our findings from the previous experiments: by knowing the scale, *Boot* and *Snap* model uncertainty much better. In contrast, *Drop* fails for this purpose.

25.3.4 Monocular+Stereo (MS) supervision

Depth. Table 25.3a reports the behavior of depth accuracy when monocular and stereo supervisions are combined. In this case, only *Self* consistently outperforms the baseline and is competitive with *Post*, which still requires two forward passes. Among empirical methods, *Boot* is the most effective. Regarding Bayesian solutions, those using *Self* are, in general, more accurate on most metrics, yet surprisingly worse than *Self* alone.

Uncertainty. Table 25.3b shows the performance of the considered uncertainties. The behavior of all variants is similar to the one observed with stereo supervision, except for *Log* and *Self*. We can notice that *Self* outperforms *Log*, similarly to what observed with M supervision. It confirms that pose estimation drives *Log* to worse uncertainty estimation, while *Self* models are much better thanks to the training on proxy labels produced by the Teacher network. Concerning Bayesian solutions, in general, *Boot* and *Snap* are improved when combined with both *Log* and *Self*, with *Self* combinations typically better than their *Log* counterparts and equivalent to standalone *Self*.

Summary. The evaluation with monocular and stereo supervision confirms that when the pose is estimated alongside with depth, *Self* proves to be a better solution compared to *Log* and, in general, other approaches to model uncertainty. Finally, empirical methods alone behave as for experiments with stereo supervision, confirming that the knowledge of the scale during training is crucial to the proper behavior of *Drop*, *Boot* and *Snap*.

Method	Sup	#Trn	#Par	#Fwd	Abs Rel	RMSE	δ <1.25
Monodepth2 [70]	MS	1×	1×	1×	0.084	3.739	0.918
Monodepth2-*Post* [70]	MS	1×	1×	2×	**0.082**	**3.666**	**0.919**
Monodepth2-*Drop*	MS	1×	1×	N×	0.172	5.885	0.679
Monodepth2-*Boot*	MS	N×	N×	1×	0.086	3.787	0.910
Monodepth2-*Snap*	MS	1×	N×	1×	0.085	3.806	0.914
Monodepth2-*Repr*	MS	1×	1×	1×	0.084	3.828	0.913
Monodepth2-*Log*	MS	1×	1×	1×	0.083	3.790	0.916
Monodepth2-*Self*	MS	(1+1)×	1×	1×	0.083	3.682	**0.919**
Monodepth2-*Boot+Log*	MS	N×	N×	1×	0.086	3.771	0.911
Monodepth2-*Boot+Self*	MS	(1+N)×	N×	1×	0.085	3.704	0.915
Monodepth2-*Snap+Log*	MS	1×	1×	1×	0.084	3.828	0.914
Monodepth2-*Snap+Self*	MS	(1+1)×	1×	1×	0.085	3.715	0.916

a) Depth evaluation

	Abs Rel		RMSE		$\delta \geq 1.25$	
Method	AUSE	AURG	AUSE	AURG	AUSE	AURG
Monodepth2-*Post*	0.036	0.018	2.498	0.655	0.044	0.031
Monodepth2-*Drop*	0.103	-0.027	7.114	-2.580	0.303	-0.081
Monodepth2-*Boot*	0.028	0.030	2.269	0.985	0.034	0.049
Monodepth2-*Snap*	0.029	0.028	2.245	1.029	0.033	0.047
Monodepth2-*Repr*	0.046	0.010	2.662	0.635	0.062	0.018
Monodepth2-*Log*	0.028	0.029	1.714	**1.562**	0.028	0.050
Monodepth2-*Self*	**0.022**	0.033	**1.654**	1.515	**0.023**	0.052
Monodepth2-*Boot+Log*	0.030	0.028	1.962	1.282	0.032	0.051
Monodepth2-*Boot+Self*	0.023	0.033	1.688	1.494	**0.023**	**0.056**
Monodepth2-*Snap+Log*	0.030	0.027	2.032	1.272	0.032	0.048
Monodepth2-*Snap+Self*	0.023	**0.034**	1.684	1.510	**0.023**	0.055

b) Uncertainty evaluation

Table 25.3: Quantitative results for monocular+stereo (MS) supervision. Evaluation on Eigen split [60] with improved ground truth [246].

25.3.5 Sparsification curves

In order to further outline our findings, we report in Figure 25.5 the RMSE sparsification error curves, averaged over the test set, when training with M, S or MS supervision. The plots show that methods leveraging on *Self* (blue) are the best to model uncertainty when dealing with pose estimation, i.e.M and MS, while those using *Log* (green) are better when training on S.

25.4 Conclusion

In this chapter, we have thoroughly investigated for the first time in literature uncertainty modelling in self-supervised monocular depth estimation. We have reviewed and evaluated existing techniques, as well as introduced a novel Self-Teaching (*Self*) paradigm. We

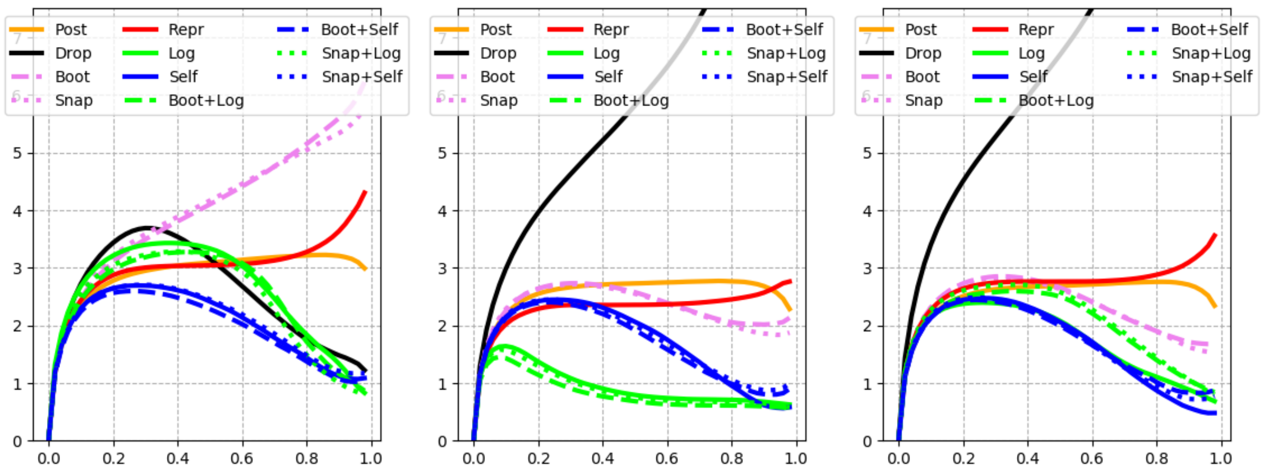


Figure 25.5: Sparsification Error curves. From left to right, average RMSE with M, S and MS supervisions. Best viewed with colors.

have considered up to 11 strategies to estimate the uncertainty on predictions of a depth-from-mono network trained in a self-supervised manner. Our experiments highlight how different supervision strategies lead to different winners among the considered methods. In particular, among empirical methods, only Dropout sampling performs well when the scale is unknown during training (M), while it is the only one failing when scale is known (S, MS). Empirical methods are affected by pose estimation, for which log-likelihood maximization gives sub-optimal results when the pose is unknown (M, MS). In these latter cases, potentially the most appealing for practical applications, the proposed *Self* technique results in the best strategy to model uncertainty. Moreover, uncertainty estimation also improves depth accuracy consistently, with any training paradigm.

Conclusion

25.5 Summary of Thesis Achievements

In this thesis, novel techniques leveraging on deep learning have been proposed in order to improve several aspects strictly related to the 3D reconstruction task from a single or multiple images.

The work was carried out by deepening the main issues in which it is possible to incur into a matching problem and, therefore, contribute to make the three-dimensional reconstruction challenging. Attention was initially paid to the investigation of methodologies able to detect the aforementioned issues through an appropriate confidence estimation procedure. After having grouped several confidence measures into specific categories, the results obtained from the ROC curves analysis show how approaches based on machine learning guarantee a better capability to identify erroneous depth assignments than other existing traditional techniques. After having deepened and analyzed a comprehensive set of confidence measures and proposed alternative solutions based on both supervised and self-supervised paradigms, the activity has evolved through the proposal of novel deep stereo matching strategies that mostly leverage on such measures. More specifically, it has been shown how few, yet accurate, sparse depth measurements are pivotal to optimize wrong 3D estimates and an essential information for deep self-supervised stereo algorithms which do not require expensive ground truth labels sourced by active sensors.

Subsequently, an in-depth study on an extremely challenging and exciting research topic such as the monocular depth estimation task has been conducted. In particular, we investigated different directions and designed several self-supervised state-of-the-art machine learning frameworks that are efficient, accurate and able to exploit additional information such as optical flow and semantic segmentation, useful for the overall 3D reconstruction improvement.

25.6 Future Work

Future research directions will include the study of other applications leveraging on depth and confidence estimation. In particular, a research area of great interest could be a tight integration of such information with multi-view stereo methodologies from both active and passive sensors. Moreover, since it is widely recognized that passive techniques are unreliable for long-range depth sensing, a key factor for navigation at even moderate speeds, higher-resolution appear to be the most promising avenue. In addition, it would be interesting to explore novel strategies that can take advantage of working on the multi-view image domain as well as on the three-dimensional space, resulting in end-to-end systems capable to directly optimize and reconstruct large scale scenes in any environments.

Bibliography

[1] Intel Real Sense camera. `https://realsense.intel.com/`.

[2] Linaro - leading software collaboration in the arm ecosystem. URL `https://www.linaro.org/`.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[4] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé. Multi-classification approaches for classifying mobile app traffic. *Journal of Network and Computer Applications*, 103:131–145, 2018.

[5] A. Adam, C. Dann, O. Yair, S. Mazor, and S. Nowozin. Bayesian time-of-flight for realtime shape, illumination and albedo. *IEEE transactions on pattern analysis and machine intelligence*, 39(5):851–864, 2016.

[6] G. Agresti, L. Minto, G. Marin, and P. Zanuttigh. Deep learning for confidence information in stereo and tof data fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Workshops)*, pages 697–705, 2017.

[7] G. Agresti, L. Minto, G. Marin, and P. Zanuttigh. Stereo and tof data fusion by learning from synthetic data. *Information Fusion*, 49:161 – 173, 2019. ISSN 1566-2535.

[8] G. Agresti, H. Schaefer, P. Sartor, and P. Zanuttigh. Unsupervised domain adaptation for tof data denoising with adversarial learning. June 2019.

[9] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia. Generative adversarial networks for unsupervised monocular depth prediction. In *15th European Conference on Computer Vision (ECCV) Workshops*, 2018.

[10] F. Aleotti, M. Poggi, F. Tosi, and S. Mattoccia. Learning end-to-end scene flow by distilling single tasks knowledge. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[11] F. Aleotti, F. Tosi, L. Zhang, M. Poggi, and S. Mattoccia. Reversing the cycle: self-supervised deep stereo through enhanced monocular distillation. In *European Conference on Computer Vision*, pages 614–632. Springer, 2020.

[12] F. Aleotti, G. Zaccaroni, L. Bartolomei, M. Poggi, F. Tosi, and S. Mattoccia. Real-time single image depth perception in the wild with handheld devices. *Sensors*, 21 (1):15, 2021.

[13] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *BMVC*, 2017.

[14] H. Alismail, B. Browning, and M. B. Dias. Evaluating pose estimation methods for stereo visual odometry on robots. In *the 11th International Conference on Intelligent Autonomous Systems (IAS-11)*, 2011.

[15] L. Andraghetti, P. Myriokefalitakis, P. L. Dovesi, B. Luque, M. Poggi, A. Pieropan, and S. Mattoccia. Enhancing self-supervised monocular depth estimation with traditional visual odometry. In *7th International Conference on 3D Vision (3DV)*, 2019.

[16] R. Anurag, V. Jampani, K. Kim, D. Sun, J. Wulff, and M. J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[18] M. Bai, W. Luo, K. Kundu, and R. Urtasun. Exploiting semantic information and deep matching for optical flow. In *European Conference on Computer Vision*, pages 154–170. Springer, 2016.

[19] D. Bailey. Space efficient division on fpgas. In *Electronics New Zealand Conference (ENZCon06)*, pages 206–211, 2012.

[20] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch. Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation. In *ICSAMOS*, pages 93–101, 2010.

[21] T. Barron and B. Poole. The fast bilateral solver. In *Proceedings of the 14th European Conference on Computer Vision*, ECCV, 2016.

[22] A. Behl, O. H. Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *International Conference on Computer Vision (ICCV)*, 2017.

[23] J.-W. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[24] C. M. Bishop. Mixture density networks. 1994.

[25] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1): 75–104, 1996.

[26] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.

[27] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012.

[28] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV)*, Oct. 2012.

[29] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.

[30] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. ShapeNet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[31] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2018.

[32] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3d tracking and forecasting with rich maps. 2019.

[33] Y.-L. Chang, Z. Yu Liu, and W. Hsu. Vornet: Spatio-temporally consistent video inpainting for object removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[34] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin. Hardnet: A low memory traffic network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3552–3561, 2019.

[35] C. Chen, X. Chen, and H. Cheng. On the over-smoothing problem of cnn based disparity estimation. 2019.

[36] L. Chen, W. Tang, and N. John. Self-supervised monocular image depth learning and confidence estimation. *arXiv preprint arXiv:1803.05530*, 2018.

[37] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[38] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[39] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *Advances in Neural Information Processing Systems*, pages 8699–8710, 2018.

[40] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[41] P.-Y. Chen, A. H. Liu, Y.-C. Liu, and Y.-C. F. Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[42] Y. Chen, C. Li, P. Ghamisi, X. Jia, and Y. Gu. Deep fusion of remote sensing data for accurate classification. *IEEE Geoscience and Remote Sensing Letters*, 14(8): 1253–1257, 2017.

[43] Y. Chen, C. Schmid, and C. Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*, 2019.

[44] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang. A deep visual correspondence embedding model for stereo matching costs. In *The IEEE International Conference on Computer Vision (ICCV)*. IEEE, December 2015.

[45] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[46] D. Ciuonzo, G. Romano, and P. S. Rossi. Channel-aware decision fusion in distributed mimo wireless sensor networks: Decode-and-fuse vs. decode-then-fuse. *IEEE Transactions on Wireless Communications*, 11(8):2976–2985, 2012.

[47] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[48] A. CS Kumar, S. M. Bhandarkar, and P. Mukta. Monocular depth prediction using generative adversarial networks. In *1st International Workshop on Deep Learning for Visual SLAM, (CVPR)*, 2018.

[49] C. Dal Mutto, P. Zanuttigh, and G. Cortelazzo. A probabilistic approach to ToF and stereo data fusion. In *Proc. of 3DPVT*, Paris, France, 2010.

[50] C. Dal Mutto, P. Zanuttigh, S. Mattoccia, and G. Cortelazzo. Locally consistent tof and stereo data fusion. In *Workshop on Consumer Depth Cameras for Computer Vision (ECCV Workshop)*, pages 598–607. Springer, 2012.

[51] C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo. Probabilistic tof and stereo data fusion based on mixed pixels measurement models. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2260–2272, 2015.

[52] C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo. Probabilistic ToF and stereo data fusion based on mixed pixels measurement models. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2260–2272, 2015.

[53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[54] L. Di Stefano, M. Marchionni, and S. Mattoccia. A fast area-based stereo matching algorithm. *Image and vision computing*, 22(12):983–1005, 2004.

[55] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[56] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[57] P. L. Dovesi, M. Poggi, L. Andraghetti, M. Martí, H. Kjellström, A. Pieropan, and S. Mattoccia. Real-time semantic stereo matching. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[58] G. Egnal, M. Mintz, and R. P. Wildes. A stereo confidence metric using single view imagery. In *PROC. VISION INTERFACE*, pages 162–170, 2002.

[59] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[60] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[61] D. Freedman, Y. Smolin, E. Krupka, I. Leichter, and M. Schmidt. Sra: Fast removal of general multipath for tof sensors. In *European Conference on Computer Vision*, pages 234–249. Springer, 2014.

[62] Z. Fu and M. Ardabilian. Stereo matching confidence learning based on multi-modal convolution neural networks. In *Representation, analysis and recognition of shape and motion FroM Image data (RFMI)*, 2017.

[63] Z. Fu and M. Ardabilian. Stereo matching confidence learning based on multi-modal convolution neural networks. In *Representation, analysis and recognition of shape and motion FroM Image data (RFMI)*, 2017.

[64] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[65] S. K. Gehrig, F. Eberli, and T. Meyer. A real-time low-power stereo vision engine using semi-global matching. In *ICVS*, pages 134–143, 2009.

[66] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012.

[67] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[68] S. Gidaris and N. Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017.

[69] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[70] C. Godard, O. Mac Aodha, and G. J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019.

[71] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[72] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *ICCV*, 2019.

[73] M. S. K. Gul, M. Bätz, and J. Keinert. Pixel-wise confidences for stereo disparities using recurrent neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA, 2019.

[74] X. Guo, H. Li, S. Yi, J. Ren, and X. Wang. Learning monocular depth by distilling cross-domain stereo networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[75] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li. Group-wise correlation stereo network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3273–3282. IEEE, 2019.

[76] R. Haeusler and R. Klette. Evaluation of stereo confidence measures on synthetic and recorded image data. In *2012 International Conference on Informatics, Electronics and Vision, ICIEV 2012*, pages 963–968, 2012.

[77] R. Haeusler, R. Nair, and D. Kondermann. Ensemble learning for confidence measures in stereo vision. In *CVPR. Proceedings*, pages 305–312, 2013. 1.

[78] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[79] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[80] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[81] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015.

[82] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[83] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE TPAMI*, 30(2):328–341, 2008.

[84] H. Hirschmüller, P. R. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *Int. J. Comput. Vision*, 47(1-3), apr 2002.

[85] H. Hirschmuller, M. Buder, and I. Ernst. Memory efficient semi-global matching. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 371–376, 2012.

[86] H. Hirschmüller and D. Scharstein. Evaluation of cost functions for stereo matching. In *CVPR*, 2007.

[87] D. Honegger, H. Oleynikova, and M. Pollefeys. Real-time and low latency embedded computer vision hardware based on a combination of fpga and mobile cpu. In *IROS*. IEEE, 2014.

[88] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17 (1-3):185–203, 1981.

[89] X. Hu and P. Mordohai. A quantitative evaluation of confidence measures for stereo vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 2121–2133, 2012.

[90] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *ICLR*, 2017.

[91] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[92] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[93] P.-Y. Huang, W.-T. Hsu, C.-Y. Chiu, T.-F. Wu, and M. Sun. Efficient uncertainty estimation for semantic segmentation in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 520–535, 2018.

[94] T.-W. Hui, X. Tang, and C. C. Loy. Lite-flownet: A lightweight convolutional neural network for optical flow estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[95] J. Hur and S. Roth. Joint optical flow and temporally consistent semantic segmentation. In *European Conference on Computer Vision*, pages 163–177. Springer, 2016.

[96] J. Hur and S. Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5754–5763, 2019.

[97] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[98] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018.

[99] E. Ilg, T. Saikia, M. Keuper, and T. Brox. Occlusions, motion and depth boundaries with a generic network for optical flow, disparity, or scene flow estimation. In *15th European Conference on Computer Vision (ECCV)*, 2018.

[100] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *arXiv preprint arXiv:1712.05877*, 2017.

[101] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[102] J. Janai, F. G"uney, A. Ranjan, M. J. Black, and A. Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *European Conference on Computer Vision (ECCV)*, volume Lecture Notes in Computer Science, vol 11220, pages 713–731. Springer, Cham, Sept. 2018.

[103] J. Y. Jason, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[104] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[105] H. Jiang, D. Sun, V. Jampani, Z. Lv, E. Learned-Miller, and J. Kautz. Sense: A shared encoder network for scene-flow estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[106] J. Jiao, Y. Cao, Y. Song, and R. Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–69, 2018.

[107] S. Joung, S. Kim, K. Park, and K. Sohn. Unsupervised stereo matching using confidential correspondence consistency. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[108] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[109] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[110] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. 2018.

[111] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weight losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[112] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft Level 5 AV Dataset 2019. url-https://level5.lyft.com/dataset/, 2019.

[113] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *15th European Conference on Computer Vision (ECCV 2018)*, 2018.

[114] S. Kim, D. g. Yoo, and Y. H. Kim. Stereo confidence metrics using the costs of surrounding pixels. In *2014 19th International Conference on Digital Signal Processing*, pages 98–103, Aug 2014.

[115] S. Kim, C. Y. Jang, and Y. H. Kim. Weighted peak ratio for estimating stereo confidence level using color similarity. In *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 196–197, Oct 2016.

[116] S. Kim, D. Min, S. Kim, and K. Sohn. Feature augmentation for learning confidence measure in stereo matching. *IEEE Transactions on Image Processing*, 26(12), Dec 2017.

[117] S. Kim, S. Kim, D. Min, and K. Sohn. Laf-net: Locally adaptive fusion networks for stereo confidence estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[118] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[119] M. Klodt and A. Vedaldi. Supervising the new with the old: learning sfm from sfm. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[120] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep

convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[121] Y. Kuznietsov, J. Stuckler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[122] H.-Y. Lai, Y.-H. Tsai, and W.-C. Chiu. Bridging stereo matching and optical flow via spatiotemporal correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019.

[123] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, 2016.

[124] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[125] S. Lefebvre, S. Ambellouis, and F. Cabestaing. A colour correlation-based stereo matching using 1D windows. In IEEE, editor, *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, SITIS'07*, pages 702–710, Shanghai, China, Dec 2007. IEEE.

[126] A. Li and Z. Yuan. Occlusion aware stereo matching via cooperative unsupervised learning. In *ACCV*. Springer, 2018.

[127] G. Li and J. Kim. Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation. In *British Machine Vision Conference*, 2019.

[128] H. Li, P. Xiong, H. Fan, and J. Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9522–9531, 2019.

[129] Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang. Learning for disparity estimation through feature constancy. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[130] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019.

[131] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 38(10):2024–2039, 2016.

[132] P. Liu, I. King, M. R. Lyu, and J. Xu. Ddflow: Learning optical flow with unlabeled data distillation. In *AAAI*, 2019.

[133] P. Liu, M. R. Lyu, I. King, and J. Xu. Selflow: Self-supervised learning of optical flow. In *CVPR*, 2019.

[134] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[135] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *PAMI*, 2019.

[136] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703. IEEE, 2016.

[137] Y. Luo, J. Ren, M. Lin, J. Pang, W. Sun, H. Li, and L. Lin. Single view stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 155–163, 2018.

[138] W.-C. Ma, S. Wang, R. Hu, Y. Xiong, and R. Urtasun. Deep rigid instance scene flow. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[139] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. J. Brostow. Learning a confidence measure for optical flow. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1107–1120, 2012.

[140] D. J. MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[141] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1): 3–15, 2017. doi: 10.1177/0278364916679498. URL `http://dx.doi.org/10.1177/0278364916679498`.

[142] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[143] O. Makansi, E. Ilg, and T. Brox. Fusionnet and augmentedflownet: Selective proxy ground truth for training on unlabeled images. *arXiv preprint arXiv:1808.06389*, 2018.

[144] R. Manduchi and C. Tomasi. Distinctiveness maps for image matching. In *Proceedings 10th International Conference on Image Analysis and Processing*, pages 26–31. IEEE, 1999.

[145] G. Marin, P. Zanuttigh, and S. Mattoccia. Reliable fusion of tof and stereo depth driven by confidence measures. In *14th European Conference on Computer Vision (ECCV 2016)*, pages 386–401, 2016.

[146] D. Martins, K. Van Hecke, and G. De Croon. Fusion of stereo and still monocular depth estimates in a self-supervised learning context. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 849–856. IEEE, 2018.

[147] L. Matthies. Stereo vision for planetary rovers: Stochastic modeling to near real-time implementation. *Int. J. Comput. Vision*, 8(1), jul 1992.

[148] S. Mattoccia. A locally global approach to stereo correspondence. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on. IEEE*, ICCV, 2009.

[149] S. Mattoccia and M. Poggi. A passive rgbd sensor for accurate and real-time depth sensing self-contained into an fpga. In *Proceedings of the 9th International Conference on Distributed Smart Cameras*, pages 146–151. ACM, 2015.

[150] S. Mattoccia and M. Poggi. A passive rgbd sensor for accurate and real-time depth sensing self-contained into an fpga. In *9th ICDSC*, 2015.

[151] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016.

[152] N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, and T. Brox. What makes good synthetic training data for learning disparity and optical flow estimation? 126(9):942–960, 2018.

[153] S. Meister, R. Nair, and D. Kondermann. Simulation of Time-of-Flight Sensors using Global Illumination. In *Vision, Modeling and Visualization.* The Eurographics Association, 2013.

[154] S. Meister, J. Hur, and S. Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, New Orleans, Louisiana, Feb. 2018.

[155] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[156] M. Menze, C. Heipke, and A. Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[157] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J. M. Frahm, R. Yang, D. Nister, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007.

[158] D. B. Min and K. Sohn. An asymmetric post-processing for correspondence problem. *Sig. Proc.: Image Comm.*, 25(2):130–142, 2010.

[159] P. Mordohai. The self-aware matching measure for stereo. In *The International Conference on Computer Vision (ICCV)*, pages 1841–1848. IEEE, 2009.

[160] C. Mostegel, M. Rumpler, F. Fraundorfer, and H. Bischof. Using self-contradiction to learn confidence measures in stereo vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[161] A. Mousavian, H. Pirsiavash, and J. Košecká. Joint semantic segmentation and depth estimation with deep convolutional networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 611–619. IEEE, 2016.

[162] R. Nair, F. Lenzen, S. Meister, H. Schäfer, C. Garbe, and D. Kondermann. High accuracy tof and stereo sensor fusion at interactive rates. In *European Conference on Computer Vision*, pages 1–11. Springer, 2012.

[163] R. Nair, K. Ruhl, F. Lenzen, S. Meister, H. Schäfer, C. S. Garbe, M. Eisemann, M. Magnor, and D. Kondermann. A survey on time-of-flight stereo fusion. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, pages 105–127. Springer, 2013.

[164] R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[165] D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[166] A. B. Owen. A robust hybrid of lasso and ridge regression. *Contemporary Mathematics*, 443(7):59–72, 2007.

[167] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[168] J. Pang, W. Sun, C. Yang, J. Ren, R. Xiao, J. Zeng, and L. Lin. Zoom and learn: Generalizing deep stereo matching to novel domains. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[169] M.-G. Park and K.-J. Yoon. Leveraging stereo matching with learning-based confidence measures. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[170] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[171] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. MIT Press, 2019.

[172] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin,

N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. 2019.

[173] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, and S. Mattoccia. Enabling energy-efficient unsupervised monocular depth estimation on armv7-based platforms. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1703–1708. IEEE, 2019.

[174] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, F. Aleotti, and S. Mattoccia. Enabling monocular depth perception at the very edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[175] V. Peluso, A. Cipolletta, M. Poggi, F. Tosi, F. Aleotti, and S. Mattoccia. Monocular depth perception on microcontrollers for edge applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[176] S. Perreault and P. Hbert. Median filtering in constant time. *IEEE Transactions on Image Processing*, 16(9):2389–2394, 2007.

[177] A. Pilzer, S. Lathuiliere, N. Sebe, and E. Ricci. Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[178] M. Poggi and S. Mattoccia. Learning a general-purpose confidence measure based on o(1) features and a smarter aggregation strategy for semi global matching. In *Proceedings of the 4th International Conference on 3D Vision, 3DV*, 2016.

[179] M. Poggi and S. Mattoccia. Learning from scratch a confidence measure. In *Proceedings of the 27th British Conference on Machine Vision, BMVC*, 2016.

[180] M. Poggi and S. Mattoccia. Deep stereo fusion: combining multiple disparity hypotheses with deep-learning. In *Proceedings of the 4th International Conference on 3D Vision, 3DV*, 2016.

[181] M. Poggi and S. Mattoccia. Learning to predict stereo reliability enforcing local consistency of confidence maps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[182] M. Poggi, S. Kim, F. Tosi, S. Kim, F. Aleotti, D. Min, K. Sohn, and S. Mattoccia. On the confidence of stereo matching in a deep-learning era: a quantitative evaluation. *arXiv preprint arXiv:2101.00431*.

[183] M. Poggi, F. Tosi, and S. Mattoccia. Even more confident predictions with deep machine-learning. In *12th IEEE Embedded Vision Workshop, CVPR 2017 workshop*, Jul 2017.

[184] M. Poggi, F. Tosi, and S. Mattoccia. Quantitative evaluation of confidence measures in a machine learning world. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[185] M. Poggi, F. Tosi, and S. Mattoccia. Efficient confidence measures for embedded stereo. In *19th International Conference on Image Analysis and Processing (ICIAP)*, September 2017.

[186] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia. Towards real-time unsupervised monocular depth estimation on CPU. In *IEEE/JRS Conference on Intelligent Robots and Systems (IROS)*, 2018.

[187] M. Poggi, F. Tosi, and S. Mattoccia. Learning monocular depth estimation with unsupervised trinocular assumptions. In *6th International Conference on 3D Vision (3DV)*, 2018.

[188] M. Poggi, G. Agresti, F. Tosi, P. Zanuttigh, and S. Mattoccia. Confidence estimation for tof and stereo sensors and its application to depth data fusion. *IEEE Sensors Journal*, 20(3):1411–1421, 2019.

[189] M. Poggi, D. Pallotti, F. Tosi, and S. Mattoccia. Guided stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 979–988, 2019.

[190] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[191] M. Poggi, F. Aleotti, F. Tosi, G. Zaccaroni, and S. Mattoccia. Self-adapting confidence estimation for stereo. *arXiv preprint arXiv:2008.06447*, 2020.

[192] M. Poggi, A. Tonioni, F. Tosi, S. Mattoccia, and L. Di Stefano. Continual adaptation for deep stereo. *arXiv preprint arXiv:2007.05233*, 2020.

[193] M. Poggi, F. Tosi, and S. Mattoccia. Good cues to learn from scratch a confidence measure for passive depth sensors. *IEEE Sensors Journal*, 20(22):13533–13541, 2020.

[194] M. Poggi, F. Tosi, and S. Mattoccia. Learning a confidence measure in the disparity domain from o (1) features. *Computer Vision and Image Understanding*, 193:102905, 2020.

[195] M. Poggi, F. Tosi, K. Batsos, P. Mordohai, and S. Mattoccia. On the synergies between machine learning and binocular stereo for depth estimation from images: a survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, April 2021. ISSN 0162-8828. doi: 10.1109/tpami.2021.3070917.

[196] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, Y. Wang, and A. Yuille.

Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017.

[197] Quanergy System, Inc. S3-8: The world's first affordable solid state lidar sensor. `https://quanergy.com/s3/`. Accessed: 2018-11-12.

[198] O. Rahnama, T. Cavalleri, S. Golodetz, S. Walker, and P. Torr. R3sgm: Real-time raster-respecting semi-global matching for power-constrained systems. In *2018 International Conference on Field-Programmable Technology (FPT)*, pages 102–109. IEEE, IEEE, 2018.

[199] O. Rahnama, T. Cavallari, S. Golodetz, A. Tonioni, T. Joy, L. Di Stefano, S. Walker, and P. H. Torr. Real-time highly accurate dense depth on a power budget using an fpga-cpu hybrid soc. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(5):773–777, 2019.

[200] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[201] H. Rashed, S. Yogamani, A. El-Sallab, P. Krizek, and M. El-Helw. Optical flow augmented semantic segmentation networks for automated driving. *arXiv preprint arXiv:1901.07355*, 2019.

[202] Z. Ren, J. Yan, B. Ni, B. Liu, X. Bin, and H. Zha. Unsupervised deep learning for optical flow estimation. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

[203] Z. Ren, O. Gallo, D. Sun, M.-H. Yang, E. Sudderth, and J. Kautz. A fusion approach for multi-frame optical flow estimation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2077–2086. IEEE, 2019.

[204] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2019.

[205] G. Riegler, Y. Liao, S. Donne, V. Koltun, and A. Geiger. Connecting the dots: Learning representations for active monocular depth estimation. 2019.

[206] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[207] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[208] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. 2019.

[209] R. Saxena, R. Schuster, O. Wasenmüller, and D. Stricker. PWOC-3D: Deep occlusion-aware end-to-end scene flow estimation. In *Intelligent Vehicles Symposium (IV)*, 2019.

[210] D. Scharstein and R. Szeliski. Stereo matching with non-linear diffusion. *International Journal of Computer Vision*, 28:155–174, 1998.

[211] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. Journal of Computer Vision*, 47(1-3):7–42, 2002.

[212] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. 2003.

[213] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, Springer, 2014.

[214] K. Schmid and H. Hirschmuller. Stereo vision and imu based real-time ego-motion and depth image computation on a handheld device. In *ICRA*. IEEE, 2013.

[215] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3260–3269. IEEE, 2017.

[216] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006.

[217] A. Seki and M. Pollefeys. Patch based confidence prediction for dense disparity map. In *British Machine Vision Conference (BMVC)*, 2016.

[218] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3889–3898, 2016.

[219] A. Shaked and L. Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. pages 4641–4650, 2017.

[220] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang. 3d photography using context-aware layered depth inpainting. 2020.

[221] W.-S. Z. Shuosen Guan, Haoxin Li. Unsupervised learning for optical flow estimation using pyramid convolution lstm. In *Proceedings of IEEE International Conference on Multimedia and Expo(ICME)*, 2019.

[222] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[223] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. 2020.

[224] N. Smolyanskiy, A. Kamenev, and S. Birchfield. On the importance of stereo for accurate depth estimation: an efficient semi-supervised deep neural network approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. IEEE, 2018.

[225] X. Song, X. Zhao, H. Hu, and L. Fang. Edgestereo: A context integrated residual pyramid network for stereo matching. In *Asian Conference on Computer Vision (ACCV)*, 2018.

[226] R. Spangenberg, T. Langner, S. Adfeldt, and R. Rojas. Large scale semi-global matching on the cpu. In *IV*, 2014.

[227] R. Spangenberg, T. Langner, S. Adfeldt, and R. Rojas. Large scale semi-global matching on the cpu. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 195–201. IEEE, 2014.

[228] A. Spyropoulos, N. Komodakis, and P. Mordohai. Learning to detect ground control points for improving the accuracy of stereo matching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1621–1628. IEEE, 2014.

[229] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[230] D. Sun, S. Liu, and J. Gaudiot. Enabling embedded inference engine with ARM compute library: A case study. *CoRR*, abs/1704.03751, 2017. URL `http://arxiv.org/abs/1704.03751`.

[231] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using

pyramid, warping, and cost volume. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[232] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. *arXiv preprint arXiv:1912.04838*, 2019.

[233] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[234] T. Taniai, S. N. Sinha, and Y. Sato. Fast multi-frame stereo scene flow with motion segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[235] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano. Unsupervised adaptation for deep stereo. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[236] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano. Unsupervised domain adaptation for depth prediction from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[237] A. Tonioni, O. Rahnama, T. Joy, L. Di Stefano, A. Thalaiyasingam, and P. Torr. Learning to adapt for stereo. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019.

[238] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. D. Stefano. Real-time self-adaptive deep stereo. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019.

[239] F. Tosi, M. Poggi, A. Tonioni, L. Di Stefano, and S. Mattoccia. Learning confidence measures in the wild. In *28th British Machine Vision Conference (BMVC 2017)*, September 2017.

[240] F. Tosi, M. Poggi, A. Benincasa, and S. Mattoccia. Beyond local reasoning for stereo confidence estimation with deep learning. In *15th European Conference on Computer Vision (ECCV)*, September 2018.

[241] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia. Learning monocular depth estimation infusing traditional stereo knowledge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[242] F. Tosi, M. Poggi, and S. Mattoccia. Leveraging confident points for accurate depth refinement on embedded systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[243] F. Tosi, F. Aleotti, P. Z. Ramirez, M. Poggi, S. Salti, L. D. Stefano, and S. Mattoccia. Distilled semantics for comprehensive scene understanding from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4654–4665, 2020.

[244] F. Tosi, Y. Liao, C. Schmitt, and A. Geiger. Smd-nets: Stereo mixture density networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[245] S. Tulyakov, A. Ivanov, and F. Fleuret. Weakly supervised learning of deep metrics for stereo reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1339–1348. IEEE, 2017.

[246] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017.

[247] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 722–729 vol.2, 1999. doi: 10.1109/ICCV.1999.790293.

[248] Velodyne LIDAR, Inc. Velodyne lidar. `https://velodynelidar.com/`. Accessed: 2019-04-3.

[249] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey. Learning depth from monocular videos using direct methods. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[250] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille. Towards unified depth and semantic prediction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2015.

[251] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang. The ApolloScape open dataset for autonomous driving and its application. 2019.

[252] R. Wang, M. Schworer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[253] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu. Unos: Unified unsupervised optical-flow and stereo-depth estimation by watching videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8071–8081, 2019.

[254] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, 13(4):600–612, Apr. 2004.

[255] J. Watson, M. Firman, G. J. Brostow, and D. Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019.

[256] A. Wedel, A. Meiner, C. Rabe, U. Franke, and D. Cremers. Detection and Segmentation of Independently Moving Objects from Dense Scene Flow. In *Proceedings of the*

7th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, pages 14–27, Bonn, Germany, August 2009. Springer.

[257] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[258] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black. Lessons and insights from creating a synthetic optical flow benchmark. In *ECCV Workshop on Unsolved Problems in Optical Flow and Stereo Estimation*, pages 168–177, 2012.

[259] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE international conference on computer vision*, pages 4705–4713, 2015.

[260] J. Xie, R. Girshick, and A. Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016.

[261] H. Xu, J. Zheng, J. Cai, and J. Zhang. Region deformer networks for unsupervised depth estimation from unconstrained monocular videos. In *IJCAI*, 2019.

[262] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia. Segstereo: Exploiting semantic information for disparity estimation. In *15th European Conference on Computer Vision (ECCV)*, 2018.

[263] G. Yang, J. Manela, M. Happold, and D. Ramanan. Hierarchical deep stereo matching on high-resolution images. 2019.

[264] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi, and B. Zhou. Drivingstereo: A large-scale dataset for stereo matching in autonomous driving scenarios. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019.

[265] N. Yang, R. Wang, J. Stückler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conference on Computer Vision*, pages 835–852. Springer, 2018.

[266] Q. Yang, R. Yang, J. Davis, and D. Nister. Spatial-depth super resolution for range images. pages 1–8, 2007.

[267] W. Yang, Y. Yang, Z. Yang, L. Zhao, and W. Xu. Occlusion aware unsupervised learning of optical flow. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[268] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia. Every pixel counts: Unsupervised geometry learning with holistic 3d motion understanding. In *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[269] Z. Yang, P. Wang, W. Yang, W. Xu, and N. Ram. Lego: Learning edge with geometry all at once by watching videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[270] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[271] Z. Yin, T. Darrell, and F. Yu. Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6044–6053. IEEE, 2019.

[272] K.-J. Yoon and I.-S. Kweon. Distinctive similarity measure for stereo matching under point ambiguity. *Computer Vision and Image Understanding*, 112(2):173–183, 2008.

[273] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmen-

tation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 325–341, 2018.

[274] L. Yu, Y. Wang, Y. Wu, and Y. Jia. Deep stereo matching with explicit cost aggregation sub-architecture. In *Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018.

[275] Y. W. J. S. Z. Ma, K. He and E. Wu. Constant time weighted median filtering for stereo matching and beyond. In *International Conference on Computer Vision*, ICCV, 2013.

[276] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Third European Conference on Computer Vision (Vol. II)*, 3rd European Conference on Computer Vision (ECCV), pages 151–158, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[277] P. Zama Ramirez, M. Poggi, F. Tosi, S. Mattoccia, and L. Di Stefano. Geometry meets semantic for semi-supervised monocular depth estimation. In *14th Asian Conference on Computer Vision (ACCV)*, 2018.

[278] P. Zanuttigh, G. Marin, C. Dal Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo. *Time-of-Flight and Structured Light Depth Cameras: Technology and Applications*. Springer, 1 edition, 2016. ISBN 978-3-319-30971-2.

[279] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17 (1-32):2, 2016.

[280] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[281] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature re-

construction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[282] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–194. IEEE, 2019.

[283] K. Zhang, J. Lu, and G. Lafruit. Cross-based local stereo matching using orthogonal integral images. *IEEE transactions on circuits and systems for video technology*, 19 (7):1073–1079, 2009.

[284] Y. Zhang, S. Khamis, C. Rhemann, J. Valentin, A. Kowdle, V. Tankovich, M. Schoenberg, S. Izadi, T. Funkhouser, and S. Fanello. Activestereonet: End-to-end self-supervised learning for active stereo systems. In *15th European Conference on Computer Vision (ECCV 2018)*, 2018.

[285] Z. Zhang, Z. Cui, C. Xu, Z. Jie, X. Li, and J. Yang. Joint task-recursive learning for semantic segmentation and depth estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 235–251, 2018.

[286] X. L. Zhang Q. and J. J. 100+ times faster weighted median filter. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, 2014.

[287] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[288] M. Zheng, C. Zhou, J. Wu, and L. Guo. Smooth deep network embedding. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[289] Y. Zhong, H. Li, and Y. Dai. Self-supervised learning for stereo matching with self-improving ability. *arXiv preprint arXiv:1709.00930*, 2017.

[290] Y. Zhong, H. Li, and Y. Dai. Open-world stereo video matching with deep rnn. In *ECCV*. Springer, 2018.

[291] C. Zhou, H. Zhang, X. Shen, and J. Jia. Unsupervised learning of stereo matching. In *The IEEE International Conference on Computer Vision (ICCV)*. IEEE, October 2017.

[292] J. Zhou, Y. Wang, N. Wang, and W. Zeng. Unsupervised high-resolution depth learning from videos with dual networks. In *Inter. Conf. on Computer Vision*. IEEE, IEEE, October 2019.

[293] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[294] J. Zhu, L. Wang, R. Yang, and J. Davis. Fusion of time-of-flight depth and stereo for high accuracy depth maps. 2008. ISBN 978-1-4244-2242-5.

[295] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

[296] Y. Zou, Z. Luo, and J.-B. Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *European Conference on Computer Vision (ECCV)*, 2018.