Machine learning techniques applied to stereo vision

PRESENTATA DA:
MATTEO POGGI

COORDINATORE DOTTORATO	RELATORE
CHIAR.MO PROF. ING.	CHIAR.MO PROF. ING.
PAOLO CIACCIA	STEFANO MATTOCCIA

Esame finale anno 2018

Abstract

Stereo is a popular technique enabling fast and dense depth estimation from two or more images. Its success is mainly due to its easiness of deployment, requiring only a couple or multiple synchronized image sensors, accurately calibrated to solve the matching problem between pixels on one of the images (named *reference*) and the other (named *target*). The absence of active technologies (e.g. pattern projection, laser scanners etc..) make this solution deployable on almost every scenario. Despite the wide literature concerning stereo, it still represents an open problem because of very challenging conditions such as poor illumination, reflective surfaces, occlusions and other elements occurring in real environments. Two main trends in stereo vision acquired popularity in the last years: confidence estimation and machine learning. Both proved to be very effective, pushing forward the state-of-the-art of dense disparity estimation.

In this thesis, we combine these two trends to improve both confidence estimation and disparity inference, by defining more effective and easier to deploy confidence measures and proposing new approaches to leverage on them for more accurate depth prediction. All the experiments are validated on three popular datasets, KITTI 2012, KITTI 2015 and Middlebury v3, following the commonly adopted methodologies and protocol to compare our proposals with previous works representing the state-of-the-art in stereo vision.

Acknowledgements

I would like to thanks all the people who supported and inspired me during my studies till this goal. In particular, special thanks to my supervisor, prof. Stefano Mattoccia, for assisting me during my PhD, for the continuous encouragements and making me enthusiastic to work on this research field, and to prof. Luigi Di Stefano for hosting me at Computer Vision lab, allowing me to work with awesome people. I also thank all these people, Alioscia Petrelli, Tommaso Cavallaro, Alessio Tonioni, Riccardo Spezialetti, Pierluigi Zama-Ramirez and Fabio Tosi, together with Alessandro Pernafini and Alessandro Zanni, for their great attitude at work and the many inspiring discussions about computer vision and other topics. Special thanks to my parents, who always supported my choice to apply for a PhD and always found good words to encourage me, even when the hardest challenges occurred. Thanks also to my mates outside the university, those who did not lose contact with despite the very different paths we follow. In particular to an awesome *friend*, whose empathy and good heart make time flying when I'm with her. I want to thank prof. Marc Pollefeys, head of CVG Lab at ETH Zurich, for hosting me 5 months and allowing an incredible growth as PhD student. In particular, this growth has been possible thanks to dr. Torsten Sattler and prof. Andreas Geiger, who welcomed me in Zurich and gave me great support, inspiration and tools for my studies. Together with all the people at CVG Lab, they made me feel home.

Publications

This thesis resulted in 9 papers plus 5 more related to other projects, for a total of 14 published works.

14. M. Poggi, F. Tosi, S. Mattoccia, ”Quantitative evaluation of confidence measures in a machine learning world”, accepted at The IEEE International Conference on Computer Vision (ICCV 2017), October 22-29, 2017, Venezia, Italy

13. A.Tonioni, M. Poggi, L. Di Stefano, S. Mattoccia, ”Unsupervised Adaptation for Deep Stereo”, accepted at The IEEE International Conference on Computer Vision (ICCV 2017), October 22-29, 2017, Venezia, Italy

12. F. Tosi, M. Poggi, A.Tonioni, L. Di Stefano, S. Mattoccia, ”Learning confidence measures in the wild”, accepted at The 28th British Machine Vision Conference (BMVC 2017), September 5-7, 2017, London, UK

11. M. Poggi, F. Tosi, S. Mattoccia, ”Efficient confidence measures for embedded stereo”, accepted at The 19th International Conference on Image Analysis and Processing (ICIAP 2017), September 11-15, 2017, Catania, Italy

10. M. Poggi, F. Tosi, S. Mattoccia, ”Even More Confident predictions with deep machine-learning”, accepted at The IEEE Embedded Vision Workshop (EVW 2017), July 21, 2017, Honolulu, Hawaii, US

9. M. Poggi, S. Mattoccia, ”Learning to predict stereo reliability enforcing local consistency of confidence maps”, accepted at The IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), July 21-26, 2017, Honolulu, Hawaii, US

8. M. Poggi, S. Mattoccia, ”Evaluation of variants of the SGM algorithm aimed at implementation on embedded or reconfigurable devices”, accepted at 6th International Conference on 3D Imaging (IC3D2016), December 13-14, 2016, Lige, Belgium

7. M. Boschini, M. Poggi, S. Mattoccia, ”Improving the reliability of 3D people tracking system leveraging on deep-learning”, accepted at 6th International Conference on 3D Imaging (IC3D2016), December 13-14, 2016, Lige, Belgium

6. M. Poggi, S. Mattoccia, ”Deep Stereo Fusion: combining multiple disparity hypotheses with deep-learning”, accepted at 4th International Conference on 3D Vision (3DV 2016), October 25-28, 2016, Stanford, California, USA

5. M. Poggi, S. Mattoccia, ”Learning a general-purpose confidence measure based on O(1) features and a smarter aggregation strategy for semi global matching”, accepted at 4th International Conference on 3D Vision (3DV 2016), October 25-28, 2016, Stanford, California, USA

4. M. Poggi, S. Mattoccia, ”Learning from scratch a confidence measure”, accepted at 27th British Machine Vision Conference (BMVC 2016), September 19-22, 2016, York, UK

3. M. Poggi, S. Mattoccia, ”A wearable mobility aid for the visually Impaired based on embedded 3D vision and deep learning”, First IEEE Workshop on ICT Solutions for eHealth (IEEE ICTS4eHealth 2016) in conjunction with the Twenty-First IEEE Symposium on Computers and Communications, June 27-30, 2016, Messina, Italy

2. M. Poggi, L. Nanni, S. Mattoccia, ”Crosswalk recognition through pointcloud processing and deep-learning suited to a wearable mobility aid for the visually impaired”, Image-based Smart City Application (ISCA2015), ICIAP Workshops 2015 : 282-289

1. S. Mattoccia, M. Poggi, ”A passive RGBD sensor for accurate and real-time depth sensing self-contained into an FPGA”, 9th ACM/SIGBED International Conference on Distributed Smart Cameras

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, progress has been made on a large number of high level computer vision tasks such as large scale reconstruction, SLAM, tracking and others thanks to the availability of 3D data. Thus, acquiring accurate and dense depth information is a crucial step of paramount importance for the success of these applications. Stereo vision represents an effective solution for 3D sensing from two or more images, suitable for real-time tasks and deployable on a wide range of scenarios thanks to its low costs and requirements compared to other, more expansive techniques. While active technologies (e.g., Kinect, etc...) work very well when acquiring indoor scenes, they usually perform poorly in outdoor environments because of some limitations. For example, active sensors such as Kinect have a limited working range and their effectiveness drastically drop when in presence of sunlight or multiple devices sensing the same area. On the other hand, stereo performs very well on such scenarios. The availability of some popular datasets, depicting real outdoor environments as well as high resolution indoor images, highlights how stereo is still an unsolved problem, because of very challenging elements such as occlusions, reflective surfaces and textureless regions, different illumination conditions and so on.

Machine learning, and more recently deep learning, proved great potential in many high level vision tasks, such as image classification, semantic segmentation etc., outperforming traditional techniques. Driven by these successes, in this thesis we will inquire about the application of such methodologies to low level vision problems like stereo matching. In particular, following the recent trends in this field deploying confidence measures for outlier detection and to improve the accuracy of stereo, we deploy machine and deep learning for better confidence estimation, as well as more accurate depth inference leveraging on it. The main topics covered by this thesis will be the possibility to predict a confidence measure from the disparity domain only without processing any cue from cost volume, leveraging on random forests (Chapter 4) or convolutional neural networks (Chapter 5), together with the definition of a complete taxonomy of the many confidence measures present in literature exhaustively evaluated on three challenging datasets using three popular stereo algorithms (Chapter 6). We will also address the implementation of some of these measures on embedded systems (Chapter 7), in order to exploit them also on custom stereo cameras and we will study how deep learning can improve the reliability of a confidence measure by using convolutional neural networks to replace random forest classifiers (Chapter 8) or to locally refine already predicted confidence maps (Chapter 9). In the final part, we will prove how confidence measures can be effectively exploited for many tasks concerning stereo vision, for example to combine results from multiple algorithms into a better disparity map (Chapter 9), to refine noisy maps and for self-supervised training of deep networks inferring dense disparity maps (Chapter 10) or for the training of confidence measures themselves (Chapter 11).

Chapter 2

Related work

The literature is rich of works addressing the stereo correspondence problem. According to the taxonomy proposed by Scharstein and Szeliski [73], conventional stereo matching algorithms can be grouped into two broad categories, *local* and *global* methods. Regardless of the category, stereo algorithms usually perform a subset of the following four steps: 1) matching cost computation 2) cost aggregation 3) disparity computation/optimization 4) disparity refinement. Local methods [85, 30, 31, 9] typically focus on steps 1 and 2, with step 2 exploiting local information (usually on a window of varying dimension), while global methods [40, 41, 39] mostly rely on 1 and 3, deploying in most cases iterative optimization strategies. The former are usually faster, exploiting information from nearby pixels to optimize pixel-wise matching costs assuming in most cases the fronto-parallel assumption [10, 73, 26, 88, 17]. While some methods simply adapt size or shape of the aggregation window [85, 26, 88] other techniques are driven by the content of reference image to better aggregate costs from pixels supposed to be on the same disparity level [17, 92, 84, 51]. In particular, the works of Rhemann et al. [31] and De-Maeztu et al. [9] proposed a cost aggregation strategy guided [46] by the left frame of the stereo pair running in constant time and achieving very accurate results on Middlebury dataset [73]. Another very fast aggregation strategy is represented by the work of Yang [91], deploying a Minimum Spanning Tree to aggregate costs over the entire image, weighted by the distance between pixels on the tree.

A good trade-off between accuracy and execution time is represented by the semi global method proposed by Hirschmuller [24]. This strategy, by means of the scanline optimization algorithm [73], enforces a *smoothness* constraint on several paths along the cost volume and sums up the outcome of each one. The optimal disparity is assigned according to a Winner Takes All (WTA) strategy applied to the final aggregated costs. Thanks to its very good trade-off between accuracy and complexity, it is one of the most popular algorithm for stereo. Original or variants of SGM have been implemented on different computing architectures such as FPGAs [3, 13, 28] and other embedded devices.

While for a long time stereo algorithms have been evaluated on the Middlebury dataset [73], containing only 4 stereo pairs with available ground-truth disparity labeling (which allowed for an explicit tuning of algorithms on the specific pairs), in the last years novel datasets became very popular, such as the KITTI 2012 [14] and KITTI 2015 [56], containing a large amount of training stereo pairs (194 and 200 respectively) with available ground-truth and more testing images (195 and 200) used for evaluation on the online benchmark, for which the ground-truth disparity is not provided in order to avoid explicit tuning. These images depict outdoor environments, mostly on the road in an autonomous driving scenario. Middlebury v3 [71] follows the same idea, providing 15 stereo pairs for training and 15 for online evaluation, collecting high-resolution frames in indoor. These new datasets highlighted how stereo matching is far from being considered solved. In the last years a popular topic concerning stereo matching is confidence measures, aimed at measuring the reliability or ambuguity of a disparity assignment. This topic has been reviewed and evaluated in [11, 12, 32]. In particular, [32] categorizes conventional confidence approaches according to the input cue processed and quantitatively evaluates their performance on two datasets by exploiting ROC curve analysis. More effective confidence measures, leveraging on machine learning techniques, significantly outperformed conventional stand-alone methods evaluated in [32]. In these methods [21, 80, 62], the reliability of disparity assignments is inferred by feeding a random forest with feature vectors containing multiple confidence measures [21, 80, 62] and hand-crafted clues extracted from disparity map [80, 62]. Compared to stand-alone confidence measures, the works by Haeusler et al. (*ENS* [21]), Spyropoulos et al. (*GCP* [80]) and Park and Yoon (*LEV* [62]) achieved significant improvements. Finally, in [75] a confidence measure (*PBCP*) is inferred with a CNN analyzing hand-crafted features extracted from *left-right* and *right-left* disparity maps, proving to outperform LEV, representing state-of-the-art. Recent works also proved the potential of confidence measures to achieve better results from stereo. Specifically, in [80] they are used as input cue for disparity inference based on MRF while in other cases to improve the effectiveness of the SGM algorithm modulating its raw matching costs [62] or dynamically adjusting P1 and P2 parameters [75]. Finally, in [48, 57], confidence measures have been deployed for sensor fusion.

The recent success of deep learning on high level computer vision tasks also reflected into low level problems. In the last few years, many authors addressed stereo vision by applying machine learning and deep learning. In particular, Zbontar and LeCun [95, 96] were the first to train a CNN to compute the matching costs replacing traditional and robust techniques used before such as AD-CENSUS [94]. Plugging this novel cost function into a well-designed stereo pipeline, made of iterative local aggregation steps [97], SGM and refinement post-processing resulted to rank first on KITTI and Middlebury v3 datasets at that time. Despite the effectiveness of this approach, the matching cost computation required several seconds to be carried out. In [96] a more efficient architecture based on dot product between features representations was proposed, resulting in much faster (nearly real-time) cost computation, with a negligible loss of accuracy when deployed into the established stereo pipeline. Similarly, Chen et al. [6] and Luo et al. [45] followed this approach. This strategy was further improved by Shaked and Wolf [77], deploying a three steps pipeline for matching cost computation, disparity prediction and refinement, introducing a self-learned confidence measure for this purpose. Disparity post-processing has been tackled similarly by Gidaris and Komodakis [19], training a CNN to process a disparity map and its reference image in order to detect outliers, replacing them and finally refining the map.

A first ground-breaking contribution to deep learning stereo is represented by the work of Mayer et al. [54], training an end-to-end network in charge of disparity prediction given the two rectified images, skipping any of the known steps commonly adopted by stereo matching algorithms. Then, Kendall et al. proposed GC-net [35], a deep architecture trained in end-to-end fashion as well, but implementing the well-known steps by leveraging 3D convolutions to globally optimize the cost volume and infer more accurate disparity maps, currently achieving the best results on KITTI datasets.

Chapter 3

How to compare with state-of-the-art

In this chapter, we introduce standard datasets and protocols used to evaluate both disparity and confidence estimation. Moreover, we also define three stereo algorithms involved on most of the experiments concerning confidence measures.

3.1 Reference datasets

Evaluation of newly proposed techniques with respect to previous works is possible thanks to some popular datasets providing ground-truth disparity maps. In this chapter we present three, popular benchmarks representing the most adopted datasets to measure the accuracy of stereo algorithms, which are KITTI 2012, 2015 and Middlebury v3.

3.1.1 KITTI 2012

Proposed by Geiger et al. [15], the first KITTI dataset for stereo matching was released on 2012. It is made of 194 training pairs acquired by synchronized cameras at a resolution of about 375×1242 pixels, provided together with ground-truth disparity maps acquired by a Lidar system. Because of the nature of such sensor, ground-truth maps are sparse, providing disparity assignments for about 33% of the total pixels. Values are 16 bit unsigned integers, encoding disparities between 0 and 255

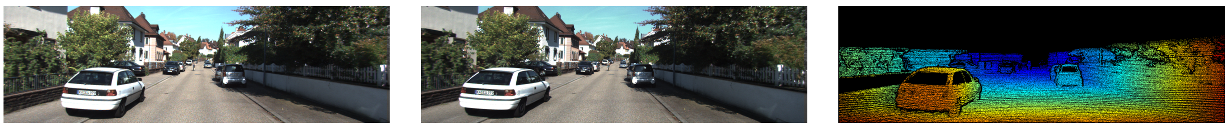


Figure 3.1: Stereo pair sample from KITTI 2012 dataset (training image 000006). From left to right, reference and target images, with ground-truth disparity map (warm color for higher values. Black pixels have no ground-truth available).

with sub-pixel precision. The 0 value is assigned to pixels with no ground-truth available. These samples can be used to tune algorithms before running them on the 195 test images, provided without ground-truth and used for the online evaluation[1]. All stereo pairs are made by grayscale frames (more recently, released also as color images), depicting static environments, on which camera ego-motion represents the only source of movement in the scene. Figure 3.1 shows an example from KITTI 2012 training set.

3.1.2 KITTI 2015

In 2015, a new KITTI benchmark as been proposed [56]. Following the successful 2012 dataset, it provides 200 training and 200 testing color pairs with the same resolution of the previous set. Some differences make this new set more diverse compared to the previous. In particular, most of the scenes are no longer static, making the estimation of optical flow much more challenging in presence of multiple objects moving with self motion. Moreover, dense disparity estimation is provided for cars by using CAD models, making it possible to provide depth information also for reflective surfaces which are challenging to acquire for a Lidar sensor. Figure 3.2 shows an example from KITTI 2015 training set, on which we can notice the much more dense disparity provided on the car compared to Figure 3.1. Moreover, this dataset allows for 3D scene flow evaluation, providing also ground-truth for disparity changes occurring between the reference frame and its correspondent image in the future, mapped on the reference system of the first.

[1] *cvlibs.net/datasets/kitti/eval_stereo.php*

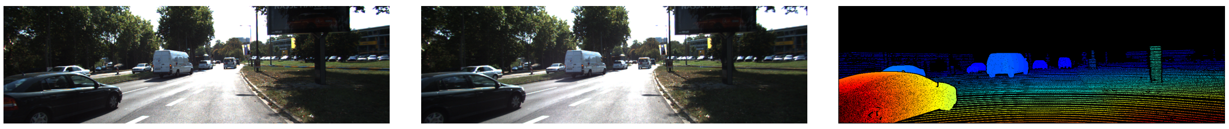


Figure 3.2: Stereo pair sample from KITTI 2015 dataset (training image 000095). From left to right, reference and target images, with ground-truth disparity map (warm color for higher values. Black pixels have no ground-truth available).

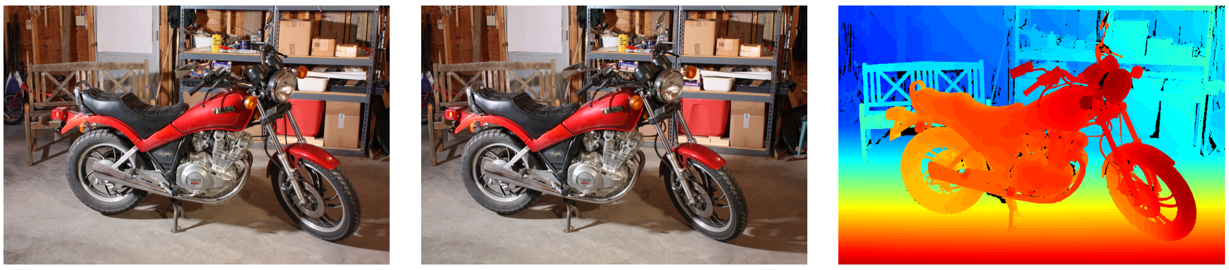


Figure 3.3: Stereo pair sample from Middlebury v3 dataset (training image *Motorcycle*). From left to right, reference and target images, with ground-truth disparity map (warm color for higher values. Black pixels have no ground-truth available).

3.1.3 Middlebury v3

The Middlebury v3 dataset [71] represents the most recent of a series of stereo benchmarks. It is made of 15 training and 15 testing pairs, with a maximum resolution of 2000×3000 pixels and a maximum disparity of 800, and an additional evaluation set made of 13 stereo pairs. Dense ground-truth data is provided for training and evaluation set, while is not available for frames evaluated by the online benchmark[2]. Because of the high-resolution setup (leading to a large disparity range) and the dense ground-truth available, it currently represent a very challenging benchmark for most of the stereo algorithms. In particular, its low number of training samples make hard for end-to-end deep learning methods to achieve state-of-the-art accuracy. Experimental results reported in this thesis are obtained processing quarter resolution images, to have a disparity range more consistent with KITTI datasets (being Middlebury v3, most of the time, used for cross-validation). Figure 3.3 show an example taken from the Middlebury v3 training set.

[2] $vision.middlebury.edu/stereo/eval3/$

3.2 Evaluation protocols

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

3.2.1 Disparity estimation

Three common metrics are used to evaluate the accuracy of disparity estimations compared to ground-truth maps. Among them, the badτ error rate represents the percentage of pixels in the disparity map having an error larger than τ. Usually, **bad3** rate is used on KITTI 2012 and KITTI 2015 datasets to rank algorithms on the online benchmarks, while Middlebury v3 usually adopt a **bad1** score.

Other statistics are the end point error, usually referred to as **EPE** or **MAE** (mean average error), which consists into the disparity error averaged over all pixels in the image. Similarly, **RMSE** (root mean square error) is obtained by averaging the disparity root square error.

The two kinds of errors measures very different behaviors. In particular, we may have a disparity map with a MAE of 1.1, which is relatively low, having an error of 1.1 on all pixels, resulting into a 100% bad1 error rate, while we can have the same MAE on a disparity map having a bad1 of about 1% and an error of magnitude 110 on these pixels.

3.2.2 Confidence measures

To evaluate the effectiveness of a confidence measure, in particular for the outlier detection task, we follow the ROC curve analysis protocol proposed by Hu and Mordohai [32]. Given a confidence map, it sequentially extract pixels from it and measures the amount of outliersg amon them, being an outlier a point having a disparity error larger than a certain threshold τ. In particular, pixels are sorted according to their confidence in descending order and a subset of them equal to 5% of the total is sampled and the error rate is plotted, then the subset is increased to 10% of the total and so on until 100%. Ties are solved by taking into the subset all points with the same confidence value. This results into plotting a curve, by measuring the ***Area Under the Curve*** **(AUC)** we can compare different confidences and assess when one more effectively detects outliers. Given the error rate ε as the percentage of wrong pixels on the entire image, we can obtain the optimal AUC as follows

$$AUC_{opt} = \int_{1-\varepsilon}^{\varepsilon} \frac{p-(1-\varepsilon)}{p} dp = \varepsilon + (1-\varepsilon)\ln(1-\varepsilon) \tag{3.1}$$

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. According to the dataset, τ threshold is usually set to 3 for KITTI 2012, 2015 and to 1 for Middlebury v3.

3.3 Stereo algorithms for confidence evaluation

To evaluate and compare multiple confidence measures following the AUC protocol, all of them must refer to the same disparity map(s) to allow for a fair confrontation. Maps obtained from different algorithms usually have different degrees of accuracy, thus we want to assess the effectiveness of confidence measures on accurate methods, as well as on fast but noisy techniques. Our experiments deploy three popular stereo algorithm, described below.

- AD-CENSUS [94], by applying a 5×5 binary census transform on input images, then computing matching costs according to Hamming distance, further aggregated on 5×5 support windows following the block matching principle before applying WTA. This very fast algorithm represents a popular choice to compute matching costs inside stereo pipelines deployed for real-time applications. Confidence measures effective on this algorithm can improve the accuracy of the overall pipeline [62, 75].
- Semi Global Matching (SGM) [24], it is the most popular algorithm for real-time applications and embedded systems. By filtering matching costs provided by AD-CENSUS along eight

scanlines, it provides much smoother disparity maps. It relies on two smoothing penalties P1 and P2, to discourage large disparity changes.

- MC-CNN [96], a siamese-CNN in charge of computing matching costs by processing 9×9 patches from input images, followed by WTA strategy, much more accurate than AD-CENSUS. We used the trained networks provided by the authors running MC-CNN-*fst*, providing a slightly higher error rate (about 2%), but much faster for experiments (about 100 times). Plugged into an SGM pipeline, it achieved the first rank on both KITTI and Middlebury benchmarks, further improved by subsequent works [75, 77, 76]. It represents the first, seminal method introducing deep learning to tackle stereo matching.

Chapter 4

Learning a confidence measure extracting features from the disparity domain

The content of this chapter has been presented at the 4th International Conference on 3D Vision (3DV 2016) - ”Learning a general-purpose confidence measure based on O(1) features and a smarter aggregation strategy for semi global matching”. Most relevant to the work shown in this chapter are the following papers: [32, 21, 80, 62, 94, 24, 25].

4.1 Introduction

The Semi Global Matching (SGM) algorithm [24] represents one of most popular techniques to infer a dense disparity map thanks to its good trade-off between accuracy and computation requirements. For this reason it has been implemented, according to different strategies and simplifications, on almost any computing architecture. SGM relies on multiple disparity optimization steps performed along different paths, typically 8 or 16. Disparity optimization is performed, by means of the Scanline Optimization (SO) [73] algorithm, minimizing an energy function. Although SO is very fast, disparity optimization on a 1D domain may lead to well-known *streaking* artifacts. SGM partially attenuates this problem by aggregating energy computed along different paths and by selecting, by means of a winner-takes-all (WTA) strategy, the disparity label with the minimum cost.

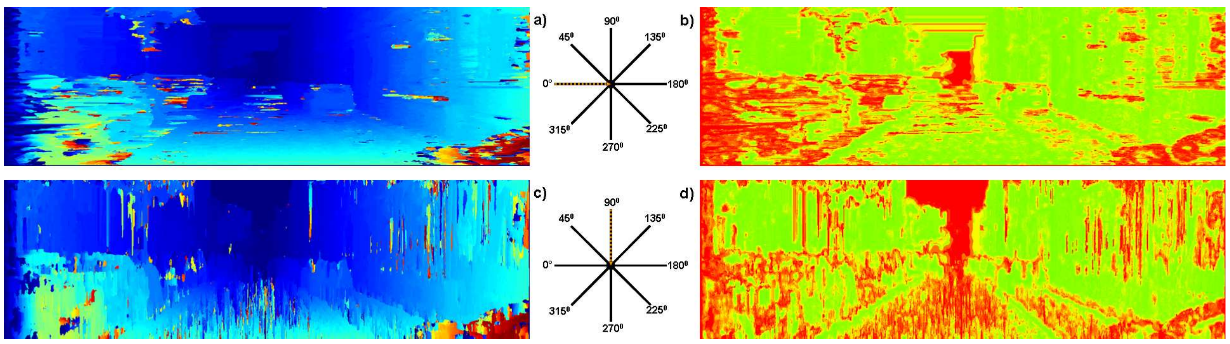


Figure 4.1: Streaking detection by O1 confidence measure.

In this work we take a deeper look at SGM with the aim to improve its accuracy by softening the propagation of streaking artifacts induced by SO. For this purpose, we propose a framework based on a random forest (RF) classifier that allows us to obtain a very effective and general-purpose confidence measure by processing features extracted from a disparity map. Then, we apply our confidence measure to the output of each single SO of SGM in order to detect streaking and thus softening its effect when aggregating costs from different paths.

In particular, focusing on SGM, we extract from the disparity map obtained along each path, with a WTA strategy, a pool of $O(1)$ features processed by our framework to obtain a confidence measure that encodes the degree of uncertainty of each SO. The outcome of this analysis is then fed to a smart aggregation step that, conversely from SGM, weights each path according to the estimated uncertainty in order to obtain a more accurate overall disparity map. We thoroughly evaluate effectiveness of our general-purpose confidence measure as well as the disparity accuracy achieved by our overall proposal, referred to as *smart*-SGM ($sSGM$), on KITTI 2012 [14]. Moreover, to avoid *overfitting* and to prove that it can generalize its behavior to different scenes, we cross-validated our method on KITTI 2015 [56] and Middlebury v3 [71] performing training on eight stereo stereo pairs from the KITTI 2012 dataset. In both evaluations, experimental results confirm that our proposal increases the accuracy of the original SGM algorithm with a minimal overhead and, by adopting appropriate strategy discussed later, enables obtaining better results with a reduced execution time and at a fraction of the original memory footprint. Moreover, to validate our $O(1)$ feature set, we compare the performance of our proposal when fed with such features and with the features proposed in [62]. This evaluation shows that our general-purpose confidence measure outperforms previous state-of-the-art.

4.2 Semi Global Matching

Semi Global Matching [24] represents a good trade-off between accuracy and computational complexity and for this reason it is very popular. Most of the top-performing algorithms in the literature rely on such method to obtain state-of-art results according to standard evaluation datasets [14, 56, 71]. For each pixel p, SGM combines the outcome of multiple energy minimizations computed by independent SO [73] algorithm on different paths $s \in S$, typically 8 or 16 according to [24]. For the 8 path version, referred to as SGM_8, the paths $S = \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$ are depicted in Figure 4.1. Each SO, within the disparity range $[0, d_{max}]$ and along each path $s \in S$, performs for each pixel p a disparity optimization according to the following energy term $E_s(p, d)$,

$$\begin{aligned} E_s(p,d) = C(p,d) + \min\Big\{ & E_s(p',d), E_s(p',d-1) + P1, \\ & E_s(p',d+1) + P1, \min_{i \in [0,d_{max}]} (E_s(p',i) + P2) \Big\} \\ & - \min_{i \in [0,d_{max}]} (E_s(p',i)) \end{aligned} \tag{4.1}$$

where p' represents the previous pixel along the path and $C(p, d)$ the *point-wise* or aggregated matching cost computed, for each disparity $d \in [0, d_{max}]$, between reference and target corresponding points along epipolar lines. Terms $P1$ and $P2$ ($P1 < P2$) in (4.1) enforce *smoothing* by penalizing disparity variations along each path s. According to [23], among the many cost functions proposed for stereo *non-parametric* approaches such as *census* perform very well in challenging environments. Compared to global approaches that enforce a smoothness term on a grid (i.e., 2D domain) SO is less computationally demanding. However, it is well-known that it is prone to streaking artifacts along the direction of the path. SGM softens this effect by summing up, for each point p, the results yielded by multiple SO as follows

$$E(p,d) = \sum_{s \in S} E_s(p,d) \tag{4.2}$$

and a selects the optimal disparity assignment according to a WTA strategy. The SGM algorithm requires to store the entire Disparity Space Image (DSI) [73] resulting in a high memory footprint. Moreover, strategy (4.2) attenuates streaking artifacts only partially. Our proposal aims at tackling both issues by learning a smarter aggregation strategy with respect to (4.2) driven by an analysis of the outcome of the SOs computed along each path $s \in S$.

4.3 Proposed method

In this work, we introduce a novel step within the stereo pipeline of the SGM algorithm to detect streaking artifacts occurring on each path with the aim to soften their propagation in the final disparity map. Streaking detection for each SO is carried out by means of a RF-based framework and then used to weight, accordingly, the contribution brought in by each scanline.

In this section, we introduce the feature vector adopted for our streaking detection module and we discuss the importance of the variables obtained through the training process. Then, we introduce a smart scanline aggregation approach that takes into account such confidence values to refine the final DSI.

4.3.1 Features extraction

We process a feature vector, through a RF framework, in order to infer, for each pixel p and path $s \in S$, a value $C_s(p)$ that encodes its degree of reliability ($\in [0, 1]$). Five cues are computed on four patches of increasing size $\Omega = \{5 \times 5, 7 \times 7, 9 \times 9, 11 \times 11\}$ centered on p. By observing the behavior of the streaking effect, which typically occurs near depth discontinuities, we extract features that enable to encode the statistical dispersion of disparity in the neighborhood of p. We define H_p^N the histogram of disparity within patch $N \in \Omega$ centered in p, $H_p^N(d)$ the amount of points at disparity d within N, and the cardinality N as:

$$\bar{\bar{N}} = \sum_{d \in [0, d_{max}]} H_p^N(d) \tag{4.3}$$

Given a patch N, centered on p, we extract the following cues from the disparity map:

1. Disparity agreement (DA), encodes the number of neighboring pixels with the same disparity of the central point p:

$$DA_p^N = H_p^N(d(p)) \tag{4.4}$$

A large amount of pixels sharing the same disparity of p stands for a higher likelihood of correctness with respect to circumstances where p has slighter support from its neighbors.

2. Disparity scattering (DS), encodes how many different disparity hypotheses appears in the neighborhood of p

$$DS_p^N = -log \frac{\sum_{d \in [0, d_{max}]} 1 - \delta(H_p^N(d), 0)}{\bar{\bar{N}}} \tag{4.5}$$

where δ is Kronecker's delta function (1 if $H_p^N(d)$ value is zero, 0 otherwise). According to such definition, a patch of $\bar{\bar{N}}$ pixels in complete disagreement with $d(p)$ yields to a DS value equal to zero. The lower the number of different hypotheses within the patch, the higher the value of the DS score is.

3. Median disparity (MD)

$$MD_p^N = median(H_p^N) \tag{4.6}$$

4. Variance of the disparity values (VAR),

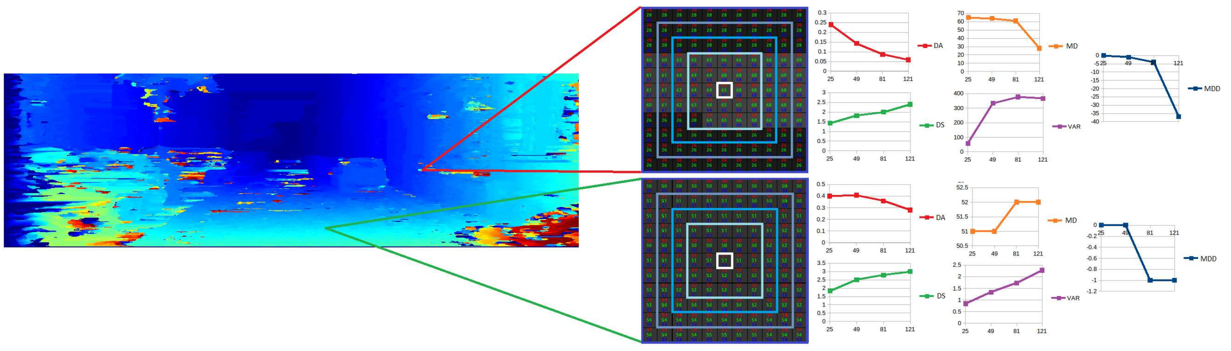


Figure 4.2: Example of hand-crafted features extracted from disparity map.

$$VAR = \frac{1}{\overline{\overline{N}}} \sum_{q \in N} (d(q) - \mu(p))^2 \tag{4.7}$$

with

$$\mu(p) = \frac{1}{\overline{\overline{N}}} \sum_{q \in N} d(q) \tag{4.8}$$

5. Disparity deviation from median (MDD), as proposed in [80, 62], which is the negative of the absolute difference between the disparity in p and the median disparity value in the patch N,

$$MDD = - \mid d(p) - MD(H_p^N) \mid \tag{4.9}$$

For each disparity map estimated by SO on path p, we combine these 5 features at four scales $N \in \Omega$ obtaining the following features vector, $f_{20} = [f_1, f_2, \ldots, f_{19}, f_{20}]^T$. By leveraging on a multi-scale approach, more information is provided to the RF to identify potentially erroneous matches. In particular, in presence of a streaking, with larger patches the magnitude of the features encoding the statistical dispersion decreases. Figure 4.2 gives an overview of the multi-scale approach described, emphasizing the different behavior of each feature and for each patch size in two completely different circumstances (with streaking, on top, and without streaking). It worth to note that the proposed features can be computed in constant time exploiting $O(1)$ techniques such as *integral images* for VAR and histogram-based optimization techniques [63, 7, 34] for DA, DS, MD and MDD. Compared to features extracted analyzing the behavior of the cost curve [80, 62] with complexity $O(d_{max})$, all our features are independent of the disparity range as well as of the patch size and hence turn out to be $O(1)$.

We train an ensemble regression trees classifier that provides a confidence value $C_s(p)$ for each path. It is worth observing that, according to the proposed strategy, we can specialize the RF for each path s or we can train the RF on multiple paths obtaining a more general RF suited for any path. We will provide a detailed discussion of two strategies, respectively referred to as *multiple* (M) and *single*, in the experimental results section. Moreover, we point out that the computation of the disparity map for each $s \in S$ required by our approach introduces a negligible overhead being, substantially, the outcome of SO.

Finally, differently from [62], we do not consider false positives, false negatives, true positive and true negatives to rescale our confidence, in order to maintain the gap between lower and higher values. In fact, during the experimental evaluation we tested either raw and rescaled values, obtaining no substantial difference between the two approaches. Moreover, the former strategy allows us to enhance more effectively the costs of reliable scanlines.

4.3.2 Smart aggregation

Given a point p, the smart aggregation approach proposed aims at replacing the cost aggregation performed by SGM on each path computed by the SO algorithm with a strategy that takes into account the reliability $C_s(p)$ of each path $s \in S$ estimated by the RF. Specifically, for each point p, we aggregate the SO costs according to the following weighted sum:

$$E^*(p,d) = \frac{\sum\limits_{s \in S} C_s(p) E_s(p,d)}{\frac{1}{S} \sum\limits_{s \in S} C_s(p)} \tag{4.10}$$

in (4.10) the average confidence value at the denominator allows us to further enhance the dynamic of the cost curve whenever a path s is expected to be more reliable with respect to the others. Although it never occurred in our experimental evaluation, if all the $C_s(p)$ are zero we replace $E^*(p,d)$ with $E(p,d)$ and hence assign the disparity according to the conventional SGM approach.

In the next section we prove that the learned aggregation strategy outlined so far enables to improve the effectiveness of the SGM algorithm. Moreover, with a subset of appropriate paths $s \in S$, we are able to obtain better results with respect to the standard SGM approach. This strategy also enables us to reduce the execution time and the memory footprint of SGM making our proposal suited to higher resolution stereo pairs and computing architectures with constrained resources.

4.4 Experimental results

In this section, we provide an exhaustive evaluation of our proposal on standard dataset KITTI 2012 detailing the methodology adopted to train the RF in two different configurations (i.e., single and multiple RF) and evaluating, by means of Area Under the Curve (AUC) [32], the confidence measure $C_s(p)$ provided by our framework. Then, we report on the same dataset, the improvements yielded by our framework with respect to SGM_8. Moreover, to prove that our framework is able to generalize to different scenarios, we provide additional experimental results, in terms of AUC and improvements with respect to SGM_8, regarding the cross-validation on KITTI 2015 and Middlebury v3 with the RF trained on KITTI 2012. Finally, we show that our method outperforms state-of-the-art [62] and also report an experimental evaluation combining the two approaches.

4.4.1 Framework configuration and training

In our experiments, we adopt as baseline the SGM_8 [24] algorithm computed on the 8 paths belonging to S depicted in Figure 4.1. As matching cost function we use the Hamming distance, aggregated on 5×5 patches, computed on images obtained according to a binary census transform considering 5×5 neighborhood points. We set parameters $P1$ and $P2$ of the SGM algorithm to 30 and 300, respectively. According to [4] we do not change these parameters being the target datasets quite structured.

We tuned an ensemble classifier made of 10 regression trees, maximum depth equals to 25 and minimum number of samples in each node to split equal to 20. To generate the training data, we processed eight challenging stereo pairs from KITTI 2012 commonly adopted on related works [21, 62], which

Dataset	*Optimal*	PKRN	LRD	LEV [62]	Proposed	LEV [62] (M)	Proposed (M)
KITTI 2012	0.038202	0.182604	0.155302	0.075122	0.072264	0.092018	**0.071020**
KITTI 2015	0.043930	0.193883	0.160993	0.098198	0.092410	0.111211	**0.089296**
Middlebury v3	0.050107	0.181431	0.172787	0.093343	**0.084257**	0.112117	0.084539

Table 4.1: AUC analysis, comparison between PKRN, LRD, LEV [62] and O1 [65] on the single scanlines, averaged over all directions. We mark with (M) results obtained by training a different random forest for each scanline direction.

are 43, 71, 82, 87, 94, 120, 122, and 180^{th}. For each of these stereo pairs, eight independent SOs provide a disparity map for each path according to the WTA strategy. We evaluate the performance of our proposal with a single RF as well as with one RF for each path. It is worth observing that, in this latter case, the amount of training samples on the same images is reduced by a factor 8. Moreover, we trained two versions of our framework: one with our features and the other with the features proposed in [62] in order to provide a comparison with state-of-art. The evaluation was carried out on the remaining images of the KITTI 2012 dataset and also cross-validated (with the same training) on KITTI 2012, KITTI 2015 and Middlebury v3 datasets. For the latter case we used images at quarter resolution.

4.4.2 Confidence evaluation

We compute AUC, a common method [32, 80, 21, 62] to evaluate the effectiveness of a confidence measure. Table 4.1 reports average AUCs computed on KITTI 2012 (first row), KITTI 2015 (second row) and Middlebury v3 (third row) datasets, evaluating it over the results of each SO and averaging. The tables report optimal values and AUCs related to PKRN [32], LRD [32], LEV [62], our proposal, LEV trained on configuration M and our proposal trained on configuration M.

The numbers show that our feature vector f_{20} significantly outperforms in most cases state-of-the-art [62]. Moreover, we can notice that configuration M yields even more accurate results in terms of confidence prediction. On average, on all the eight paths, the relative improvement in terms of AUC with respect to [62] adopting our features is 22% on KITTI 2012, 20.4% on KITTI 2015 and 24.6% on Middlebury v3 for configuration M and, respectively, 3%, 6.6% and 9.7% with a single RF. Regarding our proposal, the relative improvement yielded by configuration M with respect to training a single RF is 1.7% on KITTI 2012, 3.3% on KITTI 2015 and -0.3% on Middlebury v3.

Algo	Optimal	LEV [62]	Proposed	Win rate
AD-CENSUS	0.137	0.179	**0.163**	200/200
SGM_8	0.038	0.124	**0.095**	197/200
AD-CENSUS	0.093	0.114	**0.106**	13/15
SGM_8	0.042	0.093	**0.063**	15/15

Table 4.2: AUC analysis, comparison between LEV [62] and O1 on AD-CENSUS and SGM algorithms.

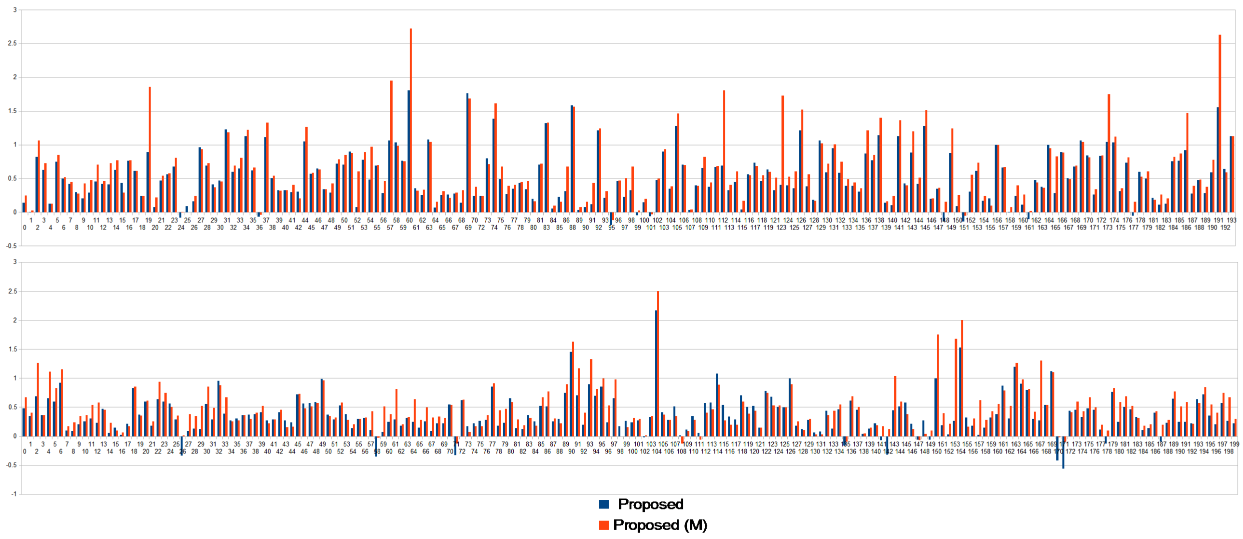


Figure 4.3: Absolute improvement of disparity accuracy yielded by [65] on KITTI datasets.

Summarizing, configuration M clearly performs better when compared to [62] with an average relative improvement of 22.3%. On the other hand, when comparing our method in the two configurations proposed, on the Middlebury dataset we do not have a dominant strategy for all the eight scanlines.

To further confirm the effectiveness of the proposed features, regardless to its application to the smarter aggregation strategy described so far, we evaluated AUC values for confidence measures yielded by our proposal and [62] with two different algorithms: AD-CENSUS and the outcome of the full SGM_8 method (i.e., not the single SOs). Table 4.2 reports average results on KITTI 2015 and Middlebury v3, showing how our general-purpose confidence measure clearly outperforms [62] even considering the output of generic stereo algorithms outside the smarter aggregation context previously proposed.

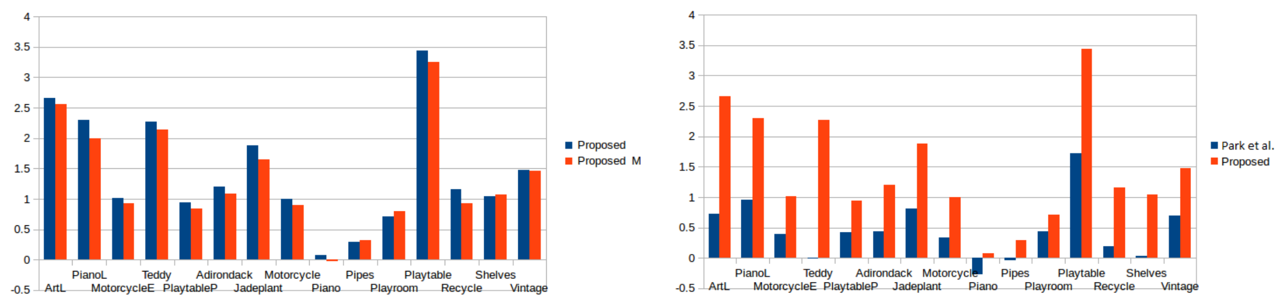


Figure 4.4: Absolute improvement of disparity accuracy yielded by [65] on Middlebury v3.

4.4.3 Disparity accuracy evaluation

In this section, we assess the performance of our proposal, referred to as $sSGM_8$, by gathering the absolute improvement in terms of error rate, with respect to the baseline SGM_8 algorithm, with our features and with the features proposed in [62]. Moreover, we include in this evaluation the results gathered by our own implementation of the DSI modulation proposed in [62].

Figures 4.3 and 4.4-left report the absolute disparity accuracy improvement on KITTI 2012, KITTI 2015 and Middlebury v3 obtained by $sSGM_8$, with our features in both configurations, with respect to baseline SGM_8. On KITTI datasets configuration M outperforms the single RF. In particular, on average, SGM_8 achieves a 9.90% error rate on KITTI 2012 and 9.56% on KITTI 2015. $sSGM_8$ in single RF configuration achieves, respectively, 9.38% and 9.14% $(-0.52\%$ and $-0.42\%)$ while configuration M achieves 9.26% and 9.04% $(-0.64\%$ and $-0.52\%)$. Conversely, on average, on the Middlebury dataset the single RF performs slightly better than configuration M. In fact, SGM_8 has an error rate of 22.93%, single RF 21.50% (-1.43%) and configuration M 21.60% (-1.33). These accuracy improvements follow the behavior of the confidence measure C_s analyzed in the previous section.

Figures 4.5 and 4.4-right report the absolute accuracy improvement yielded by our framework, in configuration M, for KITTI datasets (Figure 4.5) and single RF for Middlebury dataset (Figure 4.4-right), using our framework with the proposed features and with those of LEV. On the three datasets, our feature vector is always more effective than state-of-the-art when deployed with the smart aggregation strategy proposed. In particular, on average, with the feature vector [62] we obtain 9.40% $(+0.14)$ on KITTI 2012, 9.14% $(+0.10)$ on KITTI 2015 and 22.47% $(+0.97)$ on Middlebury.

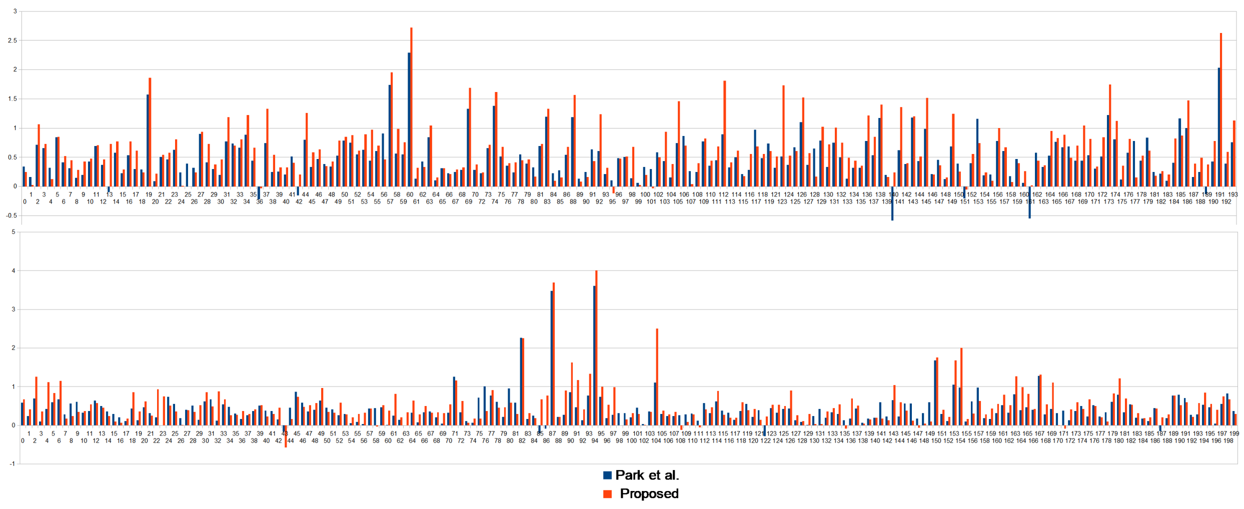


Figure 4.5: Absolute improvement of disparity accuracy yielded by [65] on KITTI datasets, comparison between [62] and [65] features.

It is worth to point out that, extending the training set by a factor 8 slightly improves the performance of configuration M on the Middlebury dataset but does not allow in the tested cases to outperform the single RF. Nevertheless, regarding the comparison of feature vector, on the three datasets our proposal outperforms [62] in any configuration and amount of training samples.

We also compared our proposal with the DSI modulation proposed in [62], referred to as $PARK_8$, applied to our baseline SGM_8 algorithm. According to this evaluation we obtained, on average, with $sSGM_8$ an absolute improvement with respect to $PARK_8$ of 1.04% on KITTI 2012 and of 0.69% on KITTI 2015. Finally, we evaluated the combination of $PARK_8$ and the proposed $sSGM_8$ obtaining an absolute improvement with respect to SGM_8 of 1.04 on KITTI 2012 and of 1.14 on KITTI 2015.

For each path, on KITTI datasets: without specific optimizations, our C++ implementation of SO requires $4s$, computing feature vector $0.08s$ and confidence measure computation $0.53s$. The final smart aggregation phase introduces a negligible overhead. Therefore, our method substantially adds an overhead of 15% to the overall execution time. Computing our feature vector is $O(1)$ and the memory footprint of the RF framework is independent of image resolution and proportional to the number of paths. The high memory footprint is a major issue of the SGM algorithm, particularly relevant with computing architectures with constrained memory resources. However, this fact may be critical with any device when dealing with high resolution stereo pairs. For instance, the full resolution

Dataset	K12	K15	M14	avg.
SGM_8	9.90%	9.59%	22.92%	**14.13%**
$sSGM_8$	9.26%	9.04%	21.49%	13.26%
SGM_4	10.65%	11.19%	23.50%	15.11%
$sSGM_4$	9.41%	9.60%	22.07%	**13.69%**

Table 4.3: Absolute improvement of disparity accuracy yielded by [65] with different numbers of directions.

Middlebury dataset has images of size $W \times H = 3000 \times 2000$ with a disparity range d_{max} = 800. In this very case, the footprint of the DSI would be $\propto$ to 9 GB, using 16 bit *short* for aggregated costs. This amount of memory might be prohibitive with any current computing device including standard PCs. On the other hand, it is worth observing that using a subset of S made of paths $0°, 45°, 90°$ and $135°$ the SGM algorithm, referred to as SGM_4, would have a memory footprint reduced by a factor $(H + 3)/3$. For the full resolution Middlebury dataset this factor is about 667 (memory footprint of SGM_4 about 13.8 MB), for KITTI datasets is about 124 (memory footprint of SGM_4 about 1.9 MB). Even compared to the memory-efficient eSGM [25] (providing results almost equivalent to the vanilla SGM_8 adding, however, a further image scan), our approach enables a notable reduction of the memory footprint improving at the same time the overall accuracy. The memory of $sSGM_4$ is reduced, with respect to eSGM, by a factor almost 10 on the KITTI dataset and by a factor almost 16 on the full-resolution Middlebury v3 dataset. Moreover, with the huge resolution stereo pairs reported in the eSGM paper [25], the memory footprint of $sSGM_4$ is 0.03 GB, 4.8 GB for eSGM and 272 GB for SGM_8 . Although the SGM_4 does not provide the same accuracy of SGM_8 (and eSGM), it has been widely adopted, at the expense of reduced performance with respect to SGM_8, when the memory footprint represents the major constraint [3, 13]. Nevertheless, on the same four paths previously highlighted, the proposed method, referred to as $sSGM_4$, clearly outperforms SGM_8 as reported in Table 4.3 on KITTI 2012, KITTI 2015 and Middlebury. This interesting fact can be exploited to reduce the execution time of SGM_8 and, more importantly, to drastically reduce the memory footprint without compromising its overall effectiveness in order to fit with a broader class of devices and image resolutions. Observing the table we can notice that, on average, on the three datasets $sSGM_4$ improves the disparity accuracy with respect to SGM_8 by 0.44% deploying only 4 paths and hence enabling a drastically reduced memory footprint.

4.5 Conclusions

In this work, leveraging on machine learning, we have: i) proposed a novel general-purpose confidence measure for stereo matching based on O(1) features uniquely computed in the disparity domain ii) focusing our attention on the popular and effective SGM algorithm, we have exploited our confidence measure to propose a smarter aggregation framework aimed at increasing the effectiveness of SGM with a negligible overhead c) the overall framework allows us to achieve, with respect to SGM, comparable or better accuracy with a notably lower memory footprint thus dealing with one of the major issues of this algorithm. Exhaustive experimental results on KITTI 2012, KITTI 2015 and Middlebury v3 confirmed the effectiveness of our proposals.

Chapter 5

Learning from scratch a confidence measure

The content of this chapter has been presented at the 27th British Conference on Machine Vision (BMVC 2016) - ”Learning from scratch a confidence measure”. Most relevant to the work shown in this chapter are the following papers: [32, 62, 94, 24].

5.1 Introduction

Most approaches, recently reviewed and evaluated by Hu & Mordohai [32], analyze intermediate results provided by stereo algorithms (i.e., the cost volume) and/or the final disparity map(s) in order to encode the uncertainty according to the behavior of an ideal approach. Recently, some authors [21, 80, 62] proved that improved results can be obtained by jointly processing a pool of CMs within a machine learning framework based on random forest (RF). In particular, state-of-the-art approach [62] proposed by Park & Yoon computes a very effective confidence measure processing, by means of a RF, a feature vector made of existing CMs and hand-crafted features obtained from the analysis of cost volume and disparity map. In the same work, the novel confidence measure was deployed to improve the overall disparity accuracy by cost volume modulation.

This work leverages on the succesful idea that the disparity domain already contains the information to effectively predict a confidence for each pixel, proved in the previous chapter. Starting from the

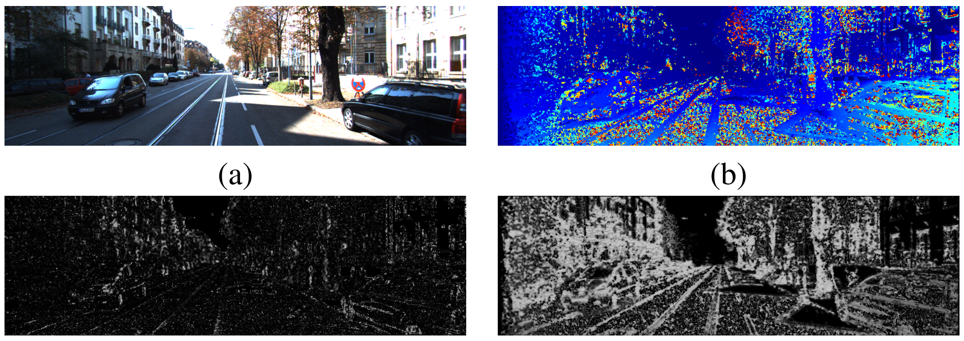

(a) (b) (c) (d)

Figure 5.1: Example of confidence maps (LRD and CCNN) on a stereo pair from KITTI dataset.

observation that correct and wrong measurements are typically characterized by recurrent patterns and considering the effectiveness of Convolutional Neural Networks (CNNs) applied to computer vision problems we decided to investigate the opportunity to obtain a confidence measure from scratch, instead of processing hand-crafted features. This means that our proposal follows a completely different strategy w.r.t. previous work in this field because it does not rely on any existing confidence measure nor it extracts hand-crafted features from the cost volume or the disparity map. Moreover, our proposal taking as input only the disparity map is also suited for out-of-the-box depth sensors (e.g., Intel Realsense [33], Zed camera [82], or FPGA-based stereo cameras [52, 28]) that in most cases do not provide the cost volume due to intellectual property issues, limited bandwidth, etc. Exhaustive experimental results and a cross-validation with different datasets and algorithms confirm that our proposal outperforms state-of-the-art. Figure 5.1 shows for frame 000000 of the KITTI 2015 dataset the reference image (a), the disparity map computed by a stereo algorithm (b), the outcome of a confidence measure known in literature (LRD) (c) and the result yielded by the confidence measure proposed in this work (d), referred to as CCNN.

5.2 Confidence measure inferred by a CNN

Our proposal starts from the observation that recurrent patterns characterize wrong and correct disparity assignments. In fact, as highlighted in Figure 5.2, local regions in the disparity map often contain recurrent patterns that enable to clearly assess the reliability of the disparity assignments. Motivated by recent work in this field [54, 60, 95, 96], we train, on a large dataset with ground-truth, a deep

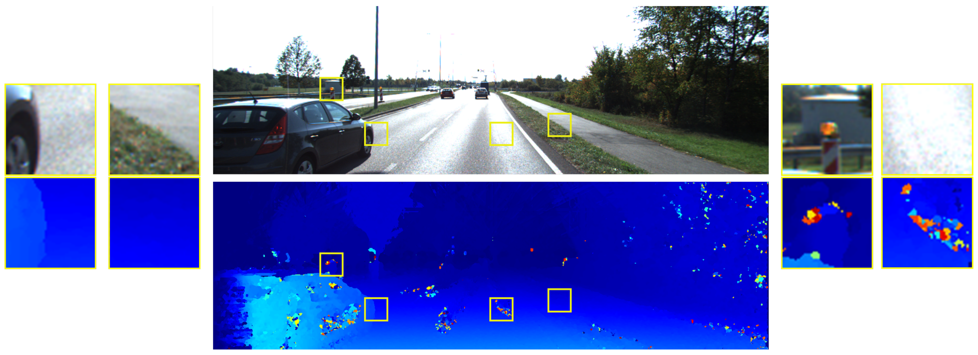


Figure 5.2: Example of outliers on a disparity map.

architecture to encode the degree of uncertainty from the disparity map. For each pixel, we extract a square patch centered on the disparity map and forward it to a CNN, trained to distinguish between patterns corresponding to correct and erroneous disparity assignments and, thus, to infer a confidence value. To this aim, we deploy a deep architecture, made of a relatively low number of layers with respect to state-of-the-art CNNs designed for higher level tasks, capable to learn such property and hence to infer an effective confidence measure.

5.2.1 Proposed architecture

The architecture of our CNN is made of a single channel network that takes as input $N \times N$ patches, each one containing disparity values normalized between zero and one, represented by a $1 \times N \times N$ tensor. Although the size of the patches is relatively small compared to the disparity map, it should provide to the CNN enough cues to infer the degree of uncertainty for each point. In our experiments we found that $N = 9$ enables to obtain quite effective results as reported in the experimental evaluation. The first part of our network is made of $\frac{N-1}{2}$ convolutional layers, each one followed by a Rectifier Linear Unit (ReLU).

$$ReLU(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases} \tag{5.1}$$

Each convolutional layer contains F filters of size 3×3. No padding or stride is applied, making the final output of the convolutional layers, a $F \times 1 \times 1$ tensor (each layer reduces the initial size N by 2

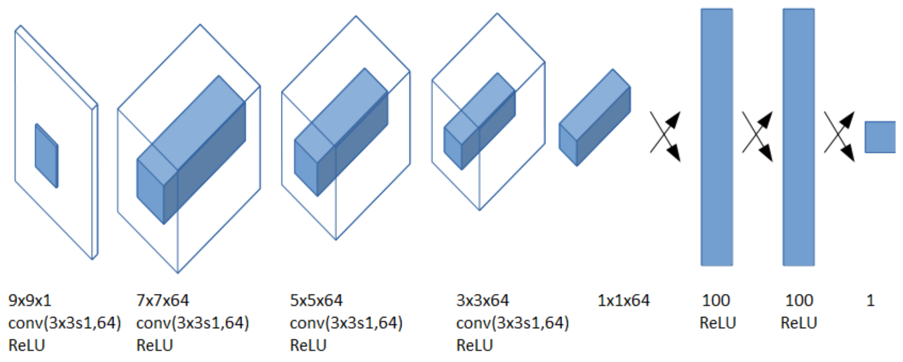


Figure 5.3: Architecture of the network predicting CCNN measure.

pixels), directly forwarded to the fully-connected part of the network deploying two layers, made of L neurons each, followed by ReLUs (11.1). The final layer collapses into a single neuron in charge of the regression.

According to a common methodology usually deployed when dealing with deep architectures, the fully-connected layers are replaced by convolutional layers made of L 1×1 kernels. This allows us to train the network on image patches (and, then, to easily handle samples generation and mini-batch dimension) as well as to compute a dense confidence map with a single forward pass of the full resolution image with a *0-padding* of $\frac{N-1}{2}$ around it, keeping for the output the same $w \times h$ size of the input disparity map due to the absence of pooling operations or stride factors inside the convolutional layers. Passing a single $w \times h$ image, which allows to reuse many intermediate results, rather than forwarding $w \times h$ patches of size $N \times N$ enables to significantly reduce the execution time [95, 96]. For instance, by running our approach on a standard Intel i7 6600K processor the time required to obtain a full confidence map (on a typical KITTI disparity map and $N = 9$) is about 5 minutes by forwarding single patches through the fully-connected network and only 630 ms with the outlined fully-convolutional architecture. Moreover, with a Titan X GPU, the same fully-convolutional network takes only 116 ms.

5.2.2 Training procedure

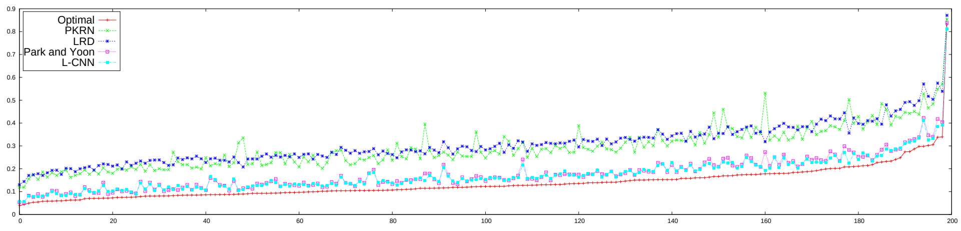
In our evaluation, we trained the proposed CNN architecture on the first 50 frames of the the KITTI 2012 dataset [14] extracting samples only centered on pixels with available ground-truth values (approximatively $\frac{1}{3}$ of the overall disparity values). This strategy provides more than 6.5 million samples to the CNN. Experiments with larger training datasets did not improve significantly the effectiveness of CCNN. Disparity maps for the training procedure are computed by the AD-CENSUS algorithm, aggregating costs on 5×5 patches. The pointwise matching costs are obtained according to the Hamming distance on census transformed images computed on 5×5 patches. The disparity map is obtained from the cost volume by means of the Winner-Takes-All strategy (WTA). We label with '1' all the confident disparity assignments (i.e., those values that differ by one or less from the ground-truth) and with '0' otherwise. According to this strategy, the average error rate of the AD-CENSUS algorithm is approximatively 50%. This fact provides a balanced distribution of samples for training the CNN.

In our evaluation, we found out that 9×9 patches enable a quite effective learning for our method. Therefore, our architecture is composed of 4 convolutional layers, each one made of $F = 64$ kernels as depicted in Figure 5.3. We deploy random connection tables, which improve learning and runtime speed and lead to superior matching prediction during the validation and cross-validation procedures. In particular, we obtained the best results with convolutional layers having a fan-in of 1 (i.e. , each kernel randomly takes as input one of the maps obtained from the previous layer), higher fan-in values did not lead to improvements. The two fully-connected layers are made of $L = 100$ neurons each (i.e., they are deployed as two 1×1 convolutional layers with 100 kernels each). During the training phase, we follow the Stochastic Gradient Descent (SGD) of the Binary Cross Entropy (BCE) between output o of the network and label t on each sample i of the mini-batch (11.3) by applying a sigmoid function S(x) (11.4) on the output of the network.

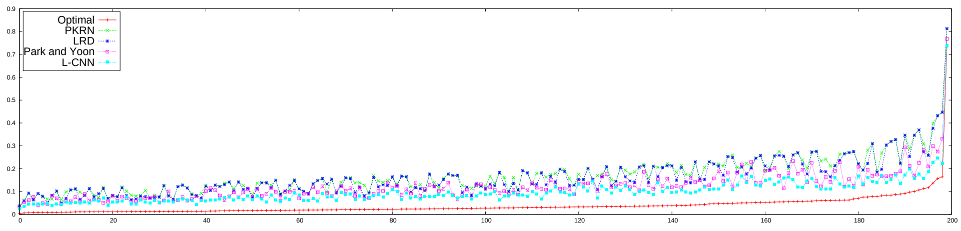
$$BCE(o,t) = -\frac{1}{n}\sum_i (t[i]\log(o[i])) + (1-t[i])(\log(1-o[i])) \tag{5.2}$$

$$S(x) = \frac{1}{1+e^{-x}} \tag{5.3}$$

We carried out 14 training *epochs*, with an initial learning rate of 0.003, increased by a factor 10 after the 10^{th} epoch, and a *momentum* of 0.9, inspired by [96] and confirmed by our experiments. To



(a)



(b)

Figure 5.4: AUC plots for PKRN, LRD, Park [62] and CCNN, on AD-CENSUS and SGM algorithms, KITTI 2015 dataset.

compare the confidence provided by our CNN with state-of-the-art, we also trained a RF as described in [62], adopting the full feature vector f_{22} described in the work in order to obtain the best results. For a fair comparison with our proposal, we trained [62] on the same 50 images of the KITTI 2012 dataset.

5.3 Experimental results

Once trained our CCNN approach on the 50 images of the KITTI 2012 dataset with the AD-CENSUS algorithm, in this section we assess its performance w.r.t. state-of-the-art with two datasets (KITTI 2015 and Middlebury v3) and with two stereo algorithms, AD-CENSUS and SGM [24]. The top performing CMs considered in our evaluation are: Park and Yoon [62], trained on the same dataset and algorithm, and two conventional, yet effective, CMs described in [32] referred to as Left Right Difference (LRD) and Peak Ratio Naive (PKRN). To do so, we follow the AUC evaluation protocol. In the remainder we compare the same four CMs on KITTI 2015 training data, a dataset with a content similar to the one adopted for training, and a second cross-validation experiment on Middlebury v3 training dataset (quarter resolution) [71] containing quite different scenes with respect to the other two datasets. In particular, this latter evaluation enables to further emphasize the ability of machine learning approaches, CCNN and Park & Yoon, to adapt not only to different algorithms but also to

Dataset/Alg.	Opt.	PKRN	LRD	Park&Yoon	CCNN	CCNN vs Park&Yoon
KITTI/AD-CENSUS	0.137	0.294	0.308	0.179	**0.175**	-2.2% (119/200)
KITTI/SGM	0.038	0.171	0.162	0.124	**0.099**	-20.2% (183/200)
Middl./AD-CENSUS	0.093	0.165	0.170	0.114	**0.107**	-6.1% (13/15)
Middl./SGM	0.042	0.095	0.098	0.093	**0.074**	-20.4% (13/15)

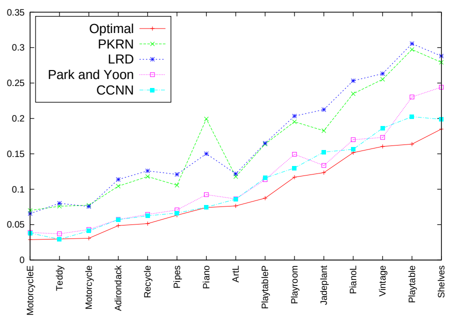
Table 5.1: AUC analysis, comparison between Park [62] and CCNN [66] on AD-CENSUS and SGM algorithms, KITTI 2015 and Middlebury v3 datasets.

quite different scene content. The outcome of this evaluation is crucial to determine if these methods, once trained, can be used as out-of-the-box CMs.

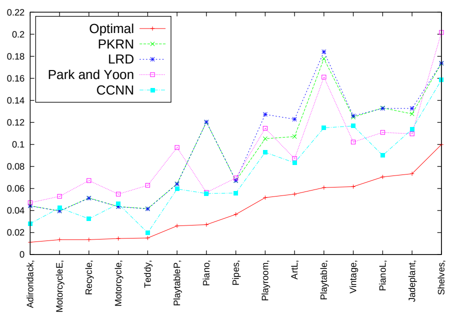
5.3.1 Validation on KITTI 2015

We perform, on the same four CMs, a first validation phase on the KITTI 2015 dataset [56] containing 200 stereo pairs with ground-truth data. Figure 5.4 depicts, for AD-CENSUS (a) and SGM (b), AUC values for each stereo pairs belonging to the KITTI 2015 training set, sorted in non-descending order with respect to their optimal values. First of all, the figure shows that, with both stereo algorithms, approaches based on machine learning techniques have significantly better performance. Observing the top of the figure, concerned with AD-CENSUS, we can notice that the proposed CCNN approach obtains slightly better results, as summarized in the first row of Table 5.1, with respect to Park & Yoon outperforming it in 119 out of 200 cases. Moreover, when dealing with disparity maps characterized by higher error rates, CCNN frequently provides results very close to optimality. Observing the bottom of the figure and the second row of the table, concerned with SGM, we can notice that the CCNN better generalizes to different input data with respect to state-of-the-art outperforming Park & Yoon in 183 out of 200 cases with an average improvement greater than 20%. This indicates that our proposal is more independent of the matching algorithm, not being based on cost volume whose content is strictly related to the stereo algorithm adopted for training. Finally, LRD and PKRN behave similarly to the previous AD-CENSUS case. However, with SGM, Park & Yoon is outperformed by LRD or PKRN in 27 out of 200 cases while this never happens for CCNN.

We also tested architectures with a lower number of convolutional kernels (i.e., 32 and 48 for each convolutional layers) obtaining higher AUC values w.r.t. the proposed architecture. In particular,



(a)



(b)

Figure 5.5: AUC plots for PKRN, LRD, Park [62] and CCNN, on AD-CENSUS and SGM algorithms, Middlebury v3 dataset.

processing AD-CENSUS disparity maps, the network with 32 kernels achieves an average AUC of 0.423, with 48 kernels 0.227 and with the final network with 64 kernels 0.175. On the disparity maps provided by SGM, we report an average AUC of 0.234 with 32 kernels, 0.110 with 48 kernels and 0.099 with the proposed network.

5.3.2 Cross-validation on Middlebury v3

In order to further stress the ability to generalize the performance of the considered CMs to more challenging conditions, we carried out a cross-validation on the Middlebury v3 dataset [71] containing 15 stereo pairs with ground-truth. This dataset depicts indoor environments, completely different w.r.t. those of the training dataset (KITTI 2012) and of the previous testing dataset (KITTI 2015) both concerned with outdoor environments. As in the previous evaluation we tested the four CMs with AD-CENSUS and SGM. Table 5.1, rows 3 and 4, summarizes the results reported in detail in Figure 5.5. With both stereo algorithms our method outperforms Park & Yoon in 13 out of 15 cases leading to an average improvement for AD-CENSUS and SGM, respectively, of 6.1% and 20.4%. Concerning AD-CENSUS, LRD and PKRN always provide worse results compared to approaches based on machine learning. On the other hand, although these latter approaches have similar performance CCNN performs better and in 4 cases out of 15 (Teddy, Pipes, Piano and PianoL) achieves results very close to optimality. With SGM, on average, LRD and PKRN provide worse results w.r.t. CCNN and Park & Yoon. However, Park & Yoon is significantly outperformed in 8 out of 15 cases by LRD or PKRN while slightly better results (MotorcycleE and Motorcycle) are obtained by these CMs w.r.t. CCNN

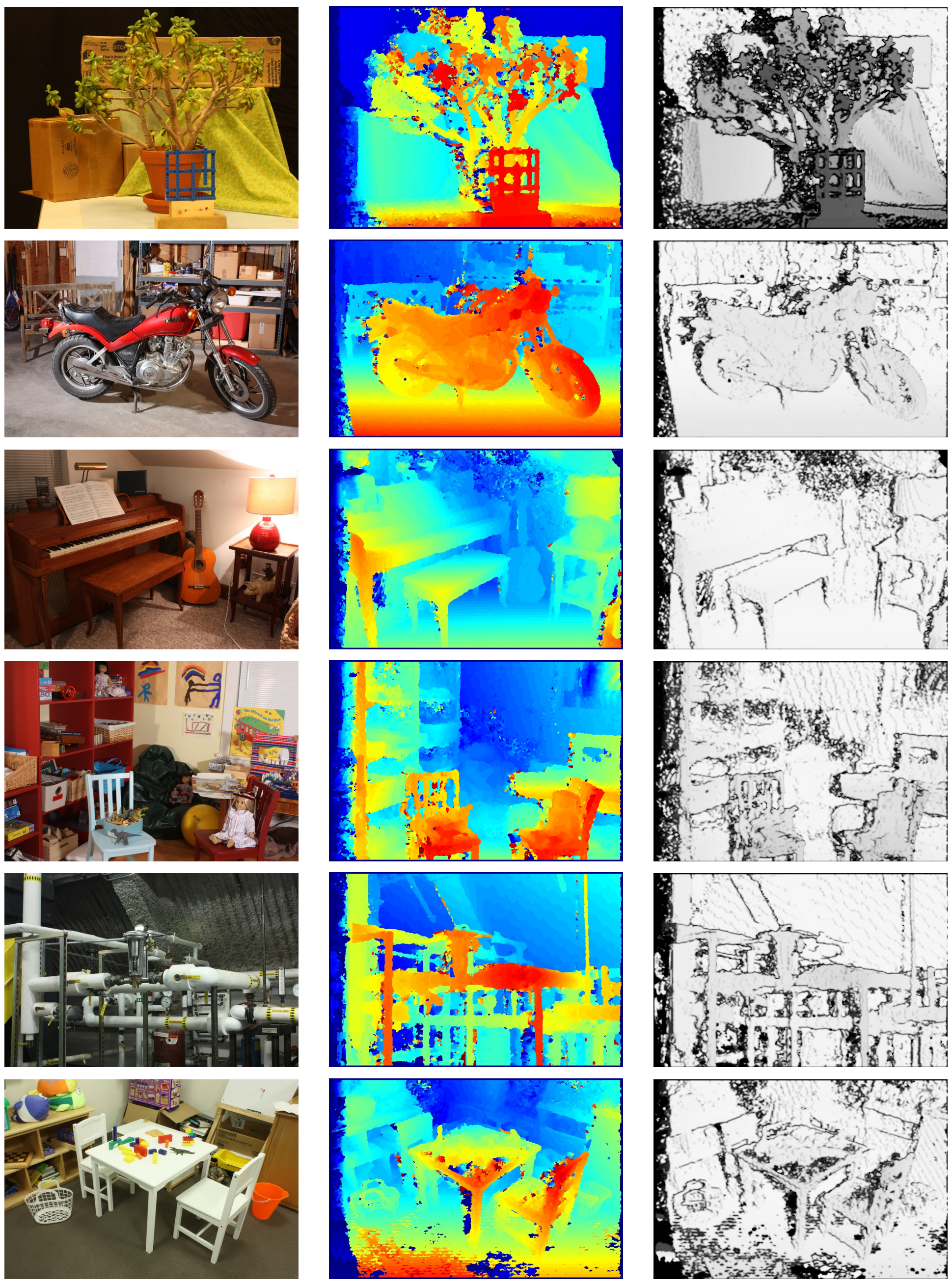


Figure 5.6: Qualitative results for CCNN, SGM algorithm, Middlebury v3 dataset.

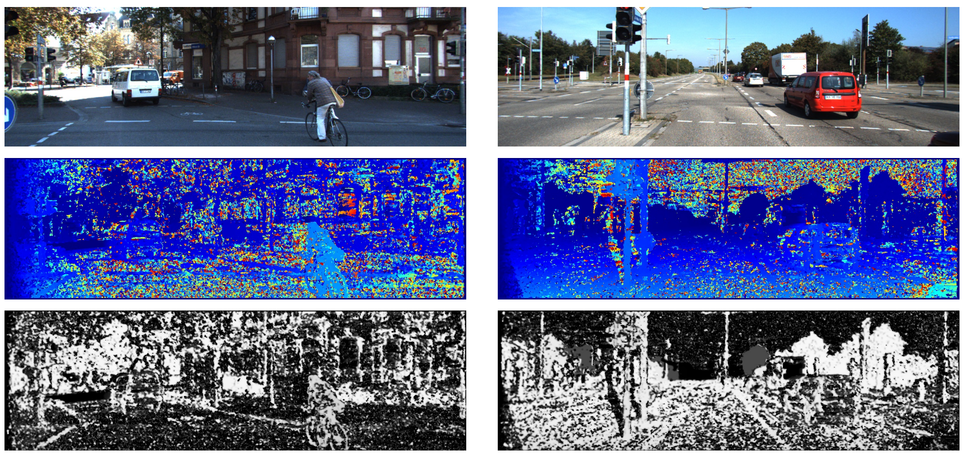


Figure 5.7: Qualitative results for CCNN, AD-CENSUS algorithm, KITTI 2015 dataset.

in only 2 cases. The evaluation on Middlebury v3 confirms that, compared to Park & Yoon, CCNN better generalizes to a different algorithm for the reason reported in the previous section.

As for the KITTI 2015 dataset, we provide experimental results with a lower number of convolutional kernels (i.e., 32 and 48 for each convolutional layers). Processing AD-CENSUS disparity maps, the network with 32 kernels achieves an average AUC of 0.367, with 48 kernels 0.159 and 0.107 with the final network with 64 kernels. On the disparity maps provided by SGM, we report an average AUC of 0.233 with 32 kernels, 0.117 with 48 kernels and 0.079 with the proposed network. These results confirm the trend previously reported on the KITTI 2015 dataset; a deeper analysis of this behaviour is left to future research.

Finally, Figure 5.6 and 5.7 depicts some examples of confidence maps generated by the proposed CCNN with 64 kernels outlined in Figure 5.3, respectively, on the Middlebury dataset with the SGM algorithm and on KITTI 2015 dataset with the AD-CENSUS algorithm.

5.4 Conclusions

In this work, arguing that disparity assignments can be classified according to recurrent patterns detectable in the disparity map, we have proposed a novel confidence measure CCNN based on a deep architecture. Experimental results, including a cross validation on different datasets, clearly confirm that our proposal outperforms state-of-art. On a GPU, CCNN delivers confidence maps at almost 9 fps. Moreover, not being based on cost volume analysis, it is more independent of the particular stereo algorithm deployed for training and also suited for out-of-the-box stereo vision systems. To the best of our knowledge, this is the first method that allows to infer from scratch, using as input cue only the disparity map, an effective confidence measure exploiting a CNN.

Chapter 6

Review and evaluation of confidence measures in a machine learning world

The content of this chapter has been presented at the IEEE International Conference on Computer Vision (ICCV 2017) - ”Quantitative evaluation of confidence measures in a machine learning world”. Most relevant to the work shown in this chapter are the following papers: [32, 21, 80, 62, 65, 66, 75, 94, 96, 24].

6.1 Introduction

Hu and Mordohai [32] exhaustively reviewed in 2012 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 [73] 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 [21, 80, 62, 65] and deep learning [66, 75]
- Much larger datasets with ground-truth depicting very challenging and realistic scenes acquired in indoor [71] and outdoor environments [14, 56]

- Novel and more effective stereo algorithms, some leveraging on deep learning techniques [96, 54], more and more often coupled with confidence measures [75, 65, 62]. Moreover, in recent years, SGM [24] became the preferred disparity optimization method for most state-of-the-art stereo algorithms (e.g., [96, 75])

Considering these facts, we believe that this field deserves a further and deeper analysis. Therefore, in this work we aim at i) extending and updating the taxonomy provided in [32] including novel confidence measures and in particular those based on machine learning techniques, ii) exhaustively assessing their performance on the larger and much more challenging datasets [56, 71] 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 (K12), (K15) and (Mv3) available today.

6.2 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 [32], 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.

6.2.1 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 pixel, the *matching score measure* **(MSM)** [32] 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)** [32] replaces c_{2m} with c_2. *Non linear margin* **(NLM)** [20] 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) [32]** and *local curve* **LC** [89] 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)** [27, 32] computes the ratio between c_{2m} and c_1. In one of its variants, *peak ratio naive* **(PKRN)** [32], c_{2m} is replaced with the second minimum c_2. In *average peak ratio* **(APKR)** [36] 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)** [37] 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 [21]. *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 [24]. 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 pixel under examination, by P1 when the difference is 1 and by P2 ($>$P1) otherwise.

6.2.2 Analysis of the entire cost curve

Differently from previous confidence measures, those belonging to this category analyze for each pixel the overall distribution of matching costs. *Perturbation* **(PER)** [21] measures the deviation of the cost curve to an ideal one. *Maximum likelihood measure* **(MLM)** [49, 32] and *attainable maximum likelihood* **(AML)** [57, 32] 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 AML. *Number of inflections* **(NOI)** [44] determines the number of local minima in the cost curve while *local minima in neighborhood* **(LMN) [36]** counts, on a patch, the number of points with local minimum at the same disparity d_1 of the examined pixel. *Winner margin measure* **(WMN)** [32] normalizes for each pixel the difference between c_{2m} and c_1 by the sum of all costs while its variant *winner margin measure naive* **(WMNN)** [32] adopts the same strategy replacing c_{2m} with c_2. Finally, *negative entropy measure* **(NEM)** [72, 32] relates the degree of uncertainty of each pixel to the negative entropy of its matching costs.

6.2.3 Left-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)** [11, 32], that assigns as confidence the negation of the absolute difference between the disparity of a pixel in L and its homologous one in R, and *left-right difference* **(LRD)** [32] 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 pixel in R. We include in this category *zero-mean sum of absolute differences* **(ZSAD)** [21] 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)** [58] and *uniqueness constraint* **(UC)** [10] 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.

6.2.4 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)** [80, 62] determines for each pixel the distance to the supposed closest depth boundary while, for the same purpose, *disparity map variance* **(DMV)** computes the disparity gradient module [21]. Remaining confidence measures belonging to this category extract features on a patch centered on the examined pixel. *Variance of disparity* **(VAR)** [62, 65] computes the disparity variance, *disparity agreement* **(DA)** [65] counts the number of points having the same disparity of the central one, *disparity deviation from median* **(MDD)** [80, 62, 65] computes the difference between disparity and its median and *disparity scattering* **(DS)** [65] encodes the number of different disparity assignments on the patch.

6.2.5 Reference image features

Confidence measures belonging to this category use as domain only the reference image. *Distance to border* **(DB)** [80, 62] 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)** [21, 62] 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.

6.2.6 Image distinctiveness

The idea behind these confidence measures is to exploit the notion of distinctiveness of the examined pixel within its neighborhoods along the horizontal scanline of the same image. *Distinctiveness* **(DTS)** [47, 32] exactly leverages on such definition by assuming as confidence for a given pixel the lowest *self-matching* cost computed within a certain prefixed range excluding the one under examination. *Distinctive similarity measure* **(DSM)** [93, 32] assigns as confidence value to a given pixel 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 pixel, divided by the square of c_1 [32] or c_1 [93]. For a given pixel the *self-aware matching measure* **(SAMM)** [59, 32] 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.

6.2.7 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 [21, 80, 62, 65] or deploying for the same purpose deep learning architectures [75, 66]. 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 [60, 86], to infer the degree of uncertainty of disparity assignments.

Random forest approaches

In this category a seminal approach is represented by *ensemble learning* **(ENS$_c$)** [21]. 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) [80]** 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 **(LEV) [62]** 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) [65]** 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$_r$, in GCP and LEV the feature vector is fed to a random forest trained in regression mode. We conclude this section observing that ENS [21] and LEV [62] 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 [21] and [62], 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)** [75] 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. Confidence map computation is pretty demanding, because each patch need to be recomputed from scratch for each pixel. 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) [66]**. In fact, in this approach confidence prediction is regressed by a CNN without manually 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 [65, 66].

6.2.8 SGM specific

This category groups two approaches intrinsically related to SGM [24]. 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)** [48] combines the cues available in the cost curve before and after semi-global optimization, while *sum of consistent scanlines* **(SCS)** [25] counts for each pixel the number of scanlines voting for the same disparity assigned by the full SGM pipeline.

6.3 Evaluation protocol and 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

(a)

	K12 (ε = 38.82%)			K15 (ε = 35.41%)			Mv3 (ε = 37.78%)		
Category	measure	rank	AUC	measure	rank	AUC	measure	rank	AUC
6.2.1	$APKR_{11}$	4^{12}	0.1806	$APKR_{11}$	4^{12}	0.1541	$APKR_{11}$	4^{7}	0.1355
6.2.2	WMNN	7^{34}	0.2215	WMN	7^{34}	0.2024	WMN	6^{23}	0.1579
6.2.3	LRD	5^{20}	0.1946	LRD	6^{28}	0.1825	LRD	5^{21}	0.1519
6.2.4	DA_{11}	3^{8}	0.1668	DA_{11}	3^{7}	0.1399	DA_{11}	3^{4}	0.1294
6.2.5	DB	8^{65}	0.3446	DB	8^{66}	0.3103	DLB	8^{69}	0.3333
6.2.6	SAMM	6^{25}	0.2030	SAMM	5^{20}	0.1715	DSM	7^{40}	0.1798
6.2.7	O1	2^{3}	0.1309	O1	2^{3}	0.1128	O1	2^{3}	0.1211
6.2.7	**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 6.2.7 and 6.2.7			
Measure	K12	K15	Mv3
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 (ε = 17.10%)			K15 (ε = 15.37%)			Mv3 (ε = 26.70%)		
Category	measure	rank	AUC	measure	rank	AUC	measure	rank	AUC
6.2.1	$APKR_{11}$	4^{11}	0.0566	$APKR_{11}$	4^{11}	0.0508	$APKR_{11}$	3^{5}	0.0728
6.2.2	WMN	6^{30}	0.0748	WMN	6^{31}	0.0654	WMN	4^{13}	0.0763
6.2.3	LRD	7^{31}	0.0748	LRD	7^{32}	0.0712	UCC	5^{22}	0.0896
6.2.4	DS_{9}	3^{8}	0.0542	DS_{9}	3^{8}	0.0477	DS_{11}	6^{35}	0.1061
6.2.5	DLB	8^{66}	0.1543	HGM	8^{67}	0.1439	DLB	8^{68}	0.2260
6.2.6	SAMM	5^{16}	0.0598	SAMM	5^{21}	0.0557	DSM	7^{40}	0.1228
6.2.7	O1	2^{2}	0.0317	O1	2^{2}	0.0324	O1	2^{3}	0.0680
6.2.7	**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 6.2.7 and 6.2.7			
Measure	K12	K15	Mv3
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 (ε = 16.78%)			K15 (ε = 13.68%)			Mv3 (ε = 25.91%)		
Category	measure	rank	AUC	measure	rank	AUC	measure	rank	AUC
6.2.1	$APKR_{11}$	3^{7}	0.0492	$APKR_{11}$	3^{7}	0.0457	$APKR_{9}$	2^{2}	0.0739
6.2.2	WMN	4^{11}	0.0554	WMN	5^{12}	0.0502	WMN	4^{8}	0.0.779
6.2.3	UCC	6^{21}	0.0735	UCC	6^{19}	0.0640	UCC	6^{23}	0.0959
6.2.4	DS_{11}	5^{12}	0.0554	DS_{11}	4^{11}	0.0501	DS_{11}	5^{13}	0.0884
6.2.5	DB	9^{67}	0.1378	DB	9^{68}	0.1265	DLB	9^{70}	0.2157
6.2.6	DSM	7^{36}	0.0811	DSM	7^{28}	0.0679	DSM	7^{32}	0.1041
6.2.7	LEV	2^{2}	0.0358	O1	2^{2}	0.0323	O1	3^{6}	0.0777
6.2.7	**CCNN**	$\mathbf{1^{1}}$	**0.0358**	**CCNN**	$\mathbf{1^{1}}$	**0.0302**	**CCNN**	$\mathbf{1^{1}}$	**0.0736**
6.2.8	SCS	8^{41}	0.0851	SCS	8^{48}	0.0790	SCS	8^{36}	0.1080
Optimal			0.0227			0.0184			0.0431

Categories 6.2.7 and 6.2.7			
Measure	K12	K15	Mv3
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 6.1: AUC analysis for top-performing measures of each category on AD-CENSUS, MC-CNN and SGM algorithms on KITTI 2012, KITTI 2015 and Middlebury v3 datasets.

Mv3 (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 in LEV and O1 features) being these scales effective according to [62, 65]. 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. In Section 6.3.1 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 [32] and widely adopted in this field [21, 80, 62, 65, 66, 75]. Moreover, since machine learning is the key technology behind most recent approaches, in Section 6.3.2 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 [80, 62, 65, 75], in Section 6.3.3 we assess the performance of the most effective confidence measures when plugged in one of such state-of-the-art methods [62].

6.3.1 Detection of correct matches

The ability to distinguish correct disparity assignments from wrong ones is the most desirable property of a confidence measure. To quantitatively measure this we follow the AUC protocol defined by Hu and Mordohai [32] to evaluate the 76 confidence measures on K12, K15 and Mv3 with AD-CENSUS, MC-CNN, 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 [62] and adopting the same termination criteria (e.g., linked to the maximum 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 [66, 75]. 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 6.3.2. 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 Mv3[1] as suggested in the corresponding papers.

In Table 6.1 we summarize results in terms of AUC averaged on each dataset (K12, K15 and Mv3) 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 6.2 and its absolute ranking and, on the right table, the absolute ranking for confidence measures based on machine learning. Observing tables 6.1 (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 6.2.7 and 6.2.7, we can notice that in most cases PBCP, O1 and LEV perform very well with the exception of the SGM algorithm and Mv3 (Table 6.1(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 6.1 (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 Mv3. 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 Mv3 and SGM as already highlighted. Other interesting confidence measures are those belonging to category 6.2.4 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 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

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

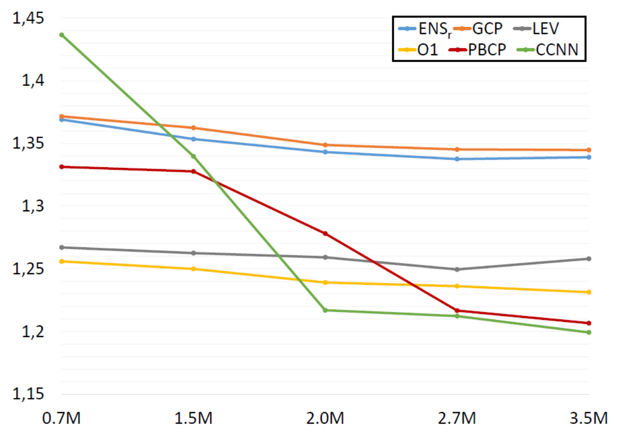


Figure 6.1: Impact of the amount of training data on AUC values.

category 6.2.2, 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 6.2.3 and 6.2.6, such measures on the three datasets do not grant reliable confidence prediction.

6.3.2 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,

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

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 in Section 6.3.1 to compare the behavior of the same forest fed with different feature vectors when more samples are available. Figure 6.1 reports Δ_k, as a function of the number of training samples, for the best six measures 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 Mv3, processing an almost equivalent amount of training samples with respect to the training configuration adopted in Section 6.3.1. Then, we compared the results achieved with this configuration to the one used in Section 6.3.1 with AD-CENSUS on the remaining data from Mv3, computing Δ_k as defined in Equation 6.1. A confidence measure achieving similar Δ_k in the two configuration is able to generalize well between the two very different scenarios. Figure 6.2 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.

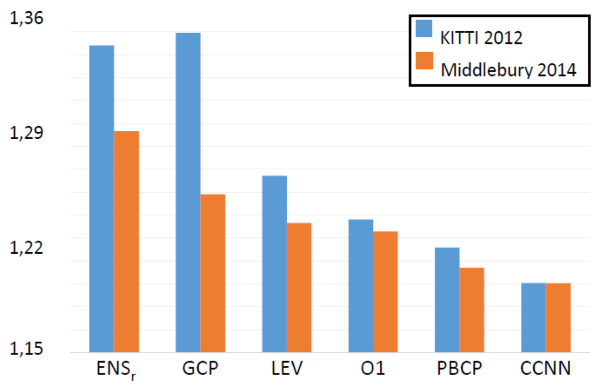


Figure 6.2: Impact of training on different datasets.

6.3.3 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., [80, 62, 65, 75]). For this evaluation we choose the cost modulation proposed by Park and Yoon [62]. The reason is that differently from [65], which is specific for SGM algorithm, and [80, 75], based on parameters potentially different from measure to measure, [62] is suited for any stereo algorithm and parameter-free. SGM was tuned as reported in Section 6.3.1. We plugged in [62] the machine learning based measures, as well as three standalone measures (i.e., APKR, SAMM and DA$_{11}$). On the three datasets K12, K15 and Mv3, from Table 6.2 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 Mv3. 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 is observed in terms of AUC.

	K12		K15		Mv3	
	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 6.2: Absolute improvement of disparity accuracy yielded by cost modulation [62] using different measures, on KITTI 2012, KITTI 2015 and Middlebury v3 datasets.

6.4 Conclusions

In this work 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.

Chapter 7

Efficient confidence measures for embedded stereo

The content of this chapter has been presented at the 19th International Conference on Image Analysis and Procesing (ICIAP 2017) - ”Efficient confidence measures for embedded stereo”. Most relevant to the work shown in this chapter are the following papers: [32, 94, 24].

7.1 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. Given evidence of the benefits granted by using passive techniques for this task, many custom devices deploy popular stereo matching algorithms, some of them particularly suited for hardware implementation, thus enabling the design of compact, low-powered and real-time depth sensors [29, 53, 3, 13, 78, 74, 87]. Nevertheless, it is essential to detect when the camera fails and to filter-out unreliable pixels 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 our work proved that confidence prediction can also leverage on CNNs [66], [75]. 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) [73]).

Therefore, in this work 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.

7.2 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 [2]. Fixed point represents an efficient and hardware-friendly way to express and manipulate fractional numbers with a fixed number of bits [2]. 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 negligible 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 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 pre-computed 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.

7.3 Confidence measures suited for hardware implementation

In this section we describe the pool of confidence measures from the literature suited for implementation on target embedded devices. Figure 7.1 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 pixel 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

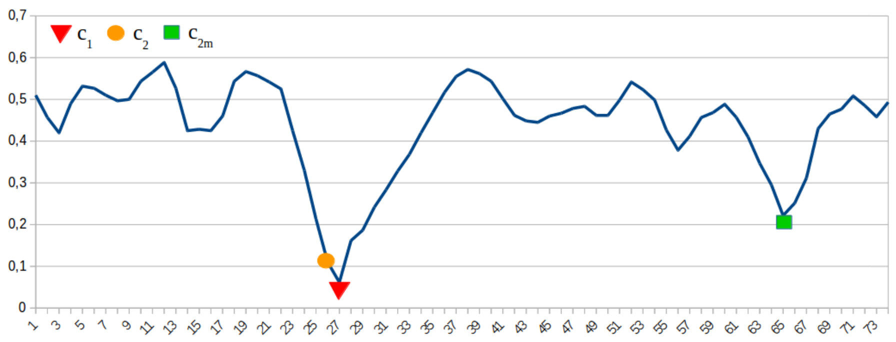


Figure 7.1: Example of cost curve.

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 considered 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.

7.3.1 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) [32] 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) [32] 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) [32] sets as confidence the negation of the absolute difference between the disparity of a pixel in L and its homologous one in R. This method represents one of the most widely adopted strategy by most algorithms even for those implemented on embedded devices. Another popular and more efficient strategy based on a single matching phase is the *uniqueness constraint* (UC) [10]: it assumes as poorly confident those pixels colliding on the same pixel of the target image (R) with the exception of the one having the lowest c_1. *Curvature* (CUR) [32] and *local curve* (LC) [89] 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) [32] simply counts the number of local minima in the cost curve assuming that the lower, the more confident is the disparity assignment.

7.3.2 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) [32] which computes the ratio between c_{2m} and c_1 and its variant *peak ratio naive* (PKRN) [32] 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) [32] which normalizes the difference between c_{2m} and c_1 by the sum of all costs. Its variant *winner margin measure naive* (WMNN) [32] follows the same strategy replacing c_{2m} with c_2. The *left-right difference* measure (LRD) [32] computes the difference between c_2 and c_1 divided by the absolute difference between c_1 and the minimum cost of the homologous pixel 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) [32] and *attainable maximum likelihood* (AML) [32] 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 computationally demanding approach *perturbation* (PER) [21], 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) [32] 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.

7.4 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 AD-CENSUS and SGM, well-suited for implementation on embedded systems.

We encode matching costs with, respectively, 6 and 8 bit integer values, being this amount enough to encode the entire ranges. We follow the AUC evaluation protocol [32] setting, as reported on Middlebury v3 and KITTI 2015 benchmarks, a threshold value on disparity error respectively of 1 and 3 for the two datasets following the guidelines. 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 [29, 3] and UC [53, 3] for confidence estimation.

7.4.1 Experimental evaluation on Middlebury v3 and KITTI 2015

In this section we report results on Middlebury v3 and KITTI 2015 datasets in terms of average AUC values achieved by confidence measures implemented in software. For hardware challenging measures of section 7.3.2 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

measure	standard	measure	standard
Opt.	0.08891	Opt.	0.08891
CUR	0.24377 (14)	AML	0.21173 (11)
LRC	0.19933 (7)	LRD	0.17004 (3)
LC	0.24377 (15)	MLM	0.22413 (12)
MM	0.17765 (6)	PER	0.20687 (9)
MMN	0.19933 (8)	PKR	0.16250 (1)
MSM	0.23182 (13)	PKRN	0.17185 (5)
NOI	0.39053 (16)	WMN	0.16503 (2)
UC	0.20974 (10)	WMNN	0.17169 (4)

(a)

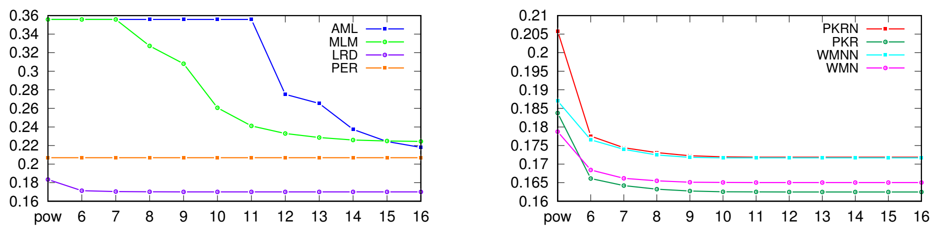
measure	standard	measure	standard
Opt.	0.04367	Opt.	0.04367
CUR	0.11602 (11)	AML	0.08843 (3)
LRC	0.16853 (15)	LRD	0.11725 (13)
LC	0.11602 (12)	MLM	0.09567 (6)
MM	0.09371 (5)	PER	0.08766 (1)
MMN	0.12920 (14)	PKR	0.08813 (2)
MSM	0.10181 (7)	PKRN	0.10527 (10)
NOI	0.32028 (16)	WMN	0.08898 (4)
UC	0.10347 (9)	WMNN	0.10232 (8)

(b)

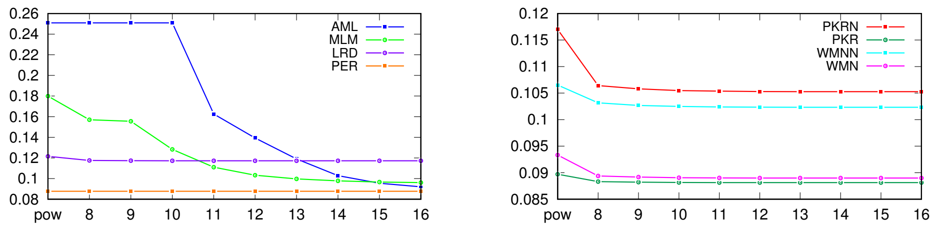
Table 7.1: AUC analysis, comparison between hardware friendly and challenging measures for AD-CENSUS (a) and SGM (b) algorithms on Middlebury v3 dataset.

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 7.1 shows for Middlebury v3 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, 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 out that top-performing confidence measures always belong



(a)



(b)

Figure 7.2: AUC analysis for different implementations of hardware challenging measures for AD-CENSUS (a) and SGM (b) algorithms on Middlebury v3 dataset. On x axis, different number of bits for fixed-point representation, on y AUC values.

to the hardware challenging category.

Therefore, in Figure 7.2 we report the performance of hardware challenging confidence measures, on Middlebury v3 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 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 on the number of bits. On the other hand, AML and MLM are significantly affected by the number of bits deployed for their implementation 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.

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)

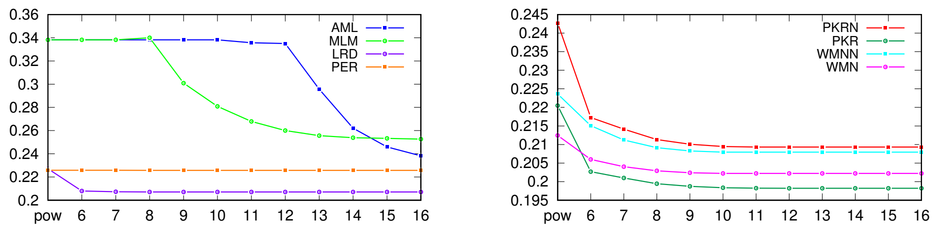
(b)

Table 7.2: AUC analysis, comparison between hardware friendly and challenging measures for AD-CENSUS and SGM algorithms on KITTI 2015 dataset.

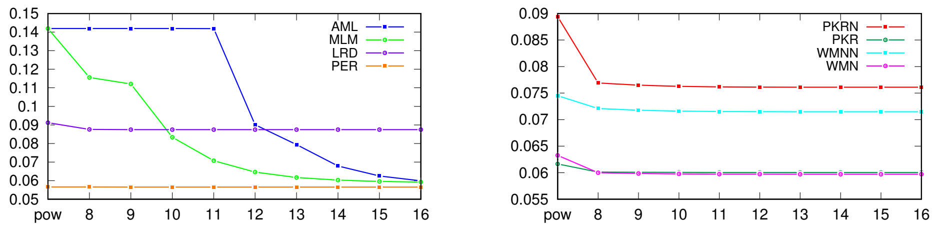
Table 7.2 reports the average AUCs for the two considered stereo algorithms on KITTI 2015 for software implementation of the 16 confidence measures. Compared to Table 7.1 we can notice a similar behavior with a notable difference. In fact, observing Table 7.2 we highlight that LRC achieves almost optimal results on AD-CENSUS but yields very poor performance with SGM. Looking at the behavior of the hardware challenging measures, reported in Figure 7.3, we observe on KITTI 2015 a substantially similar behavior with respect to Figure 7.2 concerned with Middlebury v3.

7.5 Conclusions

In this work 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



(a)



(b)

Figure 7.3: AUC analysis for different implementations of hardware challenging measures for AD-CENSUS and SGM algorithms on KITTI 2015 dataset.

evaluation on Middlebury v3 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 8

Exploiting local consistency for confidence estimation

The content of this chapter has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017) - ”Learning to predict stereo reliability enforcing local consistency of confidence maps”. Most relevant to the work shown in this chapter are the following papers: [32, 21, 80, 62, 65, 66, 94, **?**].

8.1 Introduction

After proposing and evaluating new confidence measures, establishing a new state-of-the-art in this field, we want to inquire about the possibility to further improve the outlier detection capability of confidence measures. In particular, we investigate whether a machine learning framework could be used to improve the effectiveness of confidence measures exploiting local consistency, leveraging on the information available within nearby pixels, as assumed by most computer vision algorithms. To this end, given an input confidence measure, our framework analyzes the local behavior of the examined confidence measure by means of a CNN, trained on a subset of a dataset with ground-truth, to provide a more meaningful estimation. Specifically, by learning informative patterns on confidence maps, the network is able to infer from local patches a new estimation as shown in Figure 8.1. In our

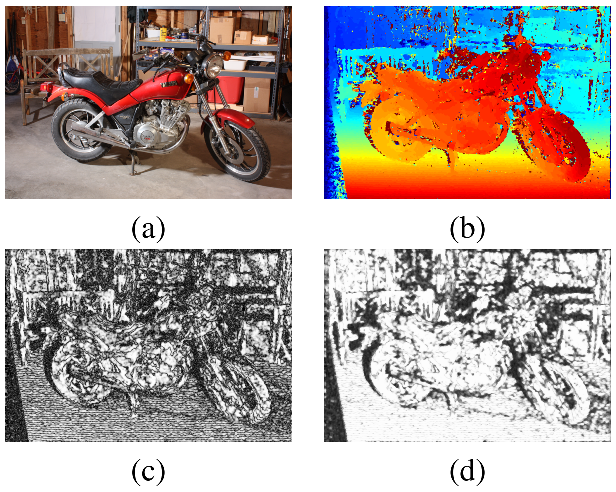


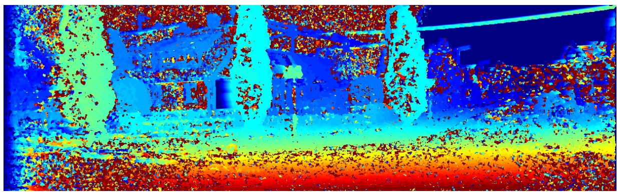
Figure 8.1: Qualitative results of confidence maps obtained by PKRN and PKRN+.

experimental evaluation, we consider 23 state-of-the-art confidence measures and, once trained the networks on 25 out of 194 images of the KITTI 2012 training dataset, we assess the improvements yielded by our method on the remaining images of the dataset. Moreover, without re-training the networks, we perform a further cross-validation on KITTI 2015 and Middlebury v3. This extensive evaluation shows that exploiting local consistency enables to dramatically improve all the 23 state-of-the-art confidence measures, including those based on machine learning, on all considered datasets and even dealing with image contents never *seen* before (e.g., on Middlebury v3 dataset).

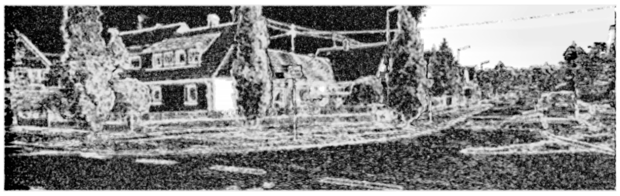
8.2 Proposed method

This work aims at improving the reliability of standalone confidence measures, learning from their local behavior effective informative patterns making the assumption that, as for most computer vision algorithms, context matters. Figure 8.1 highlights the motivations behind our proposal. Observing the two reported confidence maps, namely M$_{MLM}$ and M$_{LRC}$ [32], we can notice that, for the same disparity map, the two measures show very different information contents and local behavior. Considering that the reference image and the disparity map are locally consistent, we expect a similar behavior for the confidence maps. These observations lead us to the idea that each confidence measure exposes specific local patterns that can be identified with an *ad hoc* training. To this end we leverage on a deep network, appropriately trained on a dataset with ground-truth, aimed at learning

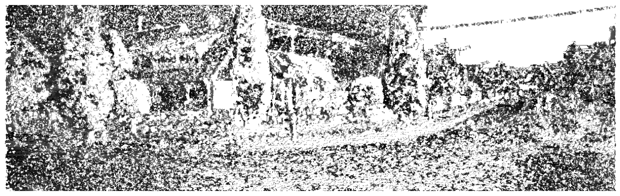
(a)



(b)



(c)



(d)

Figure 8.2: Qualitative results of confidence maps obtained by MLM and LRC.

and detecting effective informative patterns for each examined confidence measure. Exhaustive experimental results on challenging stereo pairs confirm that the proposed strategy enables to dramatically improve the effectiveness of state-of-the-art confidence measures.

8.2.1 Enforcing local consistency

A confidence measure k assigns a value to a pixel p of the disparity map computed with respect to the reference image according to C_k, a function taking as arguments one or more of the following cues: the matching cost curve c, reference left L and right image R of the stereo pair, the disparity maps D_L and D_R obtained, respectively, using as reference L and R.

$$C_k(p) = f(c(p), L, R, D_L, D_R) \tag{8.1}$$

Excluding more recent approaches based on machine-learning, a conventional confidence measure can be obtained [32] analyzing matching costs, local properties of the cost curve or of the entire curve, local minima, consistency between left and right disparity maps and distinctiveness among image pixels. Typically, a more complex analysis allows to achieve a more accurate correctness prediction. For example, the Matching Score Measure (MSM) [32], which is the simplest confidence measure, only relies on the minimum matching cost value. It has been adopted as baseline method, showing that most of the other confidence measures outperform it [32]. Another one based on very simple analysis is the Left-Right Consistency (LRC) [32], aimed at detecting inconsistent points between left and right disparity maps. This measure performs very well near depth discontinuities, and is mainly useful to detect occluded pixels. However, it is not very informative due to its discretized nature. Both measures typically fail in presence of some well-known issues of stereo matching, such as low textured areas or repetitive patterns, where multiple local minima concurring to the role of minimum would yield to high confidence according to MSM. Similarly, the absence of discontinuities might lead LRC, to label a pixel as confident even if it has wrong disparities on both maps.

In our proposal, in order to predict the correctness of a disparity assignment enforcing the locality constraint, it is useful to encode match reliability with a confidence map. That is, given a confidence measure k, for each pixel p belonging to the reference image L, the confidence map $M_k \in [0, 1]$ is obtained as follows:

$$M_k(p) = \frac{C_k(p) - \min_{p \in L} C_k(p)}{\max_{p \in L} C_k(p) - \min_{p \in L} C_k(p)} \tag{8.2}$$

Observing confidence maps we can notice that some measures apparently do not show distinctive patterns, looking like noisy images to human observers. Conversely, some others clearly present such distinctive patterns, related to particular features of the disparity map. Starting from these observations, we assume that local properties of confidence maps can be exploited to improve their reliability with respect to their original counterpart by learning specific image patterns of each measure. Such properties, within the neighborhood of a pixel p, are sought in the confidence map M_k analyzing a $N \times N$ patch centered on p with a CNN, trained to infer a new confidence estimation for the examined point.

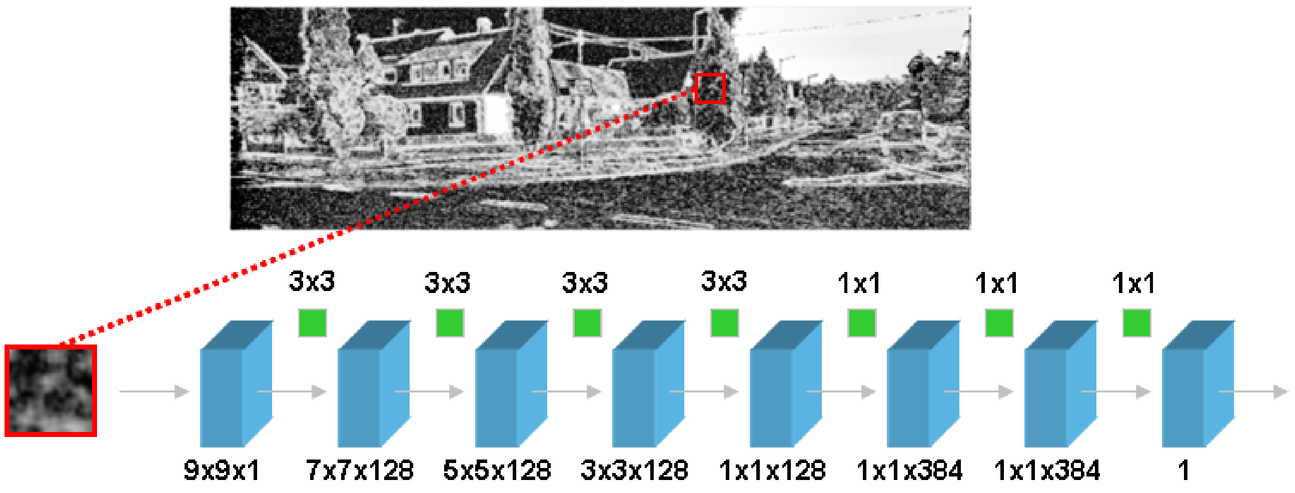


Figure 8.3: Proposed CNN architecture to prediction match reliability enforcing local consistency on the input confidence map.

8.2.2 Deep network architecture

To learn a locally consistent confidence prediction, we propose to train a custom CNN to assign the new value for the pixel under investigation, using image patches extracted from confidence maps. For this purpose we rely on a deep network architectures structured as in Figure 8.3.

In order to infer the final pixel-wise confidence score, in our experiments we evaluated different CNN architectures made of different convolutional layers, depending on the receptive field of the network, and fully-connected layers. Convolutional layers extracts f feature maps by applying 3×3 kernels from the input feature maps fed by the previous layer, fully-connected containing n neurons. The single final neuron is in charge of the regression stage. Each layer is followed by activation operators, in particular we used Rectifier Linear Units (ReLU) and we applied a Sigmoid operator on the output of the last neuron. Following the successful deployment of CNNs for stereo [96] and confidence estimation [66], we chose convolutional kernels of fixed 3×3 size and we did not include any pooling operator. The remaining hyper-parameters of our architecture, such as the size of the receptive field and the number of neurons, have been tuned during the experimental phase. Given a patch of size $N \times N$, referred to as $P^{N \times N}_{M_k(p)}$, extracted from a confidence map M_k centered on pixel p, the value predicted by the network is:

$$M_{k^+}(p) = F(P_{M_k}^{N\times N}(p)) \in [0,1] \tag{8.3}$$

where $F(P_{M_k(p)}^{N\times N})$ is the output of the network processing $P_{M_k(p)}^{N\times N}$. According to this terminology, we will refer, for example, to the learned version of the PKRN confidence measure as PKRN$^+$ (PKRN *plus*).

In testing, after the network has been trained, we replace the fully-connected layers with convolutional layers made of 1×1 kernels. This new model is functionally identical to the one used for training but, with the same network, it allows to process input of different size enabling a single forward pass of the full resolution confidence map M_k rather than forwarding all the single $P_{M_k}^{N\times N}$ patches. This strategy greatly reduces the time required to obtain the final confidence map M_{k^+}. The absence of pooling allows us to maintain full resolution output by applying zero-padding to the original M_k according to the size of the receptive field.

8.3 Experimental results

In this section we describe in detail the methodology adopted for the training phase on a subset of the KITTI 12 [14] dataset. Then, we compare, on KITTI and Middlebury datasets, the learned confidence measures to their original counterparts[1]. In particular, we evaluate the performance in terms of correctness prediction by analyzing the Area Under Curve (AUC) [32] on the remaining images of the KITTI 12 [14] dataset as well as on the whole KITTI 15 [56] and Middlebury 14 [73] datasets without re-training the networks.

Since the ground-truth is required for training and for AUC evaluation, as common in this field [21, 62, 66, 65], for each considered dataset we rely on the evaluation training sets of KITTI 12 (194 images, 25 for training and 169 for testing), KITTI 15 (200 images) and Middlebury 14 (15 images). Moreover, we compute confidence measures according to the output of two algorithms: AD-CENSUS, aggregating matching costs (computed with the Hamming distance on 5×5 census

[1]Source code and trained networks available on the author's website

transformed image patches) on a fixed support region of size 5×5, and MC-CNN fast [96].

8.3.1 Training phase

For each confidence measure we trained the CNN, on a subset of the KITTI 12 dataset, according to *stochastic gradient descend*, in order to minimize the *binary cross entropy*, with batch size set to 128 patches. Each network ran 15 training *epochs* with a *learning rate* equal to 0.003, reduced by a factor 10 after the 11^{th} epoch, a *momentum* of 0.9 and shuffled the training examples before the training phase. Network models and training phase have been implemented with the Torch 7 framework [8].

In our experiments we tested different amounts of training data to generate learned confidence maps and we achieved the best results considering 25 stereo images (i.e., from frame 000000 to 000024) of the KITTI 12 dataset [14]. Increasing the training set did not improve noticeably the quality of the learned confidence measures. From these 25 frames, we extracted patches centered on pixels with available ground-truth, obtaining approximatively 2.7 million samples for each confidence measure. Patches centered on points having a disparity error ≤ 3 (following the threshold suggested in [14, 56]) are labeled as $confident$ and encoded as ones, the remaining as zeros.

In our evaluation we considered 18 state-of-the-art stand-alone confidence measures and 5 approaches based on machine-learning. The first group includes: Matching Score Measure (MSM), Peak Ratio (PKR) and Peak Ratio Naive (PKRN), Winner Margin (WMN) and Winner Margin Naive (WMNN), Negative Entropy Measure (NEM), Number Of Inflection points (NOI), Maximum Margin Naive (MMN), Maximum Likelihood Measure (MLM), Attainable Maximum Likelihood (AML), Curvature (CUR), Local Curve (LC), Left Right Consistency (LRC), Left Right Difference (LRD), Distinctive Similarity Measure (DSM), Uniqueness Constraint (UC), Self-Aware Matching Measure (SAMM) and Perturbation (PER). Excluding PER [21], UC [10] and LC [89] the other confidence measures have been reviewed in [32]. Regarding the specific parameter setting, we set $\sigma_{MLM} = 0.3$ and $\sigma_{AML} = 0.1$ as suggested in [32]), $s_{PER} = 120$, $\gamma = 480$ for LC as suggested in [21]. SAMM has been computed in its symmetric version, within the range $[-\frac{d_{max}}{2}, \frac{d_{max}}{2}]$, as suggested by the authors.

Regarding confidence measures based on machine-learning we considered Ensemble [21] (in its more effective configuration with 23 features), GCP [80], Park [62] (in its more effective configuration with 22 features) and the two methods proposed in [65] and [66] referred, to as, respectively, *O1* and *CCNN*. We implemented these 5 approaches following exactly the guidelines reported in each paper and trained, as for our proposal, each one on the same 25 images of the KITTI 12 dataset. Before being fed to the deep network, each confidence map was normalized according to equation 8.2.

The AUC values reported in Section 8.3.2 and Section 8.3.3 for AD-CENSUS and in Section 8.3.4 for MC-CNN were obtained tuning the previously described hyper-parameters of our network as follows: 9×9 receptive field, $f = 128$ kernels per convolutional layer, $n = 384$ neurons (i.e. 1×1 kernels at test time) per fully-connected layer. The 9×9 receptive field enabled to achieve on average the best performance. The resulting CNN architecture has more than 600 thousand parameters and, with a full resolution confidence map of the KITTI dataset, it requires just 5 GB of memory and about 0.1 sec to infer a new confidence estimation with a Titan X GPU.

Finally, we stress the fact that in our experimental evaluation we performed a single training procedure on 25 images of the KITTI 12 dataset even when dealing with different datasets (ı.e, KITTI 15 and Middlebury 14) and the remaining 169 images of KITTI 12.

8.3.2 Evaluation of stand-alone confidence measures

We assess the effectiveness of confidence measures performing AUC analysis. Figure 8.4 summarizes the experimental results with AD-CENSUS on the 3 datasets involved in our evaluation. On the left we report results concerning the KITTI 12 dataset (the remaining 169 stereo pairs out of 194, being 25 used for training), in the middle concerning KITTI 15 dataset (200 stereo pairs, none involved in training), on the right concerning Middlebury 14 dataset (15 stereo pairs, none involved in training). Given a confidence measure k belonging to the pool of 18 stand-alone measures considered, two bars are depicted, related to the average AUC achieved by the original measure (AUC_k, in blue) and the one obtained after being processed by our framework (AUC_k^+, in green). The red line represents the optimal value (AUC_{opt}). The closer the AUC is to AUC_{opt}, the more effective the confidence measure is. The charts in Figure 8.4 show that our method always improves the effectiveness of each

Confidence	KITTI 12 (169/194)			KITTI 15 (200/200)			Middlebury 14 (15/15)		
measure	AUC_k	AUC_{k+}	Δ_k	AUC_k	AUC_{k+}	Δ_k	AUC_k	AUC_{k+}	Δ_k
PKRN	0.231682	**0.187407**	35.74%	0.220458	**0.154534**	49.90%	0.152359	**0.112248**	47.76%
PKR	0.251132	**0.155664**	66.61%	0.222827	**0.134693**	65.54%	0.144349	**0.101848**	55.94%
MSM	0.274919	**0.211803**	37.77%	0.260329	**0.202062**	33.88%	0.186604	**0.166312**	17.16%
MMN	0.244250	**0.167334**	56.37%	0.236990	**0.153026**	56.49%	0.162109	**0.115097**	50.15%
WMN	0.224146	**0.148876**	64.70%	0.202390	**0.130410**	63.12%	0.127015	**0.099424**	47.05%
MLM	0.273479	**0.219593**	32.52%	0.257940	**0.204421**	31.56%	0.180903	**0.164901**	14.22%
PER	0.260978	**0.210076**	33.23%	0.240324	**0.198303**	27.65%	0.171692	**0.153460**	17.65%
NEM	0.386211	**0.314742**	25.67%	0.328761	**0.295701**	13.75%	0.307148	**0.259922**	19.78%
LRD	0.240665	**0.165342**	56.69%	0.232831	**0.150244**	57.16%	0.153181	**0.110457**	50.38%
CUR	0.355582	**0.176552**	72.25%	0.316048	**0.157221**	69.76%	0.223898	**0.123904**	64.30%
DSM	0.274579	**0.211731**	37.68%	0.260062	**0.202075**	33.77%	0.186157	**0.166489**	16.70%
AML	0.287019	**0.169239**	65.72%	0.265626	**0.155299**	62.23%	0.219605	**0.116534**	68.16%
NOI	0.419441	**0.311631**	34.59%	0.345756	**0.308789**	14.36%	0.340609	**0.276457**	23.57%
SAMM	0.204491	**0.150287**	56.06%	0.171475	**0.12176**	59.81%	0.214449	**0.133298**	55.55%
WMNN	0.223139	**0.162058**	52.96%	0.211146	**0.150363**	49.50%	0.144132	**0.109271**	46.01%
LRC	0.242911	**0.159512**	61.73%	0.218156	**0.147458**	54.47%	0.174806	**0.120645**	50.89%
LC	0.335298	**0.183496**	66.73%	0.303691	**0.164670**	64.56%	0.211085	**0.121464**	62.80%
UC	0.296917	**0.165900**	69.28%	0.263651	**0.146081**	67.07%	0.215678	**0.104459**	75.50%
Optimal	0.107802			0.088357			0.068375		

Table 8.1: Average AUC for the 18 stand-alone confidence measures on the 3 considered datasets with AD-CENSUS. Last row reports the optimal AUC. The table is split into three blocks: left block reports evaluation on KITTI 12 images excluded from training (169 frames, from 000025 to 000193), middle block reports evaluation on KITTI 15 dataset (200 frames), right block reports evaluation on Middlebury 14 dataset (15 frames). Each block contains AUC for the original measure (AUC_k), its learned counterpart (AUC_{k+}) and the improvement (Δ_k) yielded by our proposal, with respect to AUC_{opt}, computed according to equation 8.4.

confidence measure, achieving a lower AUC on all the datasets.

To perceive more clearly the benefits yielded by our framework, we report in detail the AUCs in Table 8.1. Each row is related to a single stand-alone confidence measure, the final row contains AUC_{opt} values. The table is organized into three main blocks, each one related to one of the charts shown in Figure 8.4 (left: KITTI 12, middle: KITTI 15, right: Middlebury 14). For each dataset, each row reports the original confidence measure AUC_k, the learned counterpart AUC_{k+} and the the improvement Δ_k, defined in 8.4, yielded by our frameworks with respect to the optimal AUC (i.e. AUC_{opt}, last row of the table).

$$\Delta_k = \frac{AUC_k - AUC_{k+}}{AUC_k - AUC_{opt}} \tag{8.4}$$

According to 8.4, given a confidence measure, a Δ_k = 100% improvement would be achieved by our

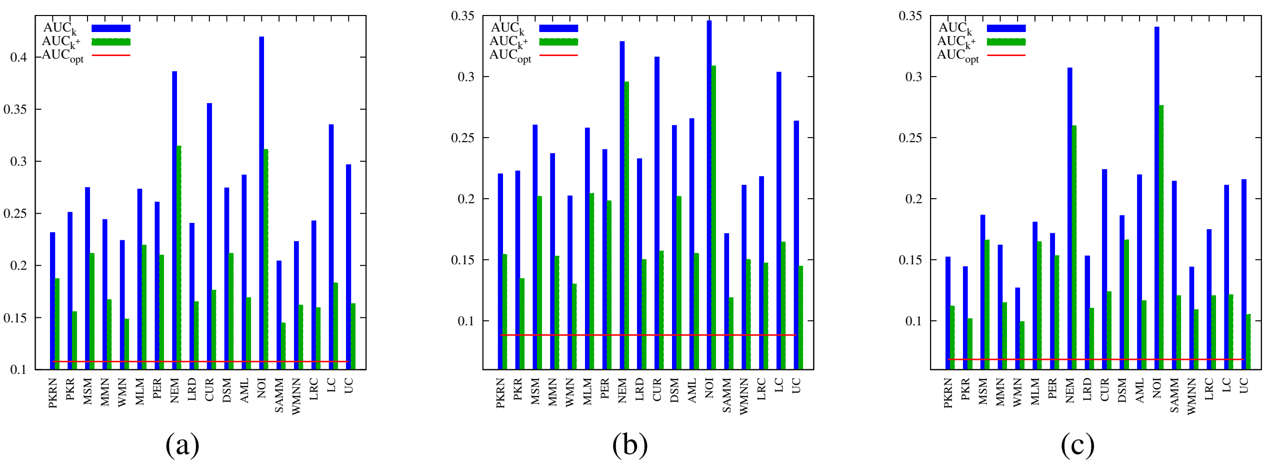


Figure 8.4: Average AUC for the 18 stand-alone confidence measures on the 3 considered datasets with AD-CENSUS. (a) Evaluation on KITTI 12 images excluded from training (169 frames, from 000025 to 000193), (b) evaluation on KITTI 15 dataset (200 frames), (c) evaluation on Middlebury 14 dataset (15 frames). In blue AUC related to the original confidence measure (e.g., AUC_{PKRN}), in green the AUC related to its learned counterpart (e.g., AUC_{PKRN^+}). The red line shows the optimal AUC value (AUC_{opt}), computed according to 8.4.

framework obtaining the optimal AUC_{opt}. Concerning the evaluation on KITTI 12 dataset, we can observe how Δ_k is always greater than 25%. In particular, the worst case is represented by NEM measure, being the AUC of NEM$^+$ 25.67% closer to AUC_{opt} with respect to the original version. For 6 measures (i.e., PKRN, MSM, MLM, PER, DSM, and NOI) our framework yields an improvement between 30% and 50% and for the remaining 11 measures we report major improvements, up to 72.25% comparing CUR with CUR$^+$. Extending the analysis to the remaining datasets, the same behavior is confirmed for all the examined confidence measures. In particular, observing the results concerning KITTI 15 dataset, NEM and NOI yield the smaller improvements, respectively with a Δ_k of 13.75% and 14.36%, PER$^+$ achieves an improvement close to 30%, 5 measures (i.e., PKRN, MSM, MLM, DSM and WMN) obtain a Δ_k between 30% and 50% and the remaining measures yield major gains, up to 69.76% deploying CUR$^+$. Finally, we report a further cross validation on Middlebury 14, the most challenging dataset being made of indoor scenes completely different from the 25 outdoor scenes of KITTI 12 *seen* during the training phase. In this case there are 6 measures (i.e., MSM, MLM, PER, NEM, DSM and NOI) with a Δ_k between 14% and 30%, PKRN, WMN and WMNN between 30 and 50% and the remaining 9 measures showing major improvements, up to 74.91% achieved of UC$^+$.

Figure 8.5 provides a qualitative comparison between PKR confidence measure and its learned coun-

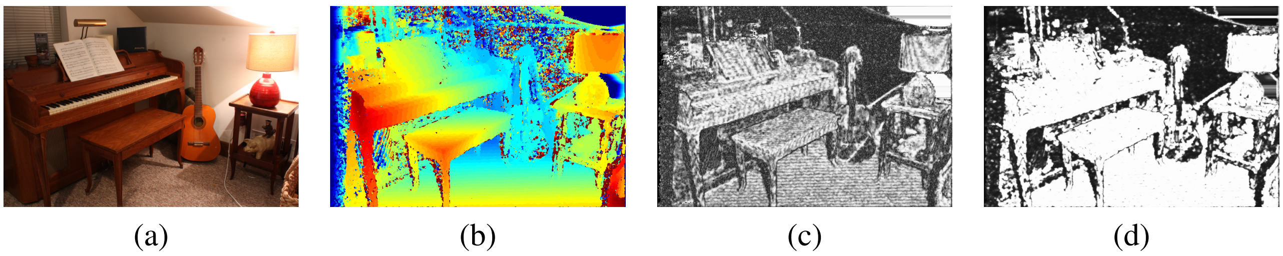


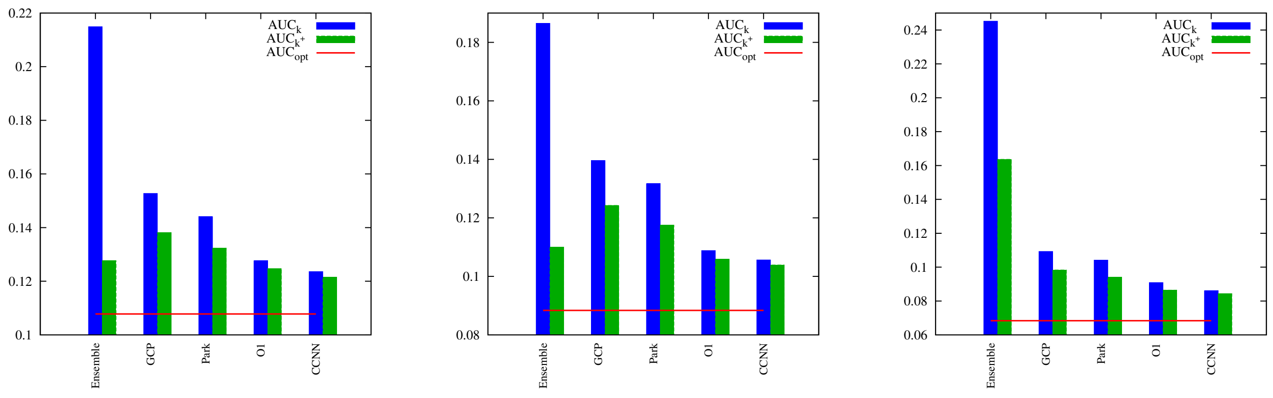
Figure 8.5: Qualitative comparison of three stand-alone confidence measures and their learned counterparts. (a) Reference image, (b) Disparity map computed by AD-CENSUS, (c) PKR and (d) learned PKR$^+$. Higher confidence values are brighter. The disparity map is encoded with colormap jet.

terparts PKR$^+$ on the *Piano* stereo pair from Middlebury 14. Observing the figure we can clearly notice the improvements yielded by our framework exploiting local consistency. Confidence values are much more smooth and consistent (e.g., the floor, the lampshade, the piano and its bench). Moreover, we can also notice how our framework can recover from gross failures of the original confidence measure (e.g., the portion of the wall at the top-right corner of the image).

8.3.3 Evaluation of confidence measures based on machine-learning

Having assessed the effectiveness of our proposal on stand-alone measures, we extended our evaluation considering 5 state-of-the-art confidence measures based on machine-learning: Ensemble [21], GCP [80], Park [62], O1 [65] and CCNN [66]. As already pointed out, we adopt for this evaluation the same protocol for training and testing. In this case, we train the original 5 considered confidence measure on the same 25 images used to train our framework (frames from 000000 to 000024 of KITTI 12).

Figure 8.6 shows the results on the three datasets with AD-CENSUS, reported in detail in Table 8.2, according to the same methodology described in Section 8.3.2. Observing the figure we can clearly notice that our proposal always outperforms significantly the 5 original confidence measures on all the three datasets. The improvements are remarkable also for top-performing confidence measures O1 and CNN being Δ_k, respectively, greater than 14% and 9% in the worst case. For the other 3 confidence measures the improvement is, in the worst case, greater than 28% for Park, almost 27% for GCP and greater than 46% for Ensemble that, in the best case, improves by more than 81% with our framework. Interestingly, the learned Ensemble$^+$ confidence measure is able to outperform the



(a) (b) (c)

Figure 8.6: Average AUC for the 5 confidence measures based on machine-learning on the 3 datasets with AD-CENSUS. (a) Evaluation on KITTI 12 images excluded from training (169 frames, from 000025 to 000193), (b) evaluation on KITTI 15 (200 frames), (c) evaluation on Middlebury 14 (15 frames). In blue the AUC for the original confidence measure (e.g., AUC_{GCP} [80]), in green the AUC related to its learned counterpart (e.g., AUC_{GCP+}).

Confidence	KITTI 12 (169/194)			KITTI 15 (200/200)			Middlebury 14 (15/15)		
measure	AUC_k	AUC_{k+}	Δ_k	AUC_k	AUC_{k+}	Δ_k	AUC_k	AUC_{k+}	Δ_k
Ensemble [21]	0.214929	**0.127682**	81.44%	0.186504	**0.109991**	77.96%	0.245227	**0.163656**	46.12%
GCP [80]	0.152764	**0.138078**	32.66%	0.139611	**0.124286**	29.90%	0.109302	**0.098367**	26.71%
Park [62]	0.144077	**0.132393**	32.21%	0.131662	**0.117529**	32.64%	0.104146	**0.094084**	28.13%
O1 [65]	0.127645	**0.124695**	14.87%	0.108812	**0.105893**	14.27%	0.090908	**0.086444**	19.81%
CCNN [66]	0.123612	**0.121257**	14.90%	0.105645	**0.103645**	11.59%	0.086082	**0.084485**	9.01%
Optimal	0.107802			0.088357			0.068375		

Table 8.2: Average AUC for the considered 5 confidence measures based on machine-learning based on the 3 datasets with AD-CENSUS. The table is split into three blocks: left block reports evaluation on KITTI 12 images excluded from training (169 frames, from 000025 to 000193), middle block reports evaluation on KITTI 15 (200 frames), right block reports evaluation on Middlebury 14 (15 frames). Each block contains AUC for the original measure (AUC_k), the outcome of our framework (AUC_{k+}) and the improvement (Δ_k) yielded by our proposal, with respect to AUC_{opt}, computed according to equation 8.4.

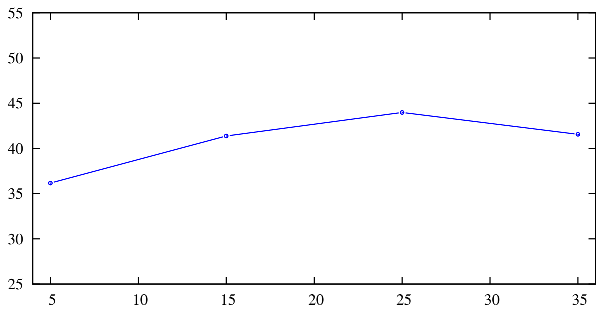


Figure 8.7: Average improvement Δ_k (%) on Middlebury 14 with different amount of training data (first 5, 15, 25 and 35 frames) from KITTI 12 with AD-CENSUS.

original GCP and LEV approaches on KITTI 12 and KITTI 15. This further evaluation confirms the effectiveness of our proposal even with the 5 confidence measures based on machine learning.

Moreover, comparing the results reported in Table 8.1 and 8.2, we can notice how with our proposal some stand-alone confidence measures are able to outperform approaches based on machine-learning. In particular, Ensemble is outperformed by all the learned confidence measures, except MLM$^+$, NEM$^+$ and NOI$^+$ on KITTI 12, MSM$^+$, MLM$^+$, PER$^+$, NEM$^+$, DSM$^+$ and NOI$^+$ on KITTI 15, NEM$^+$ and NOI$^+$ on Middlebury 14. GCP is outperformed by WMN$^+$ and SAMM$^+$ on KITTI 12, by PKR$^+$, WMN$^+$ and SAMM$^+$ on KITTI 15, by PKR$^+$, WMN$^+$, WMNN$^+$ and UC$^+$ on Middlebury 14. LEV is outperformed by WMN$^+$ and SAMM$^+$ on KITTI 15, by PKR$^+$ and WMN$^+$ on Middlebury v3. This means that the proposed framework is not only able to significantly improve the effectiveness of each considered confidence measure, but in many cases it enables to achieve even more accurate prediction by processing a single confidence measure rather than by combining multiple ones as done by the three machine-learning approaches Ensemble [21], GCP [80] and Park [62].

Finally, we report in Figure 8.7 the average improvement Δ_k achieved by our networks on Middlebury 14 as a function of the amount of training data. Observing the figure we can notice that we obtain the best performance with 25 frames and, more interestingly, our networks trained only on 5 frames achieve an average improvement greater than 35%.

Measure	KITTI 12	KITTI 15	Middlebury 14
PKRN$^+$	66.5%	60.8%	29.1%
PKR$^+$	69.2%	54.7%	23.4%
MSM$^+$	34.4%	21.9%	23.4%
MMN$^+$	52.5%	41.4%	40.6%
WMN$^+$	73.1%	59.4%	23.7%
MLM$^+$	17.8%	13.5%	14.4%
PER$^+$	43.6%	33.9%	42.3%
NEM$^+$	46.6%	32.5%	34.3%
LRD$^+$	51.8%	41.1%	44.8%
CUR$^+$	11.4%	49.9%	77.1%
DSM$^+$	36.2%	23.6%	24.3%
AML$^+$	63.5%	53.4%	51.1%
NOI$^+$	46.1%	33.9%	28.9%
SAMM$^+$	70.9%	64.0%	61.4%
WMNN$^+$	57.2%	53.0%	23.1%
LRC$^+$	73.3%	63.7%	30.9%
LC$^+$	9.8%	25.8%	65.6%
UC$^+$	75.0%	71.0%	72.3%
Ensemble$^+$	74.3%	70.5%	38.5%
GCP$^+$	27.1%	18.5%	26.0%
Park$^+$	33.5%	28.5%	36.3%
O1$^+$	26.2%	22.0%	38.9%
CCNN$^+$	15.6%	10.6%	21.5%

Table 8.3: Average improvement Δ_k yielded by our proposal on the three datasets with MC-CNN [36].

8.3.4 Evaluation with MC-CNN

In Table 8.3 we provide additional experimental results concerned with state-of-the-art cost function MC-CNN [95, 96]. We trained our networks on the same amount of data (i.e., 25 images of KITTI 12 dataset) and followed the same cross validation protocol adopted with the AD-CENSUS algorithm. Due to the lack of space, we report for MC-CNN only the average improvement Δ_k on the three datasets. The table confirms that, even with the more accurate MC-CNN algorithm, our proposal achieves notable improvements on each of the 23 examined confidence measures with Δ_k ranging from $\approx$10% (LC$^+$ in the worst case) to more than 77% (CUR$^+$ in the best case). Focusing on approaches based on machine-learning we can also notice that our proposal yields improvements from 10.6% (CCNN$^+$ in the worst case) to more than 74% (Ensemble$^+$ in the best case).

8.4 Conclusions

In this chapter we have proposed a methodology aimed at improving the effectiveness of confidence measures for stereo by exploiting local consistency. Our framework, leveraging on a deep network, is able to learn and improve the local behavior of confidence measures and, to the best of our knowledge, it is the first method to move beyond single pixel-wise confidence estimation performed by other approaches. The exhaustive experimental evaluation with two stereo algorithms, including a cross-validation on two additional datasets, shows that our method enables remarkable improvements on each of the 23 state-of-the-art confidence measures and on each dataset. This confirms the assumption made in this chapter: confidence maps are locally consistent and a deep network can learn how to exploit this fact. In particular, results reported with state-of-the-art confidence measures based on machine-learning set the bar a further step closer to optimality paving the way to further improvements in this field.

Chapter 9

Even more confident predictions with deep learning

The content of this chapter has been presented at the 13th IEEE Embedded Vision Workshop (EVW 2017) - ”Even More Confident predictions with deep machine-learning”. Most relevant to the work shown in this chapter are the following papers: [32, 80, 62, 65, 94].

9.1 Introduction

As already highlighted previosuly, one of the most popular techniques to obtain a reliable confidence measure consists of combining multiple hand-crafted features and training a random forest classifier to infer a more accurate confidence from them. Examples of works following this strategy are [21, 80, 62, 65].

These, 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 9.1, 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 by means of a state-of-the-art approach [62] based on

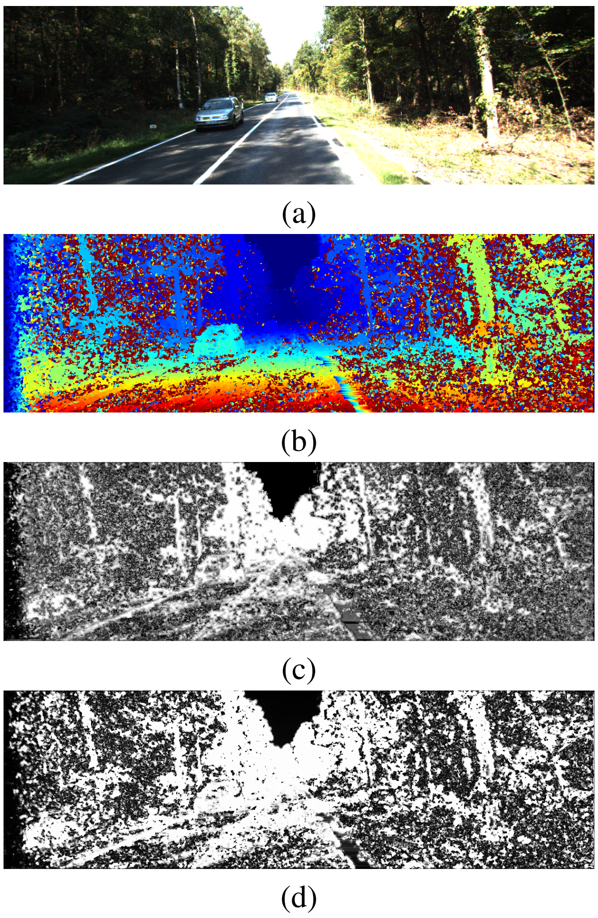


Figure 9.1: Qualitative results of confidence maps obtained by [62], random forest implementation versus EMC.

a random-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 9.3 we will report quantitative results confirming the advantages yielded of our strategy.

Differently from approaches relying on random-forest classifiers that infer, for each pixel, an estimated match reliability by processing a 1D input feature vector made of pixel-wise confidence measures and features, our proposal relies on a more distinctive 3D input domain. Such input domain, for the pixel under analysis, is made of patches extracted from multiple input confidence and feature maps around the examined pixel as shown in Figure 9.2. Leveraging on a CNN, our proposal is able to infer more meaningful confidence estimations with respect to a random forest fed with the

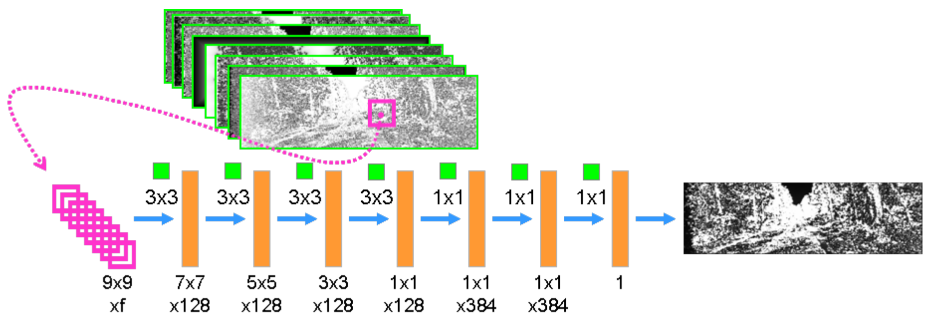


Figure 9.2: Architecture of EMC.

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.

We validate our method as follows. Once selected a subset of stereo pairs from the KITTI 2012 [14] 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 [62, 65]. 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 [80], [62] and [65] based on random-forest frameworks. Then, we evaluate the effectiveness of our proposal with respect to [80], [62] and [65] by means of ROC curve analysis [32], on the remaining portion of KITTI 2012. Moreover, we cross-validate without re-training on KITTI 2015 and Middlebury v3.

9.2 Deep learning for confidence measures

In this work, 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 9.2.

9.2.1 Hand-crafted features layer

In [21] 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 [80, 62, 65] enabled remarkable improvements, in these works 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 of input feature maps provided by such layer. For example, adopting the same input features of [80] 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 [80]: MSM, MMN, AML, LRC, LRD, distance to border, distance to discontinuities and disparity deviation from median.

9.2.2 Deep network architecture

This section describes the design of the architecture proposed to infer a learned confidence measure. Excluding the H front-end, in charge of providing multiple feature maps from the available input cues (e.g., cost curve, disparity maps, etc), we rely on a deep-network architecture made of 7 convolutional layers trained to infer a pixel-wise confidence measure processing 3D input features. Specifically, we deploy a patch-based fully-convolutional architecture, as shown in Figure 9.2.

A patch-based approach, as proposed in [96, 66], requires a significantly lower amount of data for training compared to an end-to-end deep network architecture working on full-resolution images like the one proposed in [54]. In fact, in this second case, the dataset required to train such deep-network for the same purpose would be much more larger. Considering this fact, our model is made of four convolutional layers, each one followed by Rectifier Linear Units (ReLU). Each layer applies 128 kernels of size 3×3, applied to each pixel (stride equal to 1). Two additional convolutional layers, made of 384 1×1 kernels followed by ReLU, increase the amount of extracted features, leading to the final output layer. This model counts more than one half million parameters and was chosen in our experiments, after a preliminary testing, as the one yielding more accurate results. According to this architecture, a single pixel-wise confidence measure is obtained by processing a 9×9 receptive field after the front-end H. According to Figure 9.2 this means that the 3D input domain processed by our network has size $9 \times 9 \times f$.

Being our architecture a fully-convolutional model, any input of size greater than the receptive field can be processed by the network. This means that it is capable of computing a full resolution confidence map by processing the feature maps forwarded by the H front-end. The deep network, excluding H, performs on a full-resolution KITTI 2012 image a confidence prediction in a few seconds on an i7 CPU, dropping to 0.8 seconds with a Titan X GPU, with an overall memory footprint of about 4.5 GB.

9.3 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 [80, 62, 65] 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 v3 datasets (15 images).

9.3.1 Training phase

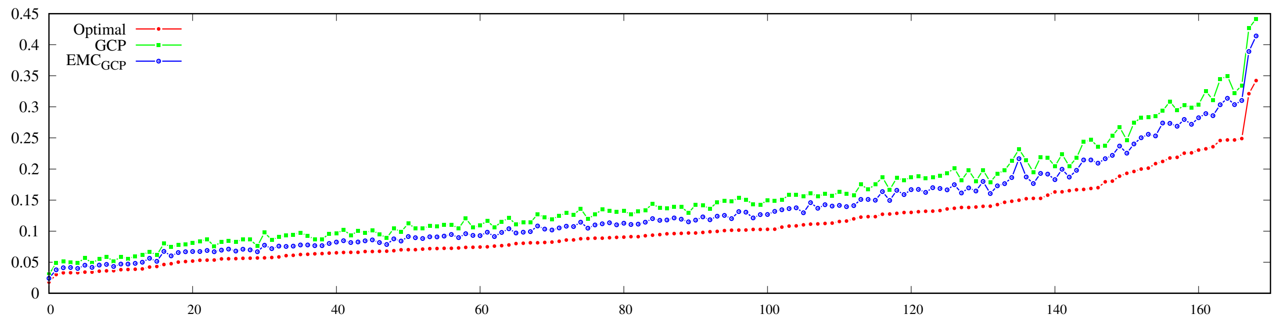
We trained our network according to *stochastic gradient descend*, we choose the *binary cross entropy* as loss function, according to the regression problem we are dealing with. We trained on nearly 3.5 million samples, obtained from the first 25 stereo pairs of the KITTI 2012 training dataset. Each 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 [14, 56]. 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 [80, 62, 65], representing state-of-the art confidence measures inferred by random-forest frameworks. During the validation, these three methods will be referred to as, respectively,

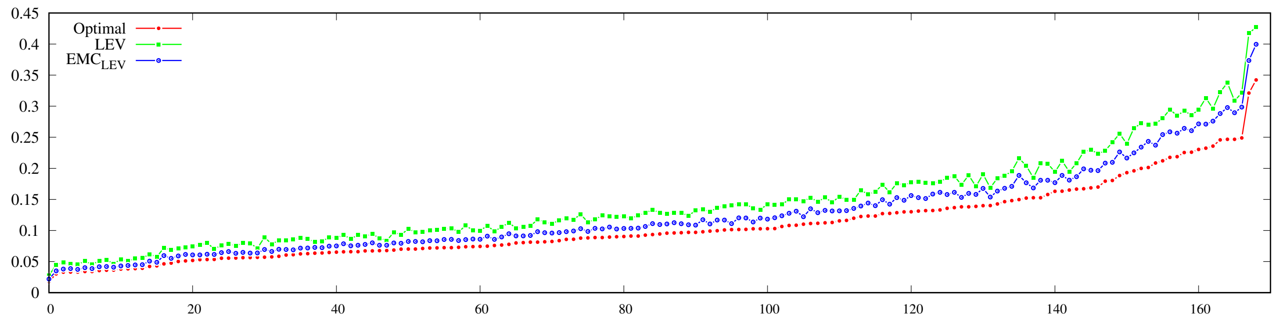
- GCP (Ground Control Point) [80], processing a feature vector of cardinality 8 by means of a random-forest. Such vector contains MSM, MMN, AML, LRC, LRD confidence measures reviewed in [32], DTB (distance to border), DTD (distance to discontinuities) and MED (disparity deviation from median) computed on a 5×5 patch.

- LEV (Leveraging-Stereo) [62], processing a feature vector of cardinality 22 by means of a random-forest. The vector contains PKR, PKRN, MSM, MMN, WMN, MLM, NEM, LRD, CUR and LRC confidence measures reviewed in [32], PER confidence measure proposed in [21], DTBL (distance to left border), DTE (distance to edges), HGM (horizontal gradient magnitude), MED (disparity deviation from median) and VAR (variance of disparity) on 5×5, 7×7, 9×9 and 11×11 neighborhood.

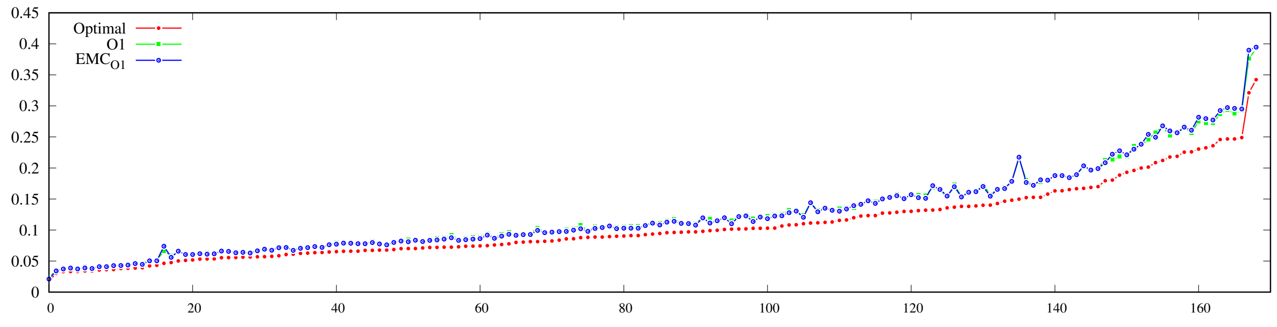
- O1 (O1) [65], processing a feature vector of cardinality 20 by means of a random-forest. The



(a)



(b)



(c)

Figure 9.3: AUC plots for GCP [80], LEV [62] and O1 [65], random forest vs EMC, AD-CENSUS algorithm, KITTI 2012 dataset.

vector contains DA (disparity agreement), DS (disparity scattering, median disparity, VAR (variance of disparity) and MED (disparity deviation from median), each one computed on 5×5, 7×7, 9×9 and 11×11 neighborhood.

9.3.2 EMC vs random-forest

We deploy AUC analysis to compare the outlier detection property of the two approaches. To be compliant with the training protocol,we fix a threshold value on disparity error of 3.

Figure 9.3 depicts three plots, containing the AUC values computed over the entire KITTI 2012 (ex-

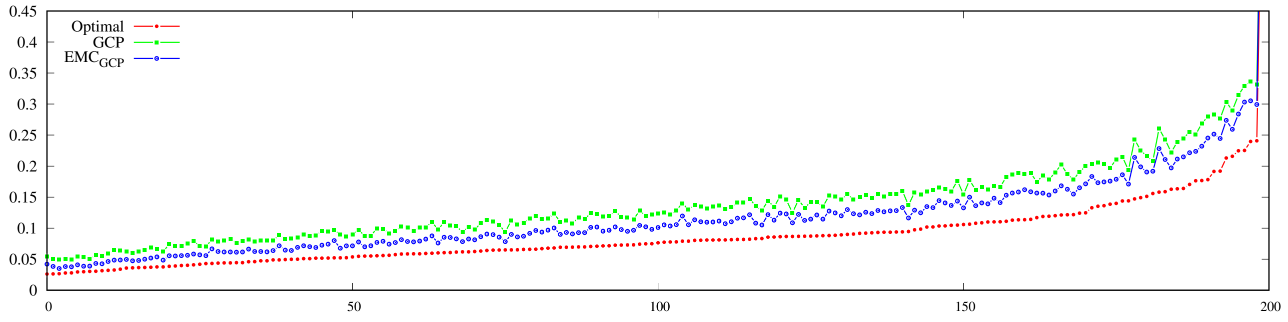
	KITTI 2012			KITTI 2015			Middlebury v3		
	GCP	LEV	O1	GCP	LEV	O1	GCP	LEV	O1
Optimal	0.107802			0.088357			0.068375		
RF	0.152764	0.144077	0.127645	0.139611	0.131662	0.108812	0.109302	0.104146	0.090908
EMC	0.133684	0.125211	0.126898	0.117551	0.107969	0.106523	0.091749	0.088473	0.089928
Δ_k	42.44%	52.01%	3.76%	43.04%	54.71%	11.19%	42.88%	43.81%	4.35%

Table 9.1: AUC analysis, comparison between GCP [80], LEV [62] and O1 [65], random forest vs EMC, AD-CENSUS algorithm, KITTI 2012, KITTI 2015 and Middlebury v3 datasets.

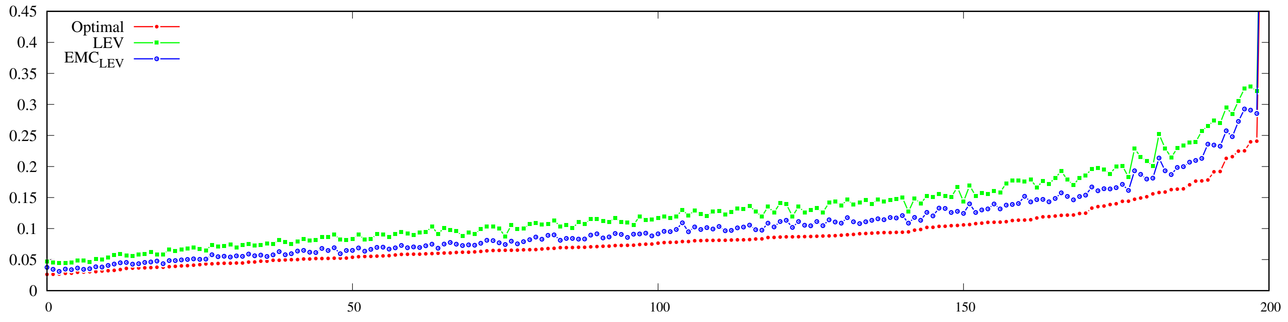
cluding the images processed during training) of both the EMC approach and the corresponding random forest counterpart, for GCP [80], LEV [62], O1 [65]. The curves are plotted in non-descending order according to optimal values (red), together with curves related to random forest implementation (referred to as GCP, LEV and O1, plotted in green) and our method processing the same inputs (referred to as EMC$_{GCP}$, EMC$_{LEV}$ and EMC$_{O1}$, plotted in blue). In particular, from top to bottom, (a) presents 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 [65], but on average it achieves a better AUC on the entire dataset.

Figure 9.4 depicts the three plots for the entire KITTI 2015, comparing the EMC approach with the corresponding random forest counterpart, for GCP [80], LEV [62], O1 [65]. 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) presents 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.

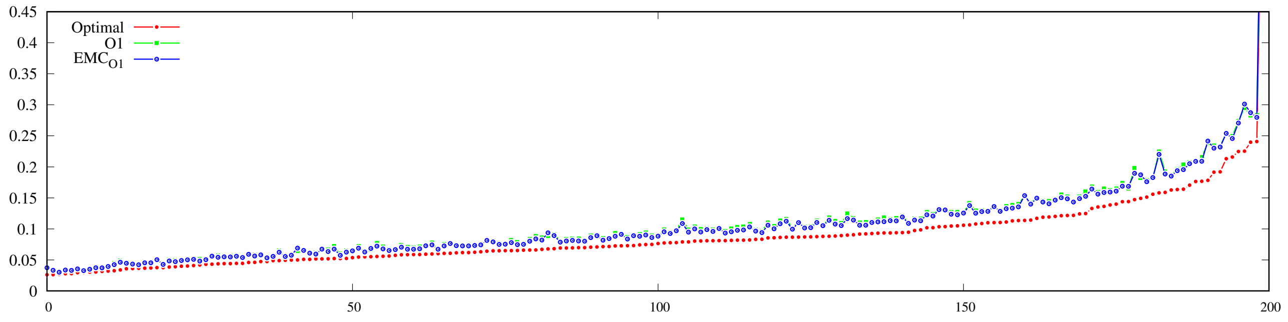
Figure 9.5 shows three plots concerning the evaluation on the Middlebury v3 dataset. As for the previous figures, optimal values are plotted in red, curves related to random forest implementation are in green (referred to as GCP, LEV and O1) and those related to EMC processing the same inputs (referred to as EMC$_{GCP}$, EMC$_{LEV}$ and EMC$_{O1}$). In particular, from left to right, (a) presents GCP versus EMC$_{GCP}$, (b) with LEV versus EMC$_{LEV}$, (c) with O1 vs EMC$_{O1}$. The three confidence



(a)

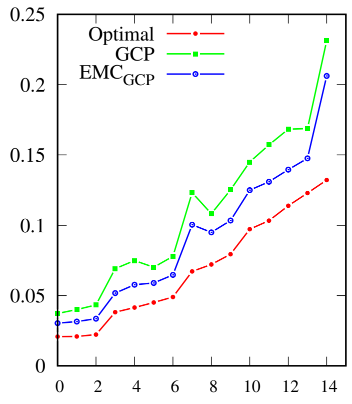


(b)

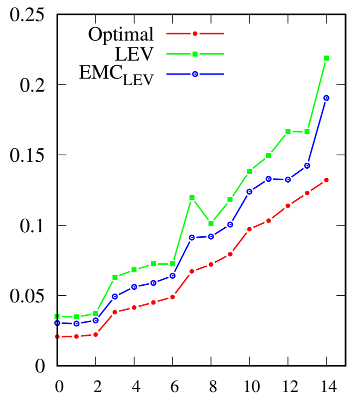


(c)

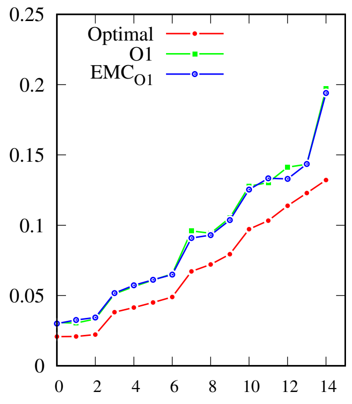
Figure 9.4: AUC plots for GCP [80], LEV [62] and O1 [65], random forest vs EMC, AD-CENSUS algorithm, KITTI 2015 dataset.



(a)



(b)



(c)

Figure 9.5: AUC plots for GCP [80], Park [62] and O1 [65], random forest vs EMC, AD-CENSUS algorithm, Middlebury 3 dataset.

	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 v3		
	GCP	LEV	O1
EMC win rate	15/15	15/15	8/15

Table 9.2: EMC win rate for GCP [80], LEV [62] and O1 [65], AD-CENSUS algorithm, KITTI 2012, KITTI 2015 and Middlebury v3 datasets.

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:

$$\Delta_k = \frac{AUC_k - AUC_{EMC_k}}{AUC_k - AUC_{opt}} \tag{9.1}$$

If delta is greater than zero, EMC improves the original confidence measure. 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 9.1 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 [32], 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%, specifically, 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 v3, on which LEV achieve a lower, yet important Δ_k value. Focusing on O1, the improvement is lower, between 3% and 12%

	KITTI 2012			KITTI 2015			Middlebury v3		
	GCP	LEV	O1	GCP	LEV	O1	GCP	LEV	O1
Optimal	0.107802			0.088357			0.068375		
baseline	0.152764	0.144077	0.127645	0.139611	0.131662	0.108812	0.109302	0.104146	0.090908
Plus	0.138078	0.132393	**0.124695**	0.124286	0.117529	**0.105893**	0.098367	0.094084	**0.086444**
EMC	**0.133684**	**0.125211**	0.126898	**0.117551**	**0.107969**	0.106523	**0.091749**	**0.088473**	0.089928

Table 9.3: AUC analysis, comparison between GCP [80], LEV [62] and O1 [65], EMC vs *Plus* (Chapter 8) frameworks, AD-CENSUS algorithm, KITTI 2012, KITTI 2015 and Middlebury v3 datasets.

(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 9.2 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 v3 data for random forest approaches and EMC. This evaluation proves how the effectiveness of the CNN-based result is maintained 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 9.6 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.

9.3.3 EMC vs *Plus* framework

We now compare the results achieved by EMC with respect to the local consistency framework presented in Chapter 8. Table 9.3 report average AUCs on the three datasets for GCP, LEV and O1 improved, respectively, by the *Plus* framework and by EMC. We can observe how EMC outperforms

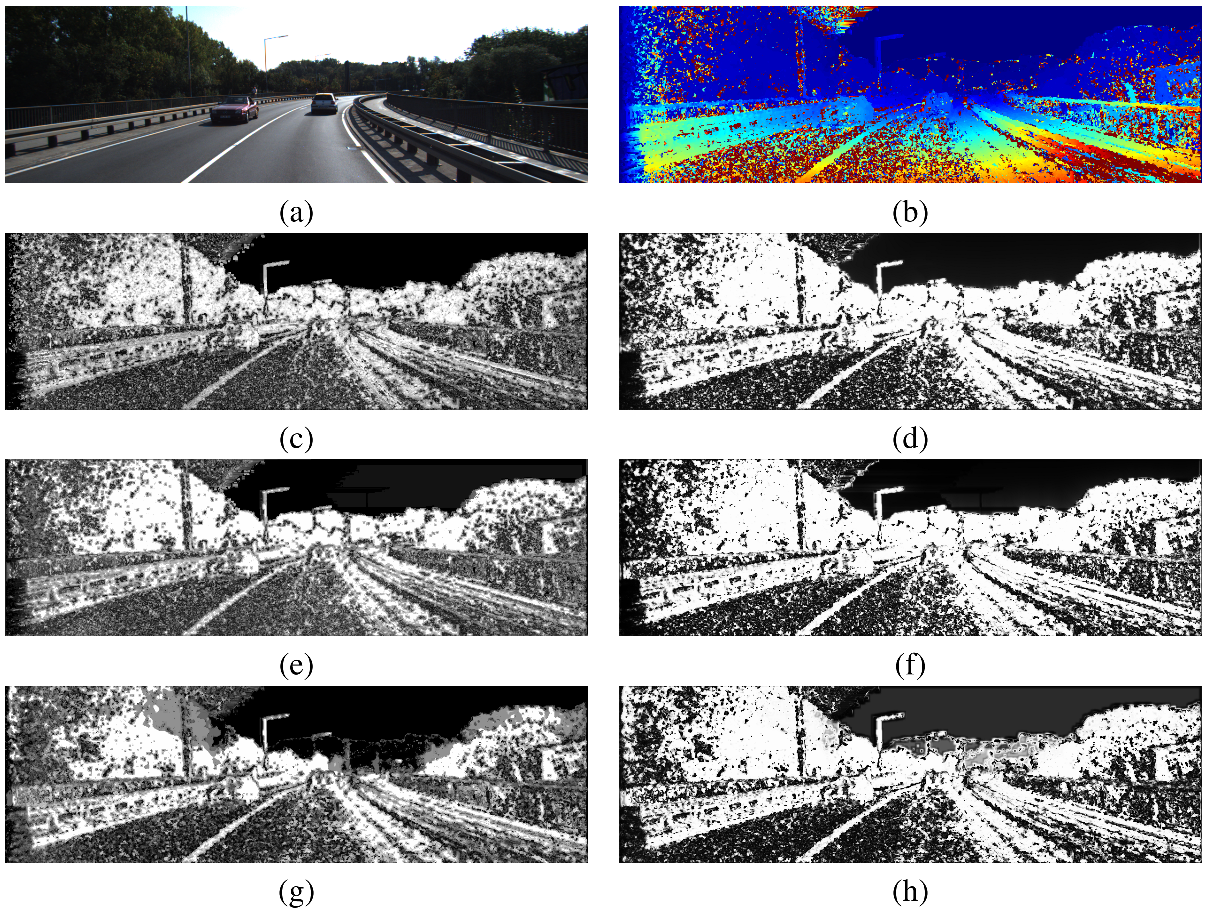


Figure 9.6: Qualitative results of confidence maps obtained by [62], GCP [80] and O1 [65], random forest implementation versus EMC.

Plus at improving GCP and LEV measures on all datasets, while O1 is better improved by the proposal described in Chapter 8. This evaluation suggests how the local content processed at features level add more information than locality properties of the inferred measure. When the latter is already processed by the original measure (as in the case of O1), the improvement given by EMC saturates, while *Plus* can still increase the effectiveness of the source measure.

9.4 Conclusions

In this work 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 10

Non-Local Anchoring for disparity refinement

Most relevant to the work shown in this chapter are the following papers: [32, 62, 75, 94, 96, 24, 63, 7, 5, 46].

10.1 Introduction

According to [73], most of the stereo algorithms rely on four common steps (except the brand new end-to-end deep learning techniques). One of them, referred to as *disparity refinement*, attempts to recover errors from the estimated 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 for detecting unreliable matches produced by stereo algorithms and, recently, machine-learning has been deployed to predict even more effective ones. Confidence measures have been involved into different steps of stereo pipelines, with the aim to further improve the overall accuracy.

In this work, we propose *Non-Local Anchoring* (NLA), a novel disparity refinement method relying on confidence measures, outlined in Figure 10.1. Given a disparity map, such measure allows for

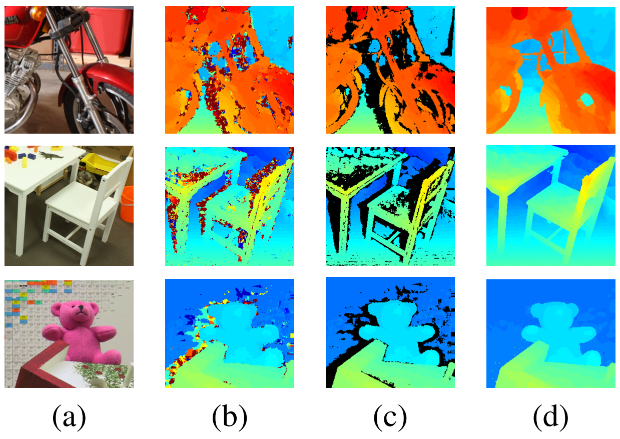


Figure 10.1: Qualitative results of NLA refinement, Middlebury v3 dataset. (a) Reference images (from top to bottom, *Motorcycle, PlayTable, Teddy*), (b) nosy disparity maps, (c) reliable pixels, (d) refined disparity maps.

removing erroneous pixels. Then, among the remaining *reliable points* (RP), a subset of *anchors* is chosen to infer, according to both spatial and color information from the reference image, a disparity value for each discarded pixel. Moreover, a machine-learning framework 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 strategy allows us to the need for an heuristic selection of a confidence threshold as often carried-out in this field [80, 75]. To assess the effectiveness of our proposal, we report an extensive evaluation on the Middlebury v3 dataset comparing our framework to conventional disparity refinement methodologies as well as with recently proposed confidence-based approaches, acting in the Disparity Space Image (DSI) domain. Differently from latter approaches, NLA acts in the disparity domain. Factors like the number of anchors deployed and a further local aggregation strategy included in our proposal are discussed and compared to state-of-the-art. Moreover, we evaluate our framework also on KITTI 2015 dataset [56] to further confirm the effectiveness of our method on indoor and outdoor data. Figure 10.2 shows the effective results lead by our proposal on frame 166 of KITTI 2015 dataset.

10.2 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

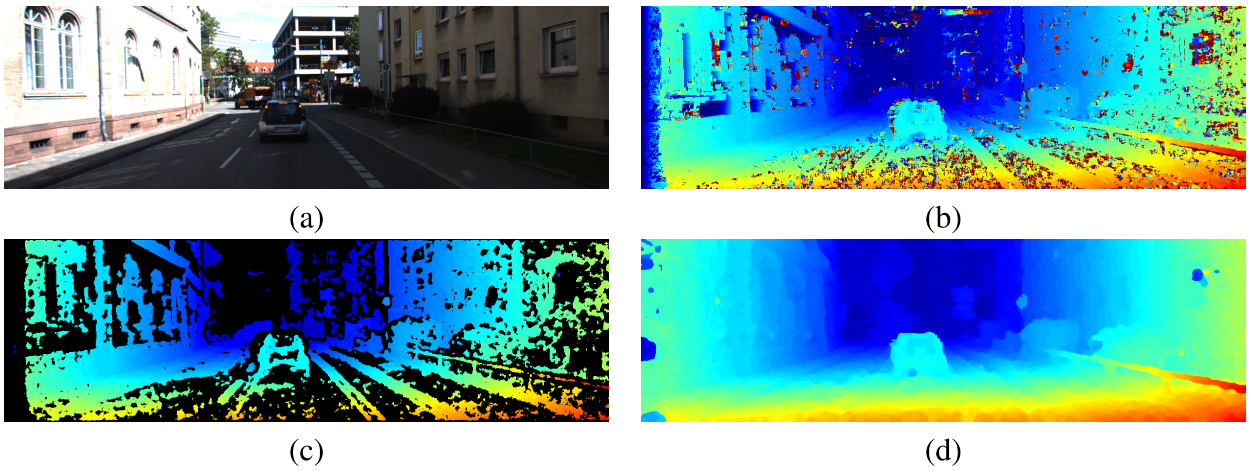


Figure 10.2: Qualitative results of NLA refinement, KITTI 2015 dataset. (a) Reference image (000166), (b) noisy disparity map, (c) reliable points, (d) refined disparity map.

reliability), infers a completely dense and more accurate map. It starts by classifying each disparity pixel belonging to $\mathcal{D}$ in two categories: *reliable* RP and *unreliable* UP points. In literature [80, 75], 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,

$$RP = \{p \in \mathcal{D}, \mathcal{C}(p) \geq \xi\} \tag{10.1}$$

consequently, the remaining ones are considered UP,

$$UP = \{p \in \mathcal{D}, \mathcal{C}(p) < \xi\} \tag{10.2}$$

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 $\mathcal{D}$. Afterwards, 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

$$P(p,d) = \{q \in \mathcal{D}, \alpha \in \mathcal{N}, q = p + \alpha d\} \tag{10.3}$$

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

$P(u,d)$

$$a(u,d) = \{v \in RP, \min_{v} |u - v|\} \tag{10.4}$$

Given a set of paths on which u lays, a set $\mathcal{A}(u)$ of anchors will contribute to compute 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 the two points as follows:

$$w(u,a) = \mathcal{G}(|\mathcal{I}(u) - \mathcal{I}(a)|) \cdot \mathcal{G}(|u - a|) \tag{10.5}$$

being $\mathcal{I}(u)$ and $\mathcal{I}(a)$ a set of features, respectively, for pixels u and a and $\mathcal{G}$ a similarity function. 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

$$\mathcal{D}(u) = \min_{k} \sum_{i=0}^{k} w(u,a_i) \geq \frac{1}{2} \sum_{i=0}^{n} w(u,a_i) \tag{10.6}$$

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, coupled with the weighted median, enables edge-preserving disparity propagation. Figure 10.3 shows an example of anchoring for an unreliable pixel (red), receiving 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. In fact, 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 to other methods, is not constrained to a restricted area (i.e., local patches). Moreover, differently from recent methodologies exploiting confidence to improve stereo accuracy [80, 62, 65, 75], our framework acts on the disparity domain hence not requiring any information from the DSI.

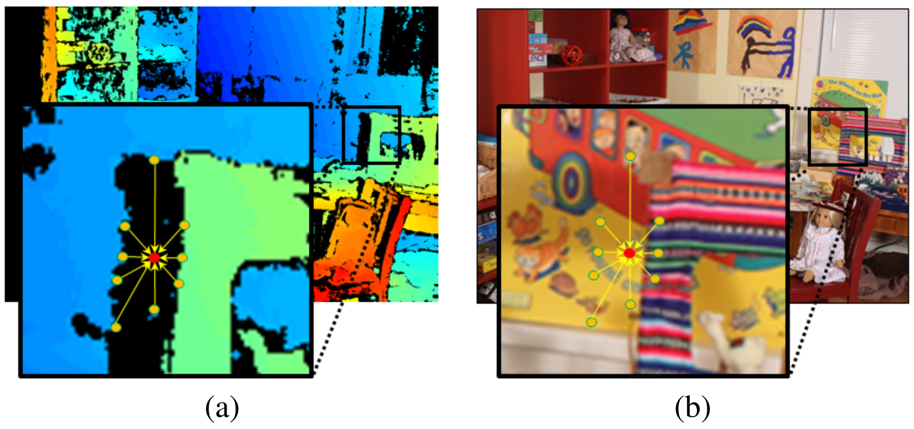


Figure 10.3: Overview of NLA technique. Selection of reliable points on the disparity map (a) and their correspondent pixel intensities on reference image (b).

Optionally, before replacing the unreliable pixel u according the outlined strategy, a further local aggregation step can improve the effectiveness of the information gathered from nearby points. This 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 for collecting additional contributions from nearby UP pixels q_k, whose set of anchors $\mathcal{A}(q_k)$ is different from $\mathcal{A}(u)$.

10.3 Threshold-free RP selection

According to the description reported in Section 10.2, classifying the disparity values in UP and RP plays a key-role for NLA to achieve optimal performance. Thus, choosing of the confidence threshold ξ is of paramount importance. This is common to other successful attempts to exploit confidence measures inside stereo algorithms [80, 75] or, in general, when we want to remove erroneous matches from the disparity map. For such tasks a proper tuning of the threshold ξ is required to achieve the best results.

To address this issue, we propose a novel machine-learning framework to effectively distinguish pixels into RP and UP according to features extracted from the confidence map. 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

$$\mu^{\mathcal{C}}_{\mathcal{N}}(p) = \frac{1}{\bar{\bar{\mathcal{N}}}} \sum_{q \in \mathcal{N}} \mathcal{C}_q \tag{10.7}$$

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

$$\sigma^{\mathcal{C}}_{\mathcal{N}}(p) = \frac{1}{\bar{\bar{\mathcal{N}}}} \sum_{q \in \mathcal{N}} [\mathcal{C}_q - \mu_{\mathcal{N}}(\mathcal{C}_p)]^2 \tag{10.8}$$

- $\delta_\mu(p)$, 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., $\mu_{\mathcal{D}}(\mathcal{C})$)

$$\delta_\mu(p) = |\mathcal{C}_p - \mu^{\mathcal{C}}_{\mathcal{D}}(\mathcal{C})| \tag{10.9}$$

- $\delta_\sigma(p)$, 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., $\sigma_{\mathcal{D}}(\mathcal{C})$)

$$\delta_\sigma(p) = |\mathcal{C}_p - \sigma^{\mathcal{C}}_{\mathcal{D}}(\mathcal{C})| \tag{10.10}$$

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 $f_9(p)$

$$f_9(p) \quad = \quad \{\mathcal{C}_p, \mu^{\mathcal{C}}_{\Omega}(p), \mu^{\mathcal{C}}_{\Theta}(p), \mu^{\mathcal{C}}_{\Gamma}(p), \sigma^{\mathcal{C}}_{\Omega}(p), \sigma^{\mathcal{C}}_{\Theta}(p), \sigma^{\mathcal{C}}_{\Gamma}(p), \delta_\mu(p), \delta_\sigma(p)\} \tag{10.11}$$

We train on such feature vector a random forest, made of 15 trees, in order to achieve an automatic RP selection without any hand-chosen threshold.

10.4 Experimental results

In this section, we evaluate the effectiveness of the proposed NLA framework with disparity maps obtained, on challenging datasets, by AD-CENSUS, MC-CNN and SGM. To exhaustively assess the effectiveness of our proposal, we compare it to state-of-the-art disparity refinements 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 [62, 65, 75] acting in the DSI domain. We also evaluate for NLA the effect yielded by 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 manual optimal choice of the ξ value by cross validation. We evaluate all this aspects on the Middlebury v3 [71] training dataset, then we evaluate the effectiveness of the overall NLA framework also on KITTI 2015 [56].

Stereo	All				Non-occ			
algorithm	bad 1(%)	bad 2(%)	RMSE	MAE	bad 1(%)	bad 2(%)	RMSE	MAE
AD-CENSUS	35.13	32.32	14.32	6.85	26.37	23.54	10.94	4.49
+ FBS [5]	33.47	28.60	12.79	5.28	24.67	19.59	8.32	2.93
+ MF [63]	27.43	23.99	10.14	4.32	18.42	14.95	5.82	2.17
+ WMF [98]	26.22	22.92	10.08	4.18	17.22	13.91	5.56	2.00
+ WMF + GF [46]	26.33	22.92	11.27	4.75	17.41	14.04	7.59	2.86
+ WMF + JBF [46]	28.03	24.93	10.95	4.68	18.86	15.75	6.87	2.44
+ LRI [96]	27.99	24.99	19.10	6.97	20.64	17.81	14.93	4.51
+ LRI + MF + BF [96]	26.02	21.53	15.78	5.94	18.68	14.23	11.18	3.58
+ LC [50]	24.23	20.00	11.68	4.74	16.30	12.23	7.93	2.65
+ NLA + CCNN	**21.16**	**17.82**	**8.11**	**3.06**	**13.36**	**10.34**	**4.78**	**1.52**
+ NLA + opt.	*6.23*	*4.07*	*3.06*	*0.85*	*2.20*	*1.21*	*1.77*	*0.44*

Table 10.1: Absolute improvement of disparity accuracy yielded by NLA, AD-CENSUS algorithm, Middlebury v3 dataset. We report bad1 and bad2 error rates, as well as RMSE and MAE, for our method and a number of competitors.

10.4.1 Evaluation of the NLA framework

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 v3 training dataset[1]. In tables 10.1, 10.2 and 10.3 we report

[1]We process Middlebury v3 stereo pairs at quarter resolution. All the results reported in this work have been computed at such resolution. We report results on training data to compare NLA with a large number of methods, without multiple submissions to online benchmark.

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
MC-CNN [96]	25.95	23.88	14.83	6.32	16.02	13.99	10.94	3.27
+ FBS [5]	25.05	21.58	10.21	4.15	15.32	11.87	5.50	1.83
+ MF	22.69	20.17	11.54	4.56	12.76	10.25	5.82	1.90
+ WMF [98]	21.49	19.14	11.92	4.59	11.62	9.30	5.56	1.78
+ WMF + GF [46]	21.68	19.30	13.13	5.18	11.96	9.67	7.59	2.69
+ WMF + JBF [46]	22.22	19.88	12.76	4.68	12.20	10.06	6.87	2.07
+ LRI [96]	20.40	17.82	15.54	4.97	12.83	10.64	10.54	2.69
+ LRI + MF + BF [96]	20.29	15.97	12.96	4.42	12.88	8.87	7.94	2.36
+ LC [50]	19.46	15.68	9.92	5.01	11.10	7.65	7.93	1.71
+ NLA + CCNN	**16.39**	**13.45**	**7.39**	**2.59**	**9.98**	**7.66**	**4.54**	**1.36**
+ NLA + opt.	*6.24*	*4.14*	*3.07*	*0.84*	*2.17*	*1.27*	*1.77*	*0.42*

Table 10.2: Absolute improvement of disparity accuracy yielded by NLA, MC-CNN fast algorithm, Middlebury v3 dataset. We report bad1 and bad2 error rates, as well as RMSE and MAE, for our method and a number of competitors.

results achieved by the following disparity refinement methods: fast bilateral solver (FBS [5]), median filter (MF [63]), weighted median filter (WMF [98]), weighted median filter together plus guided filter (WMF + GF [46]), weighted median filter plus joint bilateral filter (WMF + JBF [46]) and local consistency filter (LC [50]). All of them are refinement methods processing as only input cue the 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 [96] (LRI + MF + BF) using authors' code. 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 two different confidence measures: O1 [65] and CCNN [66]. The choice of these two measures was driven by the aim of our framework, working in the disparity domain only. We followed implementation notes, hyper-parameters tuning and code provided by the authors [65, 66], training both random forest and CCNN on a subset of images from KITTI 2012 dataset (the first 20 images). Being the effectiveness of the confidence measure crucial for our method, we also report in the final row the results achieved by NLA processing an optimal confidence measure, capable of perfectly distinguish between RP and UP. This represents the lower bound for the error rate with NLA. We report, for each algorithm, results obtained with the best performing confidence measure, respectively CCNN for AD-CENSUS, MC-CNN algorithms and O1 for SGM. The automatic selection method proposed was trained on the 13 additional images available in Middlebury v3 dataset [71] for each of the three considered algorithms. Table 10.1 report the effectiveness of disparity refinement methods with the

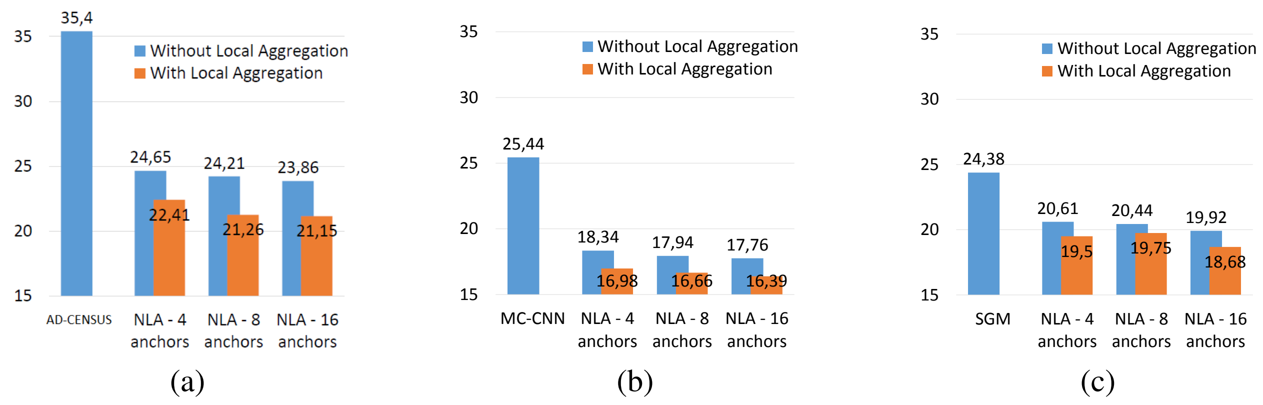


Figure 10.4: Experimental results on Middlebury v3, varying the number of anchors and enabling/disabling local aggregation with NLA framework. (a) AD-CENSUS algorithm + CCNN, (b) MC-CNN algorithm + CCNN, (c) SGM algorithm + O1.

AD-CENSUS 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 in its best configuration (i.e. with CCNN measure) it is more effective by nearly 4% on all pixels and by 3% on 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 10.2 reports evaluation concerning the refinement of disparity using MC-CNN [96]. The NLA framework still outperforms all its competitors.

Stereo	All				Non-occ			
algorithm	bad 1%	bad 2%	RMSE	MAE	bad 1%	bad 2%	RMSE	MAE
SGM [24]	24.38	22.00	13.18	5.33	14.52	12.14	7.96	2.49
+ FBS [5]	25.06	21.55	12.05	4.51	15.46	11.93	6.80	2.10
+ MF [63]	23.13	20.44	11.20	4.43	13.45	10.74	5.95	1.91
+ WMF [98]	21.88	19.29	11.32	4.34	12.26	9.67	5.69	1.74
+ WMF + GF [46]	22.22	19.56	12.54	4.96	12.72	10.10	7.96	2.68
+ WMF + JBF [46]	22.25	19.80	11.94	4.61	12.46	10.02	6.45	1.91
+ LRI [96]	21.46	18.77	14.12	4.84	13.24	10.74	8.63	2.34
+ LRI + MF + BF [96]	22.01	17.68	13.26	4.81	14.09	9.81	7.80	2.40
+ LC [50]	20.39	16.60	10.56	3.98	12.59	9.05	6.56	1.95
+ Lev.stereo* [62]	22.22	19.52	12.45	4.60	13.38	10.73	7.39	2.20
+ Lev.stereo [62]	21.69	18.66	13.63	3.74	13.63	10.05	6.13	1.96
+ Smart-SGM* [65]	22.67	19.71	11.51	4.33	13.57	10.78	6.54	2.05
+ PBCP* [75]	23.97	21.56	19.36	6.54	14.12	11.72	12.07	3.10
+ PBCP [75]	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 10.3: Absolute improvement of disparity accuracy yielded by NLA, SGM algorithm, Middlebury v3 dataset.

Table 10.3 shows the results with SGM [24]. Being our SGM implementation based on AD-CENSUS 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 with more complex approaches such as SGM. Moreover, the DSI of the map to be filtered 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* [62]. The second one is a weighted sum of the contribution of the different scanlines, according to confidence, referred to as *Smart-SGM* [65]. The last one consists of a dynamic setting of the smoothness terms P1 and P2 according to confidence, referred to as *PBCP* [75]. 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 processing O1 measure. Moreover, our proposal works in the disparity domain, not requiring intermediate results from the SGM pipeline and it is general-purpose technique suited for any stereo algorithm. To better understand the key factors enabling for such improvements, we reports results concerning the use of different amounts of anchors and without the optional local aggregation step, deployed during the previous evaluations. Figure 10.4 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 for a notable improvement, reducing the error rate by more than 2% on AD-CENSUS, about 1.4% on MC-CNN and about 1% on SGM, with negligible overhead in terms of complexity and running time. We conclude this evaluation by running NLA on the output of the full *MC-CNN-acrt* algorithm (i.e., the disparity maps obtained by the top performing stereo pipeline proposed in [96]). It achieves 15.23 % (bad 1) error rate before refinement, dropping to 12.88% with CCNN confidence measure confirming to outperform all the other techniques: 15.02% WMF, 15.18% WMF + GF, 14.72% WMF + JBF while the remaining methods do not improve the baseline algorithm

Measure	AUC	Density
CCNN ξ=0.95	0.2104	53.62%
CCNN ξ-free	**0.2024**	**61.89%**

(a)

Measure	AUC	Density
CCNN ξ=0.95	0.1302	**67.77%**
CCNN ξ-free	**0.1289**	63.40%

(b)

Measure	AUC	Density
O1 ξ=0.40	0.1428	56.03%
O1 ξ-free	**0.1345**	**65.99%**

(c)

Table 10.4: AUC evaluation using hand-tuned threshold or random forest selection.

MC-CNN-acrt.

10.4.2 Evaluation of the threshold-free module

Having 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. To carry out this comparison, we recall a protocol commonly adopted when dealing with the evaluation of confidence measures, which is the ROC curve analysis by means of its Area Under the Curve (AUC) [32]. We compare the random forest classification method with respect to using a threshold ξ obtained by cross validation, as performed by other authors of [80]. In particular, we selected for the three algorithms threshold values of 0.95 (CCNN), 0.95 (CCNN) and 0.4 (O1) on the Middlebury v3 dataset to obtain the lowest average error on the dataset. Independently of adopting ξ or the random forest, being all the pixels labeled with only two values (e.g., 0 for UP, 1 for RP), in both cases ROC curve will look like a step response. Even if CCNN and O1 are very effective for their purpose, the AUC values of step-like curves will appear way higher with respect to optimal values, because of the large number of tying pixels, usually a penalty with this kind of evaluation protocol. Nevertheless, assessing which AUC is lower represents the only relevant information in this particular case.

Table 10.4 reports the AUC comparison between manual thresholding and random forest selection for the best performing confidence measure within the NLA framework and for the three stereo algorithm previously mentioned. The table shows how, for all the considered algorithm-confidence configura-

Stereo algorithm	All				Non-occ			
	bad 1(%)	bad 2(%)	RMSE	MAE	bad 1(%)	bad 2(%)	RMSE	MAE
AD-CENSUS	35.13	32.32	14.32	6.85	26.37	23.54	10.94	4.49
+ NLA + CCNN (ξ = 0.95)	**21.13**	**17.81**	8.30	3.20	14.15	11.20	5.36	1.85
+ NLA + CCNN (ξ-less)	21.16	17.82	**8.11**	**3.06**	**13.36**	**10.34**	**4.78**	**1.52**
MC-CNN [96]	25.95	23.88	14.83	6.32	16.02	13.99	10.94	3.27
+ NLA + CCNN (ξ = 0.95)	17.68	15.07	8.39	3.04	10.22	8.02	4.83	1.51
+ NLA + CCNN (ξ-less)	**16.39**	**13.45**	**7.39**	**2.59**	**9.98**	**7.66**	**4.54**	**1.36**
SGM [24]	24.38	22.00	13.18	5.33	14.52	12.14	7.96	2.49
+ NLA + O1 (ξ = 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 10.5: Absolute improvement of disparity accuracy yielded by NLA with hand-tuned threshold or random forest selection, AD-CENSUS, MC-CNN and SGM algorithms, Middlebury v3 dataset.

tions, the random forest framework is able to achieve a lower AUC and, thus, a better detection of correct matches. The table also reports the density of RP selected on the whole image. This more effective selection method also improves the overall NLA approach. Table 10.5 shows comparison between the results achieved by the manually selected threshold through cross validation, highlighting how the random forest selection strategy also increases, on average, the accuracy of the refined disparity maps.

10.4.3 Evaluation on KITTI 2015

In this section, we report experimental results on the KITTI 2015 training dataset [56], depicting outdoor environments very different from the Middlebury indoor scenes. We deploy for these experiments our full pipeline with 16 anchors, local aggregation and threshold-free selection of RP. Table 10.6 reports experimental results when refining disparity maps obtained by AD-CENSUS and MC-CNN algorithms. We compare our results with MF, WMF and LC. We can observe how, even on this very different dataset, the NLA framework is able to reduce the error rate of the raw disparity maps by nearly 26% (AD-CENSUS) and by nearly 10% (MC-CNN), notably outperforming the other refinement techniques. Table 10.7 shows refinement results by processing disparity maps of SGM. Being KITTI 2015 dataset very different with respect to Middlebury v3, we tuned P1 and P2 to 0.3 and 3 in order to obtain the most accurate results from the original SGM algorithm. Once more, we highlight how the average accuracy on the entire dataset with SGM is very similar to the one achieved by NLA processing disparity maps with AD-CENSUS. Being the environments depicted by KITTI 2015

Stereo algorithm	bad 3% - All AD-CENSUS	MC-CNN
Baseline	37.30	16.78
MF [63]	19.95	10.11
WMF [98]	21.03	10.38
LRI [96]	25.29	12.86
LRI +MF + BF [96]	18.90	10.47
LC [50]	14.92	10.47
NLA + CCNN	**11.17**	**6.90**

Table 10.6: Absolute improvement of disparity accuracy yielded by NLA, AD-CENSUS and MC-CNN algorithms, KITTI 2015 dataset.

more smooth with respect to indoor scenes considered before (e.g., the wide street plane, common term of all the stereo pairs), smoothing constraint enforced by SGM is stronger than the non-local refinement processed by NLA, being nonetheless capable of reaching a comparable degree of accuracy with significantly lower computational efforts. Focusing entirely on SGM results, we report, as for the Middlebury v3 evaluation, the improvements yielded by state-of-the-art confidence-based cost modulations proposed in [62, 65, 75], compared to NLA in its best configuration processing O1 confidence maps. Similarly to Middlebury v3, 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 outstanding results previously highlighted are confirmed even on KITTI 2015. Finally, without any particular optimization, on a standard CPU the overall execution time for the NLA requires only few seconds, which may dramatically drop with an optimized implementation granting real-time refinement.

10.5 Conclusions

In this work, 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 state-of-the-art 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. Exhaustive experimental results on challenging datasets

Stereo algorithm	bad 3% - All SGM
Baseline	10.78
MF [63]	8.73
WMF [98]	8.81
LRI [96]	10.12
LRI +MF + BF [96]	9.11
LC [50]	9.72
+ Lev.stereo* [62]	10.10
+ Lev.stereo [62]	9.52
+ Smart-SGM* [65]	8.47
+ PBCP* [75]	10.63
+ PBCP [75]	10.62
NLA + O1	**7.68**

Table 10.7: Absolute improvement of disparity accuracy yielded by NLA, SGM algorithm, KITTI 2015 dataset.

and state-of-the-art algorithms confirmed the superiority of this method with respect to any other technique. To the best of our knowledge, this is the first framework for disparity refinement leveraging on confidence measures, detecting outliers without hand-tuned threshold.

Chapter 11

Fusion of disparity maps using deep learning

The content of this chapter has been presented at the 4th International conference on 3D Vision (3DV 2016) - ”Deep Stereo Fusion: combining multiple disparity hypotheses with deep-learning”. Most relevant to the work shown in this chapter is [81] by Spyropoulos and Mordohai.

11.1 Introduction

Most stereo algorithms rely on a set of parameters or heuristics which perform very well in particular circumstances but yield poor results in others. A typical example is the size of the aggregation window used by local methods, which should be large when dealing with smooth frontal-parallel surfaces and smaller near depth discontinuities or slanted surfaces. Other methods, such as Semi Global Matching (SGM) [24] perform pretty well in smooth and slanted areas but may lead to artifacts near depth discontinuities. These observations lead some researchers [81, 61] to argue that the overall disparity accuracy can be improved exploiting redundancy in the input data by means of decision trees. In this work we follow the same intuition but following a completely different strategy. We propose Deep Stereo Fusion (DSF), a novel end-to-end methodology to predict a more reliable disparity map taking as input the output of multiple Stereo Matchers (SMs). Differently from [81], which is based on explicit features extraction from input data (e.g., confidence, matching costs, etc), DSF relies on deep learning, deploying a Convolutional Neural Network (CNN), aimed at processing only the

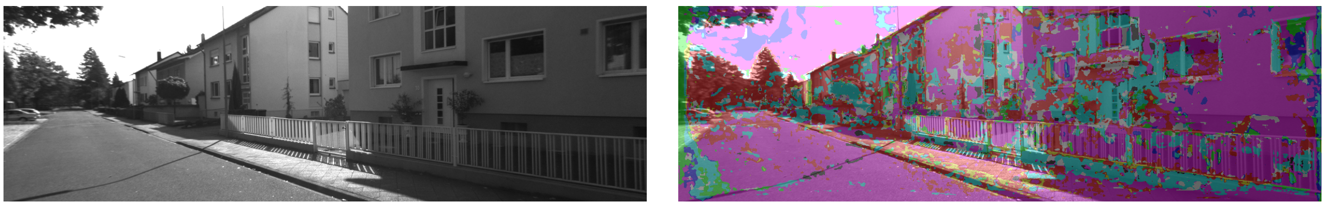

(a) (b)

Figure 11.1: Example of choice map.

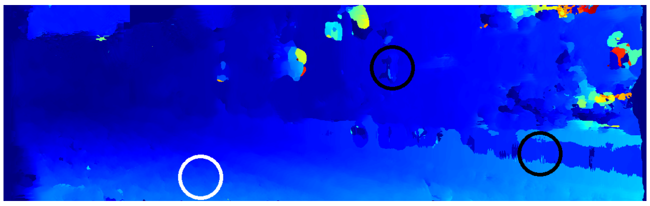
disparity maps provided by multiple SMs. This is carried out by tackling fusion as a multi-labeling classification problem, which enables to design a single classifier capable to predict, according to the the input sample, the reliability of each matcher. The outcome is a *choice map*, shown in Figure 11.1, encoding for each pixel which SM is selected at each location.

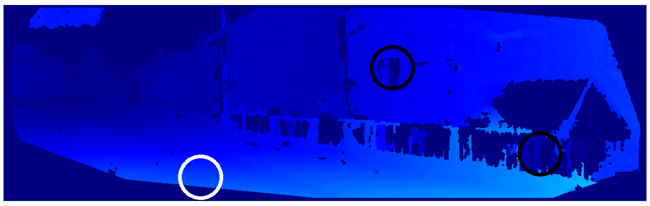
This strategy enables an elegant end-to-end training and testing procedure, conversely to other approaches which leverage on multiple classifiers (e.g., one per SM), requiring stand-alone training procedures. Moreover, it leads to a significantly faster response time. We evaluate DSF on the KITTI 2012 dataset [14, 15], comparing our results with state-of-the-art approach represented by Spyropoulos and Mordohai [81].

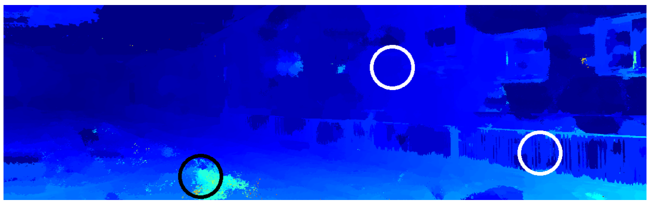
11.2 Deep Stereo Fusion

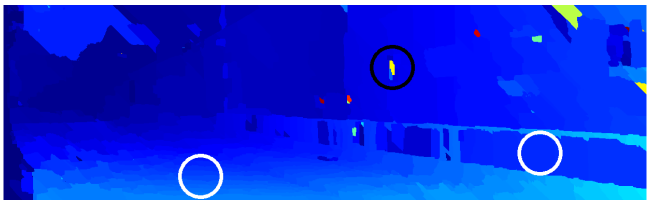
In this section we introduce the DSF framework that, given a set M of SMs, aims at combining multiple input disparity maps D_k, $k \in M$ to obtain a more accurate map D_F. By deeply analyzing this problem we decided to model it as a multi-label classification problem driven by two main assumption:

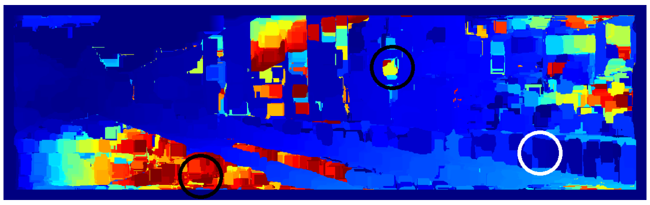
- Each SMs compute disparity assignments according to different cues. Usually, different behaviors can be observed locally on disparity maps by changing the stereo algorithm (e.g., near depth discontinuities, on low-textured areas, etc), as depicted in Figure 11.2. A framework aimed at merging different SMs should be able to distinguish, in any circumstance, the best assignment according to local properties of the input disparity maps. The different disparity maps can be seen as different *features* provided to the merging classifier.

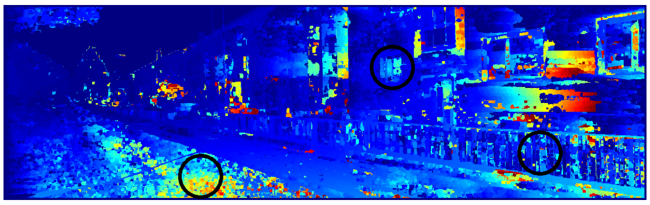

(a)

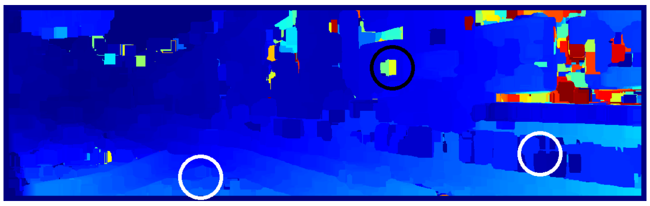

(b)


(c)


(d)


(e)


(f)


(g)

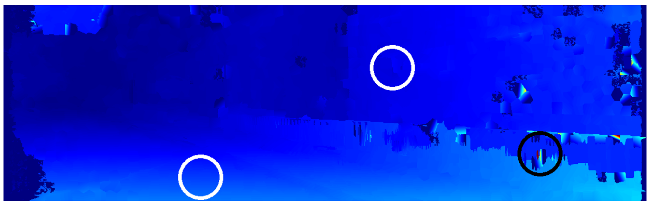

(h)

Figure 11.2: Qualitative comparison of different disparity maps obtained from different matchers.

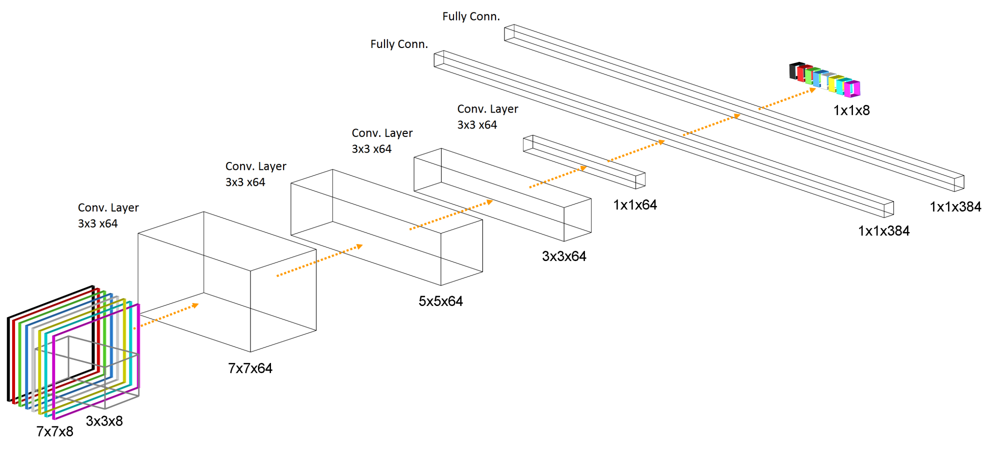


Figure 11.3: Architecture of DSF framework.

- Choosing among a pool of SMs can be casted within a classification problem: given a sample made of $m = |M|$ features, the framework choose the category (i.e., most accurate SM) it belongs to. Moreover, for a given pixel one or more matchers could possibly vote for the same, correct disparity assignment, leading to a multi-label classification problem.

According to these assumption and to the recent achievements in this field [54, 60, 95, 96], we train, on a large dataset with ground-truth, a deep architecture aimed at dealing with the outlined problem. For each pixel, we extract m square patches centered on the D_k, $k \in M$, disparity maps. These data are collected inside a 3D tensor, a Matcher Space Image (MSI). The DSF is trained on a large set of MSIs in order to distinguish the best matcher on the different samples and, thus, it provides a set of m scores. The chosen matcher will be the one with the highest score. The features representation encoded by MSI allows joint processing of data provided by the different matchers, similarly to what Spyropoulos and Mordohai achieved by an explicit computation of features encoding agreement among the SMs, but deployed within a single classifier instead of using a classifier per matcher [81]. We introduce the architecture of our network in Section 11.2.1, then we report details about our experimental evaluation, by defining the same set M of 8 matchers in Section 11.2.2 in order to compare our proposal with [81]. Section 11.2.3 provides details about the training phase and finally, in Section 11.3, we thoroughly compare our proposal to [81] on the KITTI 2012 [14, 15] dataset.

11.2.1 Proposed architecture

Figure 11.3 shows the architecture of DSF, organized as a m-channels network. The input to the network is a MSI of dimension $N \times N \times m$, with N the side of square patch extracted from the input disparity maps provided by the m methods. According to state-of-the-art methodology deployed for stereo, for our experiments we tuned N equal to 9. This means that DSF has a receptive field of 9×9, from which it will determine, for the central pixel, the optimal disparity assignment. This quite small receptive field allows us for generating a large amount of training samples from the available datasets for stereo [14, 15, 56, 71] without requiring synthetic data, needed by other techniques working on much bigger images [54]. the DSF then extracts a large number of features, by deploying four convolutional layers. Each layer is made of F convolutional kernels of size 3×3, each one followed by a Rectifier Linear Unit (ReLU).

$$ReLU(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases} \tag{11.1}$$

Since no padding or stride are applied, these four layers lead to a 1D output tensor, more precisely of size F. This feature vector is then forwarded to two fully-connected layers followed by ReLU. Finally, a classification layer is in charge of predicting which of the considered matchers propose the best disparity assignment, by a layer made of m neurons providing a 1D prediction vector C containing m values. The optimal disparity assignment for the central pixel inside the receptive field is assigned as

$$D_F(x, y) = D_w(x, y) \tag{11.2}$$

with w index of (one of) the matcher(s) achieving the highest score from the prediction layer.

By deploying our framework in fully-convolutional fashion [95, 96, 6], it can process in a single pass a full resolution MSI instead of $w \times h$ passes of 9×9 cropped data. The absence of pooling operation inside DSF leads to an output of size $(w-8) \times (h-8) \times m$, by applying a *0-padding* of size 4 around the whole input dimension w and h enables to obtain a 3D prediction tensor of size $w \times h \times m$, reducing the run time required by the DSF framework from several minutes to few seconds.

11.2.2 Combined Stereo Matchers

As SMs for the experimental evaluation we chose the same pool M of $m = 8$ of stereo algorithms deployed in [81] in order to be able to directly compare with it.

- DAISY: an approach based on a local descriptor aimed at wide baseline stereo matching [83]
- ELAS: Efficient LArge-Scale [16] stereo matching detects an initial set of reliable disparity assignments and fills remaining one with an appropriate triangulation
- FCVF: local method based on edge-preserving filtering of cost volume [31, 9] by means of the guided filter [22]
- MRF: global method expressed within a Markov Random Field framework [42]. Matching cost is Normalized Cross Correlation on 5×5 windows and smoothness penalty is modulated according to intensity difference between neighboring pixels
- SH-SOB21: Shiftable Window local aggregation on 21×21 patches. Matching cost is sum of absolute differences (SAD) of responses to vertical edge (e.g., Sobel filter on x direction)
- SH-SSD5: Shiftable Window aggregation on 5×5 boxes. Initial costs processed as sum of squared differences (SSD) of color intensities
- SH-ZNCC21: Shiftable Window aggregation on 21×21 boxes. Initial costs processed as Zero-Mean Cross-Correlation (ZNCC) on 21×21 patches
- SUPER-rSGM5: SGM [24] variant proposed by Spangerberg et al. [79]. Input images are census transformed on 5×5 patches. The output of the algorithm is further enhanced exploiting superpixels as described in [81], segmenting left image into SLIC superpixels [1] and fitting a plane for each segment with RANSAC.

11.2.3 Training procedure

In our experiments, we trained the DSF framework on the first 50 frames of the the KITTI 2012 dataset [14, 15] on cropped samples centered on pixels with available ground-truth values. This strategy provides more than 6.5 million MSIs of dimension $9 \times 9 \times 8$ for the training set. For each sample, a label vector of dimension 8 is assigned, encoding the correctness of a disparity assignment for each of the 8 SMs belonging to set M. If a given matcher provides a disparity assignment which differs from the ground-truth value for more than 3, it is labeled as wrong assignment, encoded as '0' in the label vector, '1' otherwise. Differently from the strategy adopted in [90], we directly trained our model on multi-label samples, in order to the reduce the amount of *single-label sample*s (i.e., pixels having only a single matcher proposing the correct assignment) due to the high overlap between the correct matches predicted by the different SMs. Otherwise, the amount of training data would be drastically reduced. We tuned DSF hyper-parameters, achieving the best results with $F = 64$ kernels for each convolutional layer and 384 neurons for the fully-connected layers (thus, 384 kernels 1×1 in the fully-convolutional model). During the training phase, we followed the Stochastic Gradient Descent (SGD). We optimized the Binary Cross Entropy loss function (BCE), extended to the multi-label classification problem as in [18, 90, 43], between output o of the network and label t on each sample i of the mini-batch B

$$BCE(o,t) = -\frac{1}{n}\sum_{i\in B}\sum_{k\in M}\Bigg(t[i][k]\log\left(o[i][k]\right) \\ + \left(1 - t[i][k]\right)\log\left(1 - o[i][k]\right)\Bigg) \tag{11.3}$$

by adding a sigmoid function S(x) (11.4) as final layer of the network

$$S(x) = \frac{1}{1 + e^{-x}} \tag{11.4}$$

Algorithm	Out-Noc	Out-All
DAISY	10.88%	12.86%
ELAS	19.90%	21.68%
FCVF	21.59%	22.46%
MRF	10.60%	12.58%
SH-SOB21	43.64%	44.86%
SH-SSD5	55.08%	56.04%
SH-ZNCC21	30.71%	29.20%
SUPER-rSGM5	7.85%	9.90%
DSF	6.34%	8.14%

Table 11.1: Comparison between DSF and combined algorithms on KITTI 2012 dataset.

We carried out 60 training *epochs*, with an initial *learning rate* of 0.003, decreased by a factor 10 after the 10^{th} epoch and after the 30^{th}, leading to a final learning rate of 3×10^{-5} for the final 30 epochs and a *momentum* of 0.9, inspired by [96] and confirmed by our experiments. The size of each mini-batch B was 128. The whole training procedure, carried out on a i7 4720HQ CPU, took approximatively 4 days. To speed-up the training procedure, DSF was first designed with fully-connected layers, which appears to be faster with respect to 1×1 convolutional layers during this phase. Once the network was trained, we replaced fully-connected layers with fully-convolutional ones [95, 96].

11.3 Experimental results

We evaluated the proposed DSF framework[1] on the remaining 144 frames of the KITTI 2012 dataset not used during the training phase by computing: the error rate of the merged disparity map over all pixels with available ground-truth (i.e., *disp_occ* data provided by KITTI 2012) and non-occluded areas (i.e., *disp_noc* data provided by KITTI 2012), reported, respectively, as *Out-All* and *Out-Noc* rates. Then, we compared our results with the proposal of Spyropoulos and Mordohai [81] on the same latest 97 frames of KITTI 2012 dataset (in [81] the training set was made of the first 97 frames out of 194).

Figure 11.4 plots the difference in terms of error rate for Out-Noc and Out-All between the most accurate SM in the M set, which is SUPER-rSGM5, and the output provided by DSF on the KITTI 2012 dataset, excluding the first 50 stereo pairs involved in training procedure. Positive values stands

[1]Source code available on the authors' website

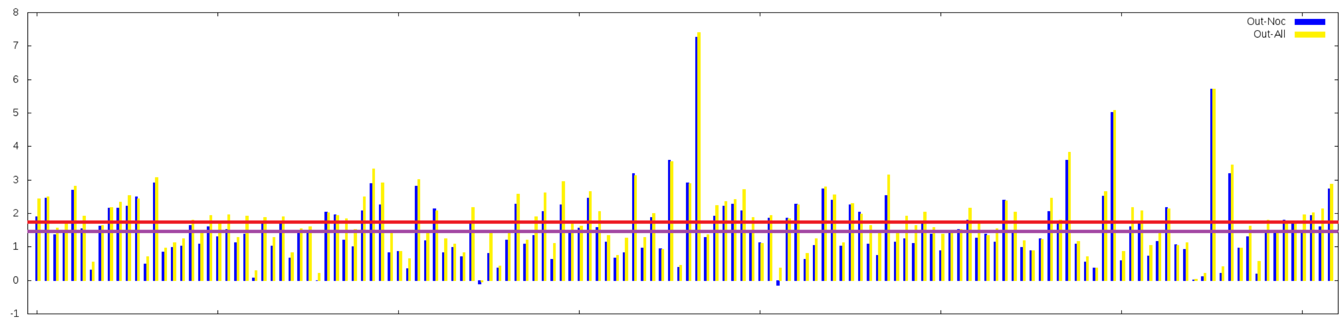


Figure 11.4: Absolute improvement of disparity accuracy yielded by DSF on KITTI 2012 dataset.

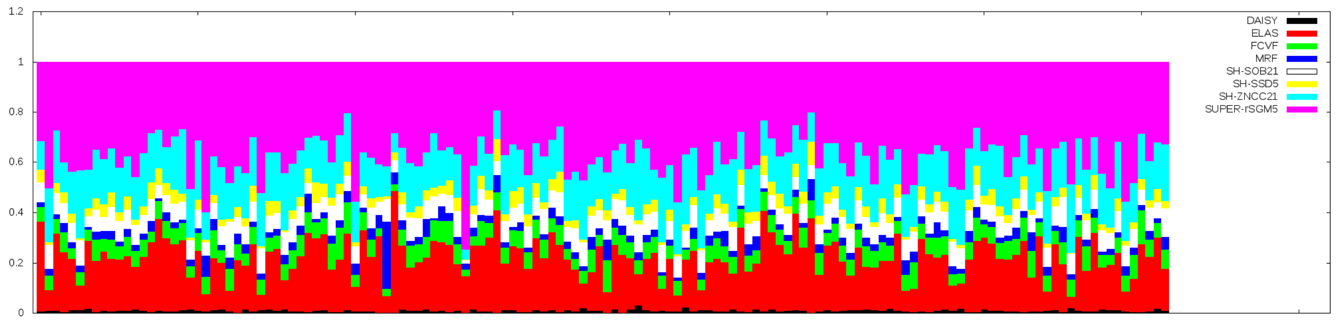


Figure 11.5: Occurrence rate of matchers.

for a reduction of the error rate carried out by our proposal. Except two cases, which are stereo pair 000099 for both Out-Noc and Out-All, 000132 for Out-Noc, DSF is able to effectively merge the 8 matchers and outperforms SUPER-rSGM5, with an absolute average error rate reduction of 1.51% Out-Noc and 1.75% Out-All. The error rate over the whole test set for SUPER-rSGM5 is 7.85% Out-Noc and 9.90% Out-All, while DSF achieves respectively 6.34% and 8.14%, with a relative improvement of 19.23% and 17.7% respectively. Table 11.1 shows average error rates for all the 8 SMs, as well for DSF.

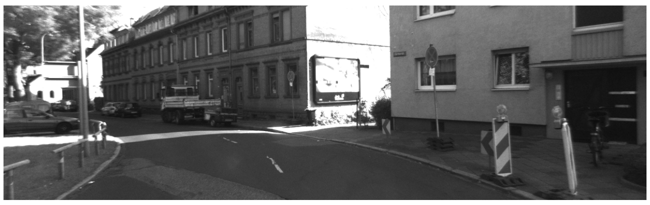
Figure 11.5 plots the occurrence rate of the matcher selected as winner by DSF. Each bar of the histogram represents a single stereo pairs from our test set, the different colors encode the 8 combined matchers according to the legend reported in the figure. We can notice how the most accurate algorithm, SUPER-rSGM5, is chosen most of the times, as we could expect, while two of the most accurate methods after SUPER-rSGM5 are not frequently selected. This fact is not necessarily inconsistent with the nature of the problem: a large subset of pixels which are correctly assigned by multiple SMs decreases the possibility of a matcher to be dominant with respect to the others. This is a direct consequence of the multi-label classification task we modeled to deal with the problem.

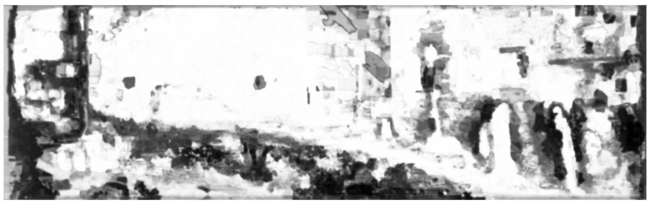
Algorithm	% of total pixels
DAISY	1.04%
ELAS	21.58%
FCVF	6.76%
MRF	3.31%
SH-SOB21	7.73%
SH-SSD5	3.84%
SH-ZNCC21	17.88%
SUPER-rSGM5	37.85%
Total	100.00%

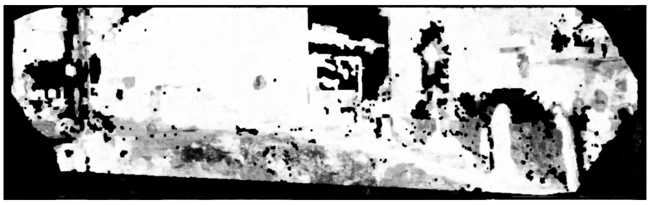
Table 11.2: Average occurrence rate of matchers.

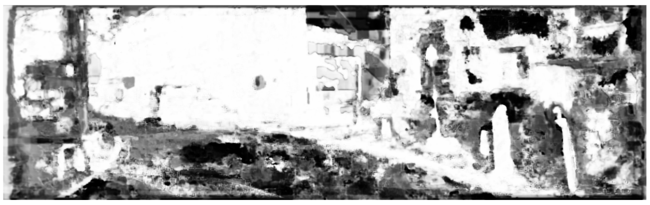
However, the reported error rates support the multi-labeling assumption adopted.

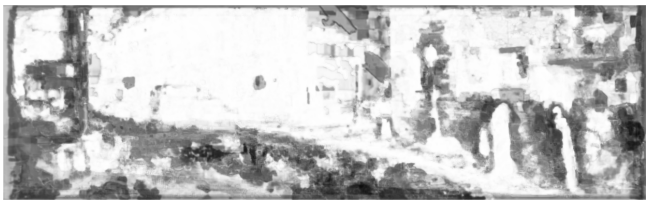
Table 11.2 summarizes these results, reporting the occurrence rate over our entire test set, confirming the trend previously highlighted with the histogram. Then, we computed the error rate on the last 97 stereo pairs from KITTI (i.e., from 000097 to 000193) in order to compare our proposal with the N8 framework proposed in [81]. Table 11.3 reports the result of this comparison. While the N8 ensemble classifiers[81] performs better when processing non-occluded pixels only, DSF slightly outperforms it when considering all pixels. Therefore, the two methods can be considered almost equivalent, but DSF performs better when dealing with occluded areas. In fact, Table 11.4 reports the error rate, on the same testing set, restricted to occluded areas only. N8 achieves a 96.28% error rate, while DSF 87.46%, outperforming it with an absolute error reduction of 8.82%. Our proposal does not rely on explicit features extraction, while N8 requires a set of features encoding matchers agreement and the Left-Right Consistency check (LRC) which might be available with the disparity maps. This latter fact makes our method suited for fusing disparity maps provided by any kind of stereo sensor, including out-of-the-box device. The LRC features available may also be responsible of the lower accuracy achieved by N8 on occluded area. According to [81], given the full disparity maps obtained by the 8 SMs, the time required to test the whole testing set (97 stereo pairs) is more than 3 hours as verified on the same CPU, leading to an average 100+ seconds per stereo pair. This means that each of the 8 classifiers takes about 12-13 seconds. On the other hand, the fully-convolutional nature of DSF, on the same CPU, makes out method much faster requiring about 10 seconds for each stereo pair (0.65 s on a Titan X GPU). Finally, Figure 11.6 depicts intermediate and final results provided by DSF, which are the single score maps related to each algorithm (b-i) and the final choice map (j).


(a)

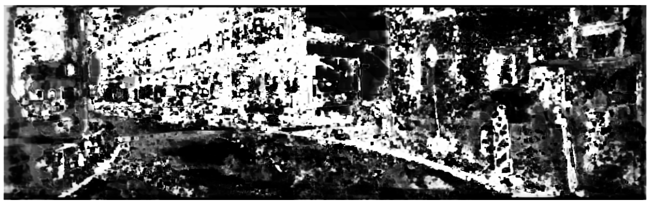

(b)

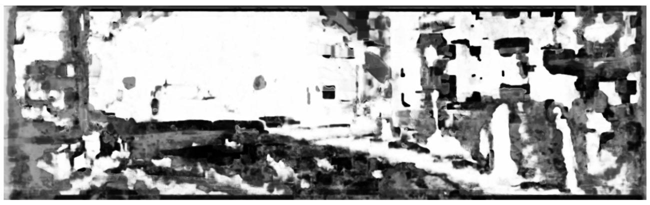

(c)

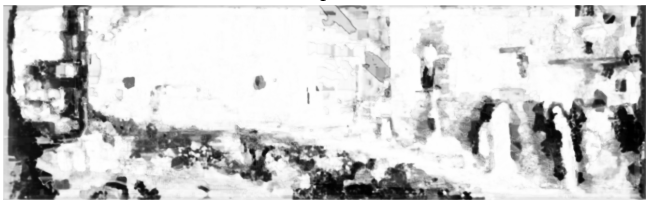

(d)


(e)

(f)


(g)


(h)


(i)

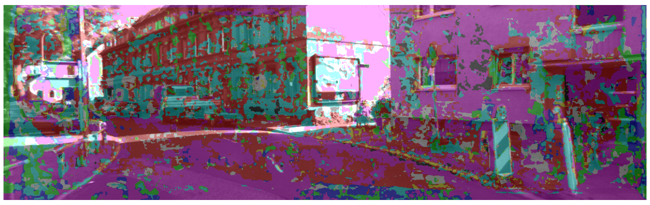

(j)

Figure 11.6: Qualitative examples of scores assigned by DSF to each matcher and final choice map.

Algorithm	SUPER-rSGM5	N8 [81]	DSF
Out-Noc	8.06%	**6.21%**	6.37%
Out-All	10.17%	8.21%	**8.18%**

Table 11.3: Comparison between error rates achieved by [81] and DSF, KITTI 2012 dataset.

Algorithm	N8 [81]	DSF
Occlusions	96.28%	**87.46%**

Table 11.4: Comparison between error rates achieved by [81] and DSF on occluded pixels, KITTI 2012 dataset.

11.4 Conclusions

In this work, we introduced Dense Stereo Fusion, a novel framework aimed at combining the output of several stereo algorithms. Our proposal allows for an elegant end-to-end training and testing of a single classifier, conversely to other approaches deploying multiple classifiers [81]. Experimental results confirm that DSF is able to outperform all the combined matchers and is almost equivalent, in terms of accuracy, to state-of-the-art framework proposed by Spyropoulos and Mordohai [81]. Nevertheless, our network clearly outperforms it on occluded pixels, proving to be more robust in such critical areas.

Acknowledgments I would like to acknowledge with thanks Dr. Aristotle Spyropoulos and Prof. Philippos Mordohai for providing disparity maps deployed for training and evaluation.

Chapter 12

Learning confidence measures in the wild

The content of this chapter has been presented at the 28th British Conference on Machine Vision (BMVC 2017) - ”Learning confidence measures in the wild”. Most relevant to the work shown in this chapter are the following papers: [32, 65, 66, 75, 94, 24, 96, 60].

12.1 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 [71], or expensive and appropriately registered active sensors, typically LIDAR, as done in [14, 56]. The first strategy provides dense (i.e., available for each pixel) 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 [54] with 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 [54]) to achieve top performance on standard datasets. Thus,

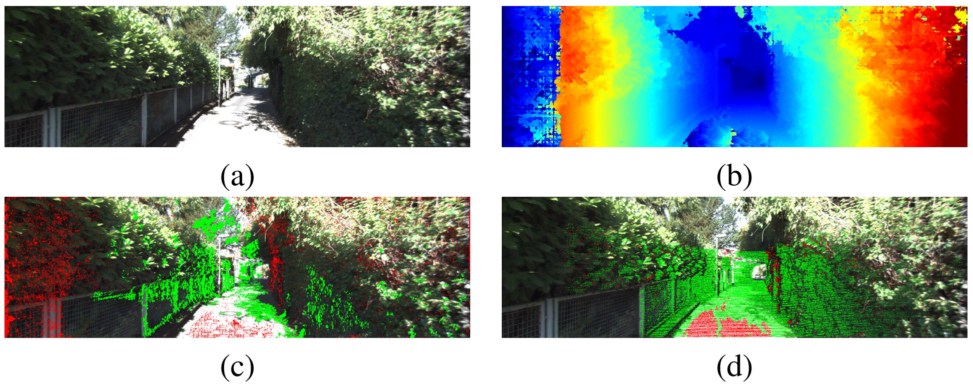


Figure 12.1: Self-supervised confidence labeling, SGM algorithm. (a) Reference image, (b) disparity map from SGM, (c) correct and wrong pixels according to ground-truth and (d) according to our method.

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. [60] 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 from two strong limitations. Firstly, it requires image sequences which are not always available. For instance, the Middlebury v3 dataset [71] 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 12.1 summarizes our proposal. Given a stereo pair (a) and the disparity map (b), we determine training labels (c) 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 [14], KITTI 15 [56] and Middlebury v3 [71], referred to as MIDD 14) with three stereo algorithms characterized by different accuracy (block-matching, MC-CNN [96] and SGM [24]) by training on labels inferred by our method three state-of-the-art confidence measures [65, 66, 75] 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 [60] but it also yields much more accurate confidence estimation.

12.2 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.

The effectiveness of a specific confidence measure is quantitatively assessed by means of a ROC curve analysis [32, 70] according to a standard procedure in this field [21, 80, 62, 65, 66, 75, 67, 68]. This strategy enables to determine how well a confidence estimator can discriminate between correct and wrong matches. The behavior of the curve itself encodes several important aspects of a confidence measure. For example, a flat portion of the curve indicates a large amount of pixels sharing the same estimated confidence. The extensive evaluation reported in [32] showed how different measures behave differently according to the processed cues as well as the adopted strategy. In particular, for the same pixel, different measures typically provide contradictory scores. This fact has been successfully exploited to infer much more effective confidence measures analyzing with random-forest [21, 80, 62, 65] or a CNN [68] a pool of not very effective confidence measures.

Our strategy relies on a set of conventional, yet according to the literature [70, 32, 10, 36, 80, 62] 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 [60], 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.

12.2.1 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 12.2.2. 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 square patch centered on pixel $\mathbf{p}$ (of size 25×25 in our experiments).

- **Average Peak Ratio (APKR)** [36]: 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.

$$APKR(\mathbf{p}) = \frac{1}{|N_\mathbf{p}|} \sum_{\mathbf{q} \in N_\mathbf{p}} \frac{c(\mathbf{q}, d_{2m}(\mathbf{p}))}{c(\mathbf{q}, d_1(\mathbf{p}))} \tag{12.1}$$

- **Left-Right Consistency (LRC)** [32, 70]: obtained by comparing the disparity of pixel $\mathbf{p}$ with the corresponding one $\mathbf{p}'$ on right disparity map.

$$LRC(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathcal{D}(\mathbf{p}) \neq \mathcal{D}^R(\mathbf{p}') \\ 1, & \text{otherwise} \end{cases} \tag{12.2}$$

- **Disparity deviation from median (MED)** [80]: represents the difference between disparity D on pixel $\mathbf{p}$ and the median disparity computed in a square neighborhood:

$$MED(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathcal{D}(\mathbf{p}) \neq median_{N_{\mathbf{p}}}(\mathcal{D}(\mathbf{p})) \\ 1, & \text{otherwise} \end{cases} \tag{12.3}$$

- **Uniqueness Constraint (UC)** [10]: 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:

$$UC(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathbf{p} \in Q \\ 1, & \text{otherwise} \end{cases} \tag{12.4}$$

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

- **Winner Margin (WMN)** [32, 70]: 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.

$$WMN(\mathbf{p}) = \frac{c_{2m}(\mathbf{p}) - c_1(\mathbf{p})}{\sum_d c_d(\mathbf{p})} \tag{12.5}$$

- **Distance to Left Border (DLB)** [62]: distance from the left border of the image, thresholded to the maximum disparity value $\mathcal{D}_{max}$ set for the stereo algorithm:

$$DLB(\mathbf{p}) = \begin{cases} 0, & \text{if } x < \mathcal{D}_{max} \\ 1, & \text{otherwise} \end{cases} \tag{12.6}$$

12.2.2 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 pixel $\in D$. 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:

$$\mathcal{C}_0 = \{\mathbf{p} \in D | 0 \leq \mathcal{C}(\mathbf{p}) \leq \delta_0\}, \quad \mathcal{C}_1 = \{\mathbf{p} \in D | 1 - \delta_1 \leq \mathcal{C}(\mathbf{p}) \leq 1\} \tag{12.7}$$

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:

$$\mathcal{G}_0 = \bigcap_{\mathcal{C}^k \in \mathcal{P}_0} \mathcal{C}_0^k, \qquad \mathcal{G}_1 = \bigcap_{\mathcal{C}^k \in \mathcal{P}_1} \mathcal{C}_1^k \tag{12.8}$$

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.

12.3 Experimental Results

In this section, we assess 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 [60] as well as using ground-truth data and comparing their performance

KITTI 12	AD-CENSUS		MC-CNN		SGM	
Method	A	D / D∩GT	A	D / D∩GT	A	D / D∩GT
SELF [60]	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 12.1: Analysis of training labels inferred by SELF [60] and confidence measures.

by means of ROC analysis. Regarding the datasets, we consider KITTI 12 [14], KITTI 15 [56] and MIDD 14 [71]. As confidence measures we choose the three top-performing methods known in literature: O1 [65], CCNN [66] and PBCP [75]. 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 [62]. Concerning the stereo algorithms, we consider AD-CENSUS, MC-CNN and SGM.

12.3.1 Evaluation protocol and training data

We follow AUC protocol [32] to quantitatively evaluate our proposal. Confidence measures are trained in most works in this field [21, 62, 65, 75] by selecting eight stereo pairs from KITTI 12 dataset: 43, 71, 82, 87, 94, 120, 122 and 180th. These images with ground-truth labels provide about 724K training samples. According to SELF [60], on the extended eight sequences available on KITTI 12 corresponding to the 8 stereo pairs 43, 71, 82, 87, 94, 120, 122 and 180th, we generate training labels following the protocol described by the authors. For all considered sequences there are 21 stereo pairs available, excluding 82th containing only 16. On such 163 stereo pairs SELF extracts a huge amount of training labels: about 25M for AD-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 AD-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 12.1 we can notice that our training samples are always more accurate. This fact

KITTI 12	AD-CENSUS (ϵ=38.6%)			MC-CNN (ϵ=16.9%)			SGM (ϵ=9.1%)		
measure	GT	[60]	Prop.	GT	[60]	Prop.	GT	[60]	Prop.
O1 [65]	0.116	0.165	0.163	0.025	0.046	0.042	0.016	0.031	0.022
CCNN [66]	0.118	0.250	0.128	0.028	0.089	0.029	0.032	0.084	0.023
PBCP [75]	0.125	0.201	0.138	0.029	0.044	0.040	0.029	0.037	0.035
APKR [36]		0.166			0.048			0.030	
opt.		0.094			0.017			0.005	

KITTI 15	AD-CENSUS (ϵ=35.4%)			MC-CNN (ϵ=15.4%)			SGM (ϵ=13.7%)		
measure	GT	[60]	Prop.	GT	[60]	Prop.	GT	[60]	Prop.
O1 [65]	0.109	0.172	0.147	0.031	0.059	0.046	0.021	0.038	0.027
CCNN [66]	0.113	0.266	0.120	0.036	0.102	0.035	0.044	0.072	0.029
PBCP [75]	0.122	0.209	0.151	0.035	0.053	0.047	0.031	0.035	0.037
APKR [36]		0.147			0.049			0.036	
opt.		0.083			0.019			0.007	

MIDD 14	AD-CENSUS(ϵ=37.8%)			MC-CNN (ϵ=26.7%)			SGM (ϵ=26.9%)		
measure	GT	[60]	Prop.	GT	[60]	Prop.	GT	[60]	Prop.
O1 [65]	0.126	0.180	0.154	0.073	0.125	0.097	0.085	0.133	0.102
CCNN [66]	0.128	0.254	0.123	0.072	0.179	0.069	0.122	0.216	0.088
PBCP [75]	0.119	0.169	0.123	0.067	0.084	0.078	0.145	0.148	0.148
APKR [36]		0.137			0.074			0.100	
opt.		0.090			0.046			0.045	

Table 12.2: AUC analysis for O1 [65], CCNN [66] and PBCP [75] trained with different strategies, AD-CENSUS, MC-CNN and SGM algorithms, KITTI 2012, KITTI 2015 and Middlebury v3 datasets.

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$\cap$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 12.1 in the left and upper side of the disparity map. Moreover as reported in Figure 12.2, 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 compared to deploy ground-truth labels for training confidence measures as will be detailed in the next section.

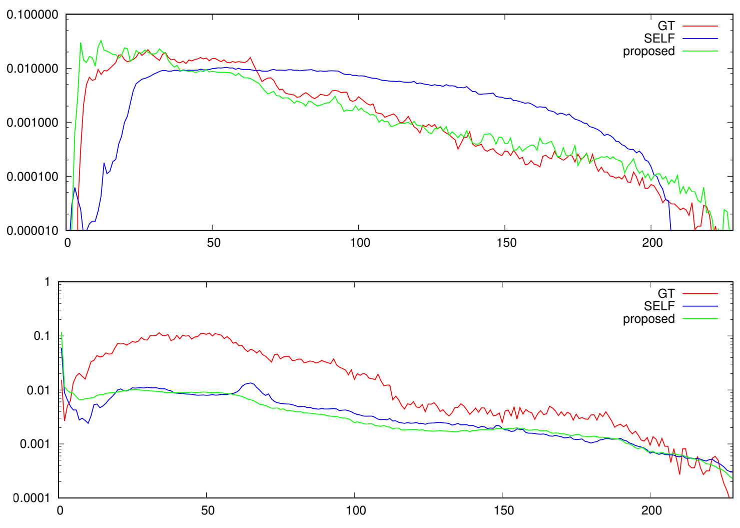


Figure 12.2: Distribution of training labels with SGM using SELF [60] or confidence measures.

12.3.2 Quantitative evaluation and analysis of training data

In this section we exhaustively compare our proposal with SELF [60] on three datasets KITTI 12, KITTI 15 and MIDD 14 and three algorithms for training the three state-of-the-art confidence measures O1 [65], CCNN [66] and PBCP [75] 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 12.2. We include in our evaluation APKR [36], the most effective confidence measure within the pool of confidence measures deployed for selecting labels as described in Section 12.2.1. Being 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.

The testing 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 compared 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 achieves better results in two cases (with CCNN) while SELF never yields better confidence estimation. The testing on MIDD 14 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 AD-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 12.2 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 12.2 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 see 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 12.3 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.

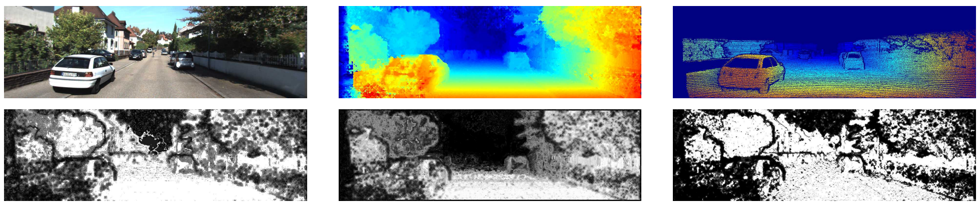


Figure 12.3: Qualitative results of confidence maps trained on groundtruth, SELF [60] or confidence measures.

12.4 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 13

Unsupervised adaptation of deep stereo

The content of this chapter has been presented at the IEEE International Conference on Computer Vision (ICCV 2017) - ”Unsupervised adaptation of deep stereo”. Most relevant to the work shown in this chapter are the following papers: [32, 66, 54, 94, 24].

13.1 Introduction

The widespread adoption of deep learning in computer vision has also affected stereo vision. In particular, Convolutional Neural Networks (CNNs) proved very effective in computing matching costs between the patches of a stereo pair [96, 6, 45], although these novel approaches still require to be plugged into well established disparity optimization and refinement pipelines (e.g., [96]) to achieve state-of-the-art accuracy. A ground-breaking forward step is DispNet , [54], a deep architecture trained from scratch to regress dense disparity measurements end-to-end from image pairs, thereby dismissing all the machinery traditionally deployed to optimize/refine disparities and speeding up the computation considerably. This approach is further improved by GC-net [35], which implements the well-known steps of a stereo pipeline casting it into end-to-end fashion. However, due to the high capacity of the model as well as the input consisting in image pairs rather than patch pairs, this approach mandates a huge amount of supervised training data not available in existing datasets (i.e. tens of thousands of stereo pairs with ground-truth). Therefore, the network is trained leveraging on

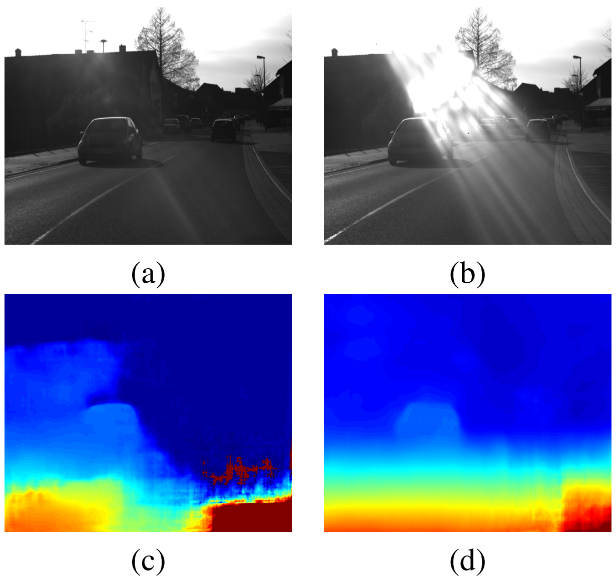


Figure 13.1: Qualitative comparison between Dispnet [54] before (c) and after (d) adaptation.

large synthetic datasets generated by computer graphics [54] and then fine-tuned on fewer available real data with ground truth [14, 56] in order to improve effectiveness in the addressed scenario [54]. Yet, the performance of a deep stereo model may deteriorate substantially when the supervised data needed to perform adaptation to a new environment are not available. For example, Figure 13.1 (c) shows how DispNet [54] yields gross errors on a stereo pair of a dataset [55] lacking the ground-truth information to fine-tune the network. Unfortunately, besides a few research datasets, stereo pairs with ground-truth disparities are quite rarely available as well as cumbersome and expensive to create in any practical settings. This state of affairs may limit deployability of deep stereo architectures significantly.

To tackle the above mentioned issue, in this work we propose a novel unsupervised adaptation approach that enables to fine-tune a deep stereo network without any ground-truth information. The first key observation in our approach is that computer vision researchers have pursued for decades the development of general-purpose stereo correspondence algorithms that do not require any adaptation to be deployed in different scenarios. The second is that, although traditional stereo algorithms exhibit well-known shortcomings in specific conditions (e.g., occlusions, texture-less areas, photometric distortions ..), recent state-of-the-art confidence measures, more often than not relying on machine learning [66, 75, 80, 62, 65], can effectively highlight uncertain disparity assignments. Thus, we propose to leverage on traditional stereo algorithms and state-of-the-art confidence measures in order to fine-tune a deep stereo model based on disparities provided by standard stereo algorithms that are deemed as highly reliable by the confidence measure. Figure 13.1 (d) shows that our unsupervised adaptation approach can improve dramatically the output provided by DispNet [54] on a dataset lacking the ground-truth to fine-tune the network with supervision. Our approach deploys a loss function that, taking as target variables the disparity measurements provided by the stereo algorithm, weighs the error contribution associated with each prediction according to the estimated confidence in the corresponding target value. Moreover, we introduce a smoothing term in the loss that penalizes dissimilar predictions at nearby spatial locations, based on the conjecture that as high confidence target disparities may turn out sparse, enforcing smoothness helps propagating the predictions from high confidence locations towards low confidence ones. The effectiveness of our unsupervised technique is demonstrated by experimental evaluation on KITTI datasets [14, 56] and Middlebury v3 [71], assessing both adaptation ability and generalization to new data. We also report qualitative results on challenging images [55], so to highlight the need for an effective unsupervised adaptation methodology.

13.2 Unsupervised Adaptation

As vouched by the experimental findings reported in Sec. 13.3.2, 13.3.3, the main issue with large networks aimed at dense disparity estimation from image pairs is robustness to different deployment scenarios. In fact, when dealing with environments quite different from those employed to train the network, the accuracy may quickly drop and the model would need to be adapted to the new settings in order to achieve comparable performance. This step requires a dataset with ground truth that is seldom available in practical applications.

Our proposal tackles this issue by enabling adaptation of the network in an unsupervised fashion by leveraging on a conventional stereo algorithm and a reliable confidence measure. Starting from a pre-trained model, we fine-tune it to minimize a novel loss function (L) made out of two terms: a *Confidence Guided Loss* ($\mathcal{C}_L$) and a *Smoothing Term* ($\mathcal{S}$), with hyper-parameter λ weighing the contribution of the latter:

$$L = \mathcal{C}_L + \lambda * \mathcal{S} \tag{13.1}$$

Such a loss function enables to adapt the pre-trained model to deal with any new environment by simply processing a pool of stereo pairs and without requiring any ground-truth information.

13.2.1 Confidence Guided Loss

Once trained on very large datasets with ground truth, end-to-end stereo networks like DispNet can predict a disparity map directly from the input stereo pair. As reported in [54], the authors firstly trained the network on a huge synthetic generated dataset of 25000 image pairs with valid disparity label for each pixel, then adapted it to a different environment through a much smaller amount of image pairs endowed with sparse ground truth labels (i.e. the nearly 200 training images of KITTI2012 [14] where only a subset of pixels have meaningful disparity values). To account for the missing values within the images used to fine-tune the network they simply set the loss function to 0 at such locations, given that, even if only a small portion of output receives meaningful gradients, the system is still able to adapt fairly well to the new scenario and hence to ameliorate its overall accuracy.

However, despite the elegance and effectiveness of such methodology, for most real world scenarios the adaptation would be impossible because we can not expect availability of enough ground truth data, even at sparse locations. On the other hand, what we could reasonably expect is availability of stereo pairs acquired in the field. Hence, the first contribution of our work is to fill this gap by providing a methodology to obtain disparity labels for the adaptation phase using conventional stereo algorithms (e.g., AD-CENSUS [94] or SGM [24]). Unfortunately a network like DispNet trained on the raw output of AD-CENSUS or SGM would, at best, learn to imitate the overall behavior of the chosen stereo algorithm, including its intrinsic shortcomings, thus leading to unsatisfactory results. However, by taking advantage of effective confidence measures recently proposed, like [66], we can discriminate between reliable and unreliable disparity measurements, to select the former and fine-tune the model using such smaller and sparse set of points as if they were ground truth labels.

Given an input stereo pair I_L and I_R, we denote as $\tilde{D}$ the disparity map predicted by the stereo network, D the disparity map computed by a conventional stereo algorithm and C a confidence map measuring the reliability of each element in D, with $C(p) \in [0,1] \forall p \in P$, with P the set of all spatial locations. We define the *Confidence Guided Loss* ($\mathcal{C}_L$) as:

$$\mathcal{C}_L = \frac{1}{|P|} \sum_{p \in P} \mathcal{E}\left(p\right) \tag{13.2}$$

$$\mathcal{E}\left(p\right) = \begin{cases} C\left(p\right) \cdot |\tilde{D}\left(p\right) - D\left(p\right)| & \text{if } C\left(p\right) \geq \tau \\ 0 & \text{if } C\left(p\right) < \tau \end{cases} \tag{13.3}$$

$\tau \in [0,1]$ being a hyper-parameter of our method that controls the sparseness and reliability of the disparity measurements provided by the stereo algorithm that act as target variables in our learning process. Higher values of τ let fewer measurements contribute to the loss but with a lower probability of injecting wrong disparities into the process. It is worth pointing out that should the confidence measure behave perfectly, minimizing such loss function with an appropriate τ might be thought of as to fine-tuning on sparse ground truth data with the same amount of samples.

13.2.2 Smoothness Term

Although fine-tuning on sparse ground truth data, as proposed in [54], does improve the disparities predicted in unseen scenarios, it may still be regarded as an approximation of the ideal optimization process that would leverage on dense labels. Therefore, to compensate for the sparsity of target measurements, we introduce in the loss function an additional smoothness term $\mathcal{S}$ that tends to penalize diverse predictions at nearby spatial locations.

Given a distance function $\mathcal{D}\left(p,q\right)$ between two spatial locations p,q, we denote as N_p the set of neighbours of spatial location p: $N_p = \{q | \mathcal{D}\left(p,q\right) < \delta\}$. We compute the average absolute difference between the disparity predicted at p and those predicted at each $q \in N_p$:

$$E\left(p\right) = \frac{1}{|N_p|} \sum_{q \in N_p} |\tilde{D}(q) - \tilde{D}(p)| \tag{13.4}$$

The smoothing term is obtained by averaging $E\left(p\right)$ across all spatial locations:

$$\mathcal{S} = \frac{1}{|P|} \sum_{p \in P} E\left(p\right) \tag{13.5}$$

The distance function, $\mathcal{D}$, as well as the radius of the neighborhood, δ, are hyper-parameters of the proposed smoothing term. It is worth observing that, optimized alone, such term would produce a uniform disparity map as output. However, when carefully weighted in conjunction with $\mathcal{C}_L$, it helps spreading the information associated with sparse target measurements towards the other spatial locations.

13.3 Experimental Results

To validate our proposal we choose DispNet-Corr1D [54], from now on referred to as DispNet, as network architecture for end-to-end disparity regression, AD-CENSUS [94] and SGM [24] as off-the-shelf stereo algorithms and CCCN [66] as confidence estimator. The choice of the confidence estimator has been driven by its top performance and broad applicability, the latter due to the method requiring only the disparity map to estimate the confidence. As for Dispnet, we modified the original authors code to incorporate our novel loss formulation and fine tuned the network starting from the publicly available weights obtained after training on synthetic data only. For CCCN we used the original implementation as well as the provided weights without any retraining or fine tuning. Lastly, we used a custom implementation of SGM and AD-CENSUS based on the original papers. We will firstly introduce the procedure used to properly tune the hyper-parameters of our learning process, then we will show that our method not only allows to effectively fine-tune the chosen disparity regression network without any labeled data but also does improve the generalization capability of the model across similar domains.

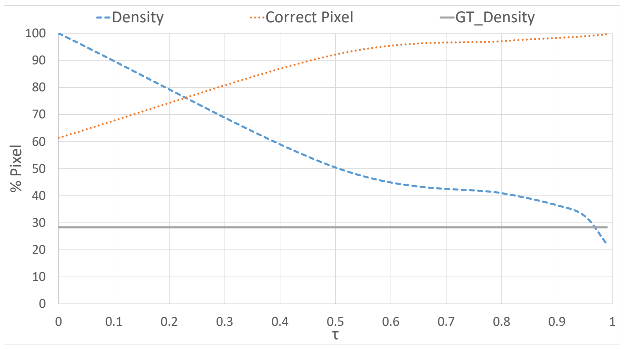


Figure 13.2: Distribution of samples with different confidence thresholds. On x axis, τ values, on y percentage of the total pixels. In blue, we report density of selected pixels , compared to ground-truth density in grey. In orange the percentage of correct pixels among those selected.

13.3.1 Learning Process

To find optimal values for the hyper-parameters of our learning machinery, we choose to rely on the commonly used KITTI datasets [14, 56]. In particular, to get insights on the training and generalization performance of our method, we have used the images from KITTI 2012 as training set and those from KITTI2015 as test set. For all our experiments we initialize DispNet according to the weights obtained after 1200000 training steps on synthetic data and publicly released by the authors. In the experiments dealing with hyper-parameters tuning, we have used AD-CENSUS [94] as stereo algorithm to compute the disparity maps that are then validated by the chosen confidence measure [66] in order to sift-out the actual target variables.

For these experiments, to obtain useful insights in an acceptable training time, we carried out just 10000 fine tuning steps for each test configuration with batch size equal to 4 on the 194 KITTI 2012 images($\sim$200 epochs) and feeding the network with random crops of the original images of size 768×384. To increase the variety of the training set, we perform random data augmentation (color, brightness and contrast transformations) as done by the authors of [54]. We use ADAM [38] as optimizer with an initial learning rate equal to 0.0001 and an exponential decay every 2000 step with $\gamma = 0.5$.

The first parameter that needs to be carefully tuned is τ, which allows for filtering out wrong disparity assignments according to the scores provided by confidence measure. Figure 13.2 shows that even for high values of τ we can get disparity maps denser than the available ground truth data for KITTI

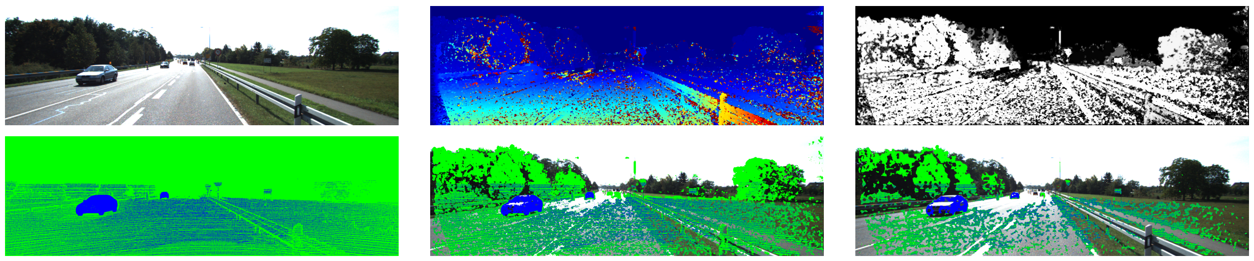


Figure 13.3: Spatial distribution of training samples with diffferent confidence thresholds. On top, reference image (000073 from KITTI 2015), disparity map from AD-CENSUS algorithm and confindence map from CCNN measure. On bottom, in green pixels selected by τ equal to 0, 0.5 and 0.99, in blue their intersection with the available ground-truth.

2012. Moreover, cross comparing such points with the available sparse ground truth, we can observe that, for quite high τ values (i.e. > 0.9), nearly 100% of the points selected by our method that appear at available ground truth locations carry correct disparities. Although we cannot assess upon the correctness of the points selected by our method that do not coincide with available ground truth locations, there seems to be no reason to believe that the confidence measure would behave much differently therein. Therefore, Figure 13.2 seems to support the intuition that high confidence disparities are very likely correct and hence may effectively act as ”surrogate” ground truth data within our unsupervised learning process. Moreover, compared to the sparse ground truth data available in the KITTI datasets, a favourable property of our selected disparities is the larger spread across the whole image. This enables our method to *look at* portions of the scene seldom included in ground truth data. From Figure 13.3 we can notice that for high values of τ, even though the density of our disparity map is similar (or slightly lower) with respect to the ground truth data, we gather samples more spread across all the image. For example, even with $\tau = 0.99$, the top of the trees on the left and one of the farthest car in the scene are always visible in our unsupervised disparity map but not included in the available ground truth data. We will show in section 13.3.3 that this property leads to better generalization performance.

Given these preliminary observations, we tried different values for τ and report the training and generalization error in Figure 13.4. We observe a perfectly smooth descending behavior of the Training and Generalization error (percentage of wrongly predicted pixel) with increasing value of τ. Given this outcome we can conclude that the higher the value of τ the better the performance of the network. Thus, we set $\tau = 0.99$. Such value selects, on this training set, 22.07% of available pixels (slightly

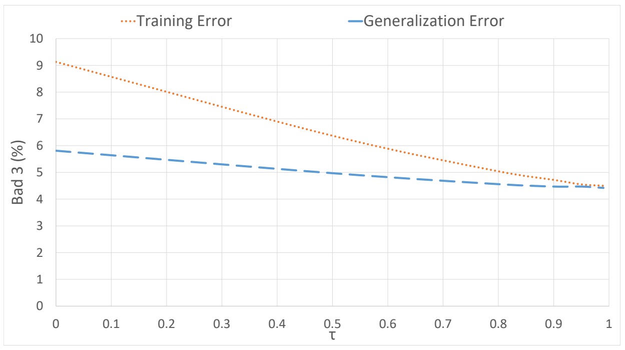


Figure 13.4: Performance of the network after 10000 steps of adaptation for different values of confidence thresholds.

less than the available ground truth points) with an accuracy of the pixels for which we have a ground truth disparity annotation equal to 99.65%. Having set τ, we evaluate how a proper tuning of the smoothing term of our loss function enables to improve the overall performance. For these experiments we choose as distance function $\mathcal{D}(p, q)$ the $L1$ distance and $\delta = 1$. Keeping the same set-up as used to tune τ (Figure 13.4), we perform experiments on the KITTI 2012 dataset with different values of $\lambda \in [0, 1]$, the results reported in Figure 13.5. Looking at the training error it is clear how our regularization term can improve the performance of the network. However the value of λ must be kept < 0.6 in order to not over-smooth predictions. More importantly, even the generalization performance of the network is influenced by the magnitude of λ, with the lowest generalization error obtained using $\lambda = 0.1$. We believe that the explanation for this behavior is that the network compensates for the missing target measurements by creating a useful training signal thanks to the smoothing factor that propagates information from existing target measurements to nearby locations. However, the value of λ must be kept low so to not overcome the contribution of the confidence guided loss.

From the careful tuning outlined so far, we found that the best configuration for our unsupervised framework is $\tau = 0.99$ and $\lambda = 0.1$ using $L1$ distance and $\delta = 1$.

13.3.2 Adaptation

Given the best configuration of hyper-parameters, we evaluate the effectiveness of our unsupervised adaptation methodology when dealing with never seen before environments. To assess performance, on one hand we assume the KITTI 2012 training dataset as a known scenario on which ground-truth

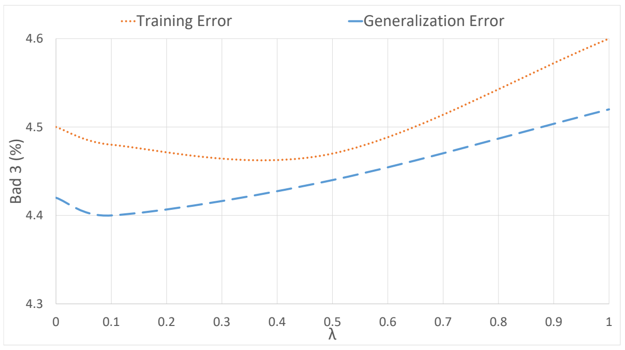


Figure 13.5: Performance of the network after 10000 steps of adaptation for different values of λ and fixed confidence threshold.

Stereo	KITTI 2015		Middlebury v3	
algorithm	bad 3(%)	MAE	bad 1(%)	MAE
AD-CENSUS [94]	35.41	20.11	30.66	10.29
SGM [24]	13.68	6.14	20.71	5.73
DispNet	7.46	1.27	32.82	2.74
DispNet K12-GT	4.58	1.15	40.21	2.94
DispNet CENSUS	**4.02**	**0.76**	25.38	**2.47**
DispNet SGM	4.21	0.85	**22.91**	2.66

Table 13.1: Absolute improvement of disparity accuracy yielded by adaptation on KITTI 2015 and Middlebury v3 datasets.

data to fine-tune DispNet are available. On the other hand, we assume KITTI 2015 and Middlebury v3 as novel environments with no ground-truth available for fine-tuning. Thus, we perform unsupervised adaptation on KITTI 2015 and Middlebury v3 and compare accuracy with respect to both the original DispNet architecture (i.e., trained on synthetic data only) as well as to DispNet fine-tuned on KITTI 2012 by the available ground truth. Following this protocol, we can prove that our unsupervised adaptation improves significantly the accuracy of the original network. i.e. that unsupervised fine-tuning is feasible and works well, and that, in absence of ground-truth data, unsupervised fine-tuning on the addressed scenario is more effective than transferring a supervised fine-tuning from another annotated (and quite similar) environment[1]. To assess the performance of our proposal with different stereo algorithms, in these experiments we use AD-CENSUS and Semi-Global Matching (SGM), the latter leveraging as data term the final cost computed by AD-CENSUS and with smoothing penalties $P1 = 0.2$ and $P2 = 0.5$, being the matching costs between 0 and 1.

Table 13.1 reports the error rate (i.e., the percentage of pixels having an error larger than ε) and the

[1]This protocol is also compliant to the KITTI submission rules, which forbid to process the test data in any manner before submitting results.

average disparity error on the entire KITTI 2015 ($\varepsilon = 3$) and Middlebury v3 ($\varepsilon = 1$) training sets. For both datasets we use the standard evaluation protocol; for Middlebury we resized the stereo pairs to quarter resolution to have a disparity range similar to the KITTI datasets. We highlight how, regardless of the chosen off-the-shelf stereo algorithm being either AD-CENSUS or SGM, our unsupervised adaptation approach achieves higher accuracy with respect to the original DispNet architecture as well as to DispNet fine-tuned supervisedly on KITTI 2012 on both datasets and according to both metrics. Table 13.1 reports also on the first two rows the accuracy of the two stereo algorithms deployed for adaptation: their very high error rates demonstrate how the proposed confidence guided loss and smoothness term can handle effectively the high number of wrong assignments within the disparity maps yielded by the stereo algorithms that provide the ”raw” target variables to the learning process.

As for the results on KITTI 2015, it is worth highlighting that our approach is able to outperform DispNet fine-tuned through the ground-truth data of a very similar dataset (i.e., KITTI 2012). Thus, despite the high similarity between the two datasets in terms of image content, which renders fine-tuning on KITTI 2012 beneficial to DispNet, as vouched by the nearly 3% decrease of the error rate and the reduced average disparity error, our proposed unsupervised adaptation turns out more effective obtaining an even higher accuracy. Moreover, we point out how our unsupervised adaptation method is effective with both the considered off-the-shelf stereo algorithms, which are characterized by quite different error rates and behaviors. This is particularly relevant to AD-CENSUS, whose average error rate is quite high (i.e., on average, more than 35% of wrong pixels in each map).

This experiment shows that our methodology can be deployed to effectively fine-tune a deep stereo network without the need of ground truth disparities. Moreover our confidence guided loss proves to be able to drastically improve the performance of a deep stereo system even if the raw target values used for the unsupervised tuning are very noisy, such as it the case of the disparity map computed by AD-CENSUS. Interestingly, DispNet adapted from such noisy data yields more accurate disparity maps compared to undergoing a fine tuned based on ground truth data from a different though similar scenario. In a further experiment we included in our usupervised fine-tuning of DispNet based on AD-CENSUS only the stereo pairs of the KITTI 2015 training dataset with available ground-truth, i.e. given the scene labeled as ”000000”, we process unspervisedly only the ”000000_10” stereo pairs

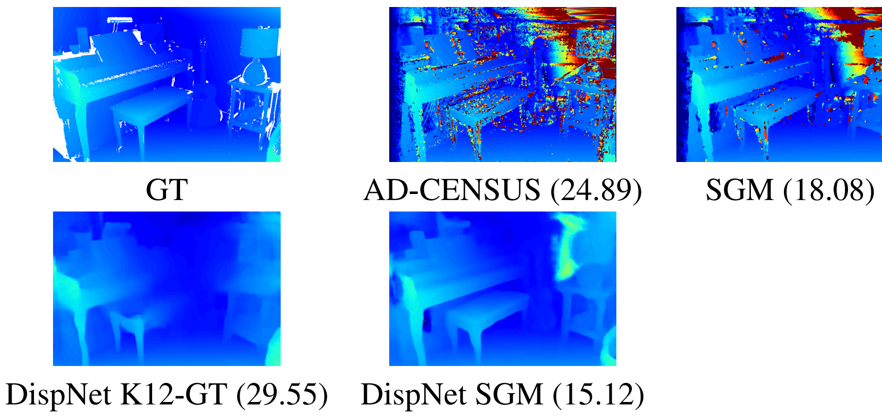


Figure 13.6: Qualitative disparity maps obtained by AD-CENSUS, SGM, Dispnet [54] and adapted Dispnet.

rather than also those labeled as ”000000_11”, so to deploy a similar number of images as DispNet fine-tuned on KITTI 2012. In these settings we observe only a modest increase of the error rate and average disparity error of about 0.09% and 0.04% respectively.

As for the evaluation on Middlebury v3, we first highlight how fine-tuning DispNet on KITTI 2012 yields a large increase of the error rate with respect to the model trained on synthetic data only and does not significantly ameliorate the average disparity error (somehow similarly to KITTI 2015). This shows that, when fine-tuned on samples depicting very different environments (such as KITTI 2012 in this case), the network can reduce the magnitude of mismatching disparities but cannot increase the overall number of correct pixels (indeed, on Middlebury such amount is vastly decreased). Conversely, adapting unsupervisedly DispNet with our technique yields a substantial reduction of both the average disparity error as well as of the error rate, in particular by more than 11% when deploying SGM as the stereo algorithm.

Overall, these results support the effectiveness of the proposed unsupervised adaptation approach even on a challenging and very varied environment such as the Middlebury dataset. In Figure 13.6 we show qualitative results on this dataset.

13.3.3 Generalization

Having assessed the superiority of unsupervised adaptation with respect to fine-tuning by ground-truth data from different datasets, we also inquire about the generalization capability of our technique

Stereo algorithm	KITTI 2012 bad 3(%)	KITTI 2012 avg	KITTI 2015 bad 3(%)	KITTI 2015 avg
DispNet	6.60	1.1399	7.46	1.27
DispNet K12-GT	**2.89**	0.93	4.58	1.15
DispNet CENSUS	4.29	**0.79**	**4.34**	**0.87**
DispNet SGM	4.12	0.80	4.35	0.88

Table 13.2: Absolute improvement of disparity accuracy yielded by adaptation on KITTI 2012 and 2015 datasets.

	AD-CENSUS		SGM	
τ	gt $\cap$ τ (%)	bad 3 (%)	gt $\cap$ τ (%)	bad 3 (%)
0.00	100.00	38.64	100.00	16.53
0.50	61.89	7.83	87.87	6.58
0.80	53.16	2.90	83.64	4.37
0.90	48.71	1.70	80.58	3.40
0.95	44.49	1.06	77.48	2.67
0.99	32.15	0.35	68.01	1.40

Table 13.3: Intersection between confident points and ground-truth data.

when dealing with the same data as deployed by traditional fine-tuning based on ground-truth. In particular, we perform both traditional fine-tuning and unsupervised adaptation on the KITTI 2012 training dataset, then we evaluate the performance of the networks also on the KITTI 2015 training dataset in order to assess generalization performance [2]. We perform unsupervised adaptation on the frames with available ground-truth only (i.e., given *000000* scene and its stereo pairs labeled as ”*_10*” and ”*_11*”, we obtain disparity and confidence only for the first pair), in order to make use of the same number of stereo pairs in the different tuning procedures for a fair comparison. Table 13.2 reports error rates (i.e., the percentage of pixels having a disparity error larger than 3) and average disparity error on both KITTI 2012 and KITTI 2015 training datasets. As we could expect, the network fine-tuned on ground-truth data (DispNet K12-GT) achieves a lower error rate with respect to the networks adapted unspervisedly. On the other hand, the unsupervised technique yields a lower average disparity error. To test the generalization property, we focus on results obtained on the KITTI 2015 dataset. Our unsupervised adaptation enables the network to outperform that fine-tuned supervisedly regarding both the error rate and the average disparity error, whatever stereo algorithm is deployed during the training phase.

These results can be explained by recalling the consideration already discussed in Section 13.3. As

[2]We follow this protocol to avoid multiple submissions to the KITTI benchmark.

shown in Figure 13.3, the pixels with a confidence higher than τ are more widely spread throughout the image than the available ground-truth pixels. Table 13.3 reports the intersection between confident (i.e., having a confidence value higher than the threshold τ) and ground-truth pixels as percentage of the total amount of available ground-truth samples; as expected, increasing τ such intersection gets smaller. In particular, with a threshold value of 0.99 and the AD-Census algorithm the subset of pixels processed during adaptation contains only 32% of the ground-truth data used by the common fine-tuning technique, while with the same threshold and the SGM algorithm this percentage rises to 68%. This means that all the remaining samples contributing to adaptation (i.e. 68 and 32% for, respectively, AD-CENSUS and SGM) encode patterns unseen using a traditional fine-tuning procedure. Thus, the network can learn from more varied and *generic* samples with respect to ground-truth which is, among other things, all contained in the lower part of the images. Moreover, the Table also reports the average error rate (bad 3) on the intersection, about 1% for both algorithms, stressing how the disparities computed on this subset of pixel are almost equivalent to ground-truth data. Assuming this property to be true for the rest of the pixels having confidence higher than τ, the unsupervised adaptation can learn many behaviors not encoded by the pixels providing the ground-truth, which is conducive to better generalization.

13.3.4 Qualitative Results on Challenging Sequences

To further test the effectiveness of the proposed approach, we adapt unsupervisedly DispNet on a set of challenging stereo sequences acquired in bad weather conditions [55]. Peculiar to these sequences is the unavailability of ground-truth data, making them a well-fitting case study for our proposal. Figure 13.7 reports some notable examples, on which the adaptation technique prove to solve most of the issues related to illumination and weather conditions.

13.4 Conclusion and Future Work

We have demonstrated that it is possible to adapt a deep learning stereo network to a brand new environment without using ground-truth disparity labels. The experimental evaluation proved that our

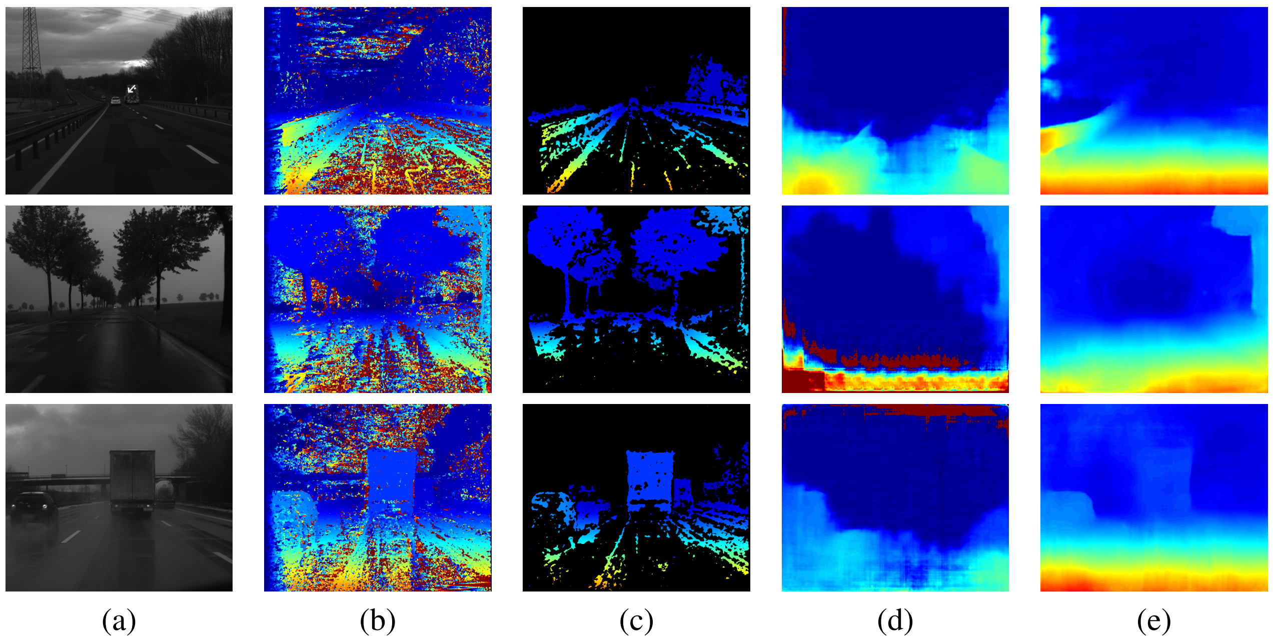


Figure 13.7: Qualitative results obtained with and without adaptation. (a) Reference image, (b) noisy disparity map from AD-CENSUS, (c) reliable points, (d) results from DispNet without adaptation, (e) results after adaptation.

proposal can better generalize when moving to similar contexts with respect to fine-tuning techniques based on sparse ground-truth data. Based on these findings, we plan to investigate on whether and how our approach may be deployed to train from scratch in a completely unsupervised manner a deep stereo network. Purposely, we may leverage jointly on different and somehow complementary stereo algorithms [81, 64] as raw target disparities to be validated by the confidence estimator. Another line of further research concerns the development of a real-time self-adaptive stereo system, which would be able to adapt autonomously and on-line to an unseen environment.

Chapter 14

Conclusion

14.1 Summary of Thesis Achievements

In this thesis, several techniques leveraging on machine learning have been proposed to improve several aspects strictly related to the stereo matching problem. In particular, two novel confidence measures have been proposed, showing that inferring such measures from the disparity domain is possible, making the cost volume no longer required to predict such cues. The proposed measures ranks among the top-performing ones according to the standard evaluation protocols, with CCNN representing the current state-of-the art following the evaluation in [70]. Deep learning techniques have been succesfully deployed to improve confidence estimation, in particular when replacing random forest classifiers [69] or to generate more accurate prediction by exploiting local patterns on confidence maps. Several techniques leveraging on confidence measures have been proposed in order to succesfully increase the accuracy of standard stereo algorithms [65], as well as for self-supervised training of both confidence and disparity estimators.

14.2 Future Work

Future research directions will include the study of other applications leveraging on confidence measures. In particular, after proving their capability to improve stereo itself, they could be deployed for other problems involving disparity computation, possibly joint with other tasks (e.g. 3D reconstruction, semantic segmantation and others). Moreover, joint disparity and confidence estimation could give benefit to both. In particular, Generative Adversarial Networks (GAN) recently proved great potential in computer vision, thus we could deploy a generator inferring dense disparity map and a discriminator predicting the correctness of disparity assignments. Also, other low level vision tasks such as optical flow or full scene flow (consisting into processing two consecutive stereo pairs and inferring both disparity maps together with optical flow between the two reference images) could benefit from this work.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2282, nov 2012.

[2] D. Bailey. Space efficient division on fpgas. In *Electronics New Zealand Conference (ENZ-Con06)*, pages 206–211, 2012.

[3] 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.

[4] C. Banz, P. Pirsch, and H. Blume. Evaluation of Penalty Functions for Semi-Global Matching Cost Aggregation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 1–6, jul 2012.

[5] T. Barron and B. Poole. The fast bilateral solver. In *Proceedings of the 14th European Conference on Computer Vision*, ECCV, 2016.

[6] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang. A deep visual correspondence embedding model for stereo matching costs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 972–980, 2015.

[7] D. Cline, K. White, and P. Egbert. Fast 8-bit median filtering based on separability. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V – 281–V – 284, Sept 2007.

[8] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[9] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza. Linear stereo matching. In *13th International Conference on Computer Vision (ICCV2011)*, November 6-13 2011.

[10] L. Di Stefano, M. Marchionni, and S. Mattoccia. A fast area-based stereo matching algorithm. *Image and vision computing*, 22(12):983–1005, 2004.

[11] G. Egnal, M. Mintz, and R. P. Wildes. A stereo confidence metric using single view imagery. In *PROC. VISION INTERFACE*, pages 162–170, 2002.

[12] G. Egnal and R. P. Wildes. Detecting binocular half-occlusions: Empirical comparisons of five approaches. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 24(8):1127–1133, 2002.

[13] 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.

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Rob. Res.*, 32(11):1231–1237, sep 2013.

[15] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] A. Geiger, M. Roser, and R. Urtasun. Efficient large-scale stereo matching. In *Asian Conference on Computer Vision (ACCV)*, 2010.

[17] M. Gerrits and P. Bekaert. Local stereo matching with segmentation-based outlier rejection. In *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, 2006.

[18] S. Ghosh, E. Laksana, S. Scherer, and L.-P. Morency. A Multi-label Convolutional Neural Network Approach to Cross-Domain Action Unit Detection. In *Proceedings of ACII 2015*, Xi'an, China, sep 2015. IEEE.

[19] 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)*, July 2017.

[20] 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.

[21] R. Haeusler, R. Nair, and D. Kondermann. Ensemble learning for confidence measures in stereo vision. In *CVPR. Proceedings*, pages 305–312, 2013.

[22] K. He, J. Sun, and X. Tang. Guided image filtering. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 1–14, Berlin, Heidelberg, 2010. Springer-Verlag.

[23] H. Hirschmller. Evaluation of cost functions for stereo matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[24] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):328–341, feb 2008.

[25] 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.

[26] H. Hirschmuller, P. R. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 2002.

[27] 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.

[28] 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 *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 4930–4935, 2014.

[29] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4930–4935, Sept 2014.

[30] A. Hosni, M. Bleyer, and M. Gelautz. Secrets of adaptive support weight techniques for local stereo matching. *Computer Vision and Image Understanding*, 117(6):620–632, jun 2013.

[31] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(2):504 – 511, 2013.

[32] 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.

[33] Intel. Realsense camera.

[34] M. Kass and J. Solomon. Smoothed local histogram filters. *ACM Trans. Graph.*, 29(4):100:1–100:10, jul 2010.

[35] 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 *International Conference on Computer Vision (ICCV 2017)*, Oct 2017.

[36] 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.

[37] 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.

[38] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, 2015.

[39] A. Klaus, M. Sormann, and K. Karner. A region based stereo matching algorithm using cooperative optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, 2008.

[40] V. Kolmogorov and Z. Ramin. Computing visual correspondence with occlusions using graph cuts. In *2001 International Conference on Computer Vision (ICCV 2001)*, 2001.

[41] N. Komodakis, G. Tziritas, and N. Paragios. Segment-based stereo matching using belief propagation and aself-adapting dissimilarity measure. In *International Conference on Pattern Recognition (ICPR 2006)*, 2006.

[42] N. Komodakis, G. Tziritas, and N. Paragios. Fast, approximately optimal solutions for single and dynamic mrfs. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*, 2007.

[43] G. Kurata, B. Xiang, and B. Zhou. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526, June 2016.

[44] S. Lefebvre, S. Ambellouis, and F. Cabestaing. A colour correlation-based stereo matching using 1D windows. In *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, SITIS'07*, pages 702–710, Shanghai, China, Dec 2007. IEEE.

[45] W. Luo, A. G. Schwing, and R. Urtasun. Efficient Deep Learning for Stereo Matching. In *Proc. CVPR*, 2016.

[46] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu. Constant time weighted median filtering for stereo matching and beyond. In *International Conference on Computer Vision*, ICCV, 2013.

[47] R. Manduchi and C. Tomasi. Distinctiveness maps for image matching. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 26–31. IEEE, 1999.

[48] G. Marin, P. Zanuttigh, and S. Mattoccia. Reliable fusion of tof and stereo depth driven by confidence measures. In *ECCV 2016*, pages 386–401, 2016.

[49] L. Matthies. Stereo vision for planetary rovers: Stochastic modeling to near real-time implementation. *Int. J. Comput. Vision*, 8(1), jul 1992.

[50] S. Mattoccia. A locally global approach to stereo correspondence. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on. IEEE*, ICCV, 2009.

[51] S. Mattoccia, S. Giardino, and A. Gambini. Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering. In *Proceedings of the 9th Asian Conference on Computer Vision - Volume Part II*, ACCV'09, 2010.

[52] 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.

[53] 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*, ICDSC '15, pages 146–151, New York, NY, USA, 2015. ACM.

[54] N. Mayer, E. Ilg, P. Häusser, 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[55] S. Meister, B. Jähne, and D. Kondermann. Outdoor stereo camera system for the generation of real-world benchmark data sets. *Optical Engineering*, 51(02):021107, 2012.

[56] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[57] P. Merrell, A. Akbarzadeh, L. Wang, J. Frahm, and R. Y. D. Nister. Real-time visibility-based fusion of depth maps. In *The IEEE Conference on Computer Vision and Pattern Recognition (ICCV)*, 2007.

[58] D. B. Min and K. Sohn. An asymmetric post-processing for correspondence problem. *Sig. Proc.: Image Comm.*, 25(2):130–142, 2010.

[59] P. Mordohai. The self-aware matching measure for stereo. In *The International Conference on Computer Vision (ICCV)*, pages 1841–1848. IEEE, 2009.

[60] 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.

[61] A. Motten, L. Claesen, and Y. Pan. *Trinocular Stereo Vision Using a Multi Level Hierarchical Classification Structure*. 2013.

[62] 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)*, 2015.

[63] S. Perreault and P. Hbert. Median filtering in constant time. *IEEE Transactions on Image Processing*, 16(9):2389–2394, 2007.

[64] 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.

[65] 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.

[66] M. Poggi and S. Mattoccia. Learning from scratch a confidence measure. In *Proceedings of the 27th British Conference on Machine Vision, BMVC*, 2016.

[67] 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.

[68] 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.

[69] M. Poggi, F. Tosi, and S. Mattoccia. Even more confident predictions with deep machine-learning. In *12th IEEE Embedded Vision Workshop (EVW2017) held in conjunction with IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[70] M. Poggi, F. Tosi, and S. Mattoccia. Quantitative evaluation of confidence measures in a machine learning world. In *International Conference on Computer Vision (ICCV 2017)*, Oct 2017.

[71] D. Scharstein, H. Hirschmller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *Pattern Recognition: 36th German Conference, GCPR 2014*, pages 31–42.

[72] D. Scharstein and R. Szeliski. Stereo matching with non-linear diffusion. *International Journal of Computer Vision*, 28:155–174, 1998.

[73] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, apr 2002.

[74] K. Schmid and H. Hirschmuller. Stereo vision and imu based real-time ego-motion and depth image computation on a handheld device. In *ICRA*, pages 4671–4678, May 2013.

[75] A. Seki and M. Pollefeys. Patch based confidence prediction for dense disparity map. In *British Machine Vision Conference (BMVC)*, 2016.

[76] A. Seki and M. Pollefeys. Sgm-nets: Semi-global matching with neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[77] A. Shaked and L. Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, July.

[78] Y. Shan, Y. Hao, W. Wang, Y. Wang, X. Chen, H. Yang, and W. Luk. Hardware acceleration for an accurate stereo vision system using mini-census adaptive support region. *ACM Trans. Embed. Comput. Syst.*, 13(4s):132:1–132:24, apr 2014.

[79] R. Spangenberg, T. Langner, S. Adfeldt, and R. Rojas. Large scale semi-global matching on the cpu. In *IV*, 2014.

[80] 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.

[81] A. Spyropoulos and P. Mordohai. Ensemble classifier for combining stereo matching algorithms. In *Proceedings of the 2015 International Conference on 3D Vision*, 3DV '15, pages 73–81, 2015.

[82] Stereolabs. Zed camera.

[83] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(5):815–830, may 2010.

[84] F. Tombari, S. Mattoccia, and L. Di Stefano. *Segmentation-Based Adaptive Support for Accurate Stereo Correspondence*. 2007.

[85] F. Tombari, S. Mattoccia, L. D. Stefano, and E. Addimanda. Classification and evaluation of cost aggregation methods for stereo correspondence. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[86] 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.

[87] C. Ttofis and T. Theocharides. Towards accurate hardware stereo correspondence: A real-time fpga implementation of a segmentation-based adaptive support weight algorithm. In *DATE*, 2012.

[88] O. Veksler. Fast variable window for stereo correspondence using integral images. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'03, 2003.

[89] 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 EMMCVPR*, pages 14–27, August 2009.

[90] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. CNN: single-label to multi-label. *CoRR*, 2014.

[91] Q. Yang. A non-local cost aggregation method for stereo matching. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12.

[92] K.-J. Yoon and I. S. Kweon. Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), 2006.

[93] 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.

[94] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the 3rd European Conference on Computer Vision, ECCV*.

[95] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[96] 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, 2016.

[97] K. Zhang, J. Lu, and G. Lafruit. Cross-based local stereo matching using orthogonal integral images. *IEEE Trans. Cir. and Sys. for Video Technol.*, 19(7):1073–1079, jul 2009.

[98] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, 2014.