ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING
CICLO XXXVI

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

Toward Practical Depth Estimation Based on Deep Learning

Presentata da: **Youmin Zhang**

Coordinatore Dottorato
Prof. **Ilaria Bartolini**

Supervisore
Prof. **Stefano Mattoccia and Matteo Poggi**

Abstract

Toward Practical Depth Estimation Based on Deep Learning

Depth estimation from images is a classic computer vision problem that has occupied researchers for decades. Obtaining dense and accurate depth estimation is pivotal to effectively address higher-level tasks in computer vision such as autonomous driving, 3D reconstruction, and robotics. It can be carried out either employing active sensors (*e.g.*, LiDAR, Time-of-Flight, structured light) or from images acquired by standard cameras. The former sensors typically provide only sparse depth measurements. Despite the tremendous progress of recent depth completion methods, the locality of the convolutional layer or graph model makes it hard for the network to model the long-range relationship between pixels. Concerning the latter category, recent end-to-end deep-learning-based models achieve unprecedented accuracy outperforming by a large margin state-of-the-art methods based on conventional approaches. In particular, self-supervised monocular depth estimation is an attractive solution that does not require hard-to-source depth labels for training. These approaches replace supervised losses on depth labels with supervisory signals derived from image reprojection across different views, by exploiting the geometric relationship between frames, *i.e.*, the scene depth itself and camera pose. However, view reconstruction based losses suffer from occlusions, dynamic objects, and photometric changes, which severely limit the performance of the network. Furthermore, the precision of relative pose across frames plays a crucial role in achieving accurate depth estimation. Unlike monocular depth estimation, stereo matching, facilitated by the known baseline distance between stereo cameras, eliminates the need for separate pose estimation and additionally furnishes metric depth/disparity predictions. Nevertheless, contemporary stereo methods typically handle each frame independently even though tens of thousands of stereo videos are publicly accessible. Exploring how to effectively incorporate relevant cues from previous matching contexts to enhance both estimates and efficiency remains a current and active research focus. Moreover, considering the limitations of physical fabrication, power consumption, and costs, the resolution of depth maps (*e.g.*, the depth from various sources such as monocular depth estimation, stereo matching, or depth sensors) usually is often insufficient to fulfill the demand of the most practical applications. Finally, globally consistent 3D reconstruction is another crucial aspect to be considered when targeting downstream applications such as augmented/virtual reality, and hence is desirable not only to infer reliable depth data but obtain high-fidelity 3D reconstruction without compromising accuracy and real-time performance.

Therefore, with a focus on diverse depth estimation setups, this thesis addresses significant challenges that impact methodologies for inferring depth from images or obtaining completed depth from hardware sensors. The goal is to develop precise and efficient frameworks tailored for accurate Simultaneous Localization And Mapping (SLAM) in challenging environments.

Acknowledgments

It's a great honor to take a Ph.D. position in Bologna, Italy. During these three years, I learned a lot and also enjoyed it very much. First of all, I would like to acknowledge my supervisors Prof. Stefano Mattoccia and Prof. Matteo Poggi for their invaluable supervision, support, and patience during the course of my PhD degree. Many thanks to Fabio Tosi, Filippo Aleotti, Chaoqiang Zhao, Xin Qiao, Huan Li, Kun Zhang, and all members of the CVLAB. It's a great time to spend with all of you together. I extend my heartfelt gratitude to Prof. Marc Pollefeys and Dr. Songyou Peng for providing me with the valuable opportunity to join your team. The internship experience was truly enriching, and I appreciate the chance to collaborate with you. Additionally, I want to express my sincere thanks to my family. Your unwavering support and selfless love have been indispensable, and without it, navigating through the Ph.D. period abroad alone would not have been possible.

Contents

Introduction

Scene depth estimation is a crucial component in computer vision, significantly enhancing the perception and comprehension of real three-dimensional environments. This advancement finds applications in various fields like robotic [182], virtual/augmented reality [268], and autonomous driving [189]. Active depth estimation methods commonly leverage Lasers, structured light, and reflections on object surfaces to generate depth point clouds, intricate surface models, and scene depth maps. While these approaches yield exceptional accuracy and precision, they are hindered by notable limitations that impede their practical deployment in real-world applications. For instance, LiDAR sensors, employing one or more laser emitters scanning the environment through mechanical rotation, may encounter challenges like misalignment, missing laser returns due to absorbing or reflective surfaces, and issues related to multi-pathing. Furthermore, the depth maps captured by other prevalent commercial depth sensors (e.g., Microsoft Kinect [138], Intel RealSense [90]) face similar issues. In challenging conditions such as transparent, shining, and dark surfaces, these sensors may exhibit highly sparse depth maps, and even fail to provide measurements, exacerbating the limitations of active depth estimation methods. To address these issues, depth completion, which targets completing and reconstructing the whole depth map from sparse depth measurements and a corresponding RGB image (*i.e.*, RGBD), has gained much attention in the latest years. Since the corresponding RGB images contain rich semantic cues that are critical for filling unknown depths, some works [27, 28, 117, 152] utilize the RGB information to guide depth completion, which directly maps sparse depth maps to dense depth maps through massive stacked convolutional layers, graph models, or pure Vision Transformer layers. In Chapter 2, we propose CompletionFormer, a pyramidal architecture coupling CNN-based local features with Transformer-based global representations for enhanced depth completion. It yields substantial improvements to depth completion compared to state-of-the-art methods, especially when the provided depth is *very* sparse, as often occurs in practical applications.

Another line of depth perception is inferring depth from single or multiple images directly. Without relying on active sensors, image-based depth estimation has become the predominant focus of research, providing a more practical and versatile solution applicable across a wide range of applications. In particular, self-supervised/unsupervised monocular depth estimation, which eliminates the need for ground truth during training, has attracted attention in recent years. This approach replaces supervised losses on depth labels with supervisory signals derived from image projection across different views, addressing a novel view synthesis problem. However, it introduces its own set of challenges. In addition to estimating depth, the model must also gauge egomotion between temporal image pairs during training. This typically involves training a pose estimation network that takes a finite sequence of frames as input and outputs the corresponding camera transformations. Furthermore, losses based on view reconstruction suf-

fer from occlusions, dynamic objects, and photometric changes, which significantly constrain the network's performance. In light of the recent successes achieved by Vision Transformer, we propose MonoViT in Chapter 3, a brand-new framework combining the global reasoning enabled by ViT models with local fine-grained details embedded by plain convolutions. As a result, our model can reason locally and globally, yielding depth prediction at a higher level of detail and accuracy, allowing MonoViT to achieve state-of-the-art performance on the several public datasets [62, 177, 255].

Conversely, using stereo data for training makes the camera-pose estimation a one-time offline calibration. Leveraging the known baseline distance between stereo cameras, stereo matching processes two rectified images as input. It computes the disparity of nearly every pixel by matching corresponding pixels along conjugate epipolar lines, facilitating metric depth estimation through triangulation. This represents a significant advantage over unsupervised monocular depth estimation approaches, which rely on multiple-frame visual odometry and encounter the challenge of scale ambiguity. However, existing methods for depth from stereo treat different stereo frames independently, leading to temporally inconsistent depth predictions. Temporal consistency is especially important for immersive AR or VR scenarios, where flickering greatly diminishes the user experience. We present TemporalStereo in Chapter 4, a coarse-to-fine stereo matching network that is highly efficient, and able to effectively exploit the past geometry and context information to boost matching accuracy. Our network leverages sparse cost volume and proves to be effective when a single stereo pair is given. However, its peculiar ability to use spatio-temporal information across stereo sequences allows TemporalStereo to alleviate problems such as occlusions and reflective regions while enjoying high efficiency also in this latter case. Notably, our model – trained once with stereo videos – can run in both single-pair and temporal modes seamlessly, allowing the deployment of the same model in both settings for downstream applications.

However, due to the limitations of physical fabrication, power consumption and costs [7], the resolution of depth maps obtained by above methods usually is often insufficient to fulfill the demand of the downstream applications [26, 60]. In contrast, collecting RGB images at much higher resolution is cheaper. As a result, the guided depth super-resolution (GDSR) task has emerged as a crucial solution to this technological limitation, allowing to obtain an accurate high-resolution (HR) depth map from a low-resolution (LR) one, guided by an HR image. Most existing GDSR algorithms are elaborately designed to exploit this piecewise smoothness contained in the low resolution depth map and take full advantage of the guidance image. The main challenge for state-of-the-art methods is still restoring precise and sharp edges near depth discontinuities and fine structures. To alleviate this issue, we propose a novel multi-stage depth super-resolution network in Chapter 5, which progressively reconstructs HR depth maps from explicit and implicit high-frequency information. The shape bias and global context of the transformer allow our model to focus on high-frequency details between objects, i.e., depth discontinuities, rather than texture within objects. To incorporate the structural details, we develop a fusion strategy that combines depth features and high-frequency information in the multi-stage-scale framework. Exhaustive experiments on the main benchmarks show that our approach establishes a new state-of-the-art.

Finally, moving forward, the last chapter (Chapter 6) will develop a dense Simultaneous Localization And Mapping (SLAM), which showcases versatility by accommodating input depth from various sources such as monocular depth estimation, stereo matching, depth sensors, or even operating without explicit depth information. Notably, the proposed SLAM system based

on Neural Radiance Fields (NeRF) capable of predicting pose and constructing a waterproofed 3D mesh. This comprehensive approach extends the applicability of our research to a broader range of scenarios and demonstrates promising outcomes in pushing the boundaries of depth estimation for practical use.

Chapter 1

Related Work

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

1.1 Depth Completion.

Scene depth completion has become a fundamental task in computer vision with the emergence of active depth sensors. Recently, following the advance of deep learning, fully-convolutional network has been the prototype architecture for current state-of-the-art on depth completion. Ma *et al.* [130, 131] utilize a ResNet [75] based encoder-decoder architecture, *i.e.*, U-Net, within either a supervised or self-supervised framework to predict the dense output. To preserve the accurate measurements in the given sparse depth and also perform refinement over the final depth map, CSPN [27] appends a convolutional spatial propagation network (SPN [122]) at the end of U-Net to refine its coarse prediction. Based on CSPN, learnable convolutional kernel sizes and a number of iterations are proposed to improve the efficiency [28], and the performance could be further improved by using unfixed local neighbors [152, 250] and independent affinity matrix for each iteration [117]. For all these SPN-based methods, while a larger context is observed within recurrent processing, the performance is limited by the capacity of the convolutional U-Net backbone. Accordingly, we strengthen the expressivity of the U-Net backbone with local and global coherent context information, proving effective in improving performance.

Rather than depending on a single branch, multi-branch networks [80, 119, 144, 165, 203, 219, 274] are also adopted to perform multi-modal fusion. The common way to fuse the multi-modal information is simple concatenation or element-wise summation operation. More sophisticated strategies like image-guided spatially-variant convolution [203, 252], channel-wise canonical correlation analysis [289], neighbour attention mechanism [277] and attention-based graph propagation [244, 286] were also proposed to enhance local information interaction and fusion. Instead of pixel-wise operation or local fusion, recently, GuideFormer [169] proposed a dual-branch fully Transformer-based network to embed the RGB and depth input separately, and an extra module is further designed to capture inter-modal dependencies. The independent design for each input source leads to huge computation costs (near 2T FLOPs with the 352×1216 input). In contrast, our CompletionFormer in one branch brings significant efficiency (559.5G FLOPs), and the included convolutional attention layer complements the disadvantage of a Transformer in local details.

1.2 Vision Transformer.

Transformers [103, 126] are first introduced in natural language processing [220], then also showing great potential in the fields of image classification [45], object detection [103, 126, 248] and semantic segmentation [242]. Tasks related to 3D vision have also benefited from the enriched modeling capability of Transformer, such as stereo matching [105, 110], supervised [111, 166] and unsupervised monocular depth estimation [284], optical flow [87, 196] and also depth completion [169]. Instead of relying on pure Vision Transformer [169], we explore the combination of Transformer and convolution into one block for depth completion. Compared to the general backbone networks (*e.g.* ResNet [75] with fully CNN-based design, Swin Transformer [126] and PVT [230] based on pure Transformer, MPViT [103] and CMT [70] using both the convolutions and Vision Transformer), our proposed joint convolutional attention and Transformer block achieves much higher efficiency and performance on public benchmarks [187, 217].

1.3 Monocular Depth Estimation.

Estimating depth from a single image is a challenging, inherently ill-posed problem. Nonetheless, the many learning-based approaches aimed at addressing it [281] enabled significant progress in the field. As fully supervised techniques [12, 55, 101, 118] for depth estimation advanced rapidly, the availability of precise depth labels in the real world became a major issue. Hence, more recent self-supervised works provided alternatives to remove it, avoiding the need for hard-to-source ground-truth depth annotations. This goal is feasible by casting the depth estimation task as a view-synthesis problem between adjacent views, in space or time, of the same observed scene. Precisely, training single-view depth estimation network with stereo images [59, 63], monocular videos [293], or a combination of both [64, 269]. The supervisory signal based on the photometric difference between real and synthesized images enables training in a self-supervised manner.

Although stereo pairs enable scale recovery, with further improvements achievable by leveraging noisy proxy labels [31, 214, 238], guidance from visual odometry [2] or trinocular assumptions [159], unlabeled video sequences represent a more flexible alternative at the expense of learning camera poses alongside depth. Several frameworks have advanced this line of research by incorporating additional losses and constraints such as those based on direct visual odometry [222], adversarial learning[282], ICP [132], normal consistency [259, 261], semantic segmentation [68, 99] and uncertainty [160, 256]. Another notable example is given in [64] where the authors introduced a minimum reprojection loss between frames and an auto-masking strategy to handle both occluded regions and static camera situations that violate the main constraints of the view-synthesis formulation and, as a consequence, cause poor network convergence. Other works, instead, directly tackle highly complex scenarios [283] and model rigid and non-rigid components present in the scene using the estimated depth, relative camera poses, and optical flow in order to handle independent motions [24, 168, 215, 260, 264, 298] or by means of scene decomposition [174].

1.4 Architecture of Monocular Depth Networks.

Playing with different architectures used as backbone showed a significant impact on the performance of monocular depth estimation itself. Yin *et al.* [264] replaced the VGG encoder used by [293] with a ResNet. Guizilini *et al.* [67] designed a novel model, PackNet, to learn detail-preserving compression and decompression of features by using 3D convolutions. Lyu *et al.* [224] worked on the features decoding, implementing an attention module for multi-scale feature fusion. Because of the limited receptive field of CNNs, the network performance still has room for further improvement. To extract the long-range relationships between features, Yan *et al.* [251] propose a channel-wise attention-based network to aggregate discriminated features in channel dimensions. Considering the ability of HRNet [224] at modeling multi-scale features, Zhou *et al.* [292] introduced HRNet for self-supervised monocular depth estimation. Other works, instead, focused on the design of efficient and fast networks suitable for low-powered embedded devices [156, 158, 240]. Despite the increased accuracy achieved by the above networks, the issue concerning long-range relationships persists [112].

1.5 Single Pair Stereo Matching.

Traditional methods [79] employ handcrafted schemes to find local correspondences [179, 257] and approximate global optimization by exploiting spatial context [77]. Recent deep learning strategies [161] adopt CNNs and can be broadly divided into two categories according to how they build the cost volume. On the one hand, 2D networks follow DispNetC [135] employing a correlation-based cost volume and applying 2D convolutions to regress a disparity map. Stacked refinement sub-networks [113] and multi-task learning [190, 253] have been proposed to improve the accuracy. GCNet [89] represents a milestone for 3D networks. Following works improve its results thanks to spatial pyramid pooling [19], group-wise correlations [71], forcing unimodal [58, 276] or bimodal [216] cost distributions. 3D methods usually outperform 2D architecture by a large margin on popular benchmarks [61, 135, 136], paying more in terms of computational requirement. In our TemporalStereo, we leverage 3D sparse cost volumes, and the peculiar proposed architecture proves to be fast and accurate at disparity estimation.

1.6 Efficient Stereo Matching with Deep-Learning.

A popular strategy to decrease the runtime consists in performing computation at a single yet small resolution (e.g., 1/4) and obtaining the final disparity through upsampling. CoEx [8] adopts this strategy. Coarse-to-fine [5, 254] design further improves efficiency, since the overall computation is split into many stages. To further reduce the disparity search range for each pixel, StereoNet [91] and AnyNet [233] add a constant offset to disparity maps produced in the previous stage to avoid checking all the disparity candidates. However, as reported in [133], the constant offset strategy is not robust against large errors in initial disparity maps. DeepPruner [46] prunes the search space using a differentiable variant of PatchMatch [9], obtaining sparse cost volumes. In contrast, methods such as [133, 185] employ uncertainty to evaluate the outcome of previous stages and then build the current cost volume accordingly. However, since the cost volume itself expresses the goodness of candidates (the better the candidates, the

more representative the cost volume), it could be used to *check* and eventually *correct* the candidates themselves. Nonetheless, previous methods in literature cannot exploit this cue since their candidates come from the earlier level and are constant through the current stage. Conversely, in TemporalStereo, we propose inferring an offset for each disparity candidate from the current aggregated cost volume, which helps to ameliorate the disparity. Moreover, to improve network efficiency on high-resolution images, we perform the coarse-to-fine predictions only at downsampled feature maps rather than at full image resolution like HITNet [207].

1.7 Multiple Pairs Stereo Matching.

When two temporally adjacent stereo pairs are available, optical flow [135, 210] or 3D scene flow [213] is often taken into account to link the images with the reconstructed 2D/3D motion field. Then, the stereo problem is tackled by leveraging a multi-task model [1, 100, 234] or by casting it as a special case of optical flow [120]. However, these methods are not able to capitalize longer stereo sequences. Conversely, OpenStereoNet [288] adopts recurrent units to capture temporal dynamics and correlations available in videos for unsupervised disparity estimation. We argue that past information could be beneficial to understand the current scene – especially in difficult areas – for *free* since past context and predictions can be cached easily, although none of the above methods make use of them explicitly in videos by taking into account geometry. On the contrary, TemporalStereo fully exploits this potential.

1.8 Depth Super-Resolution.

Initially, hand-craft models were developed for GDSR, using the edge co-occurrence between the LR depth map and its HR color counterpart as prior. [98] first utilizes a joint bilateral filter, taking guidance cues from color images. The so-called *local* methods followed this pivotal work: [258] enhances the LR depth maps by exploiting registered HR color images, [170] uses anti-alias image prefiltering built on the multi-stage joint bilateral filter, while graph-based joint bilateral upsampling [232] casts GDSR as a regularization problem.

More accurate solutions, although slower, are represented by *global* methods. The first work in this direction is [43], which employs Markov random fields (MRF) to integrate multi-modal data for LR depth map upsampling. Using the non-local mean filtering method, [151] recovers noisy LR images from a ToF camera to a high-quality image. To be more efficient, [52] exploits Total Generalized Variation (TGV) regularization for GDSR, enabling a high frame rate. [108] uses fast global smoothing (FGS) to make guided depth interpolation more robust.

More recently, deep learning-based approaches [191, 197, 300] achieved remarkable results and became the preferred choice for GDSR. [84] designs a multi-scale guided CNN using hierarchical feature extraction to gradually restores blurred edges. To reconstruct sharp edges, the works by [107, 109] learn salient features from color images using an encoder-decoder structure. In contrast, [128] casts GDSR as a pixel-to-pixel mapping from the HR RGB image to the domain of the LR source image, learned by a multi-layer perceptron. In [262], a multi-branch network with progressive refinement performs adaptive information fusion to restore depth details. [237] can quickly upsample depth maps by learning Canny edges, while [301] proposes a depth-guided affine transformation where the feature refinement is carried out iteratively. [204]

makes use of implicit neural interpolation, [92] develops a deformable kernel network whose outputs are per-pixel kernels, and [287] proposes a Discrete Cosine Transform Network (DCT-Net) to extract multi-modal features effectively. Through graph optimization, [40] combines the advantages of model-driven and deep learning-based methods. Concurrent works exploit recurrent structure attention [266] or combine deep learning with anisotropic diffusion [137].

Despite substantial advancements, these networks are not effective enough at extracting HF guidance from RGB images. Inspired by [124], we tackles GDSR leveraging both explicit and implicit HF features guidance.

1.9 Online 3D Reconstruction and SLAM.

Real-time, dense, and globally consistent 3D reconstruction is crucial in the SLAM literature. As a core element for high-quality reconstruction, the underlying representation can be approximately categorized as depth point [13, 16, 36, 53, 142, 143, 208, 211, 290], height map [56], surfel [183, 239] and volumetric representation [38, 145, 146]. Especially some of them [16, 36, 38, 142, 183], make an effort for globally consistent reconstruction by implementing global BA and LC systems. However, due to discrete and limited surface representations (*e.g.*, point-, surfel- or voxel-based), these methods suffer from the accumulation of errors in camera tracking and distortion in the reconstruction. DROID-SLAM [211] achieves impressive trajectory estimations by using neural networks to leverage richer context from images, yet it performs global bundle adjustment only offline, at the end of camera tracking. However, for some challenging cases, it gets hard to eliminate the drift error with offline refinement solely, as shown in Fig. 6.1, pointing out the importance of online BA for on-the-fly drift correction.

1.10 NeRF-based Visual SLAM.

Neural implicit fields (NeRF) have recently emerged as one of the promising and widely applicable methods for 3D representation, opening up many new research opportunities including novel view synthesis [10, 11, 54, 139, 140, 198, 227, 228, 272], multi-view 3D reconstruction [18, 20, 200], and large-scale scene reconstruction [3, 106, 148, 225, 265, 273, 299]. A common requirement of these methods is the posed images. [114, 263] tries to relax this constraint starting from imperfect camera poses on small objects. More recently, using neural implicit representation in visual SLAM [33, 104, 173, 195, 295, 296] has achieved better scene completeness, especially for unobserved regions, and it has also allowed for continuous 3D modeling at arbitrary resolution. Two pioneer works, iMAP [195] and NICE-SLAM [295], extend the neural implicit representation to RGB-D SLAM system, which learns camera pose tracking and room-scale mapping from scratch. Concurrent works [104, 296], by removing the reliance on depth sensor input, achieve visual SLAM when only RGB sequences are available. Instead of naïve pose optimization with NeRF, Orbeez-SLAM [33] resorts to visual odometry from ORB-SLAM2 [142] for accurate pose estimation. However, the lack of many of the core capabilities of modern SLAM systems, such as LC and global BA, inhibits the ability of these methods to perform large-scale reconstructions. Concurrent to our work, NeRF-SLAM [173] integrates DROID-SLAM [211] for camera tracking. However, it also inherits its limitations, *i.e.*, the absence of online loop closing and full BA, which restricts its ability to perform globally

consistent 3D reconstruction.

Chapter 2

Depth Completion with Convolutions and Vision Transformers

The content of this chapter has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2023) - "CompletionFormer: Depth Completion with Convolutions and Vision Transformers" [279].

2.1 Introduction

Active depth sensing has achieved significant gains in performance and demonstrated its utility in numerous applications, such as autonomous driving and augmented reality. Although depth maps captured by existing commercial depth sensors (*e.g.*, Microsoft Kinect [138], Intel RealSense [90]) or depths points within the same scanning line of LiDAR sensors are dense, the distance between valid/correct depth points could still be far owing to the sensor noise, challenging conditions such as transparent, shining, and dark surfaces, or the limited number of scanning lines of LiDAR sensors. To address these issues, depth completion [28, 117, 152, 169], which targets at completing and reconstructing the whole depth map from sparse depth measurements and a corresponding RGB image (*i.e.*, RGBD), has gained much attention in the latest years.

For depth completion, one key point is to get the depth affinity among neighboring pixels so that reliable depth labels can be propagated to the surroundings [27, 28, 80, 117, 152]. Based on the fact that the given sparse depth could be highly sparse due to noise or even no measurement being returned from the depth sensor, it requires depth completion methods to be capable of 1) detecting depth outliers by measuring the spatial relationship between pixels in both local and global perspectives; 2) fusing valid depth values from close or even extremely far distance points. All these properties ask the network for the potential to capture both local and global correlations between pixels. Current depth completion networks collect context information with the widely used convolution neural networks (CNNs) [27, 28, 80, 117, 152, 165, 219, 294] or graph neural network [244, 286]. However, both the convolutional layer and graph models can only aggregate within a local region, *e.g.*, square kernel in 3×3 for convolution and kNN-based neighborhood for graph models [244, 286], making it still tough to model global long-range relationship, in particular within the shallowest layers of the architecture. Recently, GuideFormer [169] resorts fully Transformer-based architecture to enable global reasoning. Unfortunately, since Vision Transformers project image patches into vectors through a single step, this

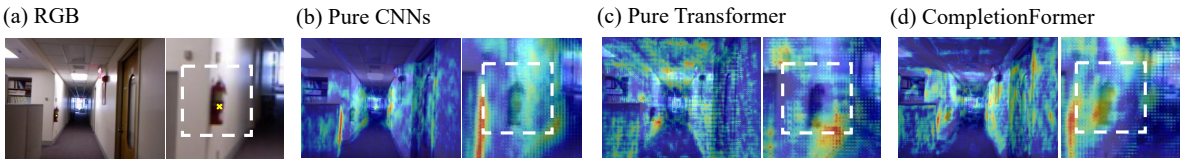


Figure 2.1: Comparison of attention maps of pure CNNs, Vision Transformer, and the proposed CompletionFormer with joint CNNs and Transformer structure. The pixel highlighted with a yellow cross in RGB image (a) is the one we want to observe how the network predicts it. Pure CNNs architecture (b) activates discriminative local regions (*i.e.*, the region on the fire extinguisher), whereas pure Transformer based models (c) activate globally yet fail on local details. In contrast, our full CompletionFormer (d) can retain both the local details and global context.

causes the loss of local details, resulting in ignoring local feature details in dense prediction tasks [157, 248]. For depth completion, the limitations affecting pure CNNs or Transformer based networks also manifest, as shown in Fig. 2.1. Despite *any* distance the reliable depth points could be distributed at, exploring an elegant integration of these two distinct paradigms, *i.e.*, CNNs and Transformer, has not been studied for depth completion yet.

In this chapter, we propose CompletionFormer, a pyramidal architecture coupling CNN-based local features with Transformer-based global representations for enhanced depth completion. Generally, there are two gaps we are facing: 1) the content gap between RGB and depth input; 2) the semantic gap between convolution and Transformer. As for the multimodal input, we propose embedding the RGB and depth information at the early network stage. Thus our CompletionFormer can be implemented in an efficient single-branch architecture as shown in Fig. 2.2 and multimodal information can be aggregated throughout the whole network. Considering the integration of convolution and Transformer, previous work has explored from several different perspectives [70, 103, 150, 157, 248] on image classification and object detection. Although state-of-the-art performance has been achieved on those tasks, high computation cost [103] or inferior performance [70, 103] are observed when these networks are directly adapted to depth completion task. To promise the combination of self-attention and convolution still being efficient, and also effective, we embrace convolutional attention and Transformer into one block and use it as the basic unit to construct our network in a multi-scale style. Specifically, the Transformer layer is inspired by Pyramid Vision Transformer [230], which adopts spatial-reduction attention to make the Transformer layer much more lightweight. As for the convolution-related part, the common option is to use plain convolutions such as Inverted Residual Block [175]. However, the huge semantic gap between convolution and the Transformer and the lost local details by Transformer require the convolutional layers to increase its own capacity to compensate for it. Following this rationale, we further introduce spatial and channel attention [241] to enhance convolutions. As a result, without any extra module to bridge the content and semantic gaps [103, 157, 169], every convolution and Transformer layer in the proposed block can access the local and global features. Hence, information exchange and fusion happen effectively at every block of our network.

To summarize, our main contributions are as follows:

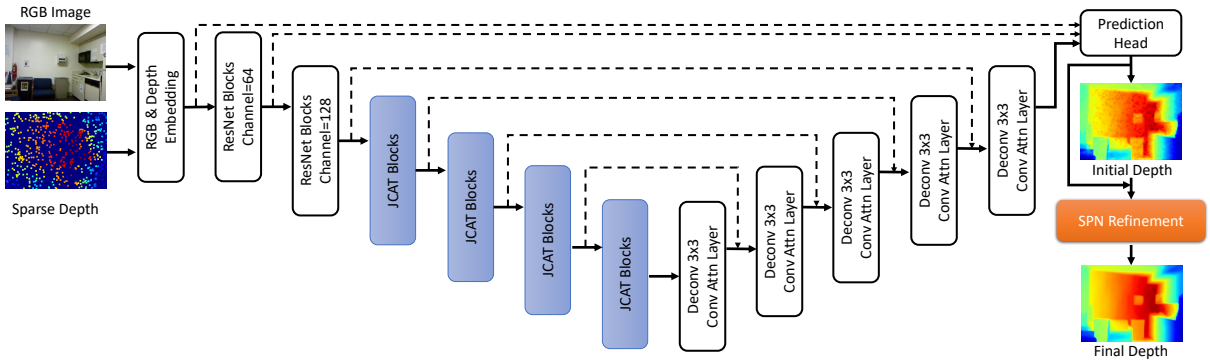
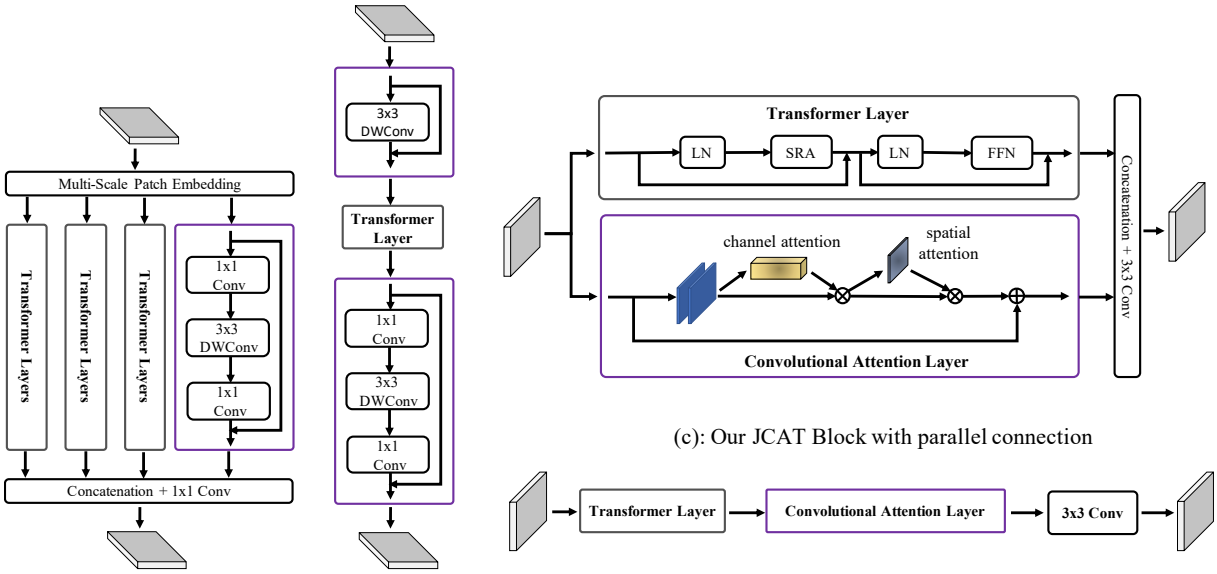


Figure 2.2: CompletionFormer Architecture. Given the sparse depth and corresponding RGB image, a U-Net backbone strengthened with JCAT block is used to perform the depth and image information interaction at multiple scales. Features from different stages are fused at full resolution and fed for initial prediction. Finally, a spatial propagation network (SPN) is exploited for final refinement.

- We propose integrating Vision Transformer with convolutional attention layers into one block for depth completion, enabling the network to possess both local and global receptive fields for multi-modal information interaction and fusion. In particular, spatial and channel attention are introduced to increase the capacity of convolutional layers.
- Taking the proposed Joint Convolutional Attention and Transformer (JCAT) block as the basic unit, we introduce a single-branch network structure, *i.e.*, CompletionFormer. This elegant design leads to a comparable computation cost to current CNN-based methods while presenting significantly higher efficiency when compared with pure Transformer based methods.
- Our CompletionFormer yields substantial improvements to depth completion compared to state-of-the-art methods, especially when the provided depth is *very* sparse, as often occurs in practical applications.

2.2 Method

In practical applications, depth maps captured by sensors present various levels of sparsity and noise. Our goal is to introduce both the local features and global context information into the depth completion task so that it can gather reliable depth hints from any distance. The overall diagram of our CompletionFormer is shown in Fig. 2.2. After obtaining depth and RGB image embedding, a backbone constructed by our JCAT block is used for feature extraction at multiple scales and the decoder provides full-resolution features for initial depth prediction. Finally, for the purpose of preserving accurate depth from the sparse input, we refine the initial estimation with a spatial propagation network.



(a): MPViT (b): CMT (c): Our JCAT Block with parallel connection (d): Our JCAT Block with cascaded connection

Figure 2.3: Example of architecture with convolutions and Vision Transformer. (a) Multi-Path Transformer Block of MPViT [103]. (b) CMT Block of CMT-S [70]. (c) Our proposed JCAT block which contains two parallel streams, *i.e.*, convolutional attention and Transformer layer respectively. (d) The variant of our proposed block with cascaded connection.

2.2.1 RGB and Depth Embedding

For depth completion, multimodal information fusion at an early stage has several advantages, 1) it makes the feature vector of each pixel possess both the RGB and depth information so that pixels with invalid depth still have a chance to be corrected by reliable depth measurements according to appearance similarity; 2) only one branch is required for the following network, which enables much efficient implementation. Therefore, we firstly use two separate convolutions to encode the input sparse depth map S and RGB image I. The outputs are concatenated and further processed by another convolution layer to get the raw feature containing contents from both sources.

2.2.2 Joint Convolutional Attention and Transformer Encoder

It has been extensively studied how to build connections between pixels to implement depth propagation from reliable pixels while avoiding incorrect ones. Recently, convolution layer [27, 28, 80, 117, 152, 165, 219, 294] or attention based graph propagation [244, 286] has been the dominant operation for this purpose. Although a fully Transformer-based network [169] has also been adopted for this purpose, it shows worse results and much higher computational cost compared to pure CNNs-based methods. Considering the complementary properties of these two-style operations, an elegant integration of these two paradigms is highly demanded for depth completion task. On the other hand, for classification and object detection tasks, MPViT [103] and CMT [70] are two representative state-of-the-art networks on the combination of self-attention and convolution as shown in Fig. 2.3 (a) and (b) respectively. Generally, the

CompletionFormer	#Layers	Params (M)	FLOPs (G)
Tiny	[2, 2, 2, 2]	41.5	191.7
Small	[3, 3, 6, 3]	78.3	231.8
Base	[3, 3, 18, 3]	142.4	301.9

Table 2.1: CompletionFormer Configurations. #Layers denotes the number of our JCAT blocks in each stage. For all model variants, the channels of 4 stages are 64, 128, 320, 512, respectively. FLOPs are measured using 480×640 input image.

integration can be implemented in a parallel or cascaded manner. Accordingly, inspired by their design, within CompletionFormer, we propose a joint design as shown in Fig. 2.3 (c) and (d). To decrease computation overhead but also get highly accurate depth completion result, our CompletionFormer only contains single rather than multiple time-consuming Transformer-based paths as in MPViT [103]. Furthermore, the representation power of convolution-based path is enhanced with spatial and channel attention.

Specifically, our encoder has five stages, allowing feature representation at different scales to communicate with each other effectively. In the first stage, to decrease the computation cost and memory overhead introduced by the Transformer layer, we use a series of BasicBlocks from ResNet34 [75] to process and finally get downsampled feature map F_1 at half resolution. For the next four stages, we introduce our proposed JCAT block as the basic unit for framework design.

Basically, for each stage $i \in \{2, 3, 4, 5\}$, it consists of a patch embedding module and L_i repeated JCAT blocks. The patch embedding module firstly divides the feature map F_{i-1} from previous stage $i-1$ into patches with size 2×2. We implement it with a 3×3 convolution layer and stride set to 2 as [230], so it actually halves resolution for features F_{i-1} and thus allows for obtaining a features pyramid $\{F_2, F_3, F_4, F_5\}$, whose resolutions are $\{1/4, 1/8, 1/16, 1/32\}$ with respect to the input image. Furthermore, position embedding is also included in the embedded patches and passed through the JCAT blocks.

Joint Convolutional Attention and Transformer Block. In overview, our JCAT block can be organized in a parallel or cascaded manner as shown in Fig. 2.3 (c) and (d) respectively. The Transformer layer is implemented in an efficient way as in Pyramid Vision Transformer [230], which contains a spatial-reduction attention (SRA) layer with multi-head mechanism and a feed-forward layer (FNN). Given input features $F \in \mathbb{R}^{H_i \times W_i \times C}$ from the patch embedding module or last joint block (with H_i and W_i height and width of features at stage i, and C number of channel), we firstly normalize it with layer normalization [4] (LN) and then flatten it into vector tokens $X \in \mathbb{R}^{N \times C}$, where N is the number of tokens and equals to $H_i \times W_i$, *i.e.*, the number of all pixels in the F. Using learned linear transformations W^Q, W^K, and $W^V \in \mathbb{R}^{C \times C}$, tokens X are projected into corresponding query Q, key K, and value vectors $V \in \mathbb{R}^{N \times C}$. Here, the spatial scale of K and V is further reduced to decrease memory consumption, and then self-attention is performed as:

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{C_{head}}})V, \tag{2.1}$$

with C_{head} the channel dimension of each attention head in SRA. According to Eq. (2.1), each token in the entire input space F is matched with any tokens, including itself. Our depth comple-

tion network benefits from the self-attention mechanism in two folds: 1) it extends the receptive field of our network to the full image in each Transformer layer; 2) as we have embedded each token with both depth and RGB image information, the self-attention mechanism explicitly compare the similarity of each pixel not only by appearance but also by depth with dot-product operation. Thus, reliable depth information can be broadcasted to the whole image, enabling to correct erroneous pixels.

We boost the representation power of the convolutional path with channel and spatial attention [241]. On the one hand, it helps to model locally accurate attention and reduce noise. On the other hand, due to the semantic gap between convolution and Transformer, the increased modeling capacity by using the attention mechanism enables this path to focus on important features provided by the Transformer layer while suppressing the unnecessary ones. Finally, by concatenating the reshaped feature from the Transformer-based path, we fuse the two paths with a 3×3 convolution and send it to the next block or stage.

Taking the proposed JCAT block as the basic unit, we build stages 2-5 with repeated configurations. As reported in Tab. 2.1, we scale-up the 4 stages in CompletionFormer from tiny, small to base scale. Our results demonstrate the superiority of JCAT design compared to recent Vision Transformers [103, 126, 230] in depth completion task.

2.2.3 Decoder

In the decoder, outputs from each encoding layer are concatenated and further processed by the corresponding decoding layers via skip connections. To accommodate diverse scale features better, the features from previous decoder layer is upsampled to current scale with a deconvolution layer and convolutional attention mechanism [241] is also exploited to strengthen feature fusion in channel and spatial dimensions. Finally, the fused result from the decoder is concatenated with features from stage one and fed to the first convolution layer of the prediction head. Its output is concatenated with the raw feature from RGB and depth embedding module (Sec. 2.2.1) and sent to another convolution, which is in charge of initial depth prediction D^0.

2.2.4 SPN Refinement and Loss Function

Considering that the accurate depth values from the sparse input may not be well preserved after going through U-Net [27, 80], spatial propagation network [122] has been a standard operation for final refinement. Recent work [27, 28, 80, 152] mainly focuses on improving the spatial propagation network from fixed-local to nonlocal propagation. While in our experiments (Tab. 2.2), we observe that, with our enhanced U-Net backbone, the network is able to provide good depth affinity and thus obtain almost the same accuracy with fixed-local [27, 28] or non-local [152] neighbours for spatial propagation. With regard to CSPN++ [28] consumes more computation cost, we adopt the non-local spatial propagation network [152] (NLSPN) for further refinement. Specifically, let $D^t = (d^t_{u,v}) \in \mathbb{R}^{H \times W}$ denotes the 2D depth map updated by spatial propagation at step t, where $d^t_{u,v}$ denotes the depth value at pixel (u, v), and H, W denotes the height and width of the D^t respectively. The propagation of $d^t_{u,v}$ at step t with its non-local neighbors $N^{NL}_{u,v}$ is defined as follows:

$$d^t_{u,v} = w_{u,v}(0,0)d^{t-1}_{u,v} + \sum_{(i,j)\in N^{NL}_{u,v}, i\neq 0, j\neq 0} w_{u,v}(i,j)d^{t-1}_{i,j}, \tag{2.2}$$

where $w_{u,v}(i,j) \in (-1,1)$ describes the affinity weight between the reference pixel at (u,v) and its neighbor pixel at (i,j) and $w_{u,v}(0,0) = 1 - \sum_{(i,j)\in N_{u,v}, i\neq 0, j\neq 0} w_{u,v}(i,j)$ stands for how much the original depth $d_{u,v}^{t-1}$ will be preserved. Moreover, the affinity matrix w is also outputted by the decoder and modulated by a predicted confidence map from decoder to prevent less confident pixels from propagating into neighbors regardless of how large the affinity is. After K steps spatial propagation, we get the final refined depth map D^K.

Finally, following [152], a combined L_1 and L_2 loss is employed to supervise the network training as follows:

$$L(\hat{D}, D^{gt}) = \frac{1}{|V|} \sum_{v\in V} \left(|\hat{D}_v - D_v^{gt}| + |\hat{D}_v - D_v^{gt}|^2 \right), \tag{2.3}$$

where $\hat{D} = D^K$, and V is the set of pixels with valid depth in ground truth D^{gt}, and $|V|$ denotes the size of set V.

2.3 Experiments

2.3.1 Datasets

NYUv2 Dataset [187]: it consists of RGB and depth images captured by Microsoft Kinect [138] in 464 indoor scenes. Following the similar setting of previous depth completion methods [152, 294], our method is trained on 50,000 images uniformly sampled from the training set and tested on the 654 images from the official labeled test set for evaluation. For both training and test sets, the original frames of size 640×480 are half down-sampled with bilinear interpolation and then center-cropped to 304×228. The sparse input depth is generated by random sampling from the dense ground truth.

KITTI Depth Completion (DC) Dataset [217]: it contains 86 898 training data, 1 000 selected for validation, and 1 000 for testing without ground truth. The original depth map obtained by the Velodyne HDL-64e is sparse, covering about 5.9% pixels. The dense ground truth is generated by collecting LiDAR scans from 11 consecutive temporal frames into one, producing near 30% annotated pixels. These registered points are verified with the stereo image pairs to eliminate noisy values. Since there is no LiDAR return at the top of the image, following [152], input images are bottom center-cropped to 240×1216 for training, validation and testing phases.

2.3.2 Implementation Details

We implement our model in PyTorch [154] on 4 NVIDIA 3 090 GPUs, using AdamW [127] as optimizer with an initial learning rate of 0.001, $\beta_1 = 0.9,\ \beta_2 = 0.999$, weight decay of 0.01. The batch size per GPU is set to 3 and 12 on KITTI DC and NYUv2 datasets, respectively. On the NYUv2 dataset, we train the model for 72 epochs and decay the learning rate by a factor of 0.5 at epochs 36, 48, 60, 72. For the KITTI DC dataset, the model is trained with 100 epochs, and we reduce the learning rate by half at epochs 50, 60, 70, 80, 90. The supplementary material outlines more details about network parameters.

	CompletionFormer	RMSE↓ (mm)	MAE↓ (mm)	Params.↓ (M)	FLOPs ↓ (G)
(A)	w/ cascaded connection	91.5	35.7	82.6	429.6
(B)	w/ parallel connection	**90.0**	**35.0**	82.6	429.6
(C)	w/o Spatial and Channel Attention	91.1	35.5	**82.5**	**429.4**
(D)	w/ dual-branch encoders	94.0	36.4	161.0	661.4

	Backbone Type	Attention Decoder	Refinement Iterations	RMSE↓ (mm)	MAE↓ (mm)	Params.↓ (M)	FLOPs ↓ (G)
(E)	ResNet34 [75]	✗	18	92.3	36.1	**26.4**	542.2
(F)	ResNet34 [75]	✓	18	91.4	35.5	28.1	582.1
(G)	Swin-Tiny [126]	✓	18	92.6	36.4	38.1	634.8
(H)	PVT-Large [230]	✓	18	91.4	35.6	68.3	419.8
(I)	MPViT-Base [103]	✓	18	91.0	35.5	83.1	1259.3
(J)	CMT-Base [70]	✓	18	92.0	35.9	47.6	**358.7**
(K)	Ours-Small	✓	18	90.1	35.2	82.6	439.1
(L)	Ours-Small	✓	6	**90.0**	35.0	82.6	429.6
(M)	Ours-Small	✓	CSPN++	90.3	**34.9**	82.7	446.4
(N)	Ours-Tiny	✓	6	90.9	35.3	45.8	389.4
(O)	Ours-Base	✓	6	90.1	35.1	146.7	499.6

Table 2.2: Ablation study on NYU Depth v2 [187]. We ablate the settings of our network in the following aspects: the backbone type, the convolutional attention mechanism in decoder and the iterations of NLSPN Refinement module. FLOPs are measured with input resolution 480×640.

2.3.3 Evaluation Metrics

Following the KITTI benchmark and existing depth completion methods [152, 294], given the prediction $\hat{D}$ and ground truth D^{gt}, we use the standard metrics for evaluation: (1) root mean square error (RMSE); (2) mean absolute error (MAE); (3) root mean squared error of the inverse depth (iRMSE); (4) mean absolute error of the inverse depth (iMAE); (5) mean absolute relative error (REL).

2.3.4 Ablation Studies and Analysis

We assess the impact of the main components of our CompletionFormer on NYUv2 dataset [187]. Following previous methods [117, 152], we randomly sample 500 depth pixels from the ground truth depth map and input them along with the corresponding RGB image for network training. Results are reported in Tab. 2.2.

Cascaded vs Parallel Connection. We can notice that the cascaded design (A) shows inferior performance compared to parallel style (B). This conclusion is also confirmed in (I) and (J), as CMT-Base gets even worse RMSE (92.0) than MPViT-Base [103] (91.0). It indicates the parallel connection is more suitable for information interaction between streams with different

	SPN	ResNet34	Swin-Tiny	PVT-Large	MPViT-Base	CMT-Base	Ours-Small
RMSE(mm)↓	✗	106.5	106.5	106.7	100.2	108.4	**99.2**
	✓	91.4	92.6	91.4	91.0	92.0	**90.0**

Table 2.3: Ablation study without SPN module. We report the accuracy of different backbone with/without SPN on NYUv2.

contents and semantics on depth completion task. Thus, we adopt parallel connection as our final scheme.

Spatial and Channel Attention in JCAT block. Previous methods combine the Transformer with plain convolutions [70, 103, 150, 157]. Here, we also ablate the case when we disable the spatial and channel attention at the convolutional path of our proposed JCAT block (C). The drop in accuracy (RMSE increases from 90.0 to 91.1) confirms that increasing the capacity of convolutions is vital when combining convolution with Vision Transformer, and it only leads to negligible FLOPs increase (0.2G FLOPs).

Single- or Dual-branch Encoder. Similar to previous methods [169, 203], we test the dual-branch architecture, which encodes the RGB and depth information separately (D). For feature communication between two branches, we include the spatial and channel attention mechanism [241] at the end of each stage in the encoder. However, the worse results compared to our single-branch design (B) demonstrates that embedding the multimodal information at the early stage is much more effective and efficient.

Comparisons with General Feature Backbones. In RMSE, our novel CompletionFormer in small scale (K) outperforms pure CNNs based method (F) and those pure Transformer-based variants (G, H) counting comparable FLOPs with respect to our model. In particular, compared to the recent MPViT-Base [103] (I) and CMT-Base [70] (J) which also integrate CNNs and Transformer for feature extraction, our network achieves higher accuracy and much lower computational overhead (429.6G FLOPs) than MPViT-Base (1259.3G FLOPS).

Decoder. Compared to our baseline (E), *i.e.*, NLSPN [152], introducing spatial and channel attention for multiscale feature fusion in the decoder (F) further improves the results, RMSE drops from 92.3 to 91.4.

SPN Refinement Iterations. Our network (L) requires as few as 6 iterations for SPN refinement to converge to the best result. Compared to baseline (E) which requires 18 iterations, our enhanced U-Net has collected information from the whole image and thus doesn't need lots of iterations to propagate to long distance. Even when the non-local refinement in NLSPN replaced with fixed-local neighbors in CSPN++ [28] (M), the accuracy remains almost the same. It indicates that our CompletionFormer can learn good affinity locally and globally, thus helping to soften the problem raised by the limited and fixed range of aggregation in CSPN++ [28].

Model Scales. Our models in various scales (L, N, O), benefiting from local and global cues, achieve significant improvement compared to the pure CNN-based baseline (E), while counting fewer FLOPs. To trade-off between accuracy and efficiency, we select our model in small scale (L) as our final architecture for the remaining experiments.

With/without SPN refinement. To conclude, in Tab. 2.3 we show the results achieved by different backbones with and without SPN refinement. Ours yields the most accurate results in both cases.

Scanning Lines	Method	RMSE↓ (mm)	MAE↓ (mm)	iRMSE↓ (1/km)	iMAE↓ (1/km)
1	NLSPN	3507.7	1849.1	13.8	8.9
	DySPN	3625.5	1924.7	13.8	8.9
	Ours-ViT	3507.2	1807.7	12.1	7.8
	Ours	**3250.2**	**1582.6**	**10.4**	**6.6**
4	NLSPN	2293.1	831.3	7.0	3.4
	DySPN	2285.8	834.3	6.3	3.2
	Ours-ViT	2241.2	795.9	5.8	2.9
	Ours	**2150.0**	**740.1**	**5.4**	**2.6**
16	NLSPN	1288.9	377.2	3.4	1.4
	DySPN	1274.8	366.4	3.2	1.3
	Ours-ViT	1268.9	360.7	3.3	1.3
	Ours	**1218.6**	**337.4**	**3.0**	**1.2**
64	NLSPN	889.4	238.8	2.6	1.0
	DySPN	878.5	228.6	**2.5**	1.0
	Ours-ViT	872.0	226.2	**2.5**	1.0
	Ours	**848.7**	**215.9**	**2.5**	**0.9**

Table 2.4: Ablation study on scanning lines of LiDAR sensor on KITTI DC [217] datasets. Ours-ViT denotes that only the Transformer layer is enabled in our proposed block.

Method		RMSE↓ (m)				
		PackNet-SAN	GuideNet	NLSPN	Ours-ViT	Ours
Sample Number	0	-	-	0.562	0.544	**0.490**
	50	-	-	0.223	0.218	**0.208**
	200	0.155	0.142	0.129	0.130	**0.127**
	500	0.120	0.101	0.092	0.091	**0.090**

Table 2.5: Sparsity Studies on the NYUv2 Dataset. Evaluation with 0, 50, 200 and 500 samples.

2.3.5 Sparsity Level Analysis

Depth maps captured by sensors such as Microsoft Kinect [138] and Velodyne LiDAR sensor are unevenly distributed and often contain outliers. To compare the effectiveness of our model with current state-of-the-art methods [69, 117, 152, 203] when dealing with this challenging input depth, we manually generate sparse data at different settings for network training and testing. As for noise, since captured depth maps are often corrupted by sensor noise or displacement between RGB camera and depth sensor [165], we do not add extra noise to the input depth for experiments.

Outdoor Scene. We conduct exhaustive experiments on the KITTI DC dataset [217] to illustrate the robustness of our CompletionFormer when input with different sparsity levels. For all experiments, we train on randomly selected 10 000 RGB and LiDAR pairs from official training data (due to resource constraints) and test on the selected validation dataset provided by KITTI. Furthermore, following [85], we sub-sample the raw LiDAR points captured by Velo-

Method	KITTI DC				NYUv2	
	MAE↓ (mm)	iMAE↓ (1/km)	iRMSE↓ (1/km)	RMSE↓ (mm)	RMSE↓ (m)	REL↓
CSPN [27]	279.46	1.15	2.93	1019.64	0.117	0.016
DeepLiDAR [165]	226.50	1.15	2.56	758.38	0.115	0.022
GuideNet [203]	218.83	0.99	2.25	736.24	0.101	0.015
NLSPN [152]	199.59	0.84	1.99	741.68	0.092	**0.012**
PENet [80]	210.55	0.94	2.17	730.08	-	-
ACMNet [286]	206.09	0.90	2.08	744.91	0.105	0.015
TWISE [85]	195.58	0.82	2.08	840.20	0.097	0.013
RigNet [252]	203.25	0.90	2.08	712.66	**0.090**	0.013
GuideFormer [169]	207.76	0.97	2.14	721.48	-	-
DySPN [117]	192.71	0.82	**1.88**	709.12	**0.090**	**0.012**
Ours (L_1)	**183.88**	**0.80**	1.89	764.87	-	-
Ours (L_1+L_2)	203.45	0.88	2.01	**708.87**	**0.090**	**0.012**

Table 2.6: Quantitative evaluation on KITTI DC and NYUv2.

dyne HDL-64e in azimuth-elevation space into 1, 4, 16 and 64 lines to simulate the LiDAR-like patterns. All methods have been fully **retrained** using the official code with variable lines of LiDAR inputs (For DySPN [117], as no public code available, the results have been provided to us by the author using the same selected training list). To get a throughout understanding of the local details encoded by convolution and global context gathered by Transformer, we present the results estimated by pure CNN-based methods (*i.e.*, NLSPN and DySPN), a fully Transformer-based variant of our network (Ours-ViT) and our CompletionFormer complete architecture (Ours), which integrates both paradigms. As reported in Tab. 2.4, thanks to the global correlations built by Transformers, our model relying on these blocks only (Ours-ViT) exhibits better results in all metrics as the LiDAR points get sparser. However, solely using Transformer layers makes it difficult to distinguish the objects from the background, as shown in Fig. 2.4. By coupling the local features and global representations, our complete model (Ours) significantly decreases the errors in all metrics.

Indoor Scene. On the NYUv2 dataset [187], we randomly sample 0, 50, 200 and 500 points from the ground truth depth map to mimic different depth sparsity levels while keeping the ground truth depth used for supervision unchanged. Both our model and NLSPN with publicly available code are retrained for a fair comparison, while the results of GuideNet [203] and PackNet-SAN [69] are taken from the original papers. In Tab. 2.5, CompletionFormer with both CNNs and Transformers consistently outperforms all other methods in any cases. Qualitative results concerning the NYUv2 dataset [187] are provided in Fig. 2.6. In both visualized cases, we can notice the improved results yielded by our CompletionFormer compared to NLSPN [152]. Especially for the transparent regions near the windows in both cases, with local details of convolution and global cues of Transformer, our complete model (Ours) predicts clear object boundaries while NLSPN and Ours-ViT give blurry estimations.

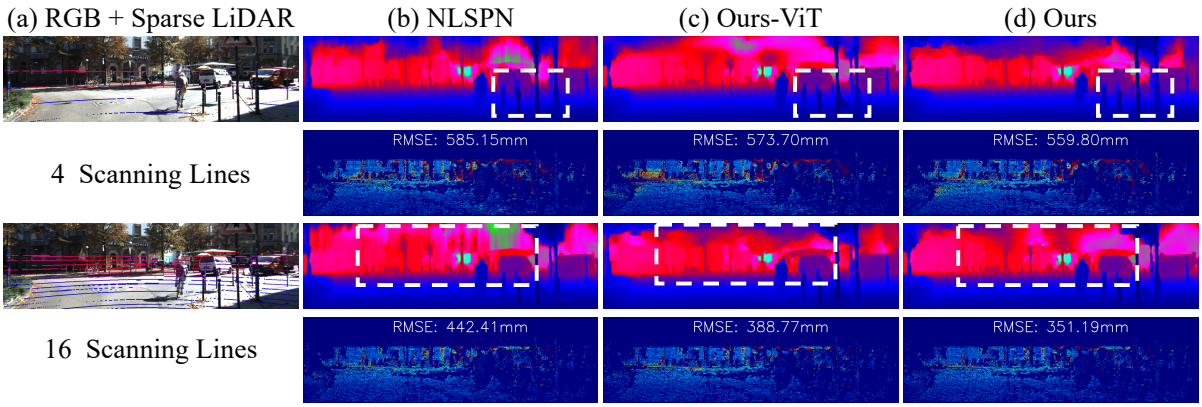


Figure 2.4: Qualitative results on KITTI DC selected validation dataset with 4 and 16 LiDAR scanning lines. We attach the subsampled LiDAR lines to the corresponding RGB image for better visualization. Ours-ViT denotes that only the Transformer layer is enabled in our proposed block. A colder color in depth and error maps denotes a lower value.

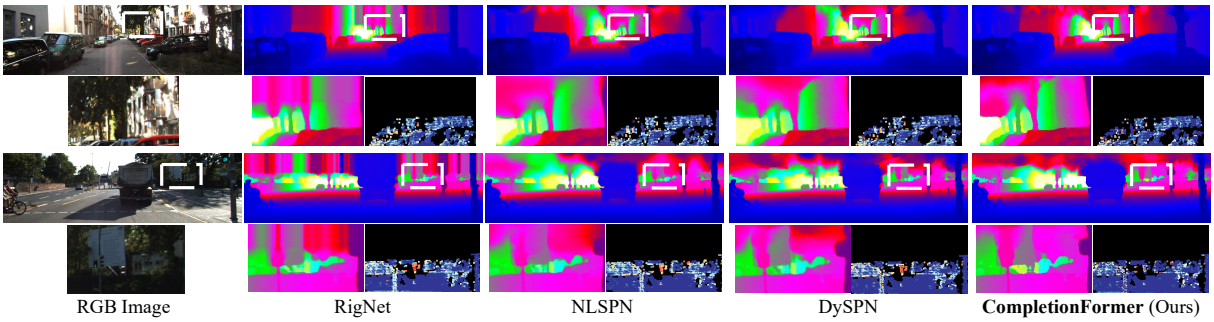


Figure 2.5: Qualitative results on the KITTI depth completion test set. Comparisons of our method against state-of-the-art methods including RigNet [252], NLSPN [152], DySPN [117] are presented. We provide RGB images, dense predictions, zoom-in views of challenging areas and corresponding error maps for better visualization.

2.3.6 Comparison with SOTA Methods

This section comprehensively assesses the performance of state-of-the-art (SOTA) methods. On **indoor**, *i.e.*, NYUv2 dataset [187], CompletionFormer achieves the best results as reported in Tab. 2.6. When moving to **outdoor** dataset, MAE, iMAE, RMSE and iRMSE are adopted for benchmark on KITTI depth completion (DC) dataset [217]. Empirically, our model trained with only L_1 loss achieved the best results on two among four metrics (MAE and iMAE). By jointly minimizing L_1 and L_2 losses, CompletionFormer ranked first on RMSE metric among published methods. Qualitative results on the KITTI DC test dataset are provided in Fig. 2.5. By integrating convolutions and Transformers, our model performed better near depth missing areas (*e.g.*, the zoom-in visualization on the second and fourth rows), textureless objects (*e.g.*, the cars on the first row) and small objects (*e.g.*, the pillars and tree stem far in the distance, on second and fourth rows).

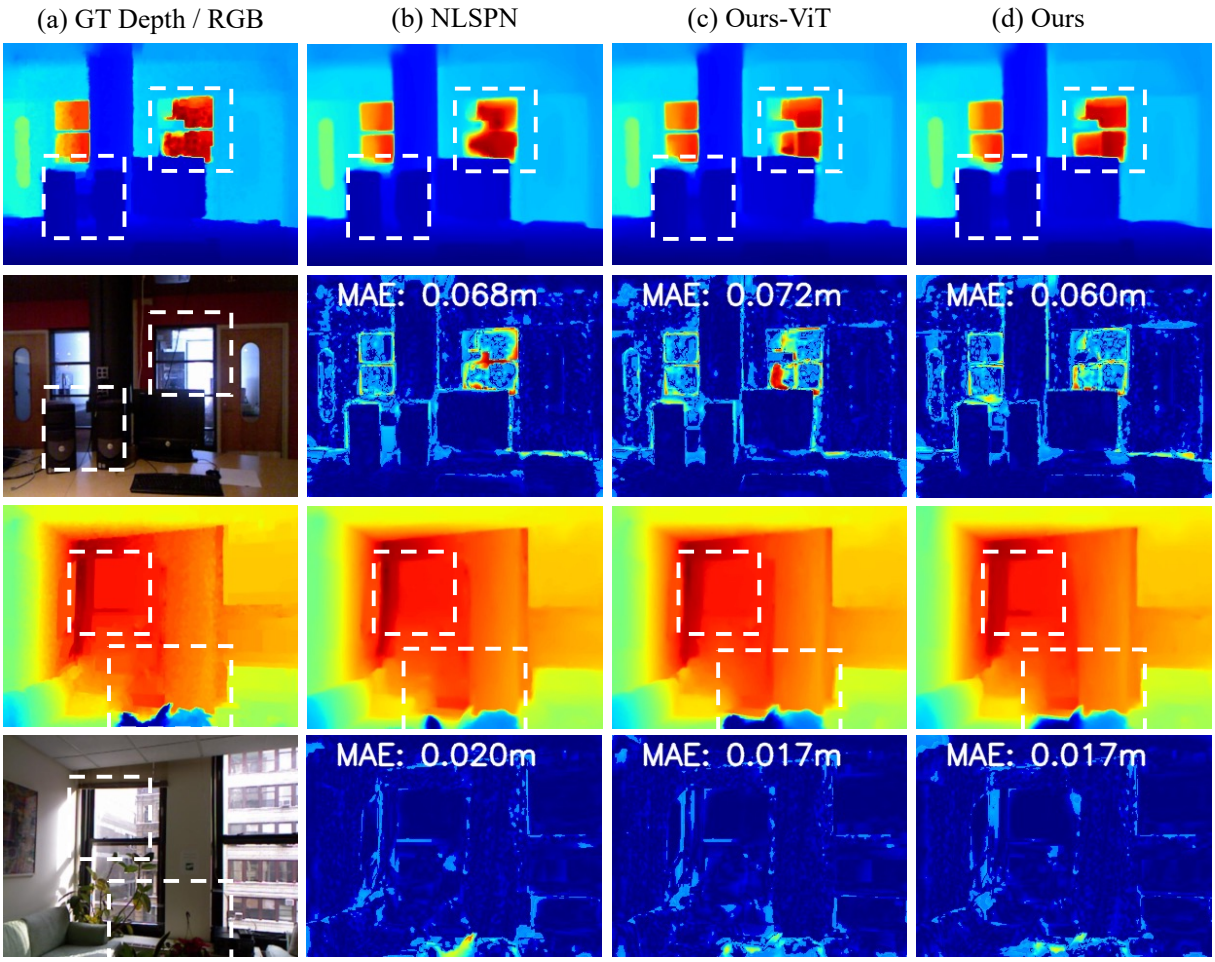


Figure 2.6: Qualitative results on NYUv2 dataset. Comparisons of our method against state-of-the-art method, *i.e.*, NLSPN [152] are presented. We provide RGB images, and dense predictions. The colder the colors of the error map, the lower the errors. Ours-ViT denotes that only the Transformer layer is enabled in our proposed block.

2.4 Conclusions

We proposed a single-branch depth completion network, CompletionFormer, seamlessly integrating convolutional attention and Transformers into one block. Extensive ablation studies demonstrate the effectiveness and efficiency of our model in depth completion when the input is sparse. This novel design yields state-of-the-art results on indoor and outdoor datasets. Currently, CompletionFormer runs at about 10 FPS: decreasing its runtime further to meet real-time requirements will be our future work.

Chapter 3

Self-Supervised Monocular Depth Estimation with a Vision Transformer

The content of this chapter has been presented at the International Conference on 3D Vision (3DV 2022) - "MonoViT: Self-Supervised Monocular Depth Estimation with a Vision Transformer" [284].

Depth perception is at the foundation of several high-level computer vision applications such as autonomous driving, robotics, and augmented reality [206]. However, despite the steady progress of active depth-sensing technologies brought by devices such as LiDARs, Time-of-Flight (ToF) cameras and more, the possibility of estimating depth from standard images is generally preferable, mainly because of three (among many) advantages: higher image resolution, lower hardware costs and potentially unconstrained working range. Although using two or more images [161] is often the preferred choice, estimating depth from a single image allows for the deployment of depth-sensing solutions on any monocular configuration, still representing the most diffused setting in most practical cases nowadays.

Deep learning paradigms favored the blooming of this latter approach [12, 48, 55, 101], at the cost of requiring extensive collections of images annotated with depth labels in order to carry out the training effectively. However, considering the high cost of collecting dense depth labels for this purpose, self-supervised monocular depth estimation [63, 293] has emerged in the literature enabling significant progress in recent years [281]. These approaches replace supervised losses on depth labels with supervisory signals derived from image reprojection across different views, by exploiting the geometric relationship between frames, *i.e.*, the scene depth itself and camera pose. Since these networks aim at learning depth, two prominent cases exist to deal with the relative pose across frames. They consist of either knowing it as a prior – for instance, by collecting stereo images and training on them [63] – or estimating it during training, allowing in this second case to train on unconstrained monocular videos [293]. The latter configuration turns out to be the preferred choice for practical deployment since it simply requires a single moving camera for gathering training data. For this reason, we stick to monocular videos for training purposes. However, view reconstruction based losses suffer from occlusions, dynamic objects and photometric changes, which severely limit the performance of the network [281]. Therefore, novel constraints [64] and additional cues [97, 264] (like semantic segmentation, optical flow and surface normals) are often used to reduce the shortcomings mentioned above.

Improving the network backbone of depth networks is another well-known effective way to gain accuracy. Recent research has shown that the encoder is crucial for achieving this [64, 293].

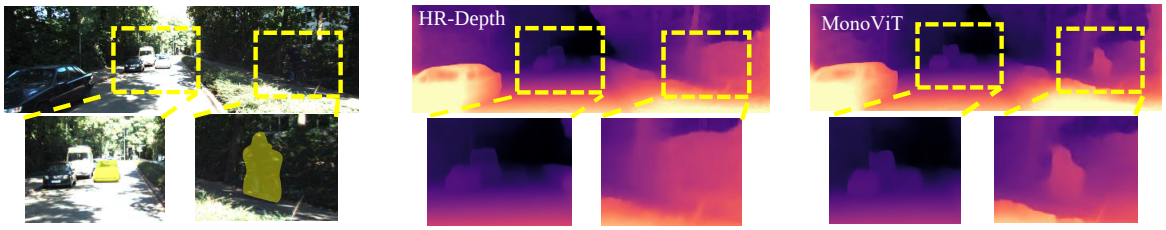


Figure 3.1: Effects of global reasoning on self-supervised monocular depth estimation. The limited receptive field of existing solutions (*e.g.*, HR-Depth [129], in the middle) often yields inaccurate depth estimation, losing fine-grained details (like the car and cyclist over imposed in yellow). On the contrary, our MonoViT architecture (right) achieves superior results.

Different kinds of backbone, such as VGGNet, ResNet, HRNet and PackNet, made their way into the self-supervised monocular depth estimation task [64, 67, 292, 293]. Moreover, to improve the feature extraction and processing ability, new frameworks like HR-Depth [129] and CADepth [251] also introduced attention modules. However, we argue that a shared shortcoming of existing self-supervised models falls in the reduced receptive field of Convolutional Neural Networks (CNNs). This fact represents an implicit bottleneck for current dense estimation methods, dampening accuracy, and the capacity to generalize to different domains. Specifically, the local nature of convolutions leads CNNs in their first layers – *i.e.*, those in charge of modeling fine-grained details – to extract features missing long-range relationships across the same image. Going deeper with convolutions makes the receptive field wider, yet it does not reach the whole image. Fig. 3.1 highlights the effect of this shortcoming. CNNs based frameworks sometimes fail to estimate foreground-background structures due to the lack of global perceiving and long-range relationship among modelled pixels. Vision Transformers (ViTs) [39, 44, 243] recently showed outstanding results on tasks such as object detection [39] and semantic segmentation [243], thanks to their capacity to model long-range relationships between pixels and thus a global receptive field. The popularity of ViTs has also reached supervised depth estimation as well [112, 167], yet being not adopted for self-supervised monocular depth estimation.

This chapter takes this missing step and explores ViTs for self-supervised monocular depth estimation by proposing the MonoViT architecture. It combines both convolutional layers and state-of-the-art (SoTA) Transformer blocks [102] within its backbone to model both the local information (objects) and global information (relationship among foreground and background, among objects) within the same image. This strategy allows us to remove the bottleneck caused by the limited perceptive fields of CNNs encoders, leading to naturally finer-grained predictions, as shown in Fig. 3.1. We evaluate the performance of MonoViT on the popular KITTI dataset, using the standard split by Eigen *et al.* [48]. The comparison to SoTA solutions highlights the constantly superior accuracy of our framework. Moreover, we also analyze the generalization capability of self-supervised monocular depth estimation networks across different datasets. Purposely, we compare MonoViT with its main competitors on the Make3D [177] and DrivingStereo [255] datasets, highlighting, even in this case, the superior generalization capacity of MonoViT.

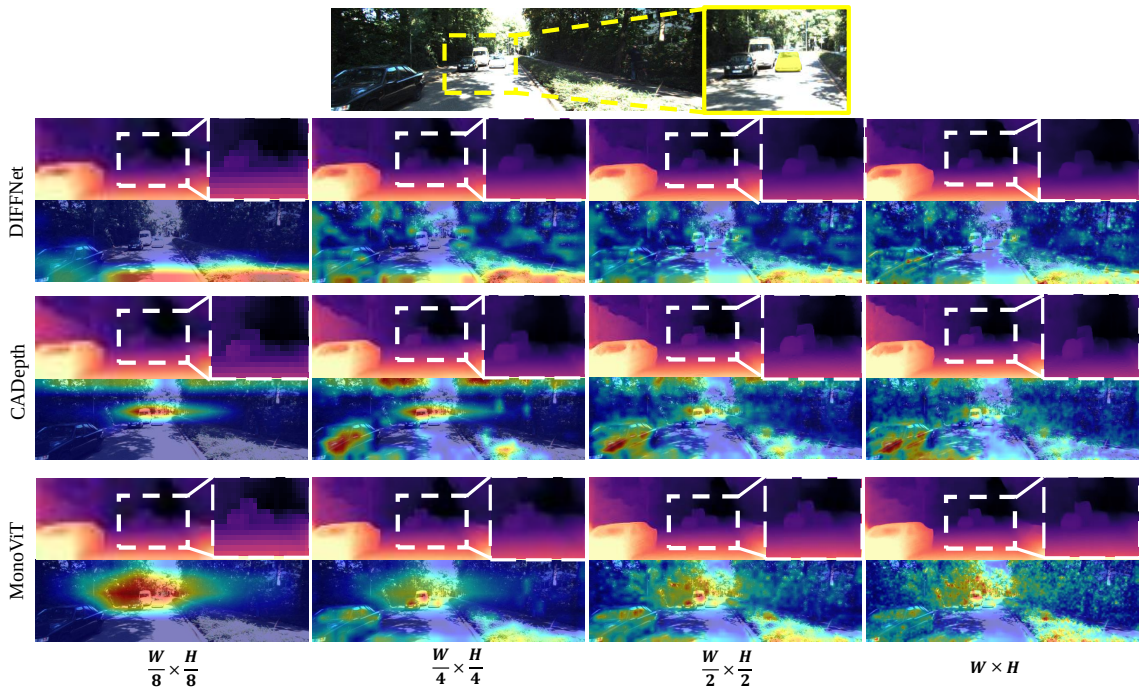


Figure 3.2: Attention maps of SoTA methods and our MonoViT. The first row shows the RGB image, and the highlighted car is the region we want to analyze. In the next two rows, we report multiscale disparity predictions and attention maps of each method. For an object that is small in size and hard to distinguish from the background, such as the car highlighted, we notice how MonoViT can predict its disparity even at the lowest resolution (*i.e.*, $\frac{H}{8} \times \frac{W}{8}$). At the same time, other methods fail to capture it.

3.1 Method

3.2 Proposed framework

This section analyzes the necessity for introducing a Transformer for self-supervised monocular depth estimation. Then, we describe our MonoViT network architecture and the loss functions used for the self-supervised training of our framework.

Unlike supervised depth estimation methods, the supervisory signal of self-supervised approaches derives from image reprojection across different, nearby viewpoints. Thus, to achieve good performance, this formulation requires the network to accurately perceive the scene structure: a challenging task, especially for regions with hard to distinguish foreground objects from the background. Current SoTA networks [129, 251] rely on traditional convolutional layers for aggregating context information and gradually lift the receptive field of the network through a cascade of layers and strided convolution [171]. However, given the intrinsic locality of the convolution operator, CNNs hardly model long-range appearance similarity among objects, in particular within the shallowest features. An example of this occurs when the foreground objects have a texture similar to the one of the background. In such a case, the feature backbone

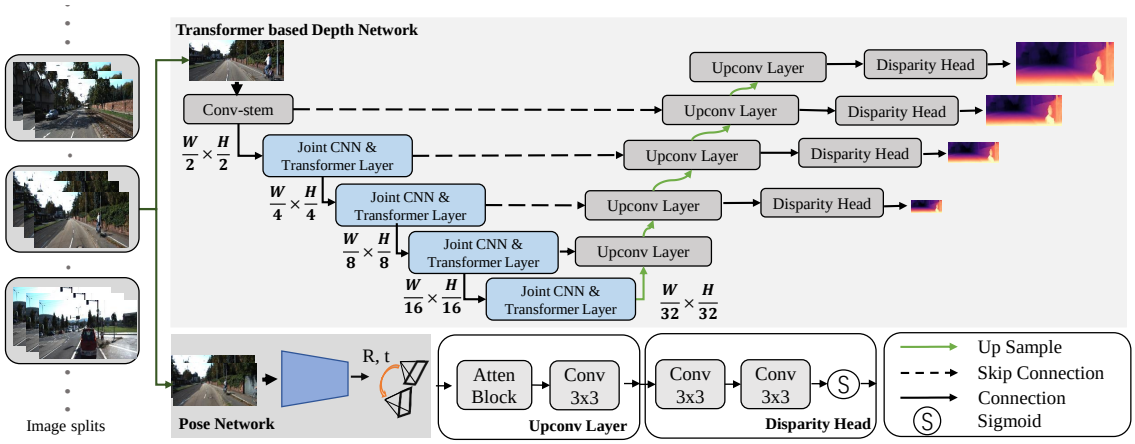


Figure 3.3: Overview of our MonoViT architecture. Our MonoViT consists of two parts, Depth Network and Pose Network. For Depth Network, both Transformer [102] and convolutional layer are adopted to enhance the feature modeling and depth inferring. For pose estimation between temporally adjacent images, we use a lightweight PoseNet as in previous works [64, 129, 251, 292].

tends to embed them in the same semantic context, and the whole architecture cannot distinguish between foreground and background depths. Fig. 3.2 shows this behaviour; we can notice that the car in the middle of the road is hard to spot from the ground due to the strong sunlight. CNNs such as CADepth [251] and DIFFNet [292] predict a depth for the car similar to the one of the ground plane. This fact is due to their encoder network paying more attention to the ground than the car itself. Hence, we propose integrating convolutions and ViT blocks to address the standard limitation of the former, since the latter has more significant potential for modeling long-range correlation.

Driven by this rationale, we design our Monocular Vision Transformer framework, **MonoViT** in short, as shown in Fig. 3.3. It includes a DepthNet and a PoseNet, respectively, designed for depth prediction of each input image and pose estimation and trained through image reconstruction losses.

3.2.1 DepthNet Architecture

As typical in previous works [64, 293], we design our DepthNet as an encoder-decoder architecture.

Depth encoder. As pointed out by recent research [64, 224, 251, 293], the encoder is crucial for effective features extraction. Inspired by one of the most recent Transformer architectures – *i.e.*, MPViT [102], in which a Multi-Path Transformer Block is proposed for simultaneously representing local and global context extracted from images – we follow such a design to build the key components of our depth encoder in five stages. Given the current input image, we adopt a Conv-stem block consisting of two convolutions with kernel size 3×3 and stride of 2 only at the first convolution, generating features with size $\frac{H}{2} \times \frac{W}{2}$. From stage two to stage five, we stack the Multi-Path Transformer Blocks in each stage, shown as "Joint CNN & Transformer Layer"

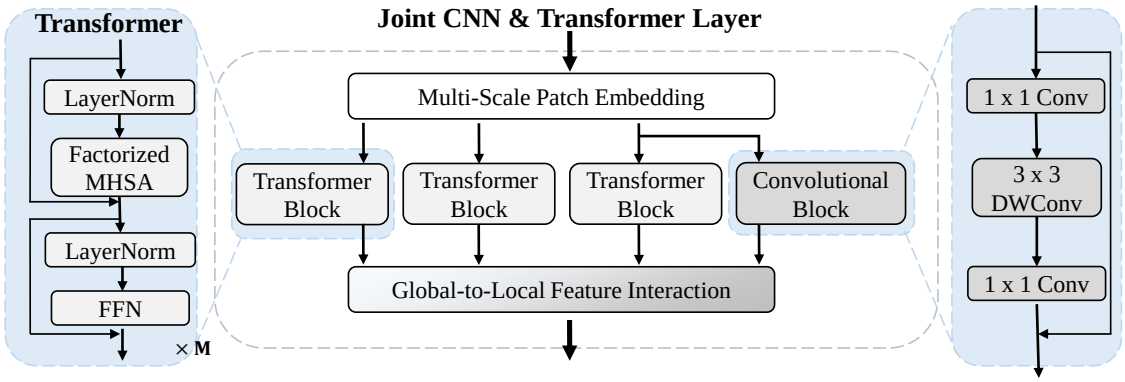


Figure 3.4: Joint CNN & Transformer Layer used in depth encoder. Each Transformer block contains M Transformer layers, consisting of a Layer Normalization (LayerNorm) module, a Factorized Multi-head Self Attention (MHSA) layer [102], another Layer Normalization and a Feed-forward Network (FFN).

in Fig. 3.3. Precisely, each "Joint CNN & Transformer Layer" (shown in Fig. 3.4) consists of a Multi-Scale Patch Embedding layer, used to embed various-sized visual tokens in parallel – in our case, four parallel convolutional blocks extract features with a receptive field of 3×3, 3×3, 5×5 and 7×7 pixels by stacking multiple 3×3 convolutional layers. Then, considering the advantage of ViT at building global dependencies while shows limitations modeling local details [102], extracted tokens are processed through both convolutional layers and Transformers blocks, in a parallel and complementary manner – *i.e.*, using the four branches shown in Fig. 3.4, respectively three parallel Transformer blocks and a convolutional block, this latter made of 1×1, 3×3 depthwise and 1×1 convolutions. While the convolutional branch constructs the local relationship between neighbors within features $\mathbf{L}$, the three Transformer Blocks model the information interaction over the whole input space within features $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$, thanks to the self-attention mechanism. Specifically, these latter take a sequence of visual tokens embedded by the Multi-Scale Patching Embedding module and project them into a query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V} \in \mathbb{R}^{N\times C}$) vectors through three separated but structure same heads (where N denotes the number of visual tokens, equal to the total number of pixels in the input space). The self-attention mechanism is implemented in an efficient factorized way [102]:

$$\text{FactorAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \frac{\mathbf{Q}}{\sqrt{C}}(\text{softmax}(\mathbf{K})^T\mathbf{V}), \tag{3.1}$$

where C refers to the embedding dimension. Finally, a feature fusion block is used to collect and further enhance the interaction between local and global features extracted by the "Joint CNN & Transformer Layer" at stage i.

$$\mathbf{A}_i = \texttt{Concat}([\mathbf{L}_i, \mathbf{G}_{i,0}, \mathbf{G}_{i,1}, \mathbf{G}_{i,2}]), \tag{3.2}$$

$$\mathbf{X}_{i+1} = \mathcal{H}(\mathbf{A}_i), \tag{3.3}$$

with $\mathbf{A}_i \in \mathbb{R}^{H_i\times W_i\times C_i}$ being the aggregated feature and $\mathcal{H}(\cdot)$ a 1×1 convolutional layer which fuses them and yields the final feature $\mathbf{X}_{i+1}$ for the next stage $i+1$.

A clear benefit of integrating convolutions with Transformer is the comprehensive – both local and global – interaction between pixels. It helps the network to perceive the structure and relative position of objects so that small foreground objects can be preserved even at the lowest resolution, rather than collapsed into similar texture background as shown in Fig. 3.2.

Depth decoder. Taking the multi-scale features from the depth encoder, cross-layer, and cross-scale connections are adopted in our depth decoder to gradually increase the spatial resolution, as shown in Fig. 3.3. Considering the context difference between features at different scales, e.g. higher resolution features favour fine-grained details, we enhance cross-scale feature fusion with both spatial and channel attention mechanisms [129, 292] (*i.e.*, our Atten Block). Finally, four heads – made of two convolutional layers and a Sigmoid activation – are in charge of disparity (inverse depth) prediction from corresponding aggregated features, outputting maps at full, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ resolution respectively.

3.2.2 PoseNet

Following [64, 129, 251, 292], our PoseNet favors a simple, yet effective implementation. Specifically, our PoseNet uses the lightweight structure of ResNet18 [75]. Receiving concatenated images $[\mathcal{I}, \mathcal{I}^{\dagger}]$ as input, it outputs a 6 DoF relative pose $\mathbf{T}$ between adjacent frames of a video sequence.

3.2.3 Self-supervised Learning

We cast depth estimation as an image reconstruction task, replacing ground truth labels with unlabeled, monocular videos at training time. The depth network takes a still (*i.e.*, target) image $\mathcal{I}$ and predict its dense inverse depth map d, from which depth $\mathcal{D}$ is derived as $1/d$ and forcing it to be in $[\mathcal{D}_{min}, \mathcal{D}_{max}]$ as in [64].

View reconstruction loss. By knowing camera intrinsics $\mathbf{k}$ and the predicted pose $\mathbf{T}$ between two nearby views, a reconstructed target image $\tilde{\mathcal{I}}$ is obtained as a function π of intrinsics, pose, source image $\mathcal{I}^{\dagger}$ and depth $\mathcal{D}$. A loss signal $\mathcal{L}_{ss}$ is computed as a function $\mathcal{F}$ of inputs $\tilde{\mathcal{I}}$ and $\mathcal{I}$:

$$\mathcal{L}_{ss} = \mathcal{F}(\tilde{\mathcal{I}}, \mathcal{I}) = \mathcal{F}(\pi(\mathcal{I}^{\dagger}, \mathbf{T}, \mathbf{k}, \mathcal{D}), \mathcal{I}). \tag{3.4}$$

$\mathcal{F}$ is usually obtained as a weighted sum between a structural similarity term and an intensity difference term. Popular choices for these two terms are the Structured Similarity Index Measure (SSIM) [236] and the L1 difference, as proposed in [64]:

$$\mathcal{F}(\tilde{\mathcal{I}}, \mathcal{I}) = \alpha \cdot \frac{1 - \text{SSIM}(\tilde{\mathcal{I}}, \mathcal{I})}{2} + (1 - \alpha) \cdot |\tilde{\mathcal{I}} - \mathcal{I}| \tag{3.5}$$

with α commonly set to 0.85 [64]. Besides, for each pixel p, the minimum among losses computed from forward and backward adjacent frames allows for softening the effect of occlusions [64] on the reprojection process

$$\mathcal{L}_{ss}(p) = \min_{i \in [1,-1]} \mathcal{F}(\tilde{\mathcal{I}}_i(p), \mathcal{I}(p)) \tag{3.6}$$

with '1' and '-1' referring to the forward and backward adjacent frames, respectively.

Smoothness loss. As in previous works [64, 292], the edge-aware smoothness loss is used to improve the inverse depth map d:

$$\mathcal{L}_{smooth} = |\partial_x d^*| e^{\partial_x I} + |\partial_y d^*| e^{\partial_y I}, \tag{3.7}$$

where $d^* = d/\hat{d}$ represents the mean-normalized inverse depth. Besides, following [64], an auto-mask μ is calculated to filter static frames and objects moving with the same motion of the camera.

Total loss. Finally, both the view reconstruction loss $\mathcal{L}_{ss}$ and the smoothness loss $\mathcal{L}_{smooth}$ are computed from outputs at each scale $s \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\}$ – brought to full resolution – and then averaged as $\mathcal{L}_{tot}$ to train MonoViT:

$$\mathcal{L}_{tot} = \frac{1}{4} \cdot \sum_{s=1}^{4} (\mu \cdot \mathcal{L}_{ss} + \lambda \cdot \mathcal{L}_{smooth}), \tag{3.8}$$

with λ being set to 10^{-3}.

3.3 Experiments

In this section, we report the outcome of our experiments, clearly supporting the superior accuracy of MonoViT at estimating depth across several benchmarks.

3.3.1 Implementation Details

We implement our MonoViT in Pytorch. The model is trained for 20 epochs on the KITTI dataset using AdamW [127] as optimizer and a batch size set to 12. The initial learning rate for PoseNet and depth decoder is 10^{-4}, while the Transformer-based depth encoder is trained with an initial learning rate of $5{\times}10^{-5}$. The number M of Transformer layers in each of the three Transformer blocks in the 'Joint CNN & Transformer Layer' is set as 1, 3, 6, 3 from stage 2 to stage 5 in the depth encoder, respectively. Both the pose encoder and depth encoder are pre-trained on ImageNet [41]. We use a single RTX 3090 GPU for the low resolution (640×192) experiments while 4 RTX 3090 GPUs for higher resolution (1024×320,1280×384) ones. Overall, network training requires about 15 hours. In our experiments, we adopt the same data augmentation detailed in [64, 129].

For evaluation, we compute the seven standard metrics (Abs Rel, Sq Rel, RMSE, RMSE log, $\delta_1 < 1.25$, $\delta_2 < 1.25^2$, $\delta_3 < 1.25^3$) proposed in [48] and used by most works in the literature.

3.3.2 Datasets

KITTI [62]. The KITTI stereo dataset contains 61 scenes, with a typical image size of 1242×375, captured using a stereo rig mounted on a moving car equipped with a LiDAR sensor. Following previous works in this field [64, 129, 292], we use the image split of Eigen *et al.* [48], which consists of 39810 monocular triplets for training and 4424 for validation. To compare with the existing solutions, we evaluate the single-view depth performance on the test split of [48] either using raw LiDAR (697 images) or improved ground truth labels [218] (652 images).

Method	Data	Resolution	lower is better				higher is better		
			Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Monodepth2 [64]	M	640×192	0.115	0.903	4.863	0.193	0.877	0.959	0.981
Sun [201]	M	640×192	0.117	0.863	4.813	0.192	0.871	0.959	0.982
SGDepth [97]	M+Se	640×192	0.113	0.835	4.693	0.191	0.879	0.961	0.981
SAFENet [32]	M+Se	640×192	0.112	0.788	4.582	0.187	0.878	0.963	0.983
VC-Depth [291]	M	640×192	0.112	0.816	4.715	0.190	0.880	0.960	0.982
PackNet‡ [67]	M	640×192	0.108	0.727	4.426	0.184	0.885	0.963	0.983
Mono-Uncertainty[160]	M	640×192	0.111	0.863	4.756	0.188	0.881	0.961	0.982
HR-Depth [129]	M	640×192	0.109	0.792	4.632	0.185	0.884	0.962	0.983
Johnston et al. [88]	M	640×192	0.106	0.861	4.699	0.185	0.889	0.962	0.982
CADepth⋆ [251]	M	640×192	0.105	0.769	4.535	0.181	0.892	0.964	0.983
DIFFNet† [292]	M	640×192	0.102	0.749	4.445	0.179	0.897	0.965	0.983
MonoFormer [6]	M	640×192	0.106	0.839	4.627	0.183	0.889	0.962	0.983
MonoViT (ours)	M	640×192	**0.099**	**0.708**	**4.372**	**0.175**	**0.900**	**0.967**	**0.984**
Monodepth2 [64]	MS	640×192	0.106	0.818	4.750	0.196	0.874	0.957	0.979
HR-depth [129]	MS	640×192	0.107	0.785	4.612	0.185	0.887	0.962	0.982
CADepth⋆ [251]	MS	640×192	0.102	0.752	4.504	0.181	0.894	0.964	0.983
DIFFNet† [292]	MS	640×192	0.101	0.749	4.445	0.179	0.898	0.965	0.983
MonoViT (ours)	MS	640×192	**0.098**	**0.683**	**4.333**	**0.174**	**0.904**	**0.967**	**0.984**

Method	Data	Resolution	lower is better				higher is better		
			Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Monodepth2 [64]	M	1024×320	0.115	0.882	4.701	0.190	0.879	0.961	0.982
Sun [201]	M	1024×320	0.110	0.791	4.557	0.184	0.887	0.964	0.983
SAFENet [32]	M+Se	1024×320	0.106	0.743	4.489	0.181	0.884	0.965	**0.984**
HR-Depth [129]	M	1024×320	0.106	0.755	4.472	0.181	0.892	0.966	**0.984**
FeatDepth [186]	M	1024×320	0.104	0.729	4.481	0.179	0.893	0.965	**0.984**
GCNDepth [134]	M	1024×320	0.104	0.720	4.494	0.181	0.888	0.965	**0.984**
CADepth⋆ [251]	M	1024×320	0.102	0.734	4.407	0.178	0.898	0.966	**0.984**
DIFFNet† [292]	M	1024×320	0.097	0.722	4.345	0.174	0.907	0.967	**0.984**
MonoViT (ours)	M	1024×320	**0.096**	**0.714**	**4.292**	**0.172**	**0.908**	**0.968**	**0.984**
PackNet‡ [67]	M	1280×384	0.104	0.758	4.386	0.182	0.895	0.964	0.982
SGDepth [97]	M+Se	1280×384	0.107	0.768	4.468	0.186	0.891	0.963	0.982
HR-Depth [129]	M	1280×384	0.104	0.727	4.410	0.179	0.894	0.966	**0.984**
CADepth⋆ [251]	M	1280×384	0.102	0.715	4.312	0.176	0.900	0.968	**0.984**
MonoViT (ours)	M	1280×384	**0.094**	**0.682**	**4.200**	**0.170**	**0.912**	**0.969**	**0.984**
Monodepth2 [64]	MS	1024×320	0.106	0.818	4.750	0.196	0.874	0.957	0.979
HR-Depth [129]	MS	1024×320	0.101	0.716	4.395	0.179	0.892	0.966	0.984
CADepth⋆ [251]	MS	1024×320	0.096	0.694	4.264	0.173	0.908	0.968	0.984
DIFFNet† [292]	MS	1024×320	0.094	0.678	4.250	0.172	0.911	0.968	0.984
MonoViT (ours)	MS	1024×320	**0.093**	**0.671**	**4.202**	**0.169**	**0.912**	**0.969**	**0.985**

Table 3.1: Results on the KITTI benchmark using the Eigen split [62]. Each method is grouped by input resolution (low: left, high: right) and training methodology (M: monocular videos, MS: binocular videos, Se: trained with semantic labels). The best scores are in **bold**. ⋆ refers to the current SoTA self-supervised method on the KITTI depth benchmark. † stands for the novel results from the official Github repository, better than published ones. ‡ refers to the model pretrained on Cityscapes [34], while the others are pretrained on ImageNet [41].

Method	lower is better			
	Abs Rel	Sq Rel	RMSE	RMSE log
Monodepth2 [64]	0.321	3.378	7.252	0.163
HR-Depth [129]	0.305	2.944	6.857	0.157
CADepth [251]	0.319	3.564	7.152	0.158
DIFFNet [292]	0.298	2.901	6.753	0.153
MonoViT (ours)	**0.286**	**2.758**	**6.623**	**0.147**

Table 3.2: Quantitative results on the Make3D Dataset [177]. Models trained on KITTI with 640×192 images.

Make3D [177]. This dataset features outdoor environments and is typically used for testing the generalization performance of monocular depth frameworks. We test MonoViT following the same image pre-processing steps and computing the evaluation metrics detailed in [64].

DrivingStereo [255]. It is a large-scale stereo dataset depicting autonomous driving scenarios. Among several sequences, we use the four image splits made available on the website, each made of 500 frames collected under different weather conditions, respectively *foggy*, *cloudy*, *rainy* and *sunny*. We use this dataset to evaluate the generalization capacity of MonoViT and its most recent competitors.

3.3.3 Depth Evaluation

Results on KITTI: We test our model by using the standard KITTI Eigen split [48], which includes 697 images coupled with raw LiDAR scans. Among them, improved ground truth labels [218] are provided for 652 images. Since monocular depth models trained on video sequences suffer from monocular scale ambiguity, the estimated depth is scaled by the per-image median ground truth [293].

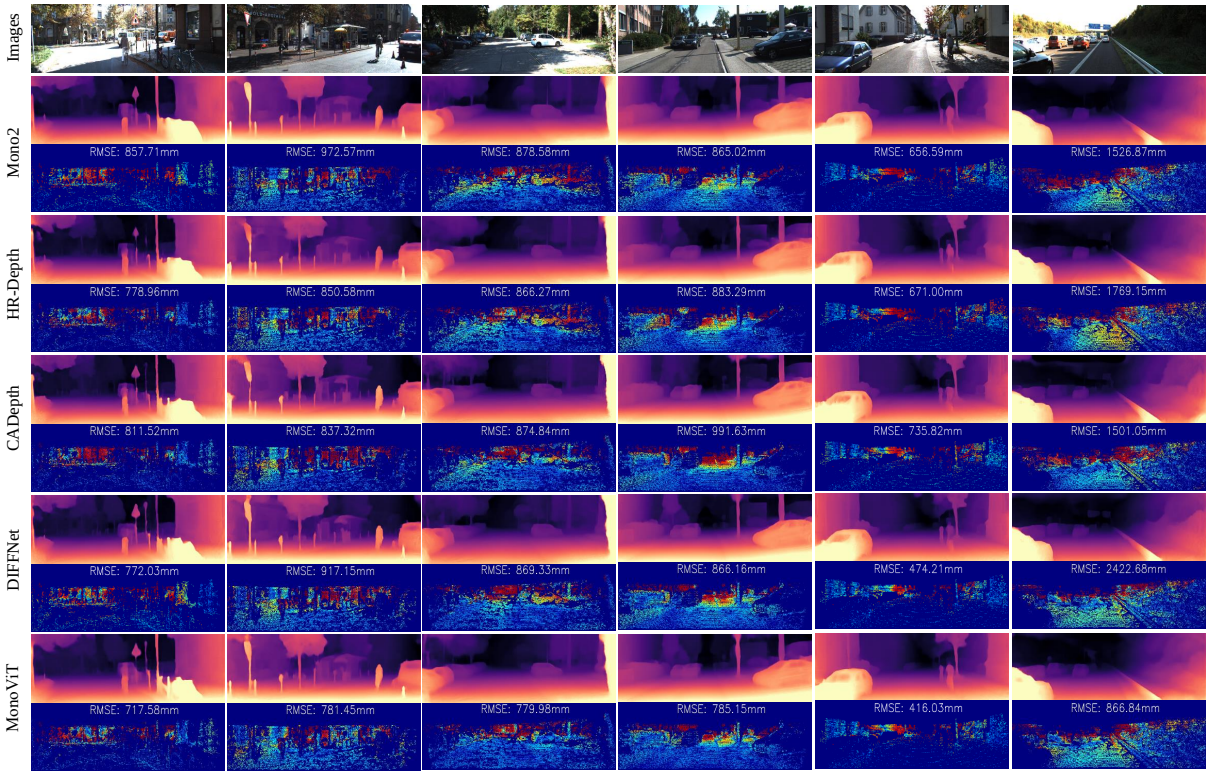


Figure 3.5: Qualitative results on KITTI. Top row, input images. Then, predictions by SoTA methods (Mono2 [64], HR-Depth [129], CADepth [251], DIFFNet [292]) and MonoViT (Ours). For each method, we show the depth map and the corresponding error map.

Tab. 3.1 collects the results achieved by SoTA self-supervised frameworks, processing either low resolution (left) or high resolution (right) images. We report results for methods trained both using monocular ('M', top), and binocular videos ('MS', bottom) for completeness. MonoViT significantly outperforms existing SoTA methods for any training resolution and setting on all metrics. In particular, we also highlight how MonoViT greatly outperforms MonoFormer [6], a concurrent attempt to deploy Transformers in self-supervised monocular depth estimation. Tab. 3.4 reports the same metrics computed over the improved ground truth labels processing 640×192 images. Again, MonoViT is constantly more accurate.

Fig. 3.5 reports a comparison between MonoViT and some of its competitors, showing that our model can get a much lower RMSE and proving that MonoViT is more powerful at modeling relations between objects than existing models.

Results on Make3D. We run experiments on the Make3D dataset [177] in order to evaluate the capability of our model to generalize on different real-world environments. By following the same protocol indicated in [64, 129, 251, 292], we firstly train our model on KITTI using images at 640×192 resolution and, then test on Make3D without a fine-tuning procedure. For fairness, we evaluate MonoViT and the existing self-supervised networks using the same evaluation code provided by [64]. Tab. 3.2 demonstrates how our proposed architecture allows us to outperform other strategies by a large margin and to achieve SoTA generalization results.

Results on DrivingStereo. Additionally, to further evaluate the generalization capacity of

Domain	Method	lower is better				higher is better		
		Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
foggy	Monodepth2 [64]	0.125	1.514	7.927	0.195	0.849	0.950	0.980
	HR-Depth [129]	0.131	1.504	8.023	0.199	0.828	0.949	0.982
	CADepth [251]	0.126	1.375	7.585	0.187	0.845	0.956	0.986
	DIFFNet [292]	0.111	1.232	7.047	0.169	0.869	0.966	0.989
	MonoViT (ours)	**0.096**	**0.934**	**6.313**	**0.150**	**0.893**	**0.974**	**0.993**
cloudy	Monodepth2 [64]	0.155	1.900	6.976	0.209	0.813	0.943	0.979
	HR-Depth [129]	0.149	1.656	6.658	0.204	0.815	0.945	0.981
	CADepth [251]	0.147	1.811	6.785	0.201	0.832	0.948	0.981
	DIFFNet [292]	0.140	1.571	6.298	0.192	0.837	0.950	0.983
	MonoViT (ours)	**0.125**	**1.300**	**5.970**	**0.177**	**0.861**	**0.958**	**0.986**
rainy	Monodepth2 [64]	0.240	3.339	11.040	0.301	0.591	0.857	0.952
	HR-Depth [129]	0.222	2.962	10.494	0.281	0.631	0.868	0.959
	CADepth [251]	0.221	3.072	10.681	0.277	0.632	0.879	0.963
	DIFFNet [292]	0.191	2.411	9.626	0.244	0.679	0.914	0.978
	MonoViT (ours)	**0.169**	**1.925**	**8.604**	**0.219**	**0.733**	**0.934**	**0.985**
sunny	Monodepth2 [64]	0.155	1.740	6.744	0.214	0.819	0.941	0.977
	HR-Depth [129]	0.153	1.546	6.505	0.212	0.812	0.942	0.978
	CADepth [251]	0.145	1.518	6.485	0.202	0.827	0.949	0.982
	DIFFNet [292]	0.142	1.457	6.165	0.197	0.835	0.950	0.982
	MonoViT (ours)	**0.130**	**1.266**	**6.109**	**0.186**	**0.851**	**0.956**	**0.985**

Table 3.3: Results on DrivingStereo Dataset [255]. Models trained on KITTI with 640×192 images and tested on four different complex scenarios (*foggy*, *cloudy*, *rainy* and *sunny*).

MonoViT, we also test it under four different weather conditions, including *foggy*, *cloudy*, *rainy* and *sunny*, from the DrivingStereo [255] dataset. In Tab. 3.3, we collect the output of this evaluation for MonoViT and SoTA frameworks. Any model has been trained on KITTI and tested on DrivingStereo without any re-training or fine-tuning. Once again, MonoViT performance always results vastly superior to any CNN competitor. In this case, the margin is even higher than what was observed for KITTI and Make3D. This fact further suggests that the ViT encoder used within our framework dramatically affects the generalization capacity of the whole depth network, thanks to the long-range relationships among features modeled by the Transformer blocks themselves.

Qualitative results. Fig. 3.6 reports some qualitative examples from the Make3D dataset, with MonoViT being able to model the structures of objects more accurately than its competitors. Fig. 3.7 shows a further qualitative comparison between MonoViT and SoTA frameworks on some challenging images from KITTI (top) and some even more challenging samples from DrivingStereo (bottom). For both datasets, we notice that MonoViT can effectively model the foreground and background because of the global receptive field, resulting in more precise, finer-grained estimation.

3.3.4 Ablation study

Finally, to further validate the effectiveness of our depth architecture, we report an ablation study in Tab. 3.5. comparing the results yielded by MPViT variants (tiny, xsmall, small, base)

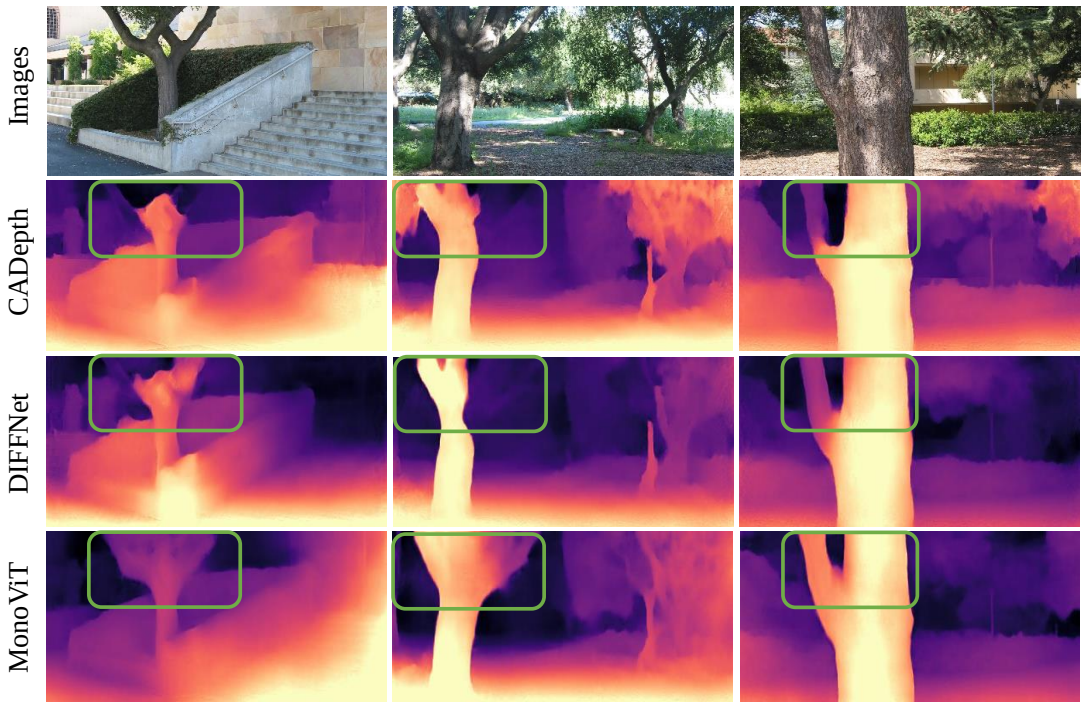


Figure 3.6: Qualitative comparison on the Make3D dataset [177]. Predictions by CADepth [251], DIFFNet [292] and our MonoViT.

Method	Data	lower is better				higher is better		
		Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Monodepth2 [64]	M	0.090	0.545	3.942	0.137	0.914	0.983	0.995
Johnston [88]	M	0.081	0.484	3.716	0.126	0.927	0.985	0.996
HR-Depth [129]	M	0.085	0.471	3.769	0.130	0.919	0.985	0.996
CADepth [251]	M	0.080	0.450	3.649	0.124	0.927	0.986	0.996
DIFFNet [292]	M	0.076	0.412	3.494	0.119	0.935	0.988	0.996
MonoViT (ours)	M	**0.075**	**0.389**	**3.419**	**0.115**	**0.938**	**0.989**	**0.997**
PackNet‡ [67]	M	0.071	0.359	3.153	0.109	0.944	0.990	0.997
MonoViT (ours)	M	**0.067**	**0.328**	**3.108**	**0.104**	**0.950**	**0.992**	**0.998**

Table 3.4: Results on KITTI using the improved ground truth [218]. Top: 640×192 input resolution, bottom: 1280×384. ‡ pretrained on Cityscapes [34] (on ImageNet [41] otherwise).

and different recent Transformer encoders, SwinT-tiny [125] and PVT [231] (which counts a similar number of parameters compared to MPViT-small) on top, also by reporting the amount of parameters and FPS for each. Benefiting from the combination of CNNs and Transformers, the MPViT backbone outperforms the other two SoTA pure Transformer backbones (SwinT [125], PVT [231]) and the pure CNN one (ResNet34 [75]) in the self-supervised monocular depth estimation task. Besides, we assess the impact of the different modules on bottom, like the Atten. Block in the decoder and the CNN path/Transformer path in the "Joint CNN &

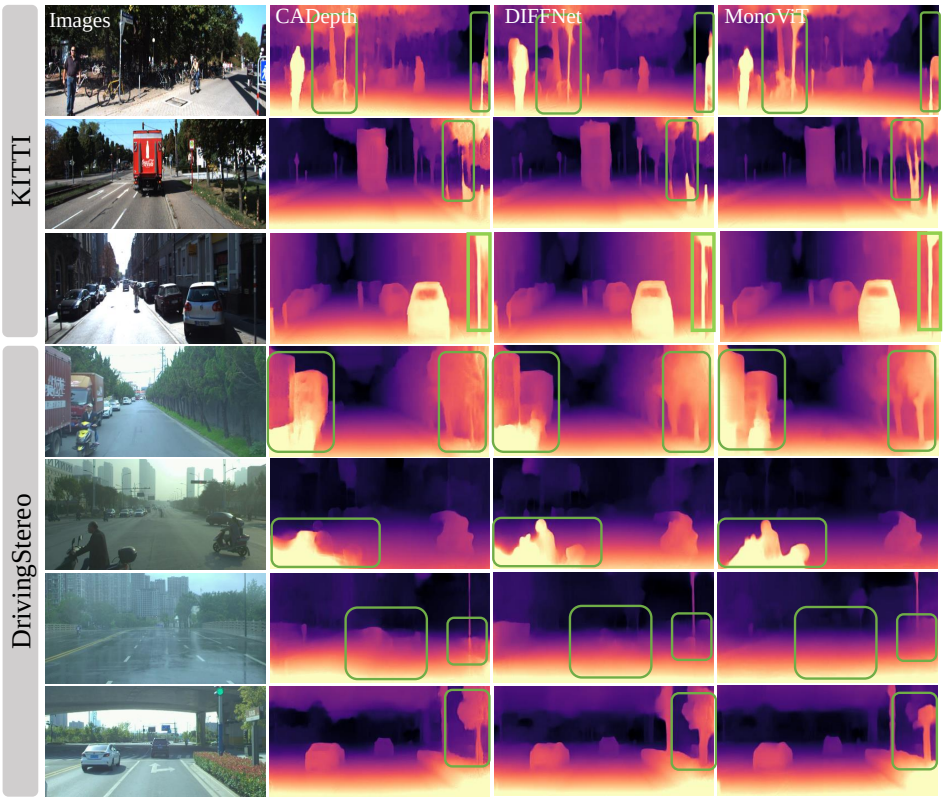


Figure 3.7: Qualitative comparison on KITTI [62] (top) and DrivingStereo [255] (bottom). Predictions by CADepth [251], DIFFNet [292] and our MonoViT.

Transformer Layer" (Fig. 3.4). As shown in the table, both CNN path, Transformer path and Atten. Blocks play an important role in the architecture.

Backbone	Params↓	FPS↑	lower is better				higher is better		
			Abs Rel↓	Sq Rel↓	RMSE↓	RMSE log ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
ResNet34 [75]	27M	**42**	0.108	0.780	4.622	0.183	0.884	0.963	0.983
SwinT-tiny [125]	34M	41	0.101	**0.698**	4.404	0.177	0.894	0.966	**0.984**
PVT-small[231]	30M	38	0.106	0.801	4.648	0.184	0.887	0.961	0.982
MPViT-tiny	**10M**	24	0.102	0.733	4.459	0.177	0.895	0.965	**0.984**
MPViT-xsmall	13M	24	0.101	0.738	4.402	**0.175**	0.898	**0.967**	**0.984**
MPViT-small	27M	18	**0.099**	0.708	**4.372**	**0.175**	0.900	**0.967**	**0.984**
MPViT-base	78M	15	0.100	0.747	4.427	0.176	**0.901**	0.966	**0.984**
MPViT-small	27M	18	**0.099**	**0.708**	**4.372**	**0.175**	**0.900**	**0.967**	**0.984**
w/o CNN Path	-	-	0.114	0.929	4.821	0.190	0.879	0.959	0.981
2 Trans. Path	-	-	0.107	0.801	4.590	0.182	0.889	0.963	0.983
1 Trans. Path	-	-	0.120	0.876	4.799	0.196	0.864	0.956	0.981
w/o Trans. Path	-	-	0.127	0.931	4.972	0.203	0.850	0.951	0.980
w/o Atten. Block	-	-	0.101	0.772	4.465	0.177	0.898	0.965	0.983

Table 3.5: Ablation study on KITTI. Trans. refers to Transformer. Input is 640×192, runtime measured on RTX 3090 GPU. Encoders are pre-trained on ImageNet.

3.4 Conclusion

We proposed MonoViT, a new architecture for self-supervised monocular depth estimation. By combining both convolutions and Transformers block inside the network encoder, MonoViT can model the local and global context of images jointly, overcoming existing solutions based on CNNs. Our proposal vastly and consistently outperforms the SoTA on the KITTI dataset. Moreover, experiments on Make3D and DrivingStereo datasets show that MonoViT achieves better generalization performance than SoTA architectures for self-supervised depth estimation.

Chapter 4

Efficient Spatial-Temporal Stereo Matching Network

The content of this chapter has been presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2023) - "TemporalStereo: Efficient Spatial-Temporal Stereo Matching Network" [278].

4.1 Introduction

Stereo reconstruction is a fundamental problem in computer vision. It aims at recovering the 3D geometry of the sensed scene by computing the disparity between corresponding pixels in a rectified pair of images. In robotic [182], virtual/augmented reality [268], autonomous driving [189], and more, depth information is crucial for scene understanding, with accurate yet fast solutions being particularly attractive.

Recently, deep learning approaches have shown tremendous improvements, especially in the supervised setting [29, 105, 245]. Common to most architectures is the cost volume, which encodes the feature similarity and plays a key role in matching pixels. In particular, features from the left image are concatenated (in 3D networks) or correlated (in 2D networks) and processed through convolutions, which are costly and prevent real-time execution. To address this issue, more efficient stereo models have been proposed, taking advantage of efficient cost aggregation [8, 246], coarse-to-fine [91, 233] and cost volume pruning [46, 133, 185] strategies to preserve computation and accuracy. Nonetheless, the sparse and constant disparity candidates used for matching cost computation can easily miss the ground truth disparity, thus deteriorating the accuracy of prediction. The case becomes even worse when dealing with challenging regions. For example, although occlusion is an ill-posed problem for stereo matching based on a single stereo pair, popular supervised stereo methods *have not ever* taken advantage of the intrinsic correlation among frames when stereo videos are available. Even if commercial cameras can easily acquire high FPS videos (*e.g.*, 30 or more), in which likely past context is strictly related to the current one, state-of-the-art stereo methods generally process each frame independently. We argue that previous disparities can be relevant cues to improve estimates, especially in complex regions where the current pair alone is not sufficiently meaningful, such as near object boundaries and occlusions.

Inspired by these observations, we propose *TemporalStereo*, a novel lightweight deep stereo

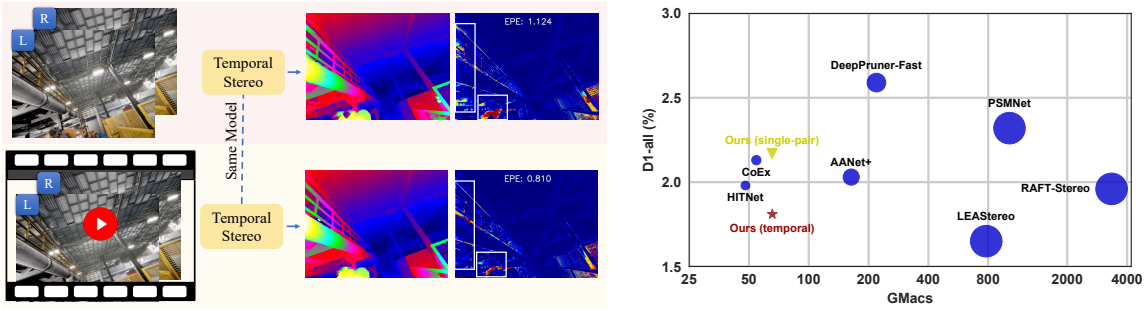


Figure 4.1: Overview of TemporalStereo. Our model can seamlessly process a stereo pair, as well as switch to temporal mode when multiple pairs are available, to increase accuracy. This allows TemporalStereo to achieve a favorable accuracy/speed trade-off with respect to existing networks. GMacs are measured with a resolution of 960×572.

model (as depicted in Fig. 4.1) being not only suited for single pair fast inference, but also able to leverage the rich spatial-temporal cues provided by stereo videos, when available. In *single-pair* mode (*i.e.*, when only the current pair is available), our network follows a coarse-to-fine scheme [66, 91, 233] to enable efficient inference. In order to increase the capacity of the constructed sparse cost volume and also the quality of sparse disparity candidates at every single stage, we enhance the cascade-based architecture in several aspects: 1) firstly, we enrich the cost of each queried disparity candidate with context from non-local neighborhoods during cost computation; 2) as the number of disparity candidates could be decreased to very few (*e.g.*, 5), in addition to exploiting pure 3D convolutions for cost aggregation, we also perform multiple statistics (*e.g.*, mean and maximum statistics) over a large window to grasp the global cost distribution of each pixel at one stage; 3) differently from previous strategies [91, 133, 185, 233], for which the candidates are constant throughout each stage, we rely on the aggregated cost volume itself to *shift* the candidates towards a better location and thus improve the quality of predicted disparity map; 4) additionally, when multiple pairs are available, our network can easily switch to *temporal* mode, in which past disparities, costs and cached features could also be aligned to the current reference frame and used to boost current estimates with a negligible runtime increase (4ms as shown in Tab. 4.7). Experimental results on synthetic (SceneFlow, TartanAir) and real (KITTI 2012, KITTI 2015) benchmarks support the following claims:

- Our method is accurate yet fast and achieves state-of-the-art results when a single stereo pair is available. In particular, the improved coarse-to-fine design based on very few (*e.g.*, 5) candidates allows TemporalStereo to unlock fast execution and high performance.
- When multiple temporally adjacent stereo pairs are available, TemporalStereo is the first supervised stereo network that can effortlessly cache the past context and use it to improve ongoing predictions, *e.g.*, in occluded and reflective regions. The model trained in temporal mode is effective also in single-pair mode, allowing the deployment of the same model in both settings.
- The proposed temporal cues can be widely applied to boost the matching accuracy of current efficient stereo methods, *e.g.*, on TartanAir [229] dataset, the improvements of

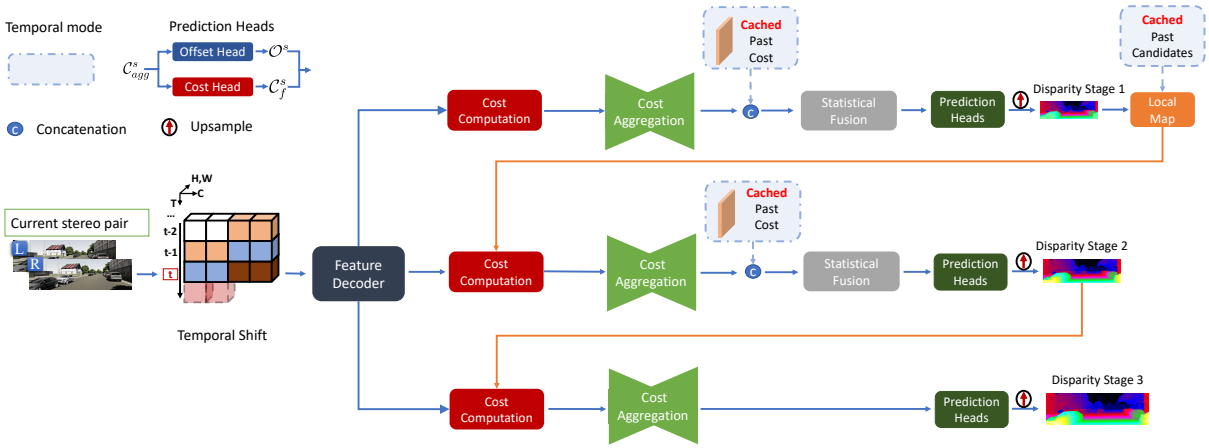


Figure 4.2: TemporalStereo Architecture. In single-pair mode, the model predicts the disparity map in a coarse-to-fine manner. If past pairs are available, the same model switches to temporal mode and employs features, costs, and candidates cached from past pairs to improve the current prediction.

CoEX [8] and StereoNet [91] are 14.6% and 26.1% respectively. Nonetheless, our TemporalStereo is still the best one for utilizing temporal information.

4.2 Method

We first describe TemporalStereo in single-pair mode, then we fully unlock the potential of our architecture enabling temporal mode. Fig. 4.2 depicts our model.

4.2.1 Single Pair Mode

Given a single stereo pair, a backbone extracts multi-scale features at $1/4$, $1/8$, and $1/16$ of the original resolution. Then, three stages predict disparity maps starting from these features. In particular, each stage performs feature decoding, cost volume computation, and aggregation.

Feature decoding In each stage $s \in \{1, 2, 3\}$, a decoder processes the current features, together with those from the previous stage F_l^{s-1}, F_r^{s-1} if $s > 1$. In particular, F_l^{s-1}, F_r^{s-1} are bilinearly upsampled by a factor 2 and concatenated with left and right features from the backbone. Then, the resulting feature maps are processed by two 2D convolutions with kernels of 3, obtaining F_l^s, F_r^s.

Cost volume computation. A 4D cost volume $\mathcal{C}^s \in \mathbb{R}^{Ch^s \times H^s \times W^s \times |\mathcal{D}^s|}$ is constructed by concatenating F_l^s with the corresponding F_r^s for each disparity d in the set $\mathcal{D}^s = \{d_1, d_2, ..., d_n\}$, with Ch^s respectively the number of channels of feature F_l^s and disparity candidates used in the stage:

$$\mathcal{C}^s_{concat}(\cdot, u, v, d) = \oplus\{F_l^s(\cdot, u, v), F_r^s(\cdot, u - d, v)\}, \tag{4.1}$$

where u, v are horizontal and vertical pixel coordinates, respectively, while $\oplus$ stands for concatenation on channel dimension. However, by doing so, $\mathcal{C}^s$ only contains local information, which is a major limitation in the case of a sparse set of candidates. To alleviate it, we enrich

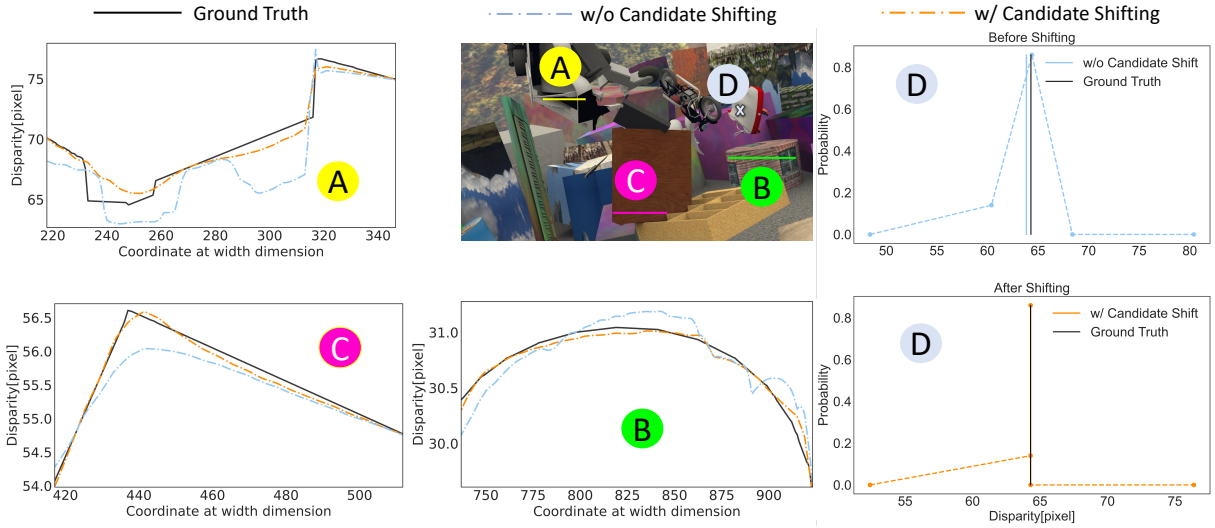


Figure 4.3: Adaptive Shifting in action. We show the ground truth disparities and the predictions with and without adaptive shifting for three horizontal regions (*i.e.*, A, B, C) and one pixel (D) in the image. For A, B, C, our strategy better estimates the actual disparity distributions. For D, the matching distribution of 5 disparity candidates switches from a flattened unimodal to one-hot distribution centered at ground truth by adaptively shifting candidates.

$\mathcal{C}^s$ with multi-level costs [210] encoding a larger neighborhood. Specifically, we build a three-level cost volume from feature maps F_l^s, F_r^s and by downsampling them with factors $2, 4$. At each level, we perform group-wise correlations [71], then costs are bilinearly upsampled to $\mathcal{C}^s$ resolution and stacked together, obtaining $\mathcal{C}_{gwc}^s$. The final multi-level cost is:

$$\mathcal{C}^s(\cdot, u, v, d) = \oplus\{\mathcal{C}_{concat}^s(\cdot, u, v, d), \mathcal{C}_{gwc}^s(\cdot, u, v, d)\}. \tag{4.2}$$

Cost aggregation. We now describe how we compute the aggregated cost volume $\mathcal{C}_{agg}^s$. In each stage, preliminary spatial cost aggregation is performed by one 3D convolution with kernel size $3 \times 3 \times 3$ followed by an hourglass network and another 3D convolution with kernel size $3 \times 3 \times 3$. Afterward, a Statistical Fusion module further improves the aggregation with mean and maximum statistics [271], especially along the disparity dimension (which contains as few as 5 candidates): 4 parallel layers (identity, convolutional with kernel size $5 \times 1 \times 1$, average pooling and max pooling both with kernel size $5 \times 5 \times 5$, respectively) extract global statistics from the preliminary aggregated volume, then their outcomes are stacked together and processed by one 3D convolution. In temporal mode, Statistical Fusion also helps to merge past costs with current ones, as we will discuss later. We employ Statistical Fusion only in stages $s = 1, 2$ to lighten the computation, while in $s = 3$ past costs are not used. Finally, a cost prediction head with two 3D convolutions with kernel size $3 \times 3 \times 3$ is in charge of predicting the final cost volume $\mathbf{C}_f^s$ from $\mathcal{C}_{agg}^s$. From $\mathcal{C}_f^s$ and $\mathcal{D}^s$ we could obtain the disparity $\hat{d}^s$ of the current stage by means of soft-argmin operator [89], but $\mathcal{C}_{agg}^s$ can help us to improve the candidates – as we will describe thereafter. For this reason, before computing $\hat{d}^s$, we have to introduce our adaptive candidate shifting strategy.

Adaptive candidate shifting. Previous works [91, 133] consider the candidates $\mathcal{D}^s$ as constants throughout the current stage s. However, the aggregated cost volume itself contains cues

about the candidates used to build it. Since it has been constructed from $\mathcal{D}^s$, it represents how well the candidates can explain the current scene. Moreover, in the cost aggregation step, each cost has been enriched with all the costs in its neighborhood, collecting matching results of nearby pixels. We propose to leverage a dedicated prediction head to infer, from $\mathcal{C}^s_{agg}$ as shown in Fig. 4.2, an offset value for each candidate, which represents how much we have to *shift* the candidate towards a better location. Thus, the candidates are pixel-wisely different and able to adaptively adjust according to the learned context. This strategy gives the network the ability to adapt $\mathcal{D}^s$ and improve $\hat{d}^s$, as depicted in Fig. 4.3. Specifically, an independent head – with the same structure of the cost prediction head, *i.e.*, two 3D convolutions, – is in charge of inferring the offset volume $\mathcal{O}^s \in \mathbb{R}^{H^s \times W^s \times |\mathcal{D}^s|}$ from $\mathcal{C}^s_{agg}$ as input. Although offset learning [123] is not new and has been used in [58] to convert the dense disparity candidates, which always tightly cover the ground truth disparity, into continuous ones so that the predicted disparity value will not be restricted to integers, in contrast, our disparity candidates are few and sparse, which could easily miss the ground truth. By enriching the context of each sparse candidate, the proposed shifting strategy is able to *correct* the constant and flawed candidates for better disparity estimation.

Disparity prediction and candidate sampling. Given $\mathcal{D}^s$, $\mathcal{O}^s$ and $\mathcal{C}^s_f$, soft-argmin operator [89] could be applied to predict the final disparity map. However, this strategy results sub-optimal in the case of multi-modal distributions [58, 216, 276]. To overcome this limitation, we predict the disparity following the strategy proposed in [8]: we select top-K values of $-\mathcal{C}^s_f$ for each pixel across the disparity dimension, and we normalize them with the softmax operator σ. The predicted disparity map $\hat{d}^s$ results:

$$\hat{d}^s = \sum_{d \in \{d_1^{top}, \cdots, d_K^{top}\}} (d + o_d) \times \sigma(-c_d), \tag{4.3}$$

where $d_k^{top} = argmax_d^k(-c_d)$, $argmax^k(\cdot)$ is the k-th maximal value for $k \in \{1, 2, ..., K\}$, c_d the cost in $\mathcal{C}^s_f$ of candidate d for each pixel, and o_d is the offset value in $\mathcal{O}^s$ for the candidate d. Convex upsampling [210] is used to double the resolution of $\hat{d}^1$ and $\hat{d}^2$, obtaining $\hat{d}^1_\uparrow$, $\hat{d}^2_\uparrow$. To bring $\hat{d}^3$ to full resolution, following [8], we bilinearly upsample $\hat{d}^3$ to full resolution and for each pixel in the upsampled resolution, a weighted average of a 3×3 superpixel surrounding it is calculated to get the final prediction $\hat{d}^3_\uparrow$. Finally, we obtain the set of candidates $\mathcal{D}^{s+1}$ for the next stage applying an inverse transform sampling. We sample according to a normal distribution $\mathcal{N}(\hat{d}^s_\uparrow, 1)$ in the range $[\hat{d}^s_\uparrow - \beta, \hat{d}^s_\uparrow + \beta]$, with β a constant value. This strategy allows to include more candidates near the predicted disparity. The first stage is initialized with $\mathcal{D}^1$ uniformly sampled across the full range to provide an overview of current scene geometry.

Loss function. Following [19], we minimise the smooth L_1 loss, *i.e.*, Huber loss at full resolution between the ground truth disparity map d^{gt} and the predictions at the three stages $\{\hat{d}^1_\uparrow, \hat{d}^2_\uparrow, \hat{d}^3, \hat{d}^3_\uparrow\}$. $\hat{d}^1_\uparrow$, $\hat{d}^2_\uparrow$ and $\hat{d}^3$ are bilinearly upsampled to full resolution and a factor λ^s weights each predicted map. The loss results:

$$\begin{aligned} \mathcal{L}_{huber} = & \frac{1}{|\mathcal{N}|} \lambda^0 \cdot smooth_{L_1}(d^{gt} - \hat{d}^3) \\ & + \sum_{s=1}^{3} \frac{1}{|\mathcal{N}|} \lambda^s \cdot smooth_{L_1}(d^{gt} - \hat{d}^s_\uparrow), \end{aligned} \tag{4.4}$$

where $\mathcal{N}$ is the number of pixels with valid ground truth.

Furthermore, to supervise the learning of offsets in three stages $\{O^1, O^2, O^3\}$, we adopt the Warsserstein loss [58] for training. Specifically, for each stage s, we downsample the ground truth disparity map d^{gt} to the resolution of corresponding offset for supervision, obtaining $d_{\downarrow}^{gt,s}$. As the offset value can range in a quite large value space, we improve the Warsserstein loss as:

$$\mathcal{L}_{war} = \sum_{s=1}^{3} \frac{1}{|\mathcal{N}|} \lambda^s \cdot \sum_{d \in \mathcal{D}^s} \left(|d + o_d - d_{\downarrow}^{gt,s}| \times (\sigma(-c_d) + \alpha) \right) \tag{4.5}$$

where c_d the cost value in $\mathcal{C}_f^s$ of candidate d for each pixel, and o_d is the offset value in $\mathcal{O}^s$ for the candidate d. We set $\alpha = 0.25$, *i.e.*, even for the candidate with extremely low matching probability $\sigma(-c_d)$, the network still learns an offset to enforce it moving towards the ground truth. With a weight factor λ_{final}, our final loss becomes:

$$\mathcal{L} = \mathcal{L}_{huber} + \lambda_{final} \cdot \mathcal{L}_{war}. \tag{4.6}$$

4.2.2 Temporal Mode

So far, we have presented the model suited for single-pair mode. Now, we illustrate how the model works in temporal mode. Differently from multi-view stereo models [82, 209], our network processes stereo videos one stereo pair at a time. This behaviour, which helps to save computation, is possible thanks to the sparse cost volume. In fact, we can easily add disparity candidates from the past inferences to the current set $\mathcal{D}^s$, thus increasing the search range with other plausible solutions. The past candidates, however, have to be aligned with the current frame to be meaningful. Optical flow/3D scene flow could be used to tackle the issue, but predicting flow fields is expensive in terms of memory footprint and time. Instead, in this work, we suppose that the camera is calibrated and the pose is given – since pose can be provided by external IMU sensors or estimated from the stereo pairs– to build the rigid motion field needed to align the candidates. Nonetheless, rigid flow formulation only holds for static objects in the scene, but since our network always relies on the current stereo pair, TemporalStereo is robust even in case of wrong flows (*e.g.*, due to bad poses) or moving objects. In the remainder, we detail how we do use temporal information to improve stereo-matching results.

Local map. As highlighted before, occlusions represent a major issue in stereo matching. However, we argue that currently occluded regions might have been visible in past frames, *e.g.*, due to camera motion. For this reason, the issue can be alleviated by adequately exploiting past estimates and at a minimal cost since they can be easily cached. Furthermore, we cache costs according to a keyframe strategy to bind the complexity in the case of long video sequences and ensure enough motion parallax. Following [199], a new incoming frame is promoted to keyframe if its relative translation and rotation are greater than $\mathbf{t}_{max}$ and $\mathbf{R}_{max}$ respectively. Then, a memory bank collects disparity $\hat{d}_{\uparrow}^3$ of the last N_{key} keyframes. Since each cached map represents the scene geometry in the past, to use it in the current computation we need to update the disparity values and their coordinates in the current image. Inspired by SLAM literature [141], we leverage a Local Map strategy to this aim. In detail, given the camera model $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, a 3D point $P(X, Y, Z)$ can be projected to a 2D pixel $p(u, v)$:

$$\pi(P) = \left(f_x \frac{X}{Z} + c_x, f_y \frac{Y}{Z} + c_y \right), \tag{4.7}$$

where (f_x, f_y, c_x, c_y) are the known camera intrinsics. Similarly, a pixel p can be back-projected to a 3D point P:

$$\pi^{-1}(p, d) = \frac{b \cdot f_x}{d} \left(\frac{u - c_x}{f_x}, \frac{v - c_y}{f_y}, 1 \right)^\top, \tag{4.8}$$

where b is the baseline between the left and the right cameras and d the disparity. With the relative extrinsic $\mathbf{T}_{j \to t} \in \mathbb{SE}(3)$ from keyframe j to the current frame t, we can update every disparity value d_j of keyframe j as:

$$d_j^{proj} = \frac{b \cdot f_x}{Z_j}, \text{ with } P_j(X_j, Y_j, Z_j) = T_{j \to t}^{-1} \cdot \pi^{-1}(p, d_j), \tag{4.9}$$

where d_j^{proj} is the updated disparity value. At this point, we obtain the coordinates of d_j^{proj} in t through forward warping. To preserve end-to-end requirement, we use the differentiable Softmax Splatting $\overrightarrow{\sigma}$ [147]:

$$\hat{d}_j^{proj} = \overrightarrow{\sigma}(d_j^{proj}, \pi(P_j) - p) \tag{4.10}$$

Finally, the Local Map is defined as follows:

$$\mathcal{D}^2 = \oplus \left\{ \mathcal{D}^2, \hat{d}_1^{proj}, \hat{d}_2^{proj}, \ldots, \hat{d}_{N_{key}}^{proj} \right\} \tag{4.11}$$

It is worth noticing that, in single-pair mode, the Local Map only contains candidates from the current pair. Moreover, it boosts candidate selection only for $s = 2$, since this stage represents the best trade-off between accuracy and speed. In fact, in $s = 1$ candidates are looked over the full search range at the lowest resolution, while in $s = 3$ a larger cost volume involves a much more expensive computation.

Temporal shift and past costs. Past semantic and matching scores are crucial for temporal stereo processing. Although Local Map effectively proposes previous depth cues about the scene, it lacks in providing the context behind these guesses. To address the problem, we introduce Temporal Shift and Past Costs modules. Temporal Shift enriches current feature maps with those computed in the past. Specifically, we adopt the TSM [115] strategy to facilitate the feature exchange among neighboring frames because it does not introduce additional computation or parameters: past features are *shifted* along the time dimension and then merged with current ones as shown in Fig. 4.2. Doing so, TSM provides *spatio-temporal* capabilities to our backbone not originally designed for temporal modeling. In practice, every feature map from the backbone is cached and used by Temporal Shift in the next prediction. Notably, as TSM does not rely on pose, we can still perform stereo matching in temporal mode and benefit from past information when pose is not available as shown in Tab. 4.3.

Similarly, Past Costs module adds past matching scores to current cost volumes. Given $\mathcal{C}_f^3$ volume computed at time $t-1$, first we update the value of its candidates to t according to Eq. (4.9). Then, cost values and candidates are forward warped to t using Eq. (4.10). Finally, the warped cost volume is downsampled by a factor of 2 and 4 and concatenated with current $\mathcal{C}_{agg}^2$ and $\mathcal{C}_{agg}^1$ respectively, to be further aggregated by the Statistical Fusion module. To preserve computation, we only warp top-K candidates and their costs to t. In single-pair mode, both Temporal Shift and Past Costs are the identity function. This strategy allows us to run the model trained in temporal mode also in single-pair mode with a limited drop in accuracy, *e.g.*, at bootstrap.

4.3 Experiments

This section describes the experimental setups used to evaluate on popular datasets, including: SceneFlow [135], TartanAir [229], KITTI 2012 [61] and KITTI 2015 [136]. As standard practice in this field [270, 276], we compute the end-point-error (EPE) and the percentage of points with a disparity error > 3 pixels (3PE, > 5 for 5PE) as error metrics in non-occluded (NOC), occluded (OCC) and both (ALL) the regions. Moreover, following the literature [136], we measure the D1 error on KITTI instead of the 3PE in background (BG), foreground (FG), and both (ALL) areas. We do so by computing the percentage of points with error > 3 pixels and $> 5\%$ than the ground truth. Runtime reported in Tab 4.2, 4.3, 4.7 is measured in the corresponding image resolution on each dataset, on a single NVIDIA 3090 GPU.

4.3.1 Dataset

SceneFlow: It is a large synthetic dataset [135] including 35,454 training and 4,370 test images with a resolution of 540×960. We only use the Flyingthings part in "finalpass" format for training and testing. Specifically, we randomly crop the image of 512×960 and set the batch size as 4. Since it does not provide ground truth camera poses, we use this dataset for single-pair mode only. The model is trained for 40 epochs with the initial learning rate of 0.001 decaying by a factor of 0.1 at epochs 30.

TartanAir: It's a challenging synthetic dataset [229] with moving objects and various light and weather conditions. To adapt it for stereo matching, we manually split the dataset with hard motion (which has 6DoF motion, the max translation and rotation are 0.5 meters and 10° respectively.) for training and testing. Specifically, the scenes consisted the testing dataset are listed in Tab. 4.1. The other parts of the corresponding scene are used for training. Overall, we collect nearly 66K stereo pairs for training and 5K for testing. The model is trained for 40 epochs with the initial learning rate of 0.001, reduced to 0.0001 at epoch 30. As for the first 20 epochs, we train the model in the single-pair mode to let the network learn to perform stereo matching. Then, we set the model in the temporal mode for the left 20 epochs. For all experiments, we keep the image at full resolution (i.e., 480×640) with batch size 16, and use the same training schedule for all experiments on the TartanAir dataset. As the TartanAir dataset targets at visual SLAM and thus ground truth poses are provided. In temporal experiments, we leverage ground truth poses for warping.

KITTI Raw Sequences: We augment the KITTI [61, 136] datasets with KITTI Raw Sequences [49]. Specifically, the KITTI Raw Sequences are composed of several outdoor scenes captured with car-mounted cameras and depth sensors. Following [49], we get 61 stereo video sequences (containing 42K pairs) with pseudo labels for pretraining, and poses are calculated from the GPS/OXTS devices on KITTI. As for pseudo label generation, we leverage the prediction results from LEAStereo [29] which has been finetuned on KITTI 2015 [136] training dataset. We also perform left-right consistency [47] to filter out outliers in the generated pseudo labels. Given the model trained on SceneFlow or TartanAir dataset, we further pretrain the network for 10 epochs with an initial learning rate of 0.001 and decrease it to 0.0001 at epochs 8. Besides, we set the batch size as 4 and randomly crop the image into 320×1024.

KITTI 2012&2015: They are both real-world datasets collected by a driving car and depicting urban scenes. KITTI 2015 [136] contains 200 training and 200 testing stereo pairs while KITTI 2012 [61] has 194 and 195 pairs for training and test, respectively. Both the splits

Scene	Part
abandonedfactory	P002
amusement	P007
carwelding	P003
endofworld	P006
gascola	P001
hospital	P042
office	P006
office2	P004
oldtown	P006
seasonforest	P002
seasonforest_winter	P015
soulcity	P008

Table 4.1: TartanAir Testing Dataset. We list Scene-Part pairs used in our testing set.

provide sparse ground truth depth labels. For temporal mode evaluation, we leverage the multi-view split, which contains, for each sample, also the 10 previous pairs. We estimate camera poses with an off-the-shelf SLAM algorithm [16]. We randomly crop the image into resolution 320×1024, use batch size as 4, and train our models on KITTI 2012 [61] and KITTI 2015 [136] with the same schedule. More specifically, for both single-pair and temporal mode, we use the pre-trained model from KITTI Raw Sequences [49] and finetune it for 16 epochs with the initial learning rate of 0.0001 and reduce it by a factor of 0.1 at epochs 12.

4.3.2 Implementation Details

We implement our network in PyTorch [154], using RMSProp as optimizer to train all models in an end-to-end fashion. Scale weights are $\lambda_0 = 1.0$, $\lambda_1 = 0.5$, $\lambda_2 = 0.7$, $\lambda_3 = 2.0$, $\lambda_{final} = 2.0$, while we set $K = 2$ for top-K selection, number of candidates $n = 12$ in stage 1 while $n = 5$ in stages $s = 2, 3$, $\beta = 4$ for sampling and $N_{key} = 3$ in Local Map. As for keyframe selection strategy, we set $|\mathbf{t}_{max}| = 0.1$m and $|\mathbf{R}_{max}| = 15^\circ$ respectively. We leverage EfficientNetV2-S [202] as backbone feature extractor, while the hourglass network proposed in [71] for cost aggregation. For all 3D convolutions with kernel size $k \times k \times k$ where $k > 1$, we implement it in depthwise [175] manner which consists two convolutions with kernels of size $k \times 1 \times 1$ and $1 \times k \times k$ respectively to save computation. For all datasets, we perform asymmetric chromatic augmentation as described in [254] to mitigate the effect of varying lighting and exposure. Furthermore, we also add random patching [254] on the right image to help the network deal with occluded areas. The maximum disparity value is $D_{max} = 192$. The smooth L_1 loss, *i.e.*, Huber loss is defined as:

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1, \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{4.12}$$

4.3.3 Ablation Study

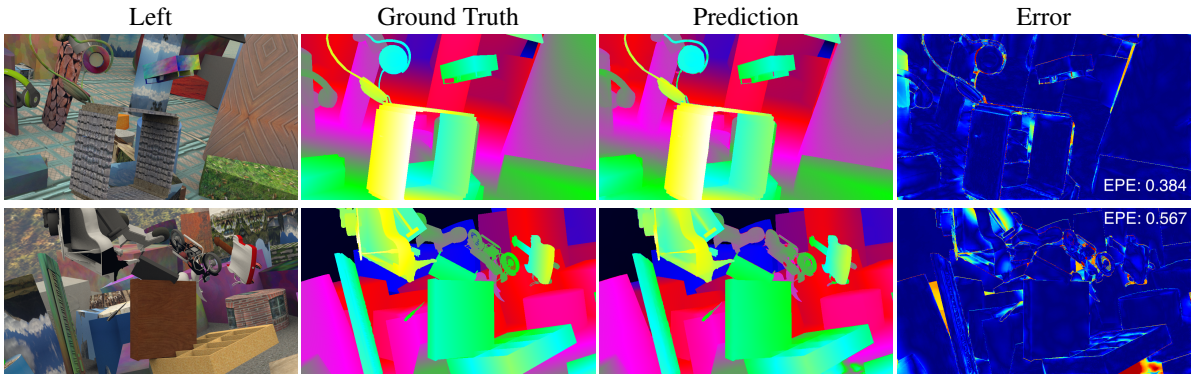


Figure 4.4: Qualitative results on SceneFlow. From left to right, the reference/left image, the ground truth disparity map, the prediction in single-pair mode and its error (darker the color, lower the error).

	Multi-Level Cost	Statistical Fusion	Adaptive Shifting	EPE ALL	EPE OCC	EPE NOC	3PE ALL	3PE OCC	3PE NOC	Runtime (ms)
(A)	✗	✗	✗	0.600	1.949	0.369	2.85	11.62	1.36	**40.62**
(B)	✓	✗	✗	0.587	1.924	0.360	2.79	11.44	1.33	43.41
(C)	✓	✓	✗	0.581	1.908	0.356	2.78	11.39	**1.31**	44.54
(D)	✗	✗	✓	0.564	1.923	0.334	2.87	11.78	1.36	41.58
(E)	✓	✓	✓	**0.532**	**1.830**	**0.315**	**2.75**	**11.37**	**1.31**	45.42

	Candidates Number	Disparity Range	EPE	Runtime (ms)
(F)	3	[-4, 4]	0.565	**41.61**
(G)	9	[-4, 4]	**0.531**	55.84
(H)	5	[-4, 4]	0.532	45.42
(I)	5	[-2, 2]	0.543	45.42
(J)	5	[-8, 8]	0.568	45.42

	Model Variants	Depth-wise 3DCNN	EPE	Runtime (ms)
(K)	Baseline	✗	0.589	63.91
(L)	Baseline	✓	0.600	**40.62**
(M)	Full	✗	0.535	75.42
(N)	Full	✓	**0.532**	45.42

Table 4.2: Single-pair mode ablation. We assess on SceneFlow the key components of the proposed architecture.

Single-pair mode. We leverage SceneFlow [135] to assess the impact of main components of TemporalStereo in single-pair mode. Tab. 4.2 reports this ablation study, witnessing how each module helps to improve the results consistently. In particular, the multi-level cost computation with group-wise convolutions (B) is effective in enriching the cost volume of each stage, and the baseline model (A) also benefits from the further aggregation of Statistical Fusion (C). Nonetheless, the adaptive shifting strategy (E) provides without any doubt the main boost in

performance. To be noticed, without the enriched context of each sparse candidate, the adaptive shifting strategy shows limitations on disparity correcting and its performance gain (D) decreases a lot. Furthermore, results shown in (F, G, H) illustrates our TemporalStereo is able to predict accurate disparity with as few as 5 candidates. With the ability to shift the candidates towards a better solution, our model can search in a quite large space (*e.g.*, $\beta = 2,\ 4,\ 8$ as reported in H, I, J) to get the best result when $\beta = 4$. 3D convolutions are the common operations to aggregate the cost in recent methods [8, 91, 276]. Our baseline model (A) benefits from the 3D convolutions (K) when compared to depth-wise [175] 3D convolutions (L), but the runtime increases a lot. In contrast, the full model (E) gives much higher improvement (N) with negligible runtime increase (4.8ms), and the time-consuming 3D convolutions are not necessary (M) for accuracy improvement. Besides, we can notice that our network achieves extremely low error in both examples as shown in Fig. 4.4.

	Temporal Shift	Local Map	Past Costs	EPE			3PE			Runtime (ms)
				ALL	OCC	NOC	ALL	OCC	NOC	
(A)	✗	✗	✗	0.647	1.899	0.420	3.96	14.21	2.08	**36.85**
(B)	✓	✗	✗	0.643	1.842	0.435	4.00	14.55	2.15	**36.85**
(C)	✓	✓	✗	0.624	1.799	**0.413**	3.81	13.60	**2.02**	38.22
(D)	✓	✓	✓	**0.610**	**1.637**	**0.413**	**3.73**	**12.60**	2.03	40.13

Table 4.3: Temporal mode ablation. We evaluate the temporal components on TartanAir, with $W_{tr} = W_{test} = 4$.

Temporal mode. Before presenting the results achieved in temporal mode, we illustrate the protocol adopted at training and test time on TartanAir dataset [229]. *Given a temporal window containing W frames*, initial $f = \{1, 2, ..., W-1\}$ frames are processed by the network sequentially without computing error metrics and blocking the gradients. Specifically, each outcome is cached and used in the next frame prediction. When $f = W$, we compute errors (and backpropagation at training time). Tab. 4.3 reports the ablation conducted on TartanAir [229], with $W = 4$ both at train and test time, aimed at evaluating the importance of each temporal module. Specifically, we train the model in the single-pair mode for 20 epochs (*i.e.*, the model learns how to solve stereo matching task), then we enable temporal components for 20 more epochs (*i.e.*, the model now focuses on how to use past information). We can notice how the baseline (A), *i.e.*, the model trained in single-pair mode for 40 epochs, could be improved using Temporal Shift to fuse past features with current ones (B). However, the benefit due to Temporal Shift is much lower than the gain provided by Local Map, which largely improves the performance (C). Finally, including cached past costs as well (D) provides an additional boost in accuracy.

Impact of temporal window. Tab. 4.4 reports the results achieved by TemporalStereo using different values of W. In particular, we could have two different values for W, that are W_{tr} and W_{test} for train and test respectively. In addition to the baseline $W_{tr} = 1$ and the temporal $W_{tr} = 4$ models, we also train a $W_{tr} = 8$ model following the same configuration as for $W_{tr} = 4$. When compared with existing models such as PSMNet [19] and CoEx [8], our baseline outperforms them by a large margin in EPE metric. As for temporal mode, in general, the more frames joining the training or testing phases, the better result we can get in all regions. Notably, $W_{test} = 4$ and $W_{test} = 8$ are always beneficial in OCC, witnessing that TemporalStereo can

effectively exploit more frames to deal with such difficult areas. Moreover, when $W_{test} > 1$, the results consistently overcome baseline with single stereo pair. It indicates our network can benefit from past information with only a few frames (*e.g.*, 4, 8) after the network startup. Finally, we can notice how temporal models tested with $W_{test} = 1$ obtain results closer to the baseline. This outcome implies that, in practical video applications, TemporalStereo can be trained once in temporal mode and used in single-pair way for the first inference (yet providing a good initial estimate) and in the temporal mode for all the others. Fig. 4.5 visualises the benefits of temporal model ($W_{tr} = 8$, $W_{test} = 8$) to alleviate occlusion errors. The temporal mode (2) largely outperforms the single-pair one (1). Furthermore, we highlight how TemporalStereo is also robust against inaccurate poses: although in (3) the camera pose is set to an identity matrix and only Temporal Shift module could help, it still outperforms (1). Eventually, the proposed temporal cues can be plugged into recent efficient methods (*e.g.*, CoEx [8] and StereoNet [91]) for further improvement. Although CoEX with a very different architecture compared to ours, its EPE still decreases from 0.714 to 0.610, with near 14% improvement. Nonetheless, our TemporalStereo utilizes past cues more effectively.

	Method	W_{test}	EPE			3PE		
			ALL	OCC	NOC	ALL	OCC	NOC
$W_{tr} = 1$	PSMNet [19]	1	0.866	2.654	0.558	4.80	18.63	2.48
	StereoNet [91]	1	0.888	2.647	0.578	5.15	19.34	2.68
	CoEx [8]	1	0.714	2.074	0.463	**3.83**	14.57	**1.93**
	Ours (single-pair)	1	**0.647**	**1.899**	**0.420**	3.96	**14.21**	2.08
$W_{tr} = 4$	Ours (single-pair)	1	0.665	1.731	0.459	3.95	13.34	2.16
	Ours (temporal)	4	0.610	1.637	0.413	3.73	12.60	2.03
	Ours (temporal)	8	**0.607**	**1.634**	**0.409**	**3.71**	**12.59**	**2.01**
$W_{tr} = 8$	Ours (single-pair)	1	0.673	1.912	0.450	4.00	14.03	2.17
	Ours (temporal)	4	0.609	1.625	0.412	3.74	12.58	2.03
	StereoNet [91] (temporal)	8	0.656	1.928	0.428	3.97	14.45	2.06
	CoEx [8] (temporal)	8	0.610	1.637	0.413	3.73	12.60	2.03
	Ours (temporal)	8	**0.601**	**1.615**	**0.405**	**3.71**	**12.54**	**2.02**

Table 4.4: Impact of different frames in temporal mode. Models are tested with $W_{test} = 1$, 4 and 8.

Pose Analysis. To assess the impact of pose input, we evaluate our model with setting $W_{tr} = 8$ and $W_{test} = 8$ on 4 scenes (2 indoor and 2 outdoor scenes) of TartanAir dataset [229] with hard motion patterns. The overall results of several pose inputs are listed in Tab 4.5. Specifically, the noise of rotation follows $\mathcal{N}(0, \sigma_R)$ in degree(°) and noise of translation follows $\mathcal{N}(0, \sigma_t)$ in meter(m). As reported, 1) poses from ground truth (GT) or estimated by DROID-SLAM [212] yield almost the same EPE and 3PE metrics. It means, when the ground truth pose is not available, an actual SLAM system like DROID-SLAM could be an alternative scheme. 2) Pose with small rotation error (*e.g.*, $\sigma_R <= 1°$) and translation error (*e.g.*, $\sigma_t <= 0.05m$) gets very close results to the one with ground truth pose, and it always surpasses the results by single-pair mode. Even when the pose error comes to the maximum level of the dataset (*i.e.*, $\sigma_R = 10°, \sigma_t = 0.5m$), our model does not crash down and provides impressive predictions,

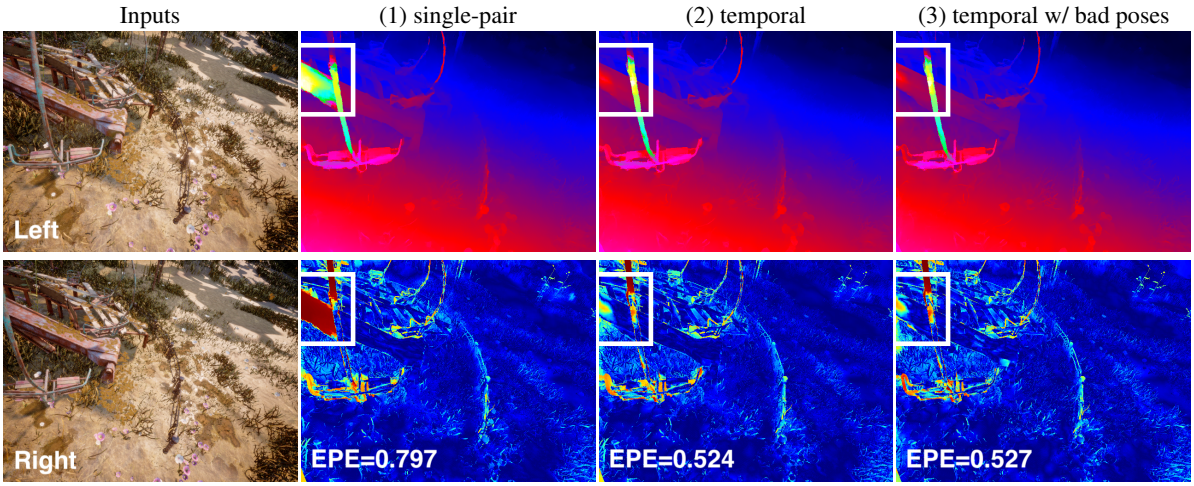


Figure 4.5: Benefits of temporal mode. Compared to single-pair mode (1), temporal mode (2) is more accurate at occlusions, even with noisy poses (3) – colder colors encode lower error.

Pose Type			Outdoor				Indoor			
			Amusement		SoulCity		Carwelding		Hospital	
			EPE	3PE	EPE	3PE	EPE	3PE	EPE	3PE
Single-pair			0.562	2.62	0.529	2.60	0.508	2.84	0.904	3.74
Identity			0.569	2.58	0.534	2.66	0.538	2.96	0.927	4.21
DROID-SLAM [212]			**0.545**	**2.51**	0.509	**2.52**	0.479	**2.63**	**0.848**	**3.30**
GT			**0.545**	**2.51**	**0.508**	**2.52**	**0.478**	**2.63**	**0.848**	**3.30**
	σ_R (°)	σ_t (m)								
GT+Noise	10	0.50	0.716	3.52	0.791	4.86	0.608	3.34	1.152	4.85
	10	0.05	0.713	3.46	0.776	4.55	0.611	3.37	1.096	4.79
	1	0.50	0.577	2.58	0.592	2.83	0.531	2.91	0.934	4.49
	1	0.05	**0.546**	**2.54**	0.518	2.64	0.506	**2.72**	0.880	3.84
	1	0.01	**0.546**	**2.54**	**0.517**	**2.62**	**0.505**	2.74	**0.873**	**3.66**

Table 4.5: Impact of different pose input in temporal mode. "Single-pair" denotes our model is in single-pair mode (no pose needed); "Identity" means identity matrix; "DROID-SLAM" means pose estimated by DROID-SLAM (ORB-SLAM3 fails on these 4 scenes); and "GT+Noise" is the ground truth pose with manually added Gaussian noise.

proving the robustness of our model to inaccurate pose. 3) For outdoor scenes, since the view change between frames is much smaller than indoor scenes, the accuracy drops negligibly even when identity transformation is applied.

4.3.4 Evaluations on KITTI Benchmarks

To conclude, we run TemporalStereo on KITTI 2012 and KITTI 2015 test data and submit to the online leaderboard.

Pretrain	SF		SF+Pseudo		TartanAir+Pseudo			
Method	CoEx	Ours	CoEx	Ours	CoEx	Ours	CoEx†	Ours†
D1-BG	1.79	2.17	1.73	1.89	1.74	1.89	1.71	**1.61**
D1-FG	3.82	2.96	3.60	2.85	3.49	3.03	**2.78**	**2.78**
D1-ALL	2.13	2.30	2.04	2.05	2.03	2.07	1.89	**1.81**

Table 4.6: Impact of pretraining. We study the importance of pretraining by evaluating the D1 metric on KITTI 2015 test dataset. Results by both CoEx and Ours in single-pair and temporal mode (†) are reported accordingly. 'SF' and 'TartanAir' denotes SceneFlow and TartanAir datasets respectively. '+Pseudo' means further training on KITTI raw sequences with our generated pseudo label.

Pretraining. As KITTI is very challenging due the lack of a big training set, pretraining on SceneFlow dataset [135] is a common training schedule for deep learning-based stereo methods [19, 29]. Recent methods [185, 254] also introduce extra data, *e.g.*, HR-VS [254], to augment KITTI during training. Following the knowledge distillation strategy proposed in AANet+ [246], we augment the KITTI dataset by leveraging the prediction results from pretrained LEAStereo [29] to generate pseudo labels on KITTI raw sequences [49]. As a result, we get 61 stereo video sequences (containing 42K pairs) with pseudo labels for pretraining, and poses are calculated from the GPS/OXTS data on KITTI. We study the influence of different pretraining strategies on the D1 metric on KITTI 2015 test dataset. As shown in Tab. 4.6, compared with CoEx [8], our model in single-pair mode performs better in D1-FG and worse in D1-ALL, D1-BG metrics when pretrained on SceneFlow dataset only. After further training on pseudo labels, we get the almost same result as CoEx, which demonstrates our model requires more data to achieve better performance. Replacing the SceneFlow dataset with TartanAir, both the result of CoEx and ours in single-pair mode do not improve and achieve almost the same accuracy, *i.e.*, D1-ALL 2.03% for CoEx and 2.07% for ours. However, by leveraging temporal information, the temporal mode can boost the accuracy of both TemporalStereo and CoEx further by a large margin, *i.e.*reducing D1-ALL to 1.81% and 1.89% respectively, supporting the major impact of our temporal paradigm over the pseudo labels training. We also point out that, despite the poses of KITTI raw sequences and KITTI 2015 are estimated by GPS/OXTS data and ORBSLAM3 [16] respectively, TemporalStereo demonstrates its robustness to the pose error again.

Tab. 4.7 collects results achieved by a variety of deep stereo models, both on Scene Flow and the KITTI online benchmarks. For KITTI, we report the error rates achieved by TemporalStereo, both in single-pair and temporal mode ($W_{tr}, W_{test} = 11$), finetuned from models pretrained on KITTI raw sequences [49]. In single-pair mode, our TemporalStereo achieves results already on par with state-of-the-art on all datasets. When switching to the temporal mode, our network surpasses all fast stereo architectures by a large margin. In particular, our result on D1-FG is even better than the one achieved by slow models [29, 270] running in hundreds of milliseconds. It is worth mentioning that TemporalStereo in temporal mode, benefiting from past semantic and geometric information, achieves better results compared to single-pair mode

	Method	SceneFlow [135]	KITTI 2012 [61]				KITTI 2015 [136]		
		EPE	Reflective		All		D1		
			3PE	5PE	3PE	5PE	BG	FG	ALL
slow	PSMNet [19]	1.09	10.18	5.64	1.89	1.15	1.86	4.62	2.32
	GwcNet-gc [71]	0.77	9.28	5.22	1.70	1.03	1.74	3.93	2.11
	GANet-Deep [270]	0.78	7.92	4.41	1.60	1.02	1.48	3.46	1.81
	AcfNet [276]	0.87	8.52	5.28	1.54	1.01	1.51	3.80	1.89
	LEAStereo [29]	0.78	**6.50**	**3.18**	1.45	0.88	1.40	**2.91**	**1.65**
	ACVNet [245]	**0.46**	7.03	4.14	**1.13**	**0.71**	**1.37**	3.07	**1.65**
	CFNet [185]	-	7.29	3.81	1.58	0.94	1.54	3.56	1.88
	DWARF‡ [1]	-	-	-	-	-	3.20	3.94	3.33
	DTF_SENSE‡ [184]	-	-	-	-	-	2.08	3.13	2.25
	SENSE‡ [86]	-	-	-	-	-	2.07	3.01	2.22
fast	StereoNet [91]	1.10	-	-	6.02	-	4.30	7.45	4.83
	DeepPruner-Fast [46]	0.97	-	-	-	-	2.32	3.91	2.59
	AANet+ [246]	0.72	9.10	5.12	2.04	1.30	1.65	3.96	2.03
	CoEx [8]	0.69	8.63	4.49	1.93	1.13	1.79	3.82	2.13
	HITNet [207]	**0.53**	7.54	4.01	1.89	1.29	1.74	3.20	1.98
	Ours (single-pair)	**0.53**	6.99	3.52	1.94	1.08	1.89	2.85	2.05
	Ours (temporal)	-	**6.14**	**3.08**	**1.61**	**0.88**	**1.61**	**2.78**	**1.81**

Table 4.7: Comparison with state-of-the-art methods – slow and fast. We report results of state-of-the-art methods on SceneFlow and KITTI. *fast* denotes models allowing real-time inference. ‡ means 3D scene flow-based methods.

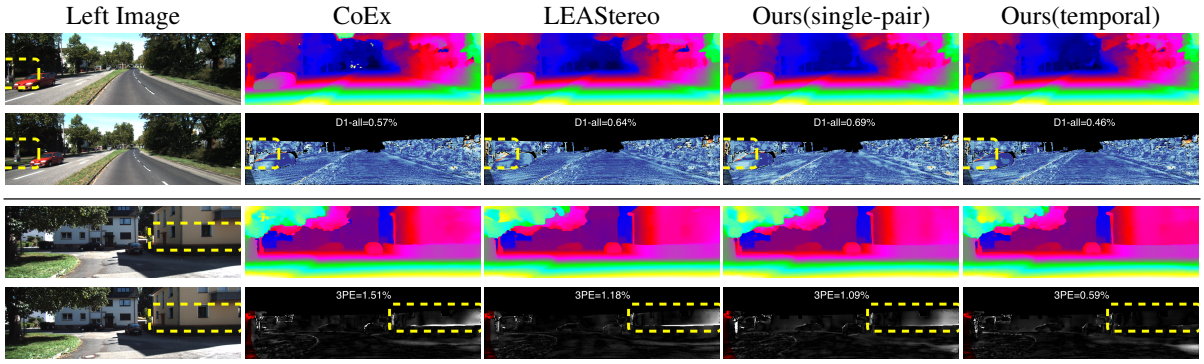


Figure 4.6: Results on KITTI 2015 and 2012 testing set. Compared to existing methods [8, 29], TemporalStereo exploits time to improve accuracy at occlusions (top) or texturless regions (bottom). For the error maps on second and forth row, colder and darker color means lower the error respectively.

on D1-FG metric (the D1 error on foreground areas, *i.e.*, moving cars), which proves the robustness of our model to **dynamic objects** as well. The same advantage is also evident on 3PE and 5PE in **reflective regions** on KITTI 2012. Finally, we also compare with **3D scene flow** based state-of-the-art methods [1, 86, 184], which project with dense 3D motion field and output disparity. In contrast, simply using camera pose, our TemporalStereo is the obvious winner

in both accuracy and efficiency on KITTI 2015. Fig. 4.6 show qualitative comparisons between existing networks [8, 29] and TemporalStereo, highlighting the benefits yielded by temporal mode.

4.4 Conclusions

We presented TemporalStereo, a novel network devoted to fast stereo matching. The enhanced coarse-to-fine design and sparse cost volumes allow for fast inference and high performance. Moreover, TemporalStereo can exploit past information to ameliorate predictions, especially in occluded regions. The same model, trained once, can handle either single or multiple stereo pairs effectively. Considering the requirement of camera poses as its main limitation, empowering our system with pose estimation will be our future research direction.

Chapter 5

Depth Super-Resolution from Explicit and Implicit High-Frequency Features

The content of this chapter has been presented at the Computer Vision and Image Understanding (CVIU) - "Depth Super-Resolution from Explicit and Implicit High-Frequency Features" [163].

5.1 Introduction

With the rise of consumer-grade depth cameras, depth maps are employed in various scenarios such as 3D reconstruction [21, 22], recognition [15] and more. Time-of-Flight (ToF) is one of the leading technologies involved in depth sensing, measuring the distance traveled by emitted rays until they reach points in the scenes. However, due to the limitations of physical fabrication, power consumption and costs [7], the resolution of depth maps usually is often insufficient to fulfill the demand of the downstream applications, such as object detection [26] and pose estimation [60]. In contrast, collecting RGB images at much higher resolution is cheaper. As a result, the guided depth super-resolution task, known as GDSR, has emerged as a crucial solution to this technological limitation, allowing to obtain an accurate high-resolution (HR) depth map from a low-resolution (LR) one, guided by an HR image.

Initially, algorithms addressing this problem were classified into local [98, 170, 232, 258] and global [43, 52, 108, 151], with the former family being faster, yet suffering in low-textured regions and the latter resulting more robust, at the expense of processing time. More recently, deep neural networks have become the preferred choice for depth super-resolution [84, 107, 109, 128, 205], although they still struggle to restore sharp and precise edges from LR depth maps reliably, especially when dealing with large upsampling factors. This is mainly due to the inadequate guidance provided by High-Frequency (HF) features, implicitly modeled by deep networks, which frequently cause texture copying effects in the upsampled depth maps. In addition, single-stage multi-scale architectures for this task [204, 237, 262, 301], at any given scale, cannot fully leverage fine details encoded at the higher ones, as they are lost due to downsampling and only partially recovered through skip connections.

In light of the two weaknesses highlighted so far, we aim to improve GDSR by explicitly countering them. For the former, we argue that explicit extraction of HF features, supported by edge detection algorithms such as the Canny operator, can play a crucial role [237]. Concerning the latter, multi-stage network design – which outperforms single-stage counterparts in

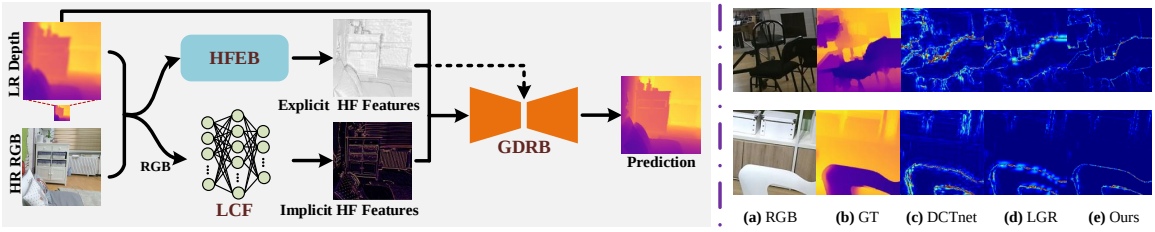


Figure 5.1: Depth Super-Resolution exploiting explicit and implicit high-frequency features. On the left, an overview of our framework, combining the power of both explicit and implicit high-frequency information extracted from the inputs. On the right, qualitative examples with (a) RGB images, (b) ground truth depth and error maps by existing methods (c – d) and ours (e).

high-level visual tasks like action segmentation [51] and pose estimation [23], as well as for low-level vision problems such as image restoration [93, 267] – can mitigate the information loss issue. However, since features extracted from RGB images need to be considered in addition to depth features, existing multi-stage networks are inadequate for GDSR and should be revised to fuse features from the two domains. Besides, only using explicit high-frequency information derived from hand-crafted models is insufficient to obtain meaningful HF features effectively. In addition, CNN models are usually more sensitive to low-frequency (LF) than HF information [247]. Hence, additional clues (e.g., features in the frequency domain) are needed for better super-solving LR depth maps.

We present a Depth Super-Resolution method leveraging both Explicit and Implicit HF information (DSR-EI), which contains two branches: the High-Frequency Extraction Branch (HFEB) and the Guided Depth Restoration Branch (GDRB). The former is designed to model **explicit** HF features – i.e., hand-crafted edges – by exploiting dynamic self-calibrated convolutions (DSP) and the power of vision transformers blocks. The latter effectively fuses the guidance from RGB features with depth features to obtain HR depth maps. This is achieved by deploying two novel modules: 1) the Adaptive Feature Fusion Module (AFFM), which counters the HF information loss due to downsampling, and 2) the Low-Cut Filtering (LCF) module, which acts in the frequency domain to improve **implicit** extraction of HF features. In contrast to explicit features, implicit features are image representations extracted by convolutional layers. Exhaustive experiments on several standard datasets show the superiority of DSR-EI. In summary, the main contributions are:

- The proposed architecture employs a novel efficient transformer for explicit, HF feature extraction. The transformer can accurately capture image details and structures from depth maps.
- In the guided depth restoration branch, we propose a low-cut filtering module that can obtain accurate, implicit HF information.
- To counter the information loss issue, we propose an Adaptive Feature Fusion Module located in the middle of the guided depth restoration branch.
- Quantitative and qualitative experimental results demonstrate that our approach establishes a new state-of-the-art in the field of guided depth super-resolution.

Fig. 5.1 provides a high-level view of our framework, followed by examples that anticipate the

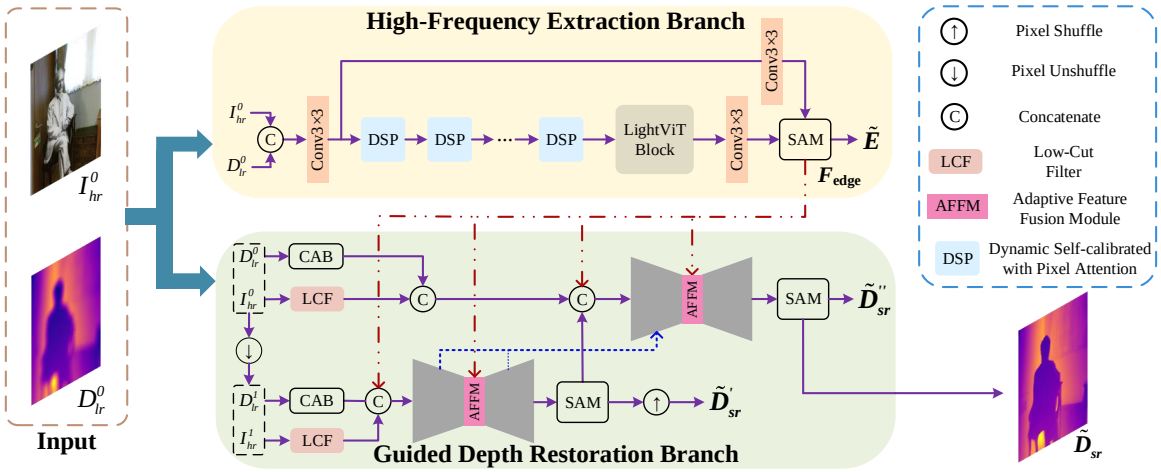


Figure 5.2: DSR-EI architecture. Rectangles with different colors depict different stages and functions in each stage. Different colored arrows and lines indicate different data flows.

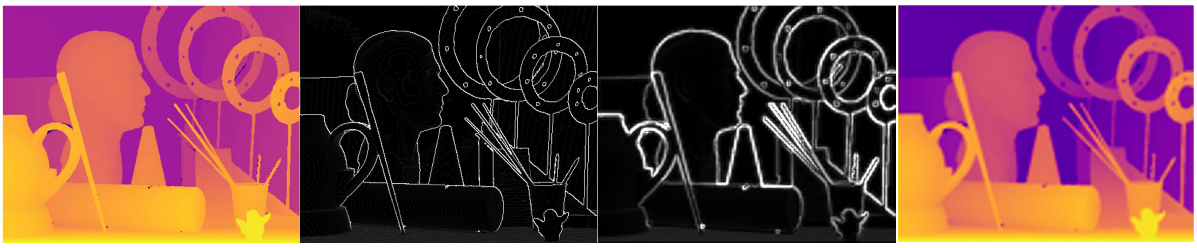


Figure 5.3: High-frequency information loss (factor $4\times$). From left to right, HR depth map and its corresponding gradient map, followed by the gradient map from bicubic upsampled LR depth map and LR depth map itself. HF information is mostly lost in the second gradient map.

superior accuracy achieved by DSR-EI compared to existing methods [40, 287].

5.2 DSR-EI Framework

In GDSR, HF information in color images – complementary to depth maps – is essential for achieving high performance, which motivates us to seek an efficient method to extract it. In this section, we present our framework that exploits explicit and implicit HF information for depth super-resolution. Then, we introduce the two branches in our network: the High-Frequency Extraction Branch (HFEB) and the Guided Depth Restoration Branch (GDRB).

Fig. 5.2 shows an overview of our architecture. Given the LR depth map $D_{lr} \in \mathbb{R}^{h\times w\times 1}$ and the corresponding HR color image $I_{hr} \in \mathbb{R}^{H\times W\times 3}$, we aim at restoring HR depth map $\tilde{D}_{sr}$. Note that $H = s \times h$ and $W = s \times w$, where s denotes the upsampling factor – e.g., $4\times, 8\times$ or even $16\times$. In our proposed network, the input depth map is firstly upsampled with bicubic interpolation to the same size as I_{hr}. At different scales, we denote the corresponding depth maps and color images as D^i_{lr} and I^i_{hr}, respectively, with $s = 2^i$. Then, according to the above notation, the input images D^0_{lr} and I^0_{hr} are fed into the two branches, respectively. Before being sent to GDRB, the RGB and depth images are respectively processed by a channel-attention

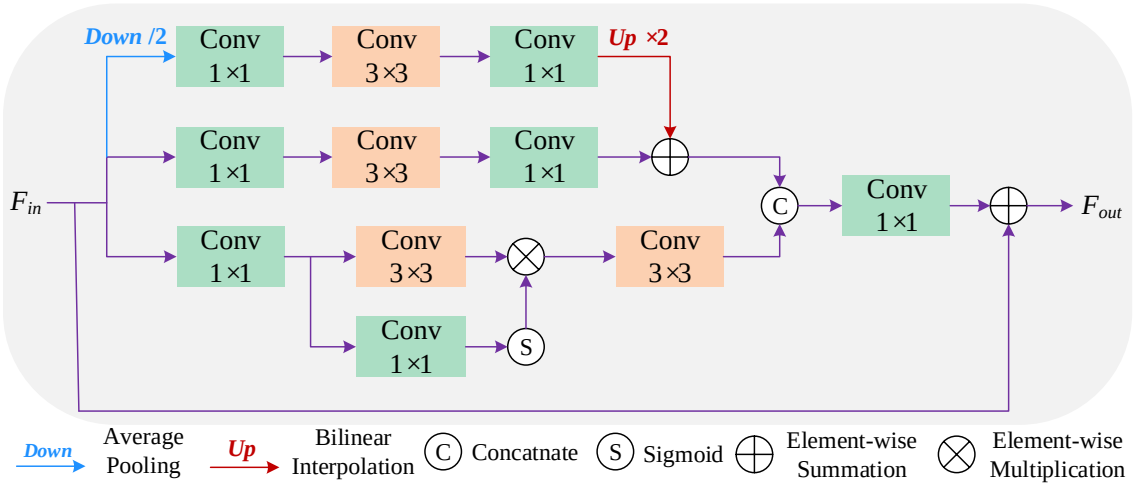


Figure 5.4: DSP architecture. Differently from SCPA [285], our module processes features at different scales, allowing to extract explicit HF information more effectively.

block (CAB) [275] and a low-cut filtering (LCF) module, which will be explained in detail in Sec. 5.2.2.

5.2.1 High-Frequency Extraction Branch (HFEB)

We argue HF information is crucial for effective super-resolving depth and is often lost by upsampling. The primary goal of HFEB is to produce an accurate gradient map from an LR depth map, with the support of a color image jointly processed with it.

Indeed, as pointed out in [237], networks for GDSR tend to focus more on depth discontinuities or object boundaries. However, from Fig. 5.3, we can notice that even with a $4\times$ factor, most high-frequency information vanishes, as shown by the gradient maps extracted from HR and upsampled LR depth maps, leading to severe degradation of the super-solved depth map. Traditional methods tend to transfer texture to depth maps rather than structural details, failing to extract accurate edges. Moreover, methods extracting binary edges [237] gather insufficient high-frequency information, yielding sub-optimal results.

The work [162] has shown that transformer-based networks can extract clear and meaningful edges by leveraging both global and local features simultaneously. Considering the sparsity of edge maps, we design an efficient transformer, inspired by dynamic scale policy [223] and self-attention [221], to obtain strong HF priors for guiding depth super-resolution. Specifically, our transformer consists of a stack of blocks called dynamic self-calibrated convolution with pixel attention (DSP) and one LightViT block [83]. To better extract HF features, we design the DSP block, which is inspired by SCPA [285] and performs self-calibrated convolution with two branches at a single scale. However, unlike SCPA, our DSP block includes an additional branch that enables the processing of features at different scales without incurring extra computational burden, as we will demonstrate empirically in our experiments. Specifically, stacked DSP blocks can be expressed as:

$$\Phi_M = \mathcal{F}^M_{DSP}(\mathcal{F}^{M-1}_{DSP}(\cdots\mathcal{F}^1_{DSP}(\Phi_0)\cdots)), \tag{5.1}$$

where $\mathcal{F}_{DSP}^{m}$ denotes the mapping of the m-th DSP block, $m \in [1, M]$, Φ_0 and Φ_M are the input/output features, respectively. As shown in Fig. 5.4, each DSP block includes three branches: the upper is the dynamic scale branch, the middle is the flat convolution branch, and the lower is the pixel attention branch. Specifically, we employ three convolutions with 1×1 kernel to split the channels, which are further processed by each branch. Note that the dynamic scale branch needs to be downsampled before 1×1 convolution. Given the input Φ_{m-1}, we obtain:

$$\Phi_{m-1}^{1} = Conv_{1\times1}((\Phi_{m-1}) \downarrow) \tag{5.2}$$

$$\Phi_{m-1}^{k} = Conv_{1\times1}(\Phi_{m-1}), \tag{5.3}$$

where Φ_{m-1}^{1} is the output from the upper dynamic scale branch, $k = 2, 3$ denotes the features of the other two branches, $Conv_{1\times1}$ is 1×1 convolution, and $\downarrow$ is the downsampling operation. Except for the pixel attention branch, which has features with half the total channels, the other two branches process features with $\frac{1}{4}$ of the channels each. Next, the pixel attention branch obtains features through the pixel attention scheme [285]. In contrast, the other two branches extract spatial information with a 3×3 flat convolution, followed by a 1×1 convolution to restore the number of channels to be the same as the pixel attention branch. Note that the dynamic scale branch needs upsampling after 1×1 convolution. Then, the features from the dynamic scale and the flat convolution branches can be fused by summation. After concatenation of the features followed by a 1×1 convolution, the DSP finally generates the output features Φ_m in a residual learning fashion. It can be written as follows:

$$\Phi_{m}^{1} = Conv_{1\times1}(Conv_{3\times3}(\Phi_{m-1}^{1})) \uparrow \tag{5.4}$$

$$\Phi_{m}^{2} = Conv_{1\times1}(Conv_{3\times3}(\Phi_{m-1}^{2})) \tag{5.5}$$

$$\Phi_{m}^{3} = Conv_{3\times3}(\Phi_{m-1}^{3}) \odot \sigma(Conv_{1\times1}(\Phi_{m-1}^{3})) \tag{5.6}$$

$$\Phi_{m}^{'} = Conv_{3\times3}(\Phi_{m}^{3}) \tag{5.7}$$

$$\Phi_{m}^{''} = Conv_{3\times3}(\Phi_{m}^{1} \oplus \Phi_{m}^{2}), \tag{5.8}$$

where σ is the sigmoid function, $\odot$ and $\oplus$ are element-wise multiplication and element-wise summation, respectively, and $\uparrow$ denotes the upsampling operation. After concatenation of the features $\Phi_{m}^{'}$ and $\Phi_{m}^{''}$ followed by a 1×1 convolution, the DSP finally generates the output features Φ_m in a residual learning manner. This process can be expressed as follows:

$$\Phi_m = Conv_{1\times1}([\Phi_{m}^{'}, \Phi_{m}^{''}]) \oplus \Phi_{m-1}, \tag{5.9}$$

where $[\cdot]$ perform concatenation.

To further enhance the feature representation of the subnetwork, we incorporate LightViT [83] as the tail module, which utilizes local-global attention broadcast to aggregate information from all tokens, allowing for the efficient integration of global dependencies of local tokens into each image token. Finally, considering that the supervised attention module (SAM) [267] can restore information progressively between stages/branches, we employ it to output the gradient map $E \in \mathbb{R}^{H\times W\times1}$ and high-frequency features $F_{edge} \in \mathbb{R}^{H\times W\times C}$, used respectively as intermediate output – allowing for explicit supervision over edges – and as guidance for GDRB. Under this lightweight design, HFEB can effectively still extract meaningful structural information with different scale receptive fields.

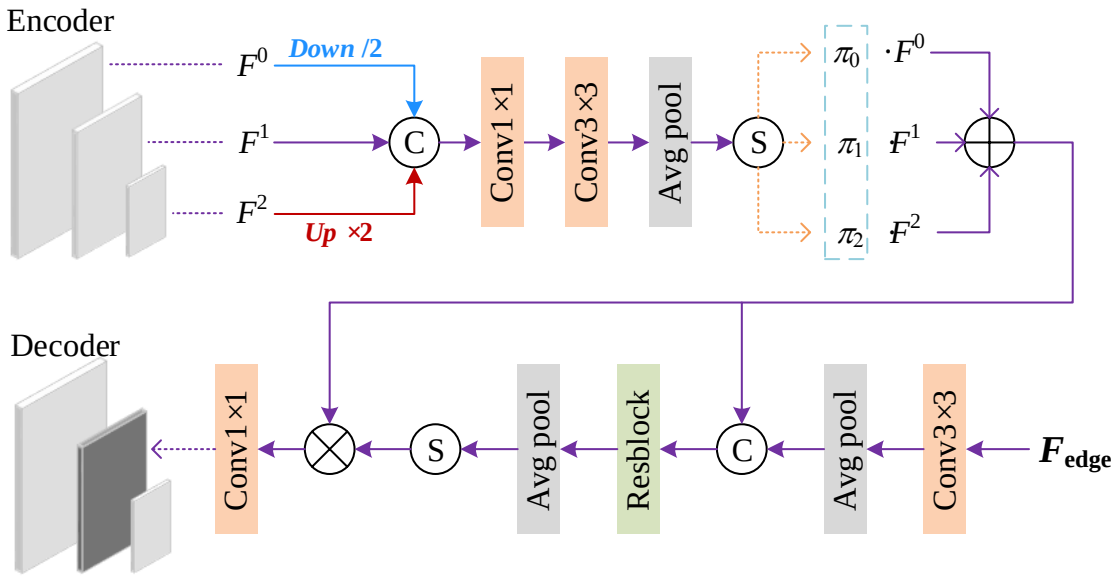


Figure 5.5: AFFM architecture, operating at middle scale. AFFMs for the remaining scales follow the same design.

5.2.2 Guided Depth Restoration Branch (GDRB)

As shown in Fig. 5.2, GDRB, which is inspired by MPRNet [267], is composed of two stages, and each one processes features at three scales, following a coarse-to-fine strategy [57, 176]. The two stages are implemented with standard U-net architectures [172]. More specifically, a cross-stage feature fusion module [267] is deployed between the two, which proved to be effective in image restoration and, in our design, allows GDRB to benefit from the intermediate features extracted by HFEB. To prevent aliasing in downsampling, we employ content-aware filtering layers (CAFL) [297] in the encoders. Besides, GDRB deploys some further SAM blocks [267], allowing valuable features to propagate to the next stage. In addition to depth features, the SAMs of the two stages also output depth maps $\tilde{D}'_{sr}$ and $\tilde{D}''_{sr}$, to which intermediate supervision is provided. Note that input images are downsampled to the lower stage using pixel unshuffling to prevent information loss. Subsequently, the depth map output of this stage is restored at high resolution by employing pixel shuffling.

Based on the above structure, we propose two novel modules: AFFM and LCF. The former fuses gradient features between each encoder/decoder, while the latter supplements additional HF information in an implicit manner.

Adaptive feature fusion module. Recent networks such as [204, 262] typically concatenate RGB and depth features directly during feature fusion, followed by additional operations such as channel attention [275] to capture useful information. In contrast, inspired by [124], we run adaptive feature fusion through AFFM in two steps to strengthen the reconstruction of HF cues, as illustrated in Fig. 5.5. We differentiate from [124] by using dynamic convolution [25] to better aggregate depth and HR RGB features. In the first step, we generate dynamic weights $\pi_i, i = 0, 1, 2$, which are then assigned to features from different scales within the current stage. Finally, we perform element-wise summation to obtain the feature maps F'. For clarity, the figure shows the module working at the middle scale of the network as an example, with the

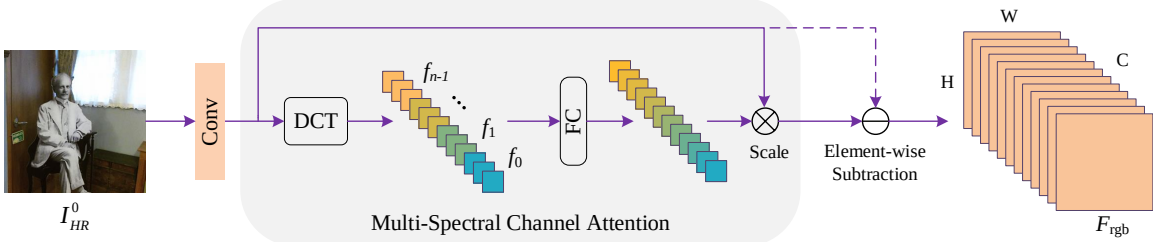


Figure 5.6: Low-cut filtering module (LCF). LF features are extracted through DCT and multi-spectral channel attention, and subtracted from the input to retain HF features.

others sharing the same design. The process is defined as follows:

$$F_{cat} = Conv_{3\times3}(Conv_{1\times1}([F^0\downarrow, F^1, F^2\uparrow])) \tag{5.10}$$

$$\{\pi_0, \pi_1, \pi_2\} = \sigma(Avgpool(F_{cat})) \tag{5.11}$$

$$F^{'} = \pi_0 \cdot F^0\downarrow + \pi_1 \cdot F^1 + \pi_2 \cdot F^2\uparrow, \tag{5.12}$$

where $F^i, i = 0, 1, 2$ denotes the feature maps from the three scales, and $\downarrow$, $\uparrow$ are respectively downsampling and upsampling operators.

In the second step, gradient features F_{edge} from HFEB are concatenated with $F^{'}$. Then, per-pixel attention maps F_{att} are generated by a ResBlock [74] followed by an average pooling operation. These attention maps are then applied directly to the adaptively fused features $F^{'}$ through element-wise multiplication operation. Finally, after 1×1 convolution, the attention-guided features F^i_{out} are delivered to the corresponding scale of the current stage. In Fig. 5.5, the output is passed to the middle scale of the decoder. AFFMs working at the other scales send their output to the corresponding scale in the decoder. This step can be formalized as follows:

$$F^{''} = [Avgpool(Conv_{3\times3}(F_{edge})), F^{'}] \tag{5.13}$$

$$F_{att} = \sigma(Avgpool(ResBlock(F^{''}))) \tag{5.14}$$

$$F^1_{out} = con_{1\times1}(F^{'} \otimes F_{att}), \tag{5.15}$$

where $\otimes$ is an element-wise multiplication operation and F^1_{out} denotes the output features at the middle scale.

Low-cut filtering module. The performance of our method greatly benefits from the explicit gradient information, but some valuable high-frequency information still vanishes. This fact motivates us to consider extracting complementary information in the frequency domain. As a common practice [17, 116], we use the low-frequency information of the discrete cosine transform (DCT) to compress images. Based on the design approach proposed in [164], we develop a filtering module utilizing feature decomposition in the frequency domain to extract low-frequency components from the input. Specifically, we apply a 1×1 convolution followed by a channel split to the input color image I^0_{hr}. Then, we can obtain assigned frequency components from the output features $[f_0, f_1, \cdots, f_{n-1}]$ after DCT. Thus, the multi-spectral channel attention maps are generated by a fully connected layer and sigmoid activation. According to [164], the low-frequency information is first assured to pass. Thus, we subtract such a low-frequency component from the input features producing the complementary high-frequency

features F_{rgb}. Fig. 5.6 illustrates LCF in detail. The high-frequency cues extracted from these features enable GDRB to progressively super-resolve LR depth maps into HR ones.

Refinement. To enhance the depth quality further, we optionally feed our final output into NLSPN [153] for refinement. This variant of the method is referred to as DSR-EI^{+}.

5.2.3 Training Loss

Our network is trained in an end-to-end fashion using two loss terms: depth loss L_d and gradient loss L_g. The depth loss is defined as:

$$\begin{aligned} L_d = \| (\tilde{D}_{sr} - D_{gt}) \odot \mathbb{I} \|_1 + \lambda_d \cdot \| (\tilde{D}'_{sr} - D_{gt}) \odot \mathbb{I} \|_1 \\ + \lambda_d \cdot \| (\tilde{D}''_{sr} - D_{gt}) \odot \mathbb{I} \|_1, \end{aligned} \tag{5.16}$$

where D_{gt} is the ground truth depth, $\tilde{D}_{sr}$, $\tilde{D}'_{sr}$ and $\tilde{D}''_{sr}$ are predicted depth maps from different stages, and $\mathbb{I}$ is pixel validity, as defined in [40]. We empirically set $\lambda_d = 0.2$. Gradient loss L_g is computed on HEFB output, as:

$$L_g = \| \tilde{E} - E_{gt} \|_1, \tag{5.17}$$

where $\tilde{E}$ is the predicted gradient map and E_{gt} is the ground truth one, extracted according to [124]. Thus, the total loss can be defined as:

$$L_{total} = L_d + \lambda_g \cdot L_g, \tag{5.18}$$

with λ_g empirically set to 0.01.

5.3 Experimental Results

In this section, we validate the effectiveness of our proposal. We first introduce datasets, metrics and implementation details involved in our evaluation. Then, we compare DSR-EI with state-of-the-art methods, conduct an ablation study on our model and, finally, discuss its limitations.

5.3.1 Datasets and Metrics

We evaluate DSR-EI on four datasets, compared with existing methods when super-solving depth maps by three different upsampling factors: $4\times$, $8\times$, and $16\times$.

Middlebury [78, 178, 180, 181]. We train all learning-based methods using 50 RGB-D images with ground truth from Middlebury 2005, 2006 and 2014 datasets. As in [40], we split the datasets into 40 images for training, 5 images for validation, and the remaining 5 for testing.

NYUv2 [188]. It contains 1449 RGB-D images in total. Following [40], we randomly split it into 849 RGB-D images for the training set, 300 for the validation set, and 300 for the test set. Compared to [121, 262], it comes with a validation set to make the comparison fairer.

DIML [30, 94–96] consists of 2 million color images and corresponding depth maps from indoor and outdoor scenes. We adopt the same strategy outlined in [40], i.e., considering only the indoor data subset, and use 1440 for training, 169 for validation, and 503 for testing.

RGBDD [76] is a new real-world dataset for GDSR, which consists of 4811 image pairs. For evaluation, we follow the protocol described in [76], using 2215 images (1586 portraits, 380 plants, 249 models) as the training set and 405 images (297 portraits, 68 plants, 40 models) as the test set.

Metrics. Following [40], we compute mean square error (MSE / cm^2) and mean absolute error (MAE / cm) as metrics on Middlebury, NYUv2 and DIML. For RGBDD, we use root mean square error (RMSE / cm) as in [76].

5.3.2 Implementation Details

During training, the HR depth maps and the color images are randomly cropped into 256×256 patches. LR depth patches are generated by bicubic interpolation at 64×64, 32×32, 16×16 resolution for $4\times$, $8\times$ and $16\times$ factors, respectively. We randomly extract about 75K, 168K, 223K and 232K patches from Middlebury, NYUv2, DIML and RGBDD for training. Before being fed to the network, depth maps and images are normalized in the [0, 1] range.

We use Pytorch [155] to implement and train DSR-EI, on a single Nvidia RTX 3090 GPU. The batch size is set to 4, using Adam as the optimizer. The learning rate is initialized to 1×10^{-4}, then performing a 5-epoch warm-up and cosine annealing. We use random rotation, horizontal/vertical flipping as data augmentation. According to the size of the four datasets, we train our network for 1505, 198, 155 and 109 epochs on Middlebury, NYUv2, DIML and RGBDD, respectively. When evaluating results on a specific dataset, we do not perform any pre-training on the others. Following [40], testing is performed by processing 256×256 patches at a time on Middlebury, NYUv2 and DIML for fairness, while full-resolution images are processed for RGBDD.

Dataset	**Middlebury**			**NYUv2**			**DIML**		
Methods	$4\times$	$8\times$	$16\times$	$4\times$	$8\times$	$16\times$	$4\times$	$8\times$	$16\times$
GF [73]	33.3 / 1.27	40.5 / 1.49	67.4 / 2.21	114 / 3.91	142 / 4.47	249 / 6.34	25.6 / 1.45	34.1 / 1.77	66.3 / 2.74
SD [72]	24.9 / 0.46	82.5 / 0.86	511 / 1.73	36.0 / 1.31	105 / 2.57	533 / 5.07	10.5 / 0.40	44.9 / 0.83	41.1 / 1.91
P2P [128]	39.8 / 0.79	32.7 / 0.82	41.5 / 1.24	112 / 3.61	122 / 3.86	219 / 5.40	20.7 / 1.15	23.0 / 1.26	39.3 / 1.78
MSG [84]	4.13 / 0.22	10.5 / 0.43	34.2 / 1.06	6.85 / 0.81	24.1 / 1.66	84.5 / 3.35	1.73 / 0.22	4.13 / 0.40	13.0 / 0.93
DKN [92]	4.29 / 0.18	11.2 / 0.38	47.6 / 1.42	11.4 / 1.03	29.8 / 1.82	115 / 4.01	3.47 / 0.33	5.47 / 0.45	19.3 / 1.20
FDKN [92]	3.60 / 0.16	10.4 / 0.37	38.5 / 1.18	9.07 / 0.85	29.9 / 1.80	113 / 3.95	2.20 / 0.23	5.95 / 0.47	20.8 / 1.24
PMBANet [262]	4.72 / 0.25	9.48 / 0.38	30.6 / 0.89	10.8 / 0.93	17.2 / 1.38	84.9 / 3.26	3.05 / 0.31	5.87 / 0.47	13.8 / 0.87
FDSR [76]	7.72 / 0.35	23.2 / 0.69	55.4 / 1.51	10.1 / 0.94	19.5 / 1.38	86.4 / 3.35	2.75 / 0.29	8.40 / 0.66	32.9 / 1.66
JIIF [204]	2.70 / 0.11	8.01 / 0.27	37.5 / 0.98	3.28 / 0.52	15.2 / 1.29	59.9 / 2.81	1.19 / 0.16	3.65 / 0.32	11.7 / 0.81
DCTNet [287]	5.00 / 0.24	15.1 / 0.57	52.3 / 1.50	3.63 / 0.68	20.9 / 1.79	77.0 / 3.61	2.09 / 0.31	7.08 / 0.65	23.4 / 1.75
LGR [40]	3.04 / 0.13	7.26 / 0.24	24.7 / 0.67	6.45 / 0.73	19.6 / 1.42	67.5 / 2.90	1.68 / 0.20	3.51 / 0.31	9.45 / 0.68
DADA [137]	2.58 / 0.11	5.68 / 0.20	16.3 / 0.48	4.83 / 0.64	16.6 / 1.30	59.0 / 2.64	1.33 / 0.17	2.93 / 0.28	7.61 / 0.59
DSR-EI	**2.46** / 0.08	6.20 / **0.18**	**15.8** / 0.47	2.82 / 0.49	**11.8** / 1.12	47.8 / 2.48	0.70 / 0.13	2.12 / **0.22**	**6.29** / 0.52
DSR-EI$^+$	2.56 / **0.07**	**5.13 / 0.18**	16.6 / **0.40**	**2.75 / 0.47**	**11.8 / 1.09**	**47.14 / 2.40**	**0.65 / 0.12**	**2.09 / 0.22**	6.31 / **0.50**

Table 5.1: Results on Middlebury, NYUv2 and DIML datasets. The lower the MSE and MAE, the better.

5.3.3 Comparison with State-of-the-Art

We compare DSR-EI to GF [73], SD [72], P2P [128], MSG [84], DKN and its fast implementation FDKN [92], PMBANet [262], FDSR [76], JIIF [204], DCTNet [287], LGR [40],

Methods	$4\times$	$8\times$	$16\times$
SDF [107]	2.00	3.23	5.16
SVLRM [149]	3.39	5.59	8.28
DJF [107]	3.41	5.57	8.15
DJFR [109]	3.35	5.57	7.99
PAC [194]	1.25	1.98	3.49
CUNet [42]	1.18	1.95	3.45
DKN [92]	1.30	1.96	3.42
FDKN [92]	1.18	1.91	3.41
FDSR [76]	1.16	1.82	3.06
DCTNet [287]	1.07	1.78	3.18
DCSR [235]	1.23	1.66	2.56
JGF-M [226]	1.21	1.81	2.91
RSAG [266]	1.14	1.75	2.96
DSR-EI	**0.91**	**1.37**	**2.10**
DSR-EI$^+$	**0.91**	<u>1.38</u>	**2.10**

Table 5.2: Results on RGBDD dataset. We report RMSE, the lower the better.

and finally to DADA [137] on Middlebury, NYUv2 and DIML datasets. We could not compare with PDRNet [121] under the same setting because the source code was unavailable at the time of testing. For the other methods, we use the results from [40] or the officially published codes, and results from [137, 266] for concurrent works. On the RGBDD dataset, the proposed network is compared to SDF [107], SVLRM [149], DJF [107], DJFR [109], PAC [194], CUNet [42], FDKN [92], DKN [92], FDSR [76], DCTNet [287], DCSR [235], JGF-M [226] and RSAG [266]. To be fair with DCTNet [287], we downsample depth maps as the LR input. When reporting results, we highlight **absolute** and <u>second</u> best methods for each metric on each dataset.

Quantitative Comparison. Tabs. 5.1 and 5.2 report the accuracy of super-solved depth maps at factors $4\times$, $8\times$ and $16\times$ on the four datasets. As expected, learning-based methods show a significant improvement over traditional methods [72, 73, 128]. DSR-EI vastly outperforms any existing network, with larger gaps in accuracy with the increasing of the upsampling factor. This can be attributed to the limitations affecting existing methods, i.e., 1) the guidance of either explicit or implicit RGB features alone being insufficient; 2) multi-modal information fusion on a single scale being not flexible enough to deal with complex scenes. Both limitations are fully addressed by DSR-EI, which consistently outperforms concurrent works [137, 266].

The margin is consistent both on perfect (Middlebury) and noisy datasets (NYUv2, DIML, RGBDD), with the latter being a more challenging, realistic benchmark. Although DSR-EI$^+$ is definitely the absolute best, its margin over DSR-EI is negligible, with tiny gains yielded by NLSPN with respect to our main modules. Indeed, DSR-EI alone consistently outperforms any other approach already.

Qualitative Comparison. Fig. 5.7 provides qualitative comparisons of the GDSR results across multiple datasets, i.e., NYUv2, Middlebury, and DIML, which cover various types of scenarios and noise levels. We can notice that our model can extract boundaries and details

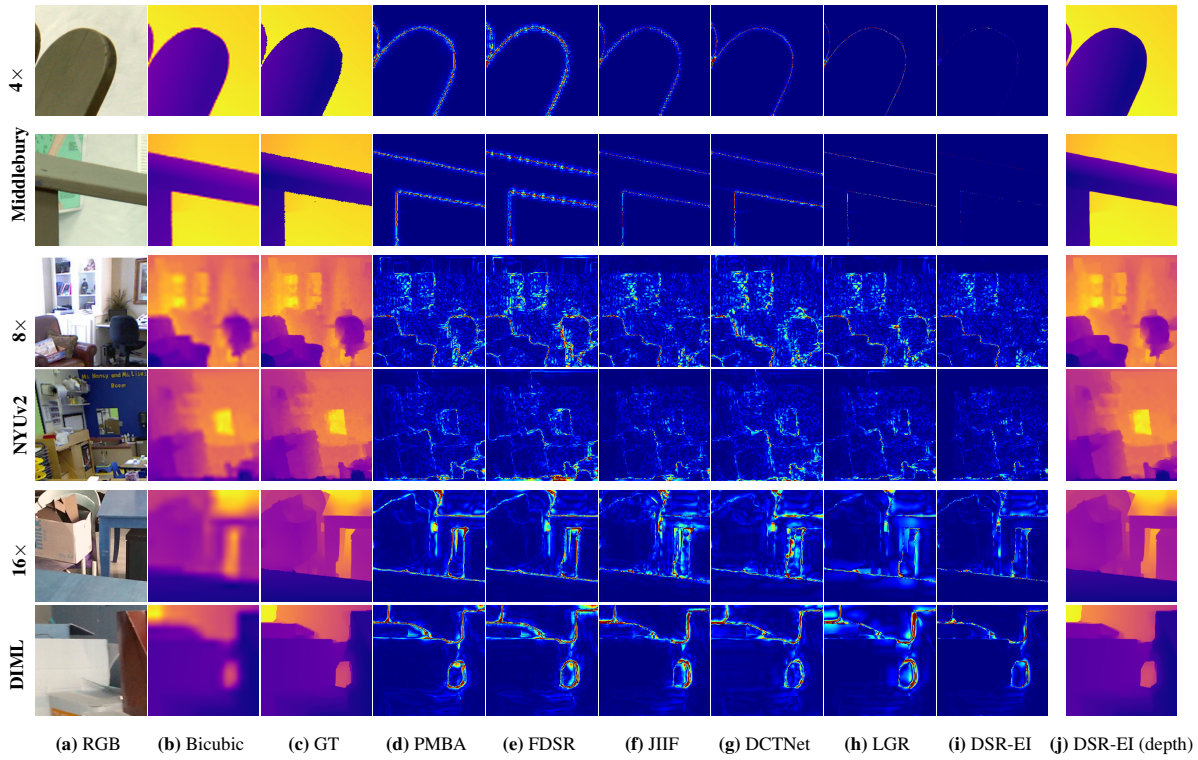


Figure 5.7: Qualitative comparison on the Middlebury, NYUv2, and DIML. From left to right: (a) RGB image, (b) Bicubic upsampled depth map, (c) GT; then, error maps achieved by selected methods: (d) PMBA [262], (e) FDSR [76], (f) JIIF [204], (g) DCTNet [287], (h) LGR [40]; finally, (i) error maps and (j) predictions by DSR-EI.

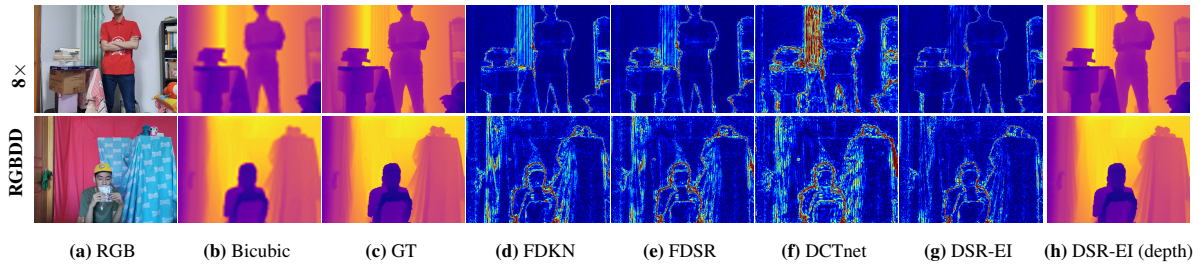


Figure 5.8: Qualitative comparison on the RGBDD dataset. From left to right: (a) RGB image, (b) Bicubic upsampled depth map, (c) GT; then, error maps achieved by selected methods: (d) FDKN [92], (e) FDSR [76], (f) DCTNet [287]; finally, (g) error maps and (h) predictions by DSR-EI.

from the RGB image more accurately. Specifically, on the depth discontinuities in the two topmost rows, DSR-EI$^+$ introduces fewer artifacts around the edges of objects where specular reflections occur, which means that our network is more robust in removing texture-copy effects from RGB images compared with other methods. On the two samples selected from NYUv2, our network produces fewer errors in recovering fine structures and details. For example, in the fourth row of this figure, there are many tiny objects whose shape and structure are degraded due

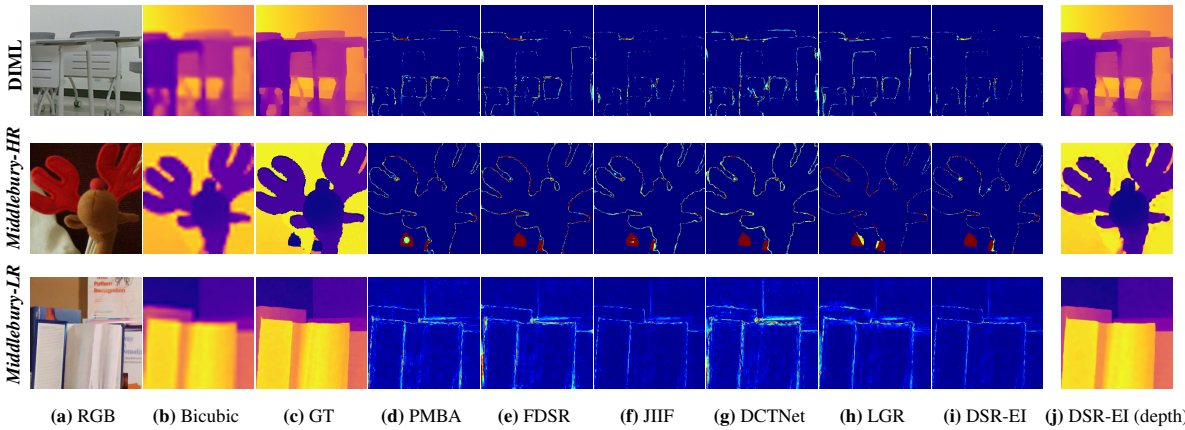


Figure 5.9: Visual comparison on cross-dataset generalization (scaling factor $8\times$). The top, middle and last row show the error maps on the DIML dataset, the *Middlebury-HR* dataset and the *Middlebury-LR* dataset, respectively. From left to right: (a) RGB image, (b) Bicubic upsampled depth map, (c) GT; then, error maps achieved by selected methods: (d) PMBA [262], (e) FDSR [76], (f) JIIF [204], (g) DCTNet [287], (h) LGR [40]; finally, (i) error maps and (j) predictions by DSR-EI.

to downsampling. Other methods may produce artifacts and inaccurate depth boundaries, while our method has a clear advantage in recovering fine-grained depth details. Fig. 5.8 also reports two examples on the RGBDD dataset. In this case, we notice fewer errors in the background, *e.g.*, on the curtain.

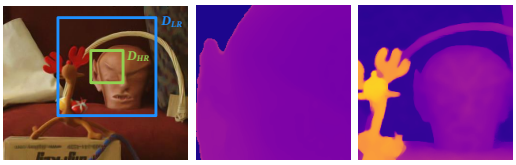
Methods	DIML	Middlebury-*HR*	Middlebury-*LR*
GF [73]	34.1 / 1.77	40.5 / 1.49	25.6 / 2.31
SD [72]	44.9 / 0.83	82.5 / 0.86	28.8 / 2.07
P2P [128]	23.0 / 1.26	32.7 / 0.82	15.8 / 1.73
MSG [84]	5.76 / 0.51	11.0 / 0.54	8.89 / 1.62
FDKN [92]	6.74 / 0.53	10.0 / 0.43	5.54 / 0.99
PMBANet [262]	7.35 / 0.59	9.62 / 0.46	4.16 / 0.91
FDSR [76]	7.73 / 0.74	18.4 / 0.73	6.92 / 1.09
JIIF [204]	4.10 / 0.38	19.3 / 0.74	4.40 / 0.92
DCTNet [287]	5.64 / 0.77	17.5 / 0.77	6.96 / 1.15
LGR [40]	4.95 / 0.40	**8.25 / 0.35**	5.94 / 1.11
DSR-EI$^+$	**3.72 / 0.36**	14.6 / 0.54	**3.44 / 0.87**

Table 5.3: Cross-dataset generalization. All methods are trained on NYUv2 and tested on DIML/Middlebury with factor $8\times$. Middlebury-*HR* is the test set defined in [40], Middlebury-*LR* is the one from [204]. The lower MSE and MAE, the better.

Cross-dataset Generalization. We conclude the comparison with existing methods by conducting cross-dataset experiments with $8\times$ factor. All methods are trained on the NYUv2

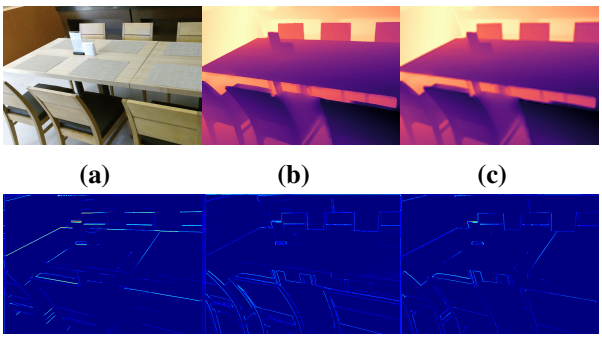
dataset and directly evaluated on DIML and Middlebury. Tab. 5.3 collects quantitative results for the 11 selected methods. Again, CNN-based methods attain better performance than traditional approaches, despite the domain gap playing a significant role in performance – as evident by comparing results with Tab. 5.3. Nonetheless, DSR-EI outperforms any other framework on DIML.

When considering the Middlebury dataset, we evaluate using the setting proposed in [40] – Middlebury-*HR* in the table. In this case, our results are slightly less accurate compared to a few existing methods. However, given the very high resolution of Middlebury images, we argue that this testing protocol – i.e., consisting of processing 256×256 crops at a time – penalizes our network's ability to leverage the global context in the input that results irremediably reduced to a very local area in these images. Therefore, we also evaluate on Middlebury test set defined by [204] – Middlebury-*LR* in the table. Note that different subsets of images are used in Middlebury-*HR* and Middlebury-*LR* splits. Besides, Middlebury-*LR* images are resized and processed without cropping, i.e., used at full-size after resizing, allowing to fully exploit global context, while this is not feasible with Middlebury-*HR* due to memory constraints. In this case, DSR-EI attains the best performance again, confirming our previous analysis, as shown in Tab. 5.3. Such a difference in terms of context is highlighted in Fig. 5.10.



(a) RGB **(b)** D_{hr} **(c)** D_{lr}

Figure 5.10: Image context processed on Middlebury – HR vs LR. (a) RGB image and depth patches D processed when testing on (b) Middlebury-*HR* and (c) Middlebury-*LR*.



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 5.11: Visual exhibition of high-frequency features generated from HFEB. (a) RGB image, (b) GT, (c) Bicubic, (d)-(f) high-frequency features.

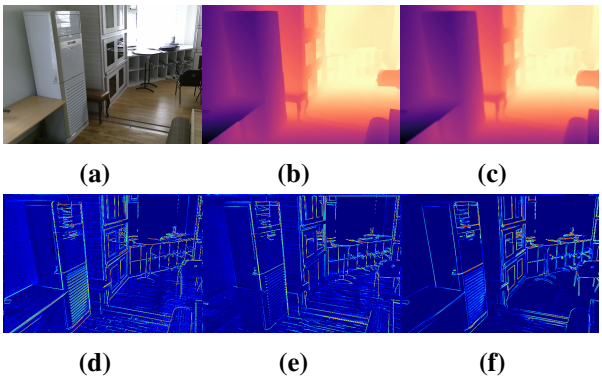


Figure 5.12: Visual exhibition of shallow high-frequency features generated from LCF. (a) RGB image, (b) GT, (c) Bicubic, (d)-(f) high-frequency features.

5.3.4 Ablation Study

We now perform a series of ablation experiments to measure the impact of key components and parameters in DSR-EI. We collects the outcome of these studies, conducted on NYUv2 test set with $8\times$ factor. Without loss of fairness, NLSPN is never used here – to fully focus on the impact of single components. The configurations marked in gray in Tab. 5.4-Tab. 5.10 correspond to our final model without NLSPN.

(a) Implicit vs Explicit High-Frequency Features. To measure the impact of both implicit and explicit HR features, we compare the performance of the proposed network and its variants when extracting either only one of the two. The quantitative results are collected in Tab. 5.4. Without the help of gradient maps (I), the performance of the network significantly degrades. We believe this is caused by the difficulty in effectively extracting fine structures or salient edges required for LR depth maps from implicit HF features alone. Moreover, explicit features highlight regions in the image that need to be focused on, avoiding DSR-EI to learn to localize them and easing its task. Fig. 5.11 shows three among the high-frequency features F_{edge} from a representative sample. We can notice how each of the three mainly emphasizes object boundaries, confirming the effectiveness of HFEB at extracting gradient information. At the same time, we can notice how the input RGB images expose very low texture, further confirming the effectiveness of HFEB at localizing high-frequency information.

Nonetheless, explicit HF features alone as guidance (II) are insufficient as well. We argue that the explicit information might neglect some RGB features, whereas implicit HF feature extraction can recover them. Furthermore, to verify the effectiveness of LCF, we replace it with ResBlock [74] (III) and CAB [275] (IV) to extract shallow features from RGB images, highlighting a negative impact on implicit features extraction – i.e., they result less accurate than (II) in terms of MAE. Fig. 5.12 shows some of the features extracted by LCF. We can notice how, in addition to the primary high-frequency information, other information is encoded, such as semantics, which can further provide support for the explicit high-frequency information extracted in parallel by HFEB and improve the guidance for the final, depth super-resolution task.

(b) Ablation on Explicit High-Frequency Features. Based on the previous analysis, HFEB can significantly improve the network. To determine which high-frequency information is more suitable as guidance for GDSR, we experiment with five kinds of edge maps used

No.	Gradient	Shallow Feature	LCF	CAB	ResBlock	MSE	MAE
(I)	✗	✓	✓			13.1	1.19
(II)	✓	✗				12.4	1.14
(III)	✓	✓			✓	12.3	1.15
(IV)	✓	✓		✓		12.2	1.15
(V)	✓	✓	✓			**11.8**	**1.12**

Table 5.4: Abaltion study – high-frequency information. Scale $8\times$.

No.	HF Information	MSE	MAE
(I)	Canny Edge	**2.82** / 12.0	**0.49** / 1.13
(II)	Gaussian Edge	2.84 / 12.1	0.52 / 1.16
(III)	DCT	**2.82** / 12.1	0.50 / 1.15
(IV)	Wavelet Transform	2.83 / 12.1	0.50 / 1.15
(V)	Gradient Map	**2.82 / 11.8**	**0.49 / 1.12**

Table 5.5: Different configurations for HR information. Scale $4\times$ / $8\times$.

as ground truth E_{gt} to train HFEB with scaling factors of $4\times$ and $8\times$: (1) the Canny edge map, (2) the Gaussian high-frequency map, (3) the high-frequency map generated by discrete cosine transform, (4) the high-frequency wavelet map and (5) the gradient map, as shown in Table 5.5. The Gaussian high-frequency map is obtained using a Gaussian filter, as detailed in [237]. Table 5.5 reports the outcome of the evaluation. From it, we can see that the Canny edge and the gradient map allow for better performance. Although DSR-EI with the gradient map attains the best results in terms of MSE and MAE with the scaling factors of $8\times$, the different types of high-frequency maps do not significantly affect the final upsampling result.

No.	Config.	Params (M)	Flops (G)	MSE	MAE
(I)	EdgeNet	5.78	95.6	12.0	**1.12**
(II)	SCPA	0.29	13.1	12.5	1.16
(III)	HFEB	**0.27**	**11.6**	**11.8**	**1.12**

Table 5.6: Effectiveness of HFEB. Scale $8\times$.

(c) Impact of HFEB. To verify the effectiveness of HFEB, we replace it with EdgeNet [124] – based on the widely-used U-net structure – and SCPA [285], which inspires our scaling strategy. As shown in Table 5.6, although the parameter size of EdgeNet is 5.6M, its performance is almost the same as our HFEB, while the parameter size of our network is only 0.7M, i.e. only $\frac{1}{8}$ of it. This fact highlights that our network based on a transformer is more efficient at feature extraction.

Besides, unlike previous works that employ fixed feature scaling rules, we adopt a dynamic scaling strategy to extract high-frequency features from depth maps. Table 5.6 also shows that our DSP with the dynamic scale strategy decreases the number of parameters while simultaneously enhancing the performance of GDSR. Compared to the original SCPA [285], DSP can perform dynamic scaling according to the characteristics of the feature map to get a more effective receptive field.

(d) Impact of AFFM. We now measure the effectiveness of AFFM. Tab. 5.7 shows results

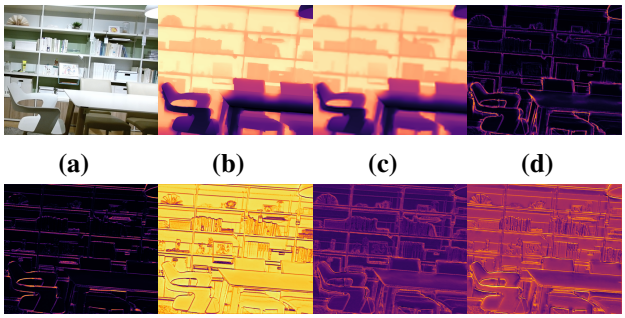


Figure 5.13: Visual exhibition of attention maps generated from AFFM. (a) RGB image, (b) GT, (c) Bicubic, (d)-(h) attention maps.

No.	Scales	Params (M)	MSE	MAE
(I)	H1	1.5	12.3	1.14
(II)	H1, H2	3.0	**11.8**	**1.12**
(III)	H1, H2, H3	4.5	**11.8**	**1.12**

Table 5.7: The impact of scales at which AFFM is applied.

obtained by deploying AFFM at different scales, respectively the highest (I), the first two (II) and all of the three scales. We can notice how performing fusion at the highest scale alone results insufficient, whereas using multi-scale features for fusion yields improvements, despite saturating already when using two scales, with the lowest one not providing additional, meaningful details to be taken into account.

No.	Config.	Params (M)	MSE	MAE
(I)	w/o AFFM	-	12.7	1.16
(II)	w/o att	1.3	12.2	1.13
(III)	Concat.	4.5	12.2	1.13
(IV)	AFFM	3.0	**11.8**	**1.12**

Table 5.8: Ablation study of AFFM. Scale $8\times$.

Furthermore, we ablate AFFM in its single components. Tab. 5.8 resumes the outcome of this evaluation. We first test the performance of DSR-EI without AFFM (I), highlighting a large drop in accuracy. By adding dynamic fusion, yet without using attention (II) vastly improves the results already, while replacing the weighted sum in the upper of Fig. 5.5 with concatenation and a ResBlock [74] (III) yields worse results compared to our full AFFM (IV).

Fig. 5.13 visualizes the attention maps produced by AFFM, highlighting how sharp and accurate they are in correspondence with depth discontinuities, tiny objects, and fine details. Thus, thanks to them AFFM can better focus on reconstructing depth boundaries and details more accurately.

(e) Ablation on LightViT and CAFL. Previous ablations allowed us to highlight the effectiveness of our designed components to some extent. Nonetheless, we conduct further experiments to thoroughly investigate the effect of existing components, LightViT and CAFL. As

No.	LightViT	CALF	MSE	MAE
(I)	✗	✗	12.6	1.17
(II)	✓	✗	12.3	1.15
(III)	✗	✓	12.0	1.14
(IV)	✓	✓	11.8	1.12

Table 5.9: Effect of LightViT and CAFL. Scale $8\times$.

shown in Table 5.9, using either LightViT (II) or CALF (III) improves the performance of the model over Case (I). Finally, the model attains the best results when using both components, further demonstrating their effectiveness in our final model.

No.	Stages	Params (M)	MSE	MAE
(I)	1	14.2	13.3	1.19
(II)	2	25.0	11.8	1.12
(III)	3	37.5	**11.6**	**1.10**

Table 5.10: Comparisons with different stage numbers. Scale $8\times$.

(f) Impact of Stages Number. To conclude, we evaluate the impact of the multi-stage design. As shown in Tab. 5.10, a single-stage architecture (I) is vastly outperformed by deploying two stages (II), yet at the expense of doubling the number of parameters. Furthermore, while the three-stage architecture (III) still yields some improvement, the benefit is minor in comparison to the significant increase in parameters. Hence, we choose two stages as the default configuration to balance accuracy and efficiency.

(g) Results on full-size images. In Tab. 5.1 and 5.2, we reported the results achieved by our model when processing 256×256 patches, to allow for a fair comparison with LGR [40] and DADA [137]. However, this irremediably reduces the global context processed by DSR-EI, hindering its capacity to exploit it enabled by the transformer blocks similar to what was observed in the generalization experiment on Middlebury (Tab. 4). In this section, we demonstrate how processing larger images allows DSR-EI to further improve its performance. Tab.5.11 compares the results achieved when switching from 256×256 patches to the full resolution images of DIML and NYUv2 – i.e., 1344×756 and 640×480, respectively. We can notice consistent improvements, particularly when dealing with larger upsampling factors.

Method	Size	DIML			Size	NYUv2		
		4×	8×	16×		4×	8×	16×
DSR-EI$^+$	256×256	0.65 / **0.12**	2.09 / 0.22	6.31 / 0.50	256×256	2.75 / 0.47	11.8 / 1.09	47.1 / 2.40
	1344×756	**0.58 / 0.12**	**1.91 / 0.20**	**5.15 / 0.45**	640×480	**1.93 / 0.39**	**8.14 / 0.89**	**33.0 / 2.02**

Table 5.11: Results on NYUv2 and DIML dataset – different input sizes. We report MSE (cm^2) / MAE (cm), the lower the better.

5.3.5 Limitations

We conclude by listing a few limitations of DSR-EI. As previously pointed out, global context is crucial for it to achieve the best performance. When this is unavailable, some accuracy is lost when generalizing across datasets. Moreover, the significant improvements over existing methods are paid for in terms of time/memory requirements. Tab. 5.12 highlights the higher runtime and, more evidently, peak memory usage. Future work will aim at reducing the overhead, while minimizing the drop in accuracy.

	Params (M)	**Runtime (ms)**	**Memory Peak (GB)**
PMBANet [262]	35.5	26.9	3.07
FDSR [76]	0.67	1.03	2.05
JIIF [204]	10.8	89.8	2.36
DCTNet [287]	0.04	9.03	0.26
LGR [40]	32.5	26.4	0.19
Ours	25.0	51.5	18.6

Table 5.12: Computational requirements at inference. Experiments on Nvidia RTX 3090 GPU, with 256×256 input and $8\times$ factor.

5.4 Conclusion

We proposed DSR-EI, a depth super-resolution network, which includes a high-frequency extraction branch (HFEB) and a guided depth restoration branch (GDRB). Specifically, implemented as an efficient transformer, HFEB extracts explicit HF features. Then, GDRB deploys a two-stage encoder-decoder network to recover HR depth maps progressively, by adaptively fusing discriminative features while supplementing additional, implicit HF information. Exhaustive experiments demonstrate that DSR-EI sets a new state-of-the-art for guided depth super-resolution.

Chapter 6

Global Optimization for Consistent 3D Instant Reconstruction

The content of this chapter has been presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2023) - "GO-SLAM: Global Optimization for Consistent 3D Instant Reconstruction" [280].

6.1 Introduction

The demand for high-fidelity 3D object and scene reconstructions grows continuously in various fields, including robotics and augmented/virtual reality applications. Thus, faithfully representing objects and scenes in three-dimensional space is paramount to modeling them as continuous surfaces rather than discrete points. However, despite the significant advancements in 3D reconstruction techniques, obtaining high-quality representations in real-time without compromising accuracy and spatial resolution remains challenging. This fact is further demanding in online reconstruction scenarios, where handling camera motions and achieving real-time performance are critical.

Dense visual Simultaneous Localization and Mapping (SLAM) systems [38, 145, 183, 239, 249] have been introduced recently, enabling real-time, dense indoor scene reconstructions using RGB-D sensors. In particular, BundleFusion [38] is the first volumetric approach that focuses on globally consistent 3D reconstruction on large-scale scenes at a real-time rate. However, consumer depth sensors have a limited working range [37, 183] and could yield extremely noisy [187] measurements. These issues make the representation mapped by RGB-D SLAM suffer from blurring or over-smoothed geometric details, degrading the accuracy of pose estimation and reconstruction. In parallel, scene reconstruction from monocular imagery is emerging as a more convenient solution compared to RGB-D or LiDAR sensors. Camera sensors are lightweight, inexpensive, and represent the most straightforward configuration. Several deep-learning approaches [13, 36, 183, 208, 209, 211, 290] have advanced monocular 3D reconstruction. However, their surface representations – point cloud, surfel-based and volumetric representations – lack flexibility at shape extraction and thus inhibit high-fidelity reconstruction.

More recently, the advent of Neural Radiance Fields (NeRFs) also impacted dense visual SLAM, offering photometrically accurate 3D representations of the world. The implicit representation yielded by continuous radiance fields allows for high-quality rendering of both visible

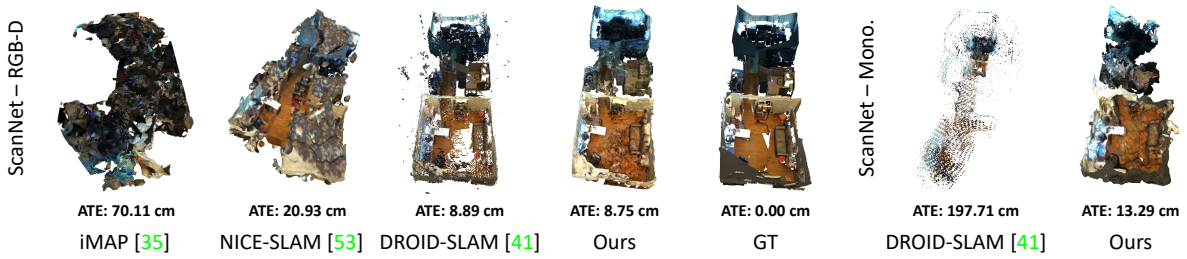


Figure 6.1: 3D Reconstruction and trajectory error on *scene0054_00* (ScanNet [37]). From left to right: RGB-D methods (iMAP [195], NICE-SLAM [295], DROID-SLAM [211] and ours), ground truth scan, and monocular methods (DROID-SLAM [211] and ours).

and occluded regions, enabling extraction of the underlying shapes at arbitrary resolution. Recent [81, 195, 295] and concurrent [33, 104, 173, 296] works demonstrate that NeRF-based visual SLAM can yield precise 3D reconstructions and camera pose estimation in small-scale scenes. However, due to the lack of global online optimization, such as loop closure (LC) and global bundle adjustment (BA), camera drift error accumulates as the number of processed frames grows, and the 3D reconstruction quickly collapses, as shown in Fig. 6.1.

Purposely, we introduce GO-SLAM, a deep-learning-based SLAM system featuring on-the-fly, globally consistent 3D reconstruction, facilitated by our robust camera tracking and real-time implicit surface updates. As real-world 3D scenes may be very complex, drifting cannot be completely avoided by only locally tracking camera motion, especially in the monocular camera setting, due to the lack of explicit depth measurements. In addition to the local registration commonly performed by current SLAM systems, we present an efficient loop closing to correct trajectory in real-time, accompanied by an online full BA module to actively optimize the 3D geometry of the complete keyframes history. In contrast to previous works performing BA or LC with sparse visual features [16, 36, 38, 142], our end-to-end global optimization procedure is naturally robust to challenging regions, thanks to richer features and geometry cues (*e.g.*, pixel-wise flow) learned by neural networks. Furthermore, GO-SLAM implements instant mapping based on a neural implicit network with multi-resolution hash encoding [140]. Its compact, multiscale representation enables updating the 3D reconstruction at high-frequency according to newly-optimized camera poses and depths from our global optimization system, thus ensuring global consistency in the dense map and capturing local details. Our contributions can be resumed as follows:

- A novel deep-learning-based, real-time global pose optimization system that considers the complete history of input frames and continuously aligns all poses.
- An efficient alignment strategy that enables instantaneous loop closures and correction of global structure, being both memory and time efficient.
- An instant 3D implicit reconstruction approach, enabling on-the-fly and continuous 3D model update with the latest global pose estimates. This strategy facilitates real-time 3D reconstructions.
- The first deep-learning architecture for joint robust pose estimation and dense 3D reconstruction suited for any setup: monocular, stereo, or RGB-D cameras.

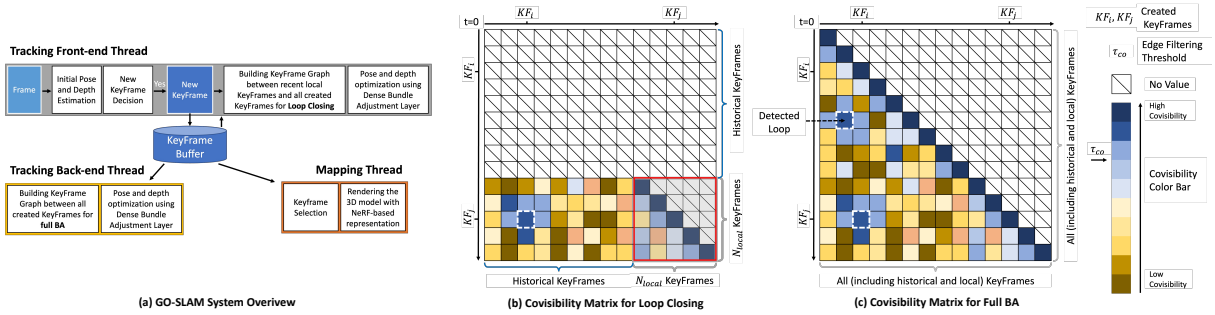


Figure 6.2: Architecture Overview. Our GO-SLAM framework consists of three parallel threads (a): front-end tracking (including keyframe initialization and loop closing), back-end tracking, and instant mapping. The front-end tracking thread uses the video stream as input and iteratively updates the pose and depth of the current frame while determining whether it should be promoted as a new keyframe. Moreover, it also actively performs efficient loop closing (b). The back-end tracking thread focuses on generating globally consistent pose and depth predictions through full bundle adjustment (c). Simultaneously, instant mapping updates the 3D reconstruction on-the-fly according to the latest geometry changes.

6.2 Method

Our GO-SLAM framework, depicted in Fig. 6.2, uses a keyframe-based SLAM paradigm to achieve real-time, globally-consistent 3D reconstruction. This is made possible by the online drift-corrected pose tracking and instant mapping capabilities of our system. By performing full bundle adjustment and loop closing, pose optimization can be carried out globally. Meanwhile, instant mapping adapts to continuous changes in optimized global poses and depths.

6.2.1 Tracking with Global Optimization

Loop closing and global bundle adjustment are crucial for robust pose estimation and long-term map consistency. In this work, we extend the tracking component of DROID-SLAM [211] by several key features. These enhancements effectively reduce the drift and pave the way for a globally optimal map. In the front-end tracking, we initialize a keyframe if sufficient motion is observed – there, we also introduce LC – while the back-end is equipped with full BA for online global refinement. Fig. 6.3 shows the effect of both LC and BA on tracking and 3D reconstruction qualitatively, correcting the large errors occurring in their absence.

Front-End Tracking. Our system takes as input a live video stream, which can be either monocular, stereo, or RGB-D, and applies a recurrent update operator based on RAFT [210] to compute the optical flow of each new frame compared to the last keyframe. If the average flow is larger than a predefined threshold τ_{flow}, a new keyframe is created out of the current frame and added to the maintained keyframe buffer for further refinement.

We use the set of keyframes $\{\mathbf{KF}_k\}_{k=1}^{N_{KF}}$ created so far to build a keyframe-graph $(\mathcal{V}, \mathcal{E})$ for performing LC. This process involves two steps: (1) select high co-visibility connections between the most recent N_{local} keyframes, and (2) detect loop closures between local keyframes and historical keyframes outside the local window. Accordingly, we compute a co-visibility

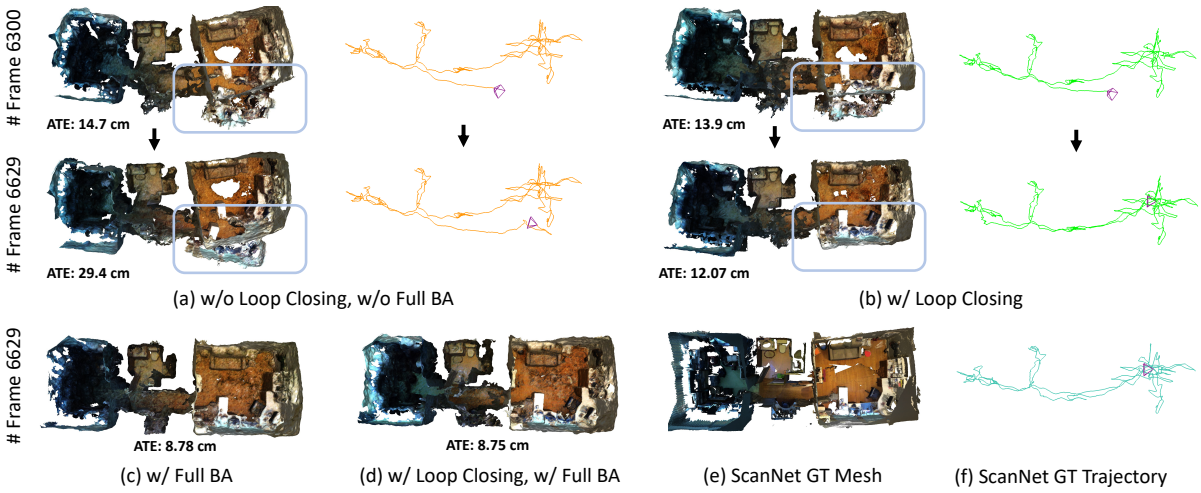


Figure 6.3: Qualitatives examples of LC and full BA on *scene0054_00* (ScanNet [37]) with a total of 6629 frames. In (a), a significant error accumulates when no global optimization is available. With loop closing (b), the system is able to eliminate the trajectory error using global geometry. Additionally, online full BA optimizes (c) the poses of all existing keyframes. The final model (d), which integrates both loop closing and full BA, achieves a more complete and accurate 3D model prediction.

matrix of size $N_{local} \times N_{KF}$ between N_{local} local keyframes and all the N_{KF} created keyframes to find valuable edge connections, as shown in Fig. 6.2 (b). In practice, the co-visibility is represented by the mean rigid flow between keyframe pairs using efficient back-projection, and those with low co-visibility, *i.e.*, mean flow larger than τ_{co}, are filtered out. Among local keyframes – the red-borders sub-matrix in Fig. 6.2 (b) – we build edges for keyframe pairs temporally adjacent or with high co-visibility. To avoid redundancy, once an edge connection (*e.g.*, $\text{KF}_i \leftrightarrow \text{KF}_j$) is added to the keyframe-graph, we suppress all possible neighboring edges between $\{\mathbf{KF}_k\}_{k=i-r_{local}}^{i+r_{local}}$ and $\{\mathbf{KF}_k\}_{k=j-r_{local}}^{j+r_{local}}$, where r_{local} is a hyper-parameter denoting a temporal radius. The loop detection step is quite similar. We sample edges from the unexplored part of the co-visibility matrix in descending order of co-visibility and suppress neighboring edges with radius r_{loop}. More strictly, to accept a loop candidate, we detect consecutively three loop candidates and validate them if their mean flow is lower than τ_{co}. The value of r_{loop} is determined empirically based on the observation that keyframes within a local window observe almost the same scene. We set r_{loop} to $\frac{N_{local}}{2}$, which allows for only one loop closure between the recent local region and one revisited region. In addition to the edge connections within local keyframes, several extra loop edges can be added to the keyframe graph depending on how many times the current local region is revisited. In general, the number of edges in the graph is linear to N_{local} with an upper-bound $N_{local} \times N_{local}$ + few loop closures. Through neighborhood suppression and co-visibility filtering, we further limit the number of edges in the keyframe-graph to $s_{edge} \cdot N_{local}$, so that the efficiency of optimization of the entire keyframe graph, *i.e.*, the whole front-end tracking, can be further ensured.

Afterward, we use the differentiable Dense Bundle Adjustment (DBA) layer proposed in [211] to solve a non-linear Least Squares optimization problem over the cost function to correct

the camera pose $\mathbf{G} \in SE(3)$ and inverse depth $\mathbf{d} \in \mathbb{R}_{+}^{H \times W}$ of each keyframe in the keyframe-graph:

$$\mathbf{E}(\mathbf{G}, \mathbf{d}) = \sum_{(i,j) \in \mathcal{E}} \left\| \mathbf{p}_{ij}^{*} - \Pi_c(\mathbf{G}_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}_i)) \right\|_{\Sigma_{ij}}^{2}, \tag{6.1}$$

where $(i, j) \in \mathcal{E}$ denotes any edge in keyframe-graph, Π_c and Π_c^{-1} are the projection and back-projection functions, $\mathbf{p}_i$ is the back-projected pixel position from keyframe $\mathbf{KF}_i$, $\mathbf{G}_{ij}$ is the pose transformation from $\mathbf{KF}_i$ to $\mathbf{KF}_j$, $\mathbf{p}_{ij}^{*}$ and $\mathbf{w}_{ij}$ are estimated flow and associated confidence map, $\Sigma_{ij} = \operatorname{diag} \mathbf{w}_{ij}$ and $\|\cdot\|_{\Sigma}$ is the Mahalanobis distance which weights the error terms based on the confidence weights $\mathbf{w}_{ij}$. The cost function allows the update of camera poses and dense per-pixel depth to maximize their compatibility with flow $\mathbf{p}_{ij}^{*}$ predicted by the recurrent update operator. For the sake of efficiency, we only compute the Jacobians with respect to the depths and poses of the local keyframes. After computing residuals and Jacobians at each iteration, a damped Gauss-Newton algorithm is applied to find the optimal poses and depths of all local keyframes.

Back-End Tracking. Simultaneously optimizing overall historical keyframes can be computationally expensive, as observed in [16, 142, 211]. To address this issue, we follow a similar approach as previous SLAM algorithms [16, 142] by running the full BA online in a separate thread, allowing the system to continue tracking new frames and loop closing. Similarly to the proposed front-end tracking, we start a new keyframe-graph and insert keyframe pairs with high co-visibility, as well as temporal adjacent keyframes, as shown in Fig. 6.2 (c). When a new edge is built, we suppress the redundant neighboring edges with radius r_{global}. As the trajectory error of the latest keyframes has been corrected with global geometry featured by loop closing, it eases the real-time requirement for the full BA. Our proposed full BA is efficient up to tens of thousands of input frames, as shown in Fig. 6.4.

6.2.2 Instant Mapping

The proposed instant mapping aims at updating the global 3D reconstruction in real-time by incorporating newly-optimized geometry from tracking. However, this goal presents two challenges for the mapping thread: i) ensuring that the updated reconstruction remains globally consistent, and ii) enabling fast rendering of the reconstructed scene. Updating all existing keyframes at once is the simplest approach to ensure global consistency, but it can quickly become impractical as the number of keyframes increases. Therefore, it is crucial to prune the keyframe candidates for updating selectively. Additionally, high-speed rendering of the scene is necessary to meet real-time requirements. To achieve these goals, we introduce our novel keyframe selection strategy and discuss our use of spatial hashing [140] and rendering networks in the remainder.

Keyframe Selection. At the beginning of each update of the 3D reconstruction, the instant mapping thread first takes a snapshot of all existing keyframe poses and depths tracked, to ensure that the geometry remains consistent during the mapping period. For keyframe selection, we prioritize those with the most relevant optimization updates, following the principle established by previous works [38]. Firstly, we ensure that the latest two keyframes and those not optimized by mapping are always included. In addition, following [38], we sort all keyframes in descending order of pose difference between the current and last updated state and select the top 10 keyframes from the sorted list when accessing. Furthermore, to prevent the mapping from

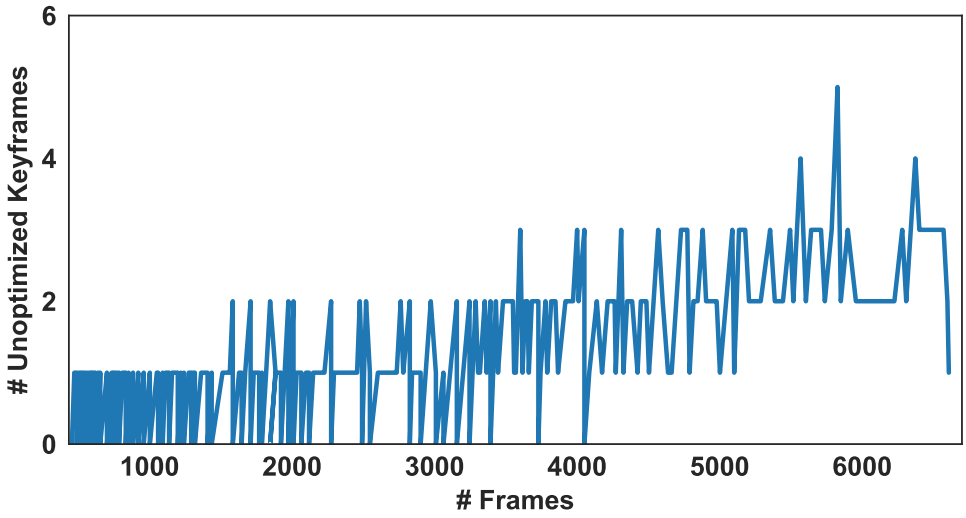


Figure 6.4: Number of unoptimized keyframes at each timestamp with full BA. Experiment on *scene0054_00* (ScanNet [37]) with 6629 frames (357 total keyframes).

forgetting previous 3D geometry, we also select 10 keyframes using a stratified sampling [139] from all the available keyframes.

Rendering. Drawing inspiration from the recent advancements in implicit neural network techniques [139] – Instant-NGP [140] in particular – we can construct 3D models from scratch with remarkable speed and accuracy. Specifically, given depth $\mathbf{D}$ (converted from inverse depth $\mathbf{d}$), pose $\mathbf{G}$, and image $\mathbf{I}$ for each selected keyframe, we randomly select M pixels for training. We then generate $N_{ray} = N_{strat}+N_{imp}$ total sampling points along the emitted ray crossing each pixel, where N_{strat} points are sampled using stratified sampling and N_{imp} points are selected near the depth value following [295]. For each 3D sampling point $\mathbf{x}$, we map it to multi-resolution hash encodings [140] $h_{\Theta_{hash}}(\mathbf{x})$ with trainable encoding parameters Θ_{hash} at each entry of the hash table. As the hash encodings $h_{\Theta_{hash}}(\mathbf{x})$ have explicitly stored the geometric and intensity information for each spatial position, we can predict a signed distance function (SDF) $\Phi(\mathbf{x})$ and color $\Omega(\mathbf{x})$ using shallow networks.

More specifically, our SDF network $f_{\Theta_{sdf}}$, which consists of a single multi-layer perceptron (MLP) with learnable parameters Θ_{sdf}, takes the point position $\mathbf{x}$ and corresponding hash encodings $h_{\Theta_{hash}}(\mathbf{x})$ as input and predicts the SDF as:

$$\Phi(\mathbf{x}),\ \mathbf{g} = f_{\Theta_{sdf}}(\mathbf{x}, h_{\Theta_{hash}}(\mathbf{x})), \tag{6.2}$$

with $\mathbf{g}$ being the learned geometry feature vector. The color network $f_{\Theta_{color}}$ processes $\mathbf{g}$, $\mathbf{x}$ and the gradient of SDF $\mathbf{n}$ with respect to $\mathbf{x}$ to estimate color $\Omega(\mathbf{x})$ as:

$$\Omega(\mathbf{x}) = f_{\Theta_{color}}(\mathbf{x}, \mathbf{n}, \mathbf{g}). \tag{6.3}$$

where Θ_{color} is the set of learnable parameters of the color network, i.e., a two-layer MLP as shown in Fig. 6.5.

The depth and color of each pixel/ray are calculated by unbiased volume rendering following NeuS [227]. Specifically, for point $\mathbf{x}_i,\ i \in \{1, \cdots, N_{ray}\}$ along a ray, given the camera center $\mathbf{o}$, view direction $\mathbf{v}$, and sampled depth D_i^{ray}, it can be formulated as $\mathbf{x}_i = \mathbf{o} + D_i^{ray}\mathbf{v}$. At the

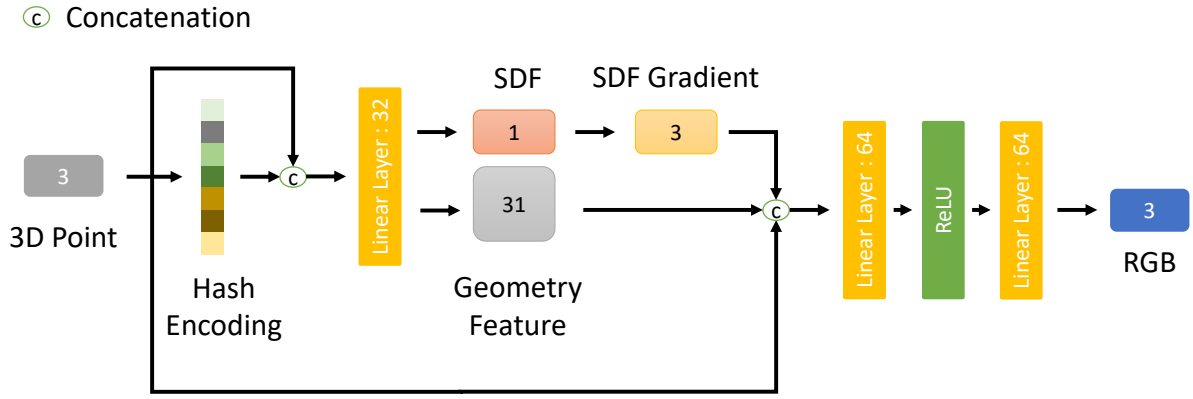


Figure 6.5: Details of rendering networks.

same time, the unbiased ray termination probability at this point is modeled as $w_i = \alpha_i \Pi_{j=1}^{i-1}(1-\alpha_j)$, where the opacity value α_i is computed as:

$$\alpha_i = \max\left(\frac{\sigma(\Phi(\mathbf{x_i})) - \sigma(\Phi(\mathbf{x_{i+1}}))}{\sigma(\Phi(\mathbf{x_i}))}, 0\right). \tag{6.4}$$

where σ is the modulated Sigmoid function [227]. With the weight w, the predictions of pixel-wise color $\hat{\mathbf{c}}$ and depth $\hat{\mathbf{D}}$ are accumulated along the ray:

$$\hat{\mathbf{c}} = \sum_{i=1}^{N_{ray}} w_i \Omega(\mathbf{x}_i), \quad \hat{\mathbf{D}} = \sum_{i=1}^{N_{ray}} w_i D_i^{ray}. \tag{6.5}$$

Training Losses. To optimize the rendering networks, taking keyframe image $\mathbf{I}$ and depth $\mathbf{D}$ as ground truth, the RGB and depth losses $\mathcal{L}_c$ and $\mathcal{L}_{dep}$ imposed on the selected M pixels are $\mathcal{L}_c = \frac{1}{M}\sum_{m=1}^{M} |\mathbf{c}_m - \hat{\mathbf{c}}_m|$, $\mathbf{c}_m \in \mathbf{I}$ and

$$\mathcal{L}_{dep} = \frac{1}{M}\sum_{m=1}^{M} \frac{|\mathbf{D}_m - \hat{\mathbf{D}}_m|}{\sqrt{\hat{\mathbf{D}}_m^{var}}}, \tag{6.6}$$

respectively, with $\hat{\mathbf{D}}_m^{var} = \sum_{i=1}^{N_{ray}} w_i (\hat{\mathbf{D}}_m - D_{m,i}^{ray})^2$ being the predicted depth variance used for down-weighting uncertain regions in the reconstructed geometry [195, 295]. Furthermore, we also introduce regularization to the predicted SDF, following [148, 225]. Specifically, to encourage the gradient of SDF to unit length, we adopt the Eikonal term [65]:

$$\mathcal{L}_{eik} = \frac{1}{M N_{ray}} \sum_{m,i} (1 - ||\mathbf{n}_{m,i}||)^2 \tag{6.7}$$

Besides, to supervise the SDF for accurate surface reconstructions, we approximate the ground truth SDF of sampling point $\mathbf{x}_i$ by computing its distance to the keyframe's depth $\mathbf{D}_m$, *i.e.*, $\mathbf{b}(\mathbf{x}_i) = \mathbf{D}_m - D_{m,i}^{ray}$. For SDF learning, we have $|\Phi(\mathbf{x}_i)| \leq |\mathbf{b}(\mathbf{x}_i)|, \forall \mathbf{x}_i$. To satisfy this bound, for near-surface points ($|\mathbf{b}(\mathbf{x}_i)| \leq \tau_{trunc}$, where τ_{trunc} is a hyper-parameter denoted the

truncation threshold, set to 16cm), the SDF loss is defined as $\mathcal{L}_{near} = |\Phi(\mathbf{x}_i) - \mathbf{b}(\mathbf{x}_i)|$, while for elsewhere, *i.e.*, free space, we apply a relaxed loss:

$$\mathcal{L}_{free} = \max\left(e^{-\beta\Phi(\mathbf{x}_i)} - 1, \Phi(\mathbf{x}_i) - \mathbf{b}(\mathbf{x}_i), 0\right) \tag{6.8}$$

where β is a hyper-parameter to apply a penalty when the predicted SDF $\Phi(\mathbf{x}_i)$ is negative in free space. Therefore, our full SDF loss is defined as:

$$\mathcal{L}_{sdf} = \frac{1}{MN_{ray}} \sum_{m,i} \begin{cases} \mathcal{L}_{near} & \textbf{if } |\mathbf{b}(\mathbf{x}_i)| \leq \tau_{trunc} \\ \mathcal{L}_{free} & \textbf{otherwise.} \end{cases} \tag{6.9}$$

Our instant mapping thread continuously optimizes the scene reconstruction over all sampled pixels of all selected keyframes. Specifically, for each set of selected keyframes, we run the mapping process for a fixed number N_{iter} of iterations. Our total loss is defined as:

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_{dep} \mathcal{L}_{dep} + \lambda_{eik} \mathcal{L}_{eik} + \lambda_{sdf} \mathcal{L}_{sdf}, \tag{6.10}$$

with $\mathcal{L}_c$, $\mathcal{L}_{dep}$, $\mathcal{L}_{eik}$, and $\mathcal{L}_{sdf}$ being loss balance weights. Given the pose and depth of each selected keyframe, our mapping thread directly uses them without refinement since our full BA and LC have exploited the global geometry to optimize both, being naturally robust in handling occluded regions and back faces.

6.3 Experimental Results

This section investigates our experimental evaluation, including implementation details, datasets, and key findings.

6.3.1 Implementation Details

Our system runs on a PC with a 3.5GHz Intel Core i9-10920X CPU and an NVIDIA RTX 3090 GPU. For tracking, we utilize pre-trained weights from DROID-SLAM [211], whereas the rendering networks are trained from scratch. The experiments are performed with the following default settings unless otherwise specified: local window size $N_{local} = 25$ for RGB-D and stereo input, $N_{local} = 50$ for monocular mode, neighboring radius $r_{local} = 1$, $r_{global} = 5$, co-visibility threshold $\tau_{co} = 25.0$, the factor used to constrain the maximum allowed edge $s_{edge} = 8$, sampling points along a ray $N_{strat} = 24$, $N_{imp} = 48$, pixel samples $M = 200$, penalty parameter $\beta = 5.0$, iterations $N_{iter} = 2$, and the loss weights $\lambda_c = 1.0$, $\lambda_{dep} = 1.0$, $\lambda_{eik} = 0.1$, $\lambda_{sdf} = 1.0$. The mesh reconstruction of a scene is achieved by running marching cubes on the SDF values of the queried points. The supplementary material provides more details about SLAM configuration and also qualitative results on various datasets.

6.3.2 Datasets

We evaluate our GO-SLAM system on several datasets with different input modalities, including the TUM RGB-D, EuRoC, ETH3D-SLAM, ScanNet, and Replica. The TUM RGB-D benchmark [193] is a small-scale indoor dataset with accurate ground truth obtained from an

	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	avg
ORB-SLAM2 [142]	✗	0.071	✗	0.023	✗	✗	✗	✗	0.010	-
ORB-SLAM3 [16]	✗	0.017	0.210	✗	0.034	✗	✗	✗	**0.009**	-
DeepV2D [209]	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
DeepFactors [36]	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
DROID-SLAM [211]	0.111	0.018	0.042	**0.021**	**0.016**	0.049	0.026	**0.048**	0.012	0.038
Ours	**0.089**	**0.016**	**0.028**	0.025	0.026	0.052	**0.019**	**0.048**	0.010	**0.035**

	fr1/desk	fr2/xyz	fr3/office
Kintinuous[145]	0.037	0.029	0.030
BAD-SLAM[183]	0.017	0.011	0.017
ORB-SLAM2[142]	0.016	**0.004**	**0.010**
iMAP [195]	0.049	0.020	0.058
NICE-SLAM [295]	0.027	0.018	0.030
GO-SLAM (ours)	**0.015**	0.006	0.013

Table 6.1: ATE[m] on the TUM RGB-D [193] benchmark. The left table shows the results of **monocular** SLAM on sequences from the freiburg1 set. The right one reports the accuracy for **RGB-D** SLAM on sequences from freiburg1, freiburg2 and freiburg3 respectively. '✗' denotes tracking failure, '-' no available data. Results of monocular SLAM [16, 36, 142, 209, 211] and RGB-D SLAM [142, 145, 183, 195, 295] are taken from [211] and [295] respectively.

external camera motion capture system. The EuRoC dataset [14] contains 11 indoor stereo sequences recorded from a micro aerial vehicle (MAV). The ETH3D-SLAM dataset [183] provides real-world RGB-D image sequences captured with synchronized global shutter cameras. The ScanNet dataset [37] features richly annotated RGB-D scans of real-world environments, including challenging short and long trajectories. Finally, the Replica dataset [192] provides high-fidelity 3D models of photo-realistic indoor scenes, enabling us to assess the reconstruction performance of our approach. For TUM RGB-D, EuRoC, and ETH3D-SLAM, images are resized to 384×512 resolution, while 240×320 and 320×640 for ScanNet and Replica datasets, respectively.

6.3.3 Evaluation Metrics

Following the common protocol in the SLAM literature [142, 211, 295], we evaluate the estimated trajectory by aligning it to the ground truth and then calculating the camera pose accuracy based on the Absolute Trajectory Error (ATE) RMSE. The reconstruction metrics include *Accuracy* [cm], *Completion* [cm], *Completion Ratio* [< 5cm %], and F-score [< 5cm %]. For the evaluation, we remove regions not observed by any camera. Furthermore, using ground truth trajectory for depth rendering, we evaluate the Depth L1 metric [295] by computing the absolute error between rendered depths from estimated and ground truth meshes.

6.3.4 Comparison with state-of-the-art SLAM

Here, we evaluate GO-SLAM in synthetic and real-world scenarios and compare it to state-of-the-art SLAM systems in monocular, stereo, and RGB-D settings.

TUM RGB-D (Monocular & RGB-D). In this dataset, we compare with monocular and RGBD SLAM systems. In Tab. 6.1 (left), we focus on the former methods: traditional SLAM [16, 142] with point-based representation fails on camera tracking on most of the challenging sequences. Among all deep-learning-based methods [36, 209, 211], GO-SLAM with online LC and full BA achieves the lowest average trajectory error. Furthermore, following [295], we also evaluate our method on RGB-D input. Tab. 6.1 (right) illustrates the clear superiority of our pose estimation compared to recent NeRF-base RGB-D SLAM [195, 295], effectively shrinking the gap with traditional SLAM systems.

	MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg
ORB-SLAM2 [142]	0.035	0.018	0.028	0.119	0.060	**0.035**	0.020	0.048	0.037	0.035	-	-
SVO [53]	0.040	0.070	0.270	0.170	0.120	0.040	0.040	0.070	0.050	0.090	0.790	0.159
ORB-SLAM3 [16]	0.029	0.019	0.024	0.085	0.052	**0.035**	0.025	0.061	0.041	0.028	0.521	0.084
DROID-SLAM [211]	**0.015**	**0.013**	0.035	0.048	**0.040**	0.037	**0.011**	**0.020**	0.018	0.015	**0.017**	**0.024**
Ours	0.016	0.014	**0.023**	**0.045**	0.045	0.037	**0.011**	0.023	**0.016**	**0.010**	0.022	**0.024**

	V103	V202	V203
ORB-SLAM3 [16]	0.037	0.022	✗
DSO [50]	0.903	0.132	1.152
DROID-SLAM [211]	0.020	0.013	**0.014**
Li *et al.* [104]	✗	0.178	✗
GO-SLAM (ours)	**0.018**	**0.011**	0.017

Table 6.2: ATE [m] on the EuRoC dataset [14]. In the left table, the results of all methods were obtained by running them on **stereo** video. In the right table, we report the trajectory error of **monocular** SLAM. '✗' denotes tracking failure. Results of [16, 50, 53, 142, 211] and [104] are adopted from [211] and [104] respectively.

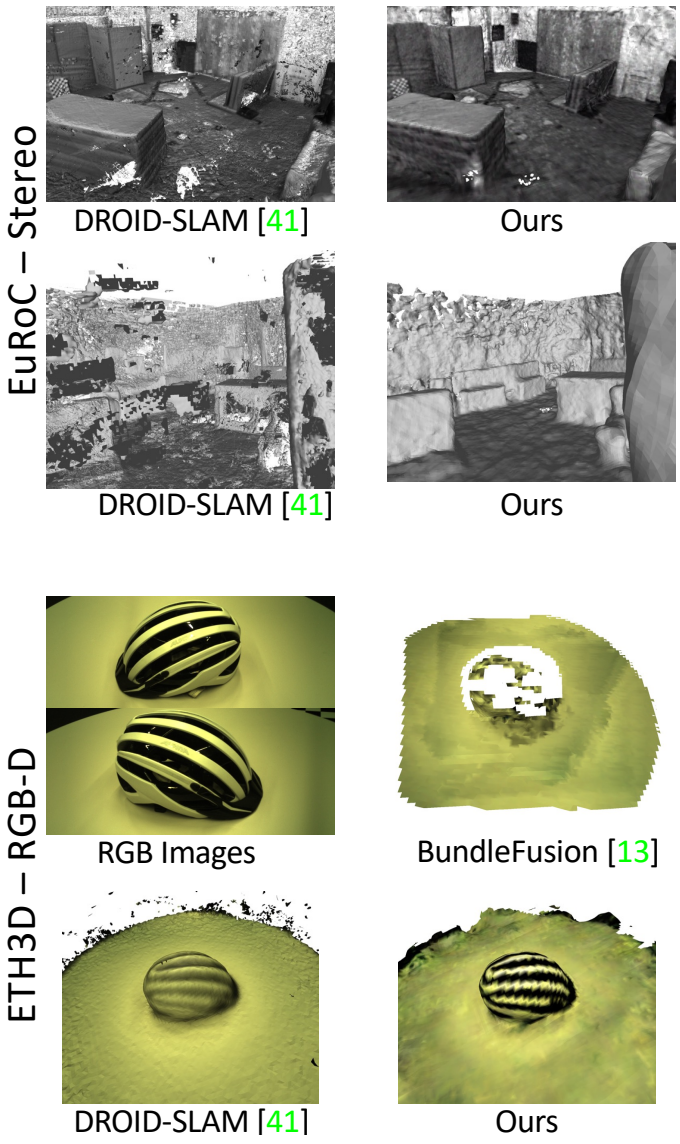


Figure 6.6: Qualitative results. Top: V1_02_Medium (EuRoC), bottom: Helmet (ETH3D). DROID-SLAM [211] reconstructions obtained with TSDF-Fusion [35].

EuRoC (Stereo & Monocular). Tab. 6.2 (left) shows the camera trajectory error of GO-SLAM for all stereo sequences compared to stereo SLAM methods. While existing systems based on neural implicit representation are designed for monocular [33, 104, 173, 296] or RGB-D [195, 295] input only, our method predicts trajectories comparable to state-of-the-art stereo SLAM [16, 53, 142, 211], while also producing globally dense and consistent 3D reconstruction, as shown in Fig. 6.6. Compared to the noisy result of DROID-SLAM with several holes and floating points, GO-SLAM produces a more complete, smoother surface and a cleaner reconstruction. In the Tab. 6.2 (right), we also report the results of monocular SLAM. Specifically, NeRF-based SLAM [104] without online BA fails in most cases, while our approach gives accurate predictions in all sequences.

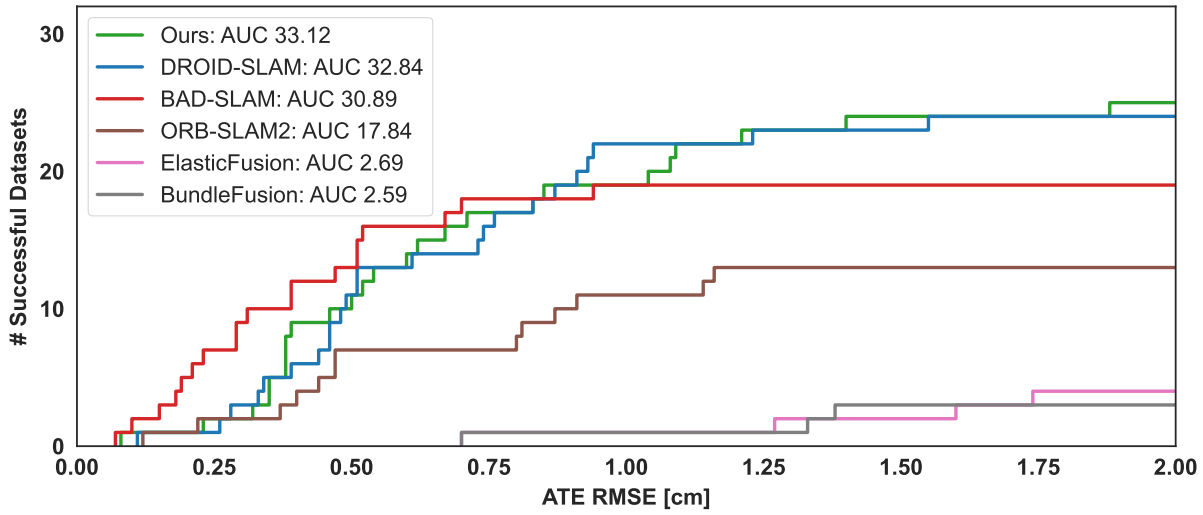


Figure 6.7: RGB-D ETH3D-SLAM test benchmark. Number of successful trajectories (y-axis) as a function of the ATE RMSE (x-axis). Maximum ATE RMSE: 2cm. Results by published SLAM systems [38, 142, 183, 211, 239] and ours.

ETH3D-SLAM (RGB-D). The ETH3D-SLAM dataset provides a public online leaderboard for RGB-D SLAM evaluation. The results in Fig. 6.7 demonstrate the significant advantage of our deep-learning-based method over published RGB-D SLAM systems with point- [142], surfel- [183, 239] or voxel-based [38] representations. Similarly to our approach, BundleFusion [38] targets globally consistent reconstruction. However, its pose estimation is highly susceptible to errors, resulting in inferior reconstruction as shown in Fig. 6.6, whereas our method yields smoother reconstruction with photometrically convincing rendering.

ScanNet (Monocular & RGB-D). For an exhaustive evaluation, we test SLAM methods on both short (sequences with less than 5000 frames) and long, complex sequences. Tab. 6.3 shows that, although DROID-SLAM [211] performs well on short sequences and RGB-D inputs, its accuracy drops dramatically when processing longer sequences in the monocular setting, even performing a final global bundle adjustment. In contrast, GO-SLAM consistently yields the best results, thanks to online loop closing and full BA. Furthermore, as anticipated in Fig. 6.1, due to the absence of global optimization to eliminate the accumulated trajectory error, NeRF-based SLAM, *e.g.*, iMAP, and NICE-SLAM [295] provide significantly poorer results for pose estimation and 3D reconstruction.

	Scene ID	0000	0054	0233	0465	0059	0106	0169	0181	Avg.
	# Frames	5578	6629	7643	6306	1807	2324	2034	2349	
RGB-D	iMAP* [195]	55.95	70.11	86.42	85.03	32.06	17.50	70.51	32.10	56.21
	NICE-SLAM [295]	8.64	20.93	9.00	22.31	12.25	8.09	10.28	12.93	13.05
	DROID-SLAM [211] (VO)	8.00	29.28	6.75	11.37	11.30	9.97	8.64	7.38	11.59
	DROID-SLAM [211]	5.36	8.89	4.90	8.32	7.72	7.06	8.01	6.97	7.15
	Ours	**5.35**	**8.75**	**4.78**	**8.15**	**7.52**	**7.03**	**7.74**	**6.84**	**7.02**
Mono.	ORB-SLAM3 [16]	73.93	243.26	25.01	181.86	90.67	178.13	60.15	104.93	119.74
	DROID-SLAM [211] (VO)	11.05	204.31	71.08	117.84	67.26	11.20	16.21	9.94	63.61
	DROID-SLAM [211]	**5.48**	197.71	72.23	114.36	9.00	**6.76**	**7.86**	**7.41**	52.60
	Ours	5.94	**13.29**	**5.31**	**79.51**	**8.27**	8.07	8.42	8.29	**17.59**

Table 6.3: ATE[cm] on ScanNet dataset [37]. For DROID-SLAM [211], we also report results for the visual odometry (VO) variant, which does not include the final global bundle adjustment. Results of iMAP* and NICE-SLAM are from [295].

	RGB-D			Mono.					
	iMAP* [195]	NICE-SLAM [295]	**Ours**	Orbeez-SLAM [33]	NeRF-SLAM [173]‡	DROID-SLAM [211]	Li *et al.* [104]	NICER-SLAM [296] ‡	**Ours**
ATE RMSE[cm] ↓	-	1.95	**0.34**	-	-	0.42	0.46	1.88	**0.39**
Depth L1[cm] ↓	7.64	3.53	**3.38**	11.88	4.49	-	-	-	**4.39**
Acc.[cm] ↓	6.95	2.85	**2.50**	-	-	5.03	4.03	**3.65**	3.81
Comp.[cm] ↓	5.33	**3.00**	3.74	-	-	8.49	4.20	**4.16**	4.79
Comp. Ratio[<5cm %] ↓	66.60	89.33	88.09	-	-	64.72	**79.60**	79.37	78.00
Avg. FPS ↑	**10**	$\ll 1$	8	≈ 20	10	**21**	3	$\ll 1$	8

Table 6.4: Reconstruction results and ATE[cm] on the Replica dataset (average over 8 scenes). The results of iMAP* [195] are adopted from NICE-SLAM [295], while results for other methods are taken from the respective original papers. ‡ denotes concurrent works yet unpublished.

Replica (Monocular & RGB-D). Finally, Tab. 6.4 shows pose and reconstruction performance achieved by GO-SLAM on the Replica dataset [192]. Our system achieves comparable accuracy with respect to existing RGB-D methods [295] and concurrent monocular [104, 296] ones. However, none of the latter runs in real-time, whereas GO-SLAM achieves a frame rate of 8 FPS with maximum GPU memory consuming 18 GB. Although iMAP [195] alone can achieve a similar speed as GO-SLAM (10 FPS), it yields much worse results. In particular, close to ours, NeRF-SLAM [173] is a concurrent, monocular SLAM system based on DROID-SLAM [211], running at nearly 10 FPS. However, their system lacks BA, which GO-SLAM provides. Moreover, NeRF-SLAM and Orbeez-SLAM [33] lack dense 3D reconstruction results, limiting the comparison. We can only compare on depth rendering quality, for which GO-SLAM achieves better results.

6.3.5 Ablation Study

We conclude by studying the impact of the novel designs in the proposed GO-SLAM with RGB-D input.

Keyframe Selection. In Tab. 6.5, we investigate the impact of different keyframe selection strategies by computing the mean F-score in eight sequences of the Replica dataset [192]. The experimental results prove that updating the 3D model with the latest keyframes or stratified sampling is not sufficient to obtain the best results. Monitoring the pose and depth of each frame continuously and updating the 3D model according to the largest change is crucial for

globally consistent reconstruction.

Sampling Strategy			Avg.
Latest	Stratified	Top-Ranked	F-score ↑
✓	✗	✗	49.55
✓	✓	✗	83.13
✓	✓	✓	**85.56**

Table 6.5: Impact of keyframe selection. We report average F-score achieved by different sampling strategies.

$\mathcal{L}_c$	$\mathcal{L}_{dep}$	$\mathcal{L}_{sdf}$	$\mathcal{L}_{eik}$	Avg. F-score ↑
✓	✗	✗	✗	34.64
✓	✓	✗	✗	83.50
✓	✗	✓	✗	84.00
✓	✓	✓	✗	84.83
✓	✓	✓	✓	**85.56**

Table 6.6: Impact of single losses. We report average F-score achieved with different losses configurations.

Skipping Frames	Speedup	Avg. F-score ↑	Avg. ATE RMSE [cm] ↓
None	$1\times$	**85.56**	**7.02**
1/2	$2\times$	84.67	7.08
3/4	$4\times$	84.80	7.16
7/8	$8\times$	84.41	7.28

Table 6.7: Impact of frames skipping. We report average F-score and ATE when running in real-time.

Losses. Tab. 6.6 evaluates on Replica the effectiveness of each loss term. Results show that the RGB loss alone cannot lead to satisfying results. Geometric supervision, such as depth or SDF loss, provides similar accuracy while integrating all terms produces the best results.

Skipping frames to run in real-time. Tab. 6.7 shows our GO-SLAM accuracy when skipping RGB-D frames to run at $2\times, 4\times, 8\times$ speed. Remarkably, both mapping accuracy (avg. F-score on Replica [192]) and tracking performance (avg. ATE RMSE on ScanNet [37]) only experience minimal degradation, proving that GO-SLAM is 1) robust to large view changes and 2) ready for real-time usage.

Loop Closing and Full BA. Finally, we study the impact of our efficient loop closing and online full BA on eight scenes of ScanNet [37]. Our baseline DROID-SLAM (VO) [211] (without LC and Full BA) achieves high speed but suffers from a large trajectory error, as shown in

	Avg. ATE RMSE [cm] ↓	avg. FPS ↑
w/o LC & w/o Full BA	11.59	**30**
w/ LC	8.83	20
w/ Full BA	7.11	12
w/ LC & w/ Full BA	**7.02**	10

Table 6.8: Impact of loop closing and full BA. We report average ATE and FPS with/without our main contributions.

Tab. 6.8. Our efficient LC significantly reduces drift with negligible speed reduction. Introducing Full BA slows down GO-SLAM, but improves global pose estimation significantly. Finally, our full system integrating LC and full BA achieves the best results in pose estimation and 3D reconstruction (see Fig. 6.3), while enabling real-time performance.

6.3.6 CPU/GPU Requirements.

We conclude by measuring hardware requirements on the Replica dataset [192]. In Tab. 6.9 we present several metrics including CPU requirements, maximum GPU memory consumption, average frames per second, and final accuracy, showcasing the performance of various SLAM algorithms during the reconstruction of an entire scene. Specifically, our SLAM system stands out by achieving an optimal balance between CPU/GPU requirements and SLAM performance, combining high accuracy with speed.

Method	CPU Processing Frequency [GHz]	GPU Consuming Memory [G]↓	Avg. FPS↑	Comp. Ratio[<5cm %] ↑
iMAP* [195]	3.80	**10.13**	10	66.60
NICE-SLAM [295]	3.80	11.72	$\ll 1$	**89.33**
DROID-SLAM [211]	3.50	14.34	**21**	70.52
Ours	3.50	15.63	8	88.09

Table 6.9: Hardware requirements and performance. All results are obtained by running on NVIDIA RTX 3090 GPU and Replica dataset [192] with RGB-D input.

6.3.7 More Qualitative Results

We provide detailed results on ScanNet [37], EuRoC [14] and Replica [192] datasets on Figs. 6.8 to 6.10 respectively.

6.4 Conclusions

We introduced a novel, real-time deep-learning-based SLAM algorithm that achieves globally consistent reconstruction with monocular, stereo, or RGB-D input. Our approach explicitly detects loop closures and performs online full BA to minimize trajectory error. Based on

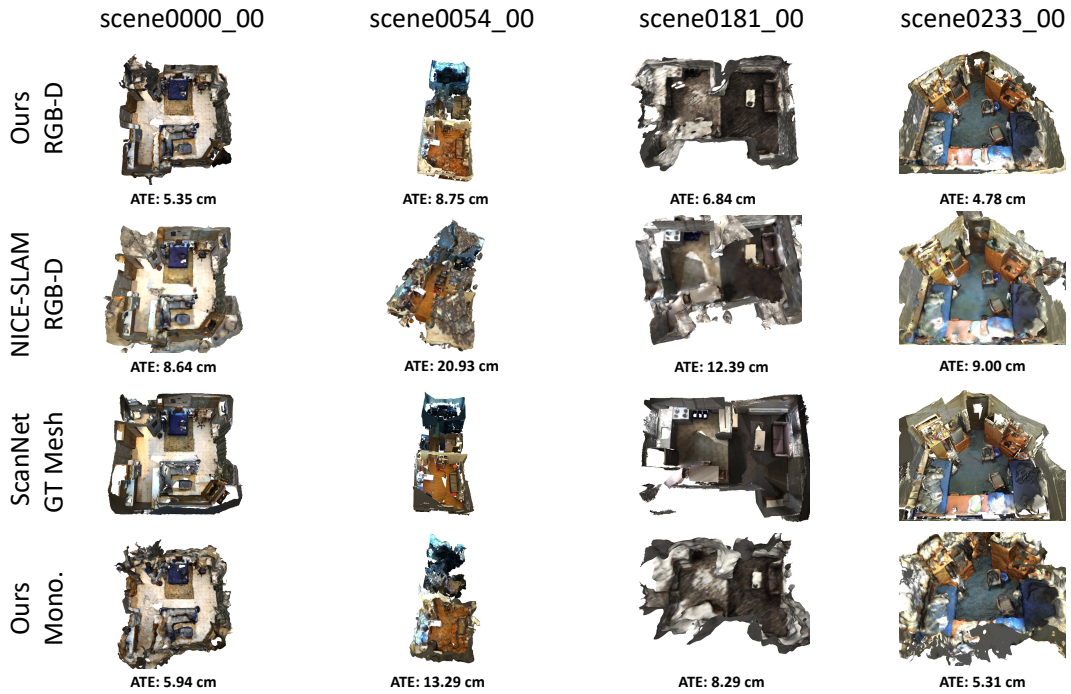


Figure 6.8: Qualitative results on ScanNet dataset [192]. Our GO-SLAM predicts globally consistent 3D reconstruction with monocular or RGB-D input at real-time speed. While NICE-SLAM [295] suffers from accumulated errors in trajectory and distortion in 3D models.

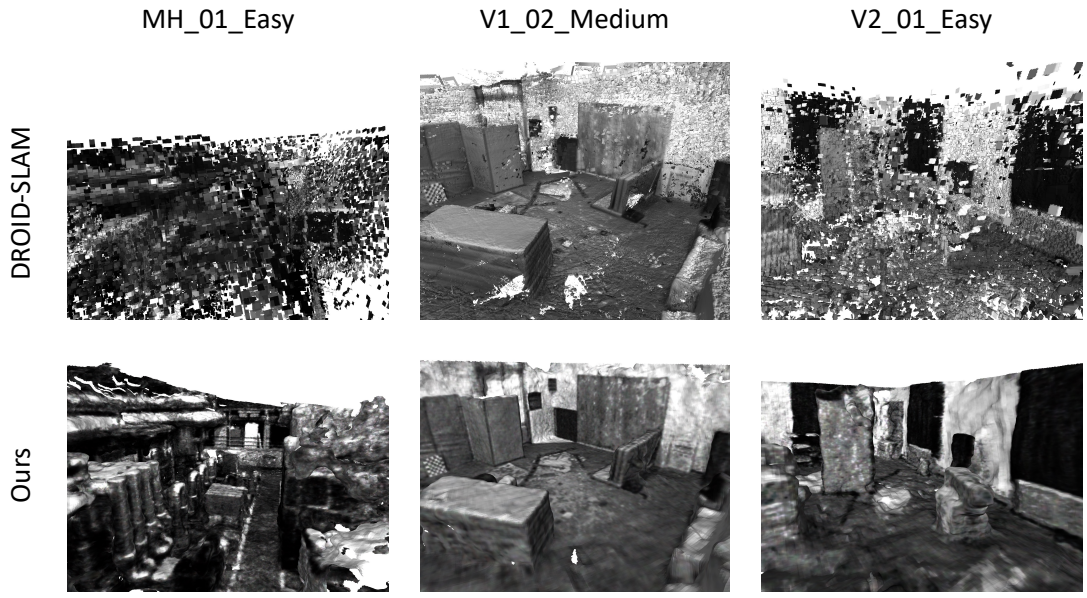


Figure 6.9: Qualitative results on EuRoC dataset [14]. DROID-SLAM [211] reconstructions obtained with TSDF-Fusion [35]. Compared to DROID-SLAM, our reconstructions are much cleaner and more complete.

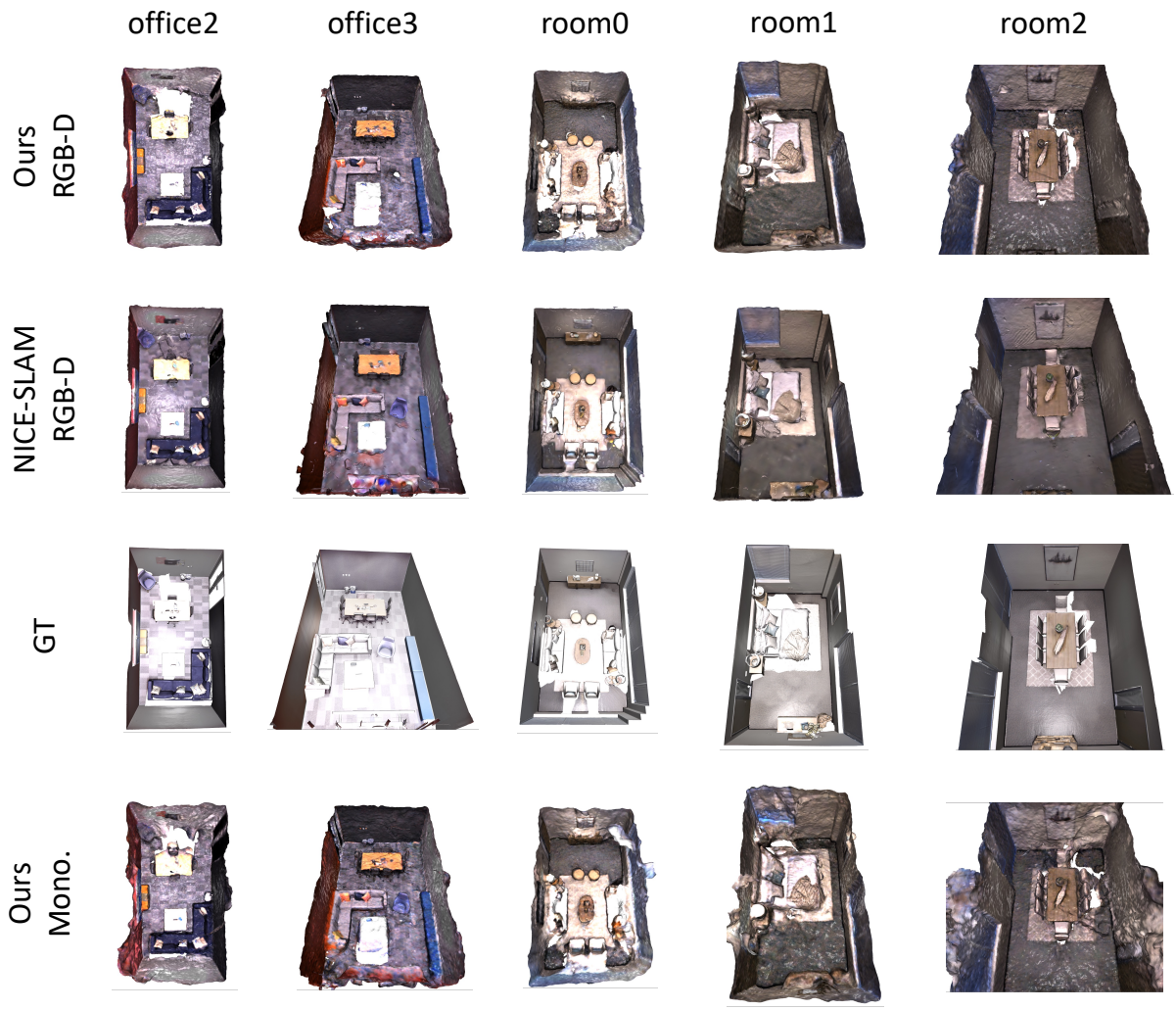


Figure 6.10: Qualitative results on Replica dataset [192]. Our GO-SLAM predicts dense 3D reconstruction with monocular or RGB-D input at real-time speed. While NICE-SLAM [295], which can only work with depth input, takes hours ($\ll$ 1 FPS) to get the reconstruction.

NeRF, our 3D reconstruction provides an efficient, compact, and multi-resolution representation. Moreover, our approach continuously updates the dense 3D reconstruction to adapt to the newly-optimized global geometry at high frequency. Our experiments demonstrate the algorithm's robustness in reliably tracking and densely mapping even in large-scale scenes, especially on long monocular trajectories with no depth information, achieving state-of-the-art performance on various datasets.

Conclusion

6.5 Summary of Thesis Achievements

In this thesis, innovative techniques rooted in deep learning are introduced to enhance the practical depth associated with the 3D reconstruction task from depth sensors and/or images.

The most straightforward method for obtaining depth information involves capturing raw depth points with various depth sensors. To address issues such as holes and noisy points, we present a depth completion network, *i.e.*, CompletionFormer, to reconstruct sparse depth measurements into a dense and complete depth map for downstream applications. In scenarios where depth cues are unavailable, a Vision Transformer-based unsupervised depth estimation network is constructed. This network relies on image reconstruction loss for depth estimation, leveraging global context information to handle challenging regions, including occlusions, dynamic objects, and photometric changes. To alleviate the challenges of accurate pose estimation and address the scale ambiguity in monocular depth estimation, a fast stereo matching network is proposed for metric depth estimation. The developed TemporalStereo stands out as the first supervised stereo network capable of seamlessly caching past context and utilizing it to enhance ongoing predictions, especially in occluded and reflective regions. Moreover, constrained by computational costs or physical device limitations, the captured or predicted depth is often presented in low resolution. To enhance its practical applications in scenarios demanding high-resolution 3D perception, we introduce an image-guided depth super-resolution network for accurate high-resolution depth prediction. The essence of this approach lies in devising an efficient method to extract both explicit and implicit high-frequency information from the guided high-resolution image, contributing to superior depth super-resolution results. At the end, we present a NeRF-based SLAM system capable of incorporating diverse depth resources, such as depth from sensors, monocular depth estimation, and stereo matching, to achieve globally consistent 3D reconstruction. This inclusive approach broadens the applicability of our research to a diverse array of scenarios and showcases promising outcomes in advancing the limits of depth estimation for practical use.

6.6 Future Work

The next step to move forward involves high-fidelity 3D reconstruction. The reconstructed surfaces yield valuable structural information for various downstream applications, including the generation of 3D assets for augmented/virtual/mixed reality or the mapping of environments for the autonomous navigation of robotics. Of notable significance is the photogrammetric surface reconstruction utilizing a monocular RGB camera. This approach holds particular appeal, as

it empowers users to casually create digital twins of the real world using ubiquitous mobile devices. To attain efficient and high-fidelity 3D reconstruction, there are still several under-explored directions. The first focal point is accurate pose estimation. Deep learning-based SLAM, particularly methods rooted in NeRF and Gaussian Splatting, demonstrate significant potential in this regard. Additionally, incorporating geometry priors, such as depth and semantic segmentation, becomes crucial for enhancing the robustness and efficiency of the 3D reconstruction network.

Bibliography

[1] F. Aleotti, M. Poggi, F. Tosi, and S. Mattoccia. Learning end-to-end scene flow by distilling single tasks knowledge. In *AAAI*, 2020.

[2] L. Andraghetti, P. Myriokefalitakis, P. L. Dovesi, B. Luque, M. Poggi, A. Pieropan, and S. Mattoccia. Enhancing self-supervised monocular depth estimation with traditional visual odometry. In *3DV*, 2019.

[3] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies. Neural rgb-d surface reconstruction. In *CVPR*, pages 6290–6301, 2022.

[4] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] A. Badki, A. Troccoli, K. Kim, J. Kautz, P. Sen, and O. Gallo. Bi3d: Stereo depth estimation via binary classifications. In *CVPR*, pages 1600–1608, 2020.

[6] J. Bae, S. Moon, and S. Im. Monoformer: Towards generalization of self-supervised monocular depth estimation with transformers. *arXiv preprint arXiv:2205.11083*, 2022.

[7] C. Bamji, J. Godbaz, M. Oh, S. Mehta, A. Payne, S. Ortiz, S. Nagaraja, T. Perry, and B. Thompson. A review of indirect time-of-flight technologies. *IEEE Transactions on Electron Devices*, 2022.

[8] A. Bangunharcana, J. W. Cho, S. Lee, I. S. Kweon, K.-S. Kim, and S. Kim. Correlate-and-excite: Real-time stereo matching via guided cost volume excitation. In *IROS*, 2021.

[9] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM TOG*, 28(3):24, 2009.

[10] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5855–5864, 2021.

[11] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pages 5470–5479, 2022.

[12] S. F. Bhat, I. Alhashim, and P. Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, pages 4009–4018, 2021.

[13] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *CVPR*, pages 2560–2568, 2018.

[14] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The euroc micro aerial vehicle datasets. *IJRR*, 35(10):1157–1163, 2016.

[15] Q. Cai, D. Gallup, C. Zhang, and Z. Zhang. 3d deformable face tracking with a commodity depth camera. In *ECCV*, pages 229–242. Springer, 2010.

[16] C. Campos, R. Elvira, J. J. Gómez, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM, 2020.

[17] C.-C. Chang, C.-C. Lin, C.-S. Tseng, and W.-L. Tai. Reversible hiding in dct-based compressed images. *Information Sciences*, 177(13):2768–2786, 2007.

[18] D. Chang, A. Božič, T. Zhang, Q. Yan, Y. Chen, S. Süsstrunk, and M. Nießner. Rc-mvsnet: unsupervised multi-view stereo with neural rendering. In *ECCV*, pages 665–680. Springer, 2022.

[19] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *CVPR*, pages 5410–5418, 2018.

[20] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, pages 14124–14133, 2021.

[21] X. Chen, K.-Y. Lin, C. Qian, G. Zeng, and H. Li. 3d sketch-aware semantic scene completion via semi-supervised structure prior. In *CVPR*, pages 4193–4202, 2020.

[22] X. Chen, Y. Xing, and G. Zeng. Real-time semantic scene completion via feature aggregation and conditioned prediction. In *ICIP*, pages 2830–2834. IEEE, 2020.

[23] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun. Cascaded pyramid network for multi-person pose estimation. In *CVPR*, pages 7103–7112, 2018.

[24] Y. Chen, C. Schmid, and C. Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*, 2019.

[25] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu. Dynamic convolution: Attention over convolution kernels. In *CVPR*, pages 11030–11039, 2020.

[26] Z. Chen, R. Cong, Q. Xu, and Q. Huang. Dpanet: Depth potentiality-aware gated attention network for rgb-d salient object detection. *TIP*, 30:7012–7024, 2021. doi: 10.1109/TIP.2020.3028289.

[27] X. Cheng, P. Wang, and R. Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *ECCV*, pages 103–119, 2018.

[28] X. Cheng, P. Wang, C. Guan, and R. Yang. Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. In *AAAI*, volume 34, pages 10615–10622, 2020.

[29] X. Cheng, Y. Zhong, M. Harandi, Y. Dai, X. Chang, T. Drummond, H. Li, and Z. Ge. Hierarchical neural architecture search for deep stereo matching. *NeurIPS*, 2020.

[30] J. Cho, D. Min, Y. Kim, and K. Sohn. Deep monocular depth estimation leveraging a large-scale outdoor stereo dataset. *Expert Systems with Applications*, 178:114877, 2021.

[31] H. Choi, H. Lee, S. Kim, S. Kim, S. Kim, K. Sohn, and D. Min. Adaptive confidence thresholding for monocular depth estimation. In *ICCV*, pages 12808–12818, October 2021.

[32] J. Choi, D. Jung, D. Lee, and C. Kim. Safenet: Self-supervised monocular depth estimation with semantic-aware feature extraction. In *NeurIPS*, 2020.

[33] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, X.-Q. Shi, Y.-H. Hua, J.-F. Yeh, W.-C. Chen, Y.-T. Chen, and W. H. Hsu. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. *arXiv preprint arXiv:2209.13274*, 2022.

[34] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016.

[35] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, pages 303–312, 1996.

[36] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE RAL*, 5(2):721–728, 2020.

[37] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017.

[38] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM TOG*, 36(4):1, 2017.

[39] Z. Dai, B. Cai, Y. Lin, and J. Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *CVPR*, pages 1601–1610, 2021.

[40] R. de Lutio, A. Becker, S. D'Aronco, S. Russo, J. D. Wegner, and K. Schindler. Learning graph regularisation for guided super-resolution. In *CVPR*, pages 1979–1988, 2022.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, Miami, FL, 2009. IEEE.

[42] X. Deng and P. L. Dragotti. Deep convolutional neural network for multi-modal image restoration and fusion. *PAMI*, 43(10):3333–3348, 2020.

[43] J. Diebel and S. Thrun. An application of markov random fields to range sensing. *NeurIPS*, 18, 2005.

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[45] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv preprint arXiv:2010.11929*, 2020.

[46] S. Duggal, S. Wang, W.-C. Ma, R. Hu, and R. Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *ICCV*, 2019.

[47] G. Egnal and R. P. Wildes. Detecting binocular half-occlusions: Empirical comparisons of five approaches. *IEEE TPAMI*, 24(8):1127–1133, 2002.

[48] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *NeurIPS*, 27, 2014.

[49] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *NeurIPS*, 27, 2014.

[50] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE TPAMI*, 40(3): 611–625, 2017.

[51] Y. A. Farha and J. Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, pages 3575–3584, 2019.

[52] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *ICCV*, pages 993–1000, 2013.

[53] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2016.

[54] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, 2022.

[55] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, pages 2002–2011, Salt Lake City, Utah, 2018. IEEE.

[56] D. Gallup, M. Pollefeys, and J.-M. Frahm. 3d reconstruction using an n-layer heightmap. In *PR*, pages 1–10. Springer, 2010.

[57] H. Gao, X. Tao, X. Shen, and J. Jia. Dynamic scene deblurring with parameter selective sharing and nested skip connections. In *CVPR*, pages 3848–3856, 2019.

[58] D. Garg, Y. Wang, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao. Wasserstein distances for stereo disparity estimation. *NeurIPS*, 2020.

[59] R. Garg, V. K. BG, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *ECCV*, pages 740–756, Amsterdam, The Netherlands, 2016. Springer.

[60] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Real-time 3d hand pose estimation with 3d convolutional neural networks. *PAMI*, 41(4):956–970, 2019. doi: 10.1109/TPAMI.2018.2827052.

[61] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361. IEEE, 2012.

[62] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 32(11):1231–1237, 2013.

[63] C. Godard, O. M. Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[64] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, pages 3828–3838, 2019.

[65] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[66] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, and P. Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, June 2020.

[67] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon. 3d packing for self-supervised monocular depth estimation. In *CVPR*, 2020.

[68] V. Guizilini, R. Hou, J. Li, R. Ambrus, and A. Gaidon. Semantically-guided representation learning for self-supervised monocular depth. In *ICLR*, 2020.

[69] V. Guizilini, R. Ambrus, W. Burgard, and A. Gaidon. Sparse auxiliary networks for unified monocular depth prediction and completion. In *CVPR*, pages 11078–11088, 2021.

[70] J. Guo, K. Han, H. Wu, Y. Tang, X. Chen, Y. Wang, and C. Xu. Cmt: Convolutional neural networks meet vision transformers. In *CVPR*, pages 12175–12185, June 2022.

[71] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li. Group-wise correlation stereo network. In *CVPR*, pages 3273–3282, 2019.

[72] B. Ham, M. Cho, and J. Ponce. Robust guided image filtering using nonconvex potentials. *PAMI*, 40(1):192–207, 2017.

[73] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV*, pages 1–14. Springer, 2010.

[74] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[75] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016.

[76] L. He, H. Zhu, F. Li, H. Bai, R. Cong, C. Zhang, C. Lin, M. Liu, and Y. Zhao. Towards fast and accurate real-world depth super-resolution: Benchmark dataset and baseline. In *CVPR*, pages 9229–9238, 2021.

[77] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE TPAMI*, 30(2):328–341, 2007.

[78] H. Hirschmuller and D. Scharstein. Evaluation of cost functions for stereo matching. In *CVPR*, pages 1–8. IEEE, 2007.

[79] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE TPAMI*, 35(2):504–511, 2012.

[80] M. Hu, S. Wang, B. Li, S. Ning, L. Fan, and X. Gong. Penet: Towards precise and efficient image guided depth completion. In *ICRA*, pages 13656–13662. IEEE, 2021.

[81] J. Huang, S.-S. Huang, H. Song, and S.-M. Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *CVPR*, pages 8932–8941, 2021.

[82] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang. Deepmvs: Learning multi-view stereopsis. In *CVPR*, pages 2821–2830, 2018.

[83] T. Huang, L. Huang, S. You, F. Wang, C. Qian, and C. Xu. Lightvit: Towards light-weight convolution-free vision transformers. *arXiv preprint arXiv:2207.05557*, 2022.

[84] T.-W. Hui, C. C. Loy, and X. Tang. Depth map super-resolution by deep multi-scale guidance. In *ECCV*, pages 353–369. Springer, 2016.

[85] S. Imran, X. Liu, and D. Morris. Depth completion with twin surface extrapolation at occlusion boundaries. In *CVPR*, pages 2583–2592, 2021.

[86] H. Jiang, D. Sun, V. Jampani, Z. Lv, E. Learned-Miller, and J. Kautz. Sense: A shared encoder network for scene-flow estimation. In *ICCV*, October 2019.

[87] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley. Learning to estimate hidden motions with global motion aggregation. In *ICCV*, pages 9772–9781, 2021.

[88] A. Johnston and G. Carneiro. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *CVPR*, pages 4756–4765, 2020.

[89] 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 *ICCV*, pages 66–75, 2017.

[90] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. Intel realsense stereoscopic depth cameras. In *CVPR Workshops*, pages 1–10, 2017.

[91] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *ECCV*, pages 573–590, 2018.

[92] B. Kim, J. Ponce, and B. Ham. Deformable kernel networks for joint image filtering. *IJCV*, 129(2):579–600, 2021.

[93] K. Kim, S. Lee, and S. Cho. Mssnet: Multi-scale-stage network for single image deblurring. *arXiv preprint arXiv:2202.09652*, 2022.

[94] S. Kim, D. Min, B. Ham, S. Kim, and K. Sohn. Deep stereo confidence prediction for depth estimation. In *ICIP*, pages 992–996. IEEE, 2017.

[95] Y. Kim, B. Ham, C. Oh, and K. Sohn. Structure selective depth superresolution for rgb-d cameras. *TIP*, 25(11):5227–5238, 2016.

[96] Y. Kim, H. Jung, D. Min, and K. Sohn. Deep monocular depth estimation via integration of global and local predictions. *TIP*, 27(8):4131–4144, 2018.

[97] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *ECCV*, pages 582–600. Springer, 2020.

[98] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ToG*, 26(3):96–es, 2007.

[99] V. R. Kumar, M. Klingner, S. Yogamani, S. Milz, T. Fingscheidt, and P. Mader. Syndistnet: Self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving. In *WACV*, pages 61–71, 2021.

[100] H.-Y. Lai, Y.-H. Tsai, and W.-C. Chiu. Bridging stereo matching and optical flow via spatiotemporal correspondence. In *CVPR*, June 2019.

[101] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, pages 239–248, Stanford University, California, 2016. IEEE.

[102] Y. Lee, J. Kim, J. Willette, and S. J. Hwang. Mpvit: Multi-path vision transformer for dense prediction. *arXiv preprint arXiv:2112.11010*, 2021.

[103] Y. Lee, J. Kim, J. Willette, and S. J. Hwang. Mpvit: Multi-path vision transformer for dense prediction. In *CVPR*, 2022.

[104] H. Li, X. Gu, W. Yuan, L. Yang, Z. Dong, and P. Tan. Dense rgb slam with neural implicit maps. *ICLR*, 2023.

[105] J. Li, P. Wang, P. Xiong, T. Cai, Z. Yan, L. Yang, J. Liu, H. Fan, and S. Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. *ArXiv preprint arXiv:2203.11483*, 2022.

[106] K. Li, Y. Tang, V. A. Prisacariu, and P. H. Torr. Bnv-fusion: dense 3d reconstruction using bi-level neural volume fusion. In *CVPR*, pages 6166–6175, 2022.

[107] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep joint image filtering. In *ECCV*, pages 154–169. Springer, 2016.

[108] Y. Li, D. Min, M. N. Do, and J. Lu. Fast guided global interpolation for depth and motion. In *ECCV*, pages 717–733. Springer, 2016.

[109] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Joint image filtering with deep convolutional networks. *PAMI*, 41(8):1909–1923, 2019.

[110] Z. Li, X. Liu, N. Drenkow, A. Ding, F. X. Creighton, R. H. Taylor, and M. Unberath. Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers. In *ICCV*, pages 6197–6206, 2021.

[111] Z. Li, Z. Chen, X. Liu, and J. Jiang. Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation. *ArXiv preprint arXiv:2203.14211*, 2022.

[112] Z. Li, Z. Chen, X. Liu, and J. Jiang. Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation. *arXiv preprint arXiv:2203.14211*, 2022.

[113] Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang. Learning for disparity estimation through feature constancy. In *CVPR*, pages 2811–2820, 2018.

[114] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, pages 5741–5751, 2021.

[115] J. Lin, C. Gan, and S. Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019.

[116] S. D. Lin, S.-C. Shie, and J. Y. Guo. Improving the robustness of dct-based image watermarking against jpeg compression. *Computer Standards & Interfaces*, 32(1-2):54–60, 2010.

[117] Y. Lin, T. Cheng, Q. Zhong, W. Zhou, and H. Yang. Dynamic spatial propagation network for depth completion. In *AAAI*, 2022.

[118] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE TPAMI*, 38(10):2024–2039, 2016.

[119] L. Liu, X. Song, X. Lyu, J. Diao, M. Wang, Y. Liu, and L. Zhang. Fcfr-net: Feature fusion based coarse- to-fine residual learning for depth completion. In *AAAI*, volume 35, pages 2136–2144, 2021.

[120] P. Liu, I. King, M. R. Lyu, and J. Xu. Flow2stereo: Effective self-supervised learning of optical flow and stereo matching. In *CVPR*, pages 6648–6657, 2020.

[121] P. Liu, Z. Zhang, Z. Meng, N. Gao, and C. Wang. Pdr-net: Progressive depth reconstruction network for color guided depth map super-resolution. *Neurocomputing*, 479:75–88, 2022.

[122] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz. Learning affinity via spatial propagation networks. *NeurIPS*, 2017.

[123] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016.

[124] Y. Liu, F. Fang, T. Wang, J. Li, Y. Sheng, and G. Zhang. Multi-scale grid network for image deblurring with high-frequency guidance. *IEEE Transactions on Multimedia*, 2021.

[125] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021.

[126] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021.

[127] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[128] R. d. Lutio, S. D'aronco, J. D. Wegner, and K. Schindler. Guided super-resolution as pixel-to-pixel transformation. In *ICCV*, pages 8829–8837, 2019.

[129] X. Lyu, L. Liu, M. Wang, X. Kong, L. Liu, Y. Liu, X. Chen, and Y. Yuan. Hr-depth: High resolution self-supervised monocular depth estimation. *AAAI*, 2021.

[130] F. Ma and S. Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *ICRA*, 2018.

[131] F. Ma, G. V. Cavalheiro, and S. Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *ICRA*, pages 3288–3295. IEEE, 2019.

[132] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*, 2018.

[133] Y. Mao, Z. Liu, W. Li, Y. Dai, Q. Wang, Y.-T. Kim, and H.-S. Lee. Uasnet: Uncertainty adaptive sampling network for deep stereo matching. In *ICCV*, pages 6311–6319, 2021.

[134] A. Masoumian, H. A. Rashwan, S. Abdulwahab, J. Cristiano, and D. Puig. Gcndepth: Self-supervised monocular depth estimation based on graph convolutional network. *arXiv preprint arXiv:2112.06782*, 2021.

[135] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016.

[136] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *CVPR*, pages 3061–3070, 2015.

[137] N. Metzger, R. C. Daudt, and K. Schindler. Guided depth super-resolution by deep anisotropic diffusion. In *CVPR*, pages 18237–18246, 2023.

[138] Microsoft. Kinect for windows. `https://developer.microsoft.com/en-us/windows/kinect/`.

[139] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[140] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):1–15, 2022.

[141] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. doi: 10.1109/TRO.2017.2705103.

[142] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[143] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[144] D. Nazir, M. Liwicki, D. Stricker, and M. Z. Afzal. Semattnet: Towards attention-based semantic aware guided depth completion. *arXiv preprint arXiv:2204.13635*, 2022.

[145] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*, pages 127–136. IEEE, 2011.

[146] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM TOG*, 32(6):1–11, 2013.

[147] S. Niklaus and F. Liu. Softmax splatting for video frame interpolation. In *CVPR*, 2020.

[148] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam. isdf: Real-time neural signed distance fields for robot perception. *arXiv preprint arXiv:2204.02296*, 2022.

[149] J. Pan, J. Dong, J. S. Ren, L. Lin, J. Tang, and M.-H. Yang. Spatially variant linear representation models for joint filtering. In *CVPR*, pages 1702–1711, 2019.

[150] X. Pan, C. Ge, R. Lu, S. Song, G. Chen, Z. Huang, and G. Huang. On the integration of self-attention and convolution. In *CVPR*, pages 815–825, 2022.

[151] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon. High quality depth map upsampling for 3d-tof cameras. In *ICCV*, pages 1623–1630. IEEE, 2011.

[152] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. So Kweon. Non-local spatial propagation network for depth completion. In *ECCV*, pages 120–136. Springer, 2020.

[153] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. So Kweon. Non-local spatial propagation network for depth completion. In *ECCV*, pages 120–136. Springer, 2020.

[154] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*, pages 8024–8035. Curran Associates, Inc., 2019.

[155] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32:8026–8037, 2019.

[156] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, and S. Mattoccia. Enabling energy-efficient unsupervised monocular depth estimation on ARMv7-based platforms. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019*, pages 1703–1708, 2019.

[157] Z. Peng, W. Huang, S. Gu, L. Xie, Y. Wang, J. Jiao, and Q. Ye. Conformer: Local features coupling global representations for visual recognition. In *ICCV*, pages 367–376, October 2021.

[158] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia. Towards real-time unsupervised monocular depth estimation on CPU. In *IROS*, 2018.

[159] M. Poggi, F. Tosi, and S. Mattoccia. Learning monocular depth estimation with unsupervised trinocular assumptions. In *3DV*, 2018.

[160] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *CVPR*, pages 3227–3237, 2020.

[161] M. Poggi, F. Tosi, K. Batsos, P. Mordohai, and S. Mattoccia. On the synergies between machine learning and binocular stereo for depth estimation from images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5314–5334, 2022. doi: 10.1109/TPAMI.2021.3070917.

[162] M. Pu, Y. Huang, Y. Liu, Q. Guan, and H. Ling. Edter: Edge detection with transformer. In *CVPR*, pages 1402–1412, 2022.

[163] X. Qiao, C. Ge, Y. Zhang, Y. Zhou, F. Tosi, M. Poggi, and S. Mattoccia. Depth super-resolution from explicit and implicit high-frequency features. *arXiv preprint arXiv:2303.09307*, 2023.

[164] Z. Qin, P. Zhang, F. Wu, and X. Li. Fcanet: Frequency channel attention networks. In *ICCV*, pages 783–792, 2021.

[165] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, and M. Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *CVPR*, pages 3313–3322, 2019.

[166] R. Ranftl, A. Bochkovskiy, and V. Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[167] R. Ranftl, A. Bochkovskiy, and V. Koltun. Vision transformers for dense prediction. In *ICCV*, pages 12179–12188, 2021.

[168] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *CVPR*, pages 12232–12241, Long Beach, California, 2019. IEEE.

[169] K. Rho, J. Ha, and Y. Kim. Guideformer: Transformers for image guided depth completion. In *CVPR*, pages 6250–6259, 2022.

[170] A. Riemens, O. Gangwal, B. Barenbrug, and R.-P. Berretty. Multistep joint bilateral depth upsampling. In *Visual Communications and Image Processing*, volume 7257, pages 192–203. SPIE, 2009.

[171] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *MICCAI*, 2015.

[172] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICAI*, pages 234–241. Springer, 2015.

[173] A. Rosinol, J. J. Leonard, and L. Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022.

[174] S. Safadoust and F. Güney. Self-supervised monocular scene decomposition and depth estimation. In *3DV*, pages 627–636. IEEE, 2021.

[175] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[176] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, pages 12716–12725, 2019.

[177] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE TPAMI*, 31(5):824–840, 2008.

[178] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR*, pages 1–8. IEEE, 2007.

[179] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

[180] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR*, volume 1, pages I–I. IEEE, 2003.

[181] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR*, pages 31–42. Springer, 2014.

[182] K. Schmid, T. Tomic, F. Ruess, H. Hirschmüller, and M. Suppa. Stereo vision based indoor/outdoor navigation for flying robots. In *IROS*, pages 3955–3962. IEEE, 2013.

[183] T. Schops, T. Sattler, and M. Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *CVPR*, pages 134–144, 2019.

[184] R. Schuster, C. Unger, and D. Stricker. A deep temporal fusion framework for scene flow using a learnable motion model and occlusions. In *WACV*, 2021.

[185] Z. Shen, Y. Dai, and Z. Rao. Cfnet: Cascade and fused cost volume for robust stereo matching. In *CVPR*, pages 13906–13915, June 2021.

[186] C. Shu, K. Yu, Z. Duan, and K. Yang. Feature-metric loss for self-supervised learning of depth and egomotion. In *ECCV*, pages 572–588. Springer, 2020.

[187] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760. Springer, 2012.

[188] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760. Springer, 2012.

[189] S. Sivaraman and M. M. Trivedi. A review of recent developments in vision-based vehicle detection. In *IV*, pages 310–315. IEEE, 2013.

[190] X. Song, X. Zhao, H. Hu, and L. Fang. Edgestereo: A context integrated residual pyramid network for stereo matching. In *ACCV*, pages 20–35. Springer, 2018.

[191] X. Song, Y. Dai, D. Zhou, L. Liu, W. Li, H. Li, and R. Yang. Channel attention based iterative residual learning for depth map super-resolution. In *CVPR*, pages 5631–5640, 2020.

[192] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[193] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012.

[194] H. Su, V. Jampani, D. Sun, O. Gallo, E. Learned-Miller, and J. Kautz. Pixel-adaptive convolutional neural networks. In *CVPR*, pages 11166–11175, 2019.

[195] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison. imap: Implicit mapping and positioning in real-time. In *ICCV*, pages 6229–6238, 2021.

[196] X. Sui, S. Li, X. Geng, Y. Wu, X. Xu, Y. Liu, R. S. M. Goh, and H. Zhu. Craft: Cross-attentional flow transformers for robust optical flow. In *CVPR*, 2022.

[197] B. Sun, X. Ye, B. Li, H. Li, Z. Wang, and R. Xu. Learning scene structure guidance via cross-task knowledge transfer for single depth super-resolution. In *CVPR*, pages 7792–7801, 2021.

[198] C. Sun, M. Sun, and H.-T. Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, pages 5459–5469, 2022.

[199] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *CVPR*, pages 15598–15607, 2021.

[200] J. Sun, X. Chen, Q. Wang, Z. Li, H. Averbuch-Elor, X. Zhou, and N. Snavely. Neural 3d reconstruction in the wild. In *SIGGRAPH*, pages 1–9, 2022.

[201] Q. Sun, Y. Tang, C. Zhang, C. Zhao, F. Qian, and J. Kurths. Unsupervised estimation of monocular depth and vo in dynamic environments via hybrid masks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[202] M. Tan and Q. Le. Efficientnetv2: Smaller models and faster training. In *ICML*, pages 10096–10106. PMLR, 2021.

[203] J. Tang, F.-P. Tian, W. Feng, J. Li, and P. Tan. Learning guided convolutional network for depth completion. *IEEE TIP*, pages 1116–1129, 2020.

[204] J. Tang, X. Chen, and G. Zeng. Joint implicit image function for guided depth super-resolution. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4390–4399, 2021.

[205] Q. Tang, R. Cong, R. Sheng, L. He, D. Zhang, Y. Zhao, and S. Kwong. Bridgenet: A joint learning network of depth map super-resolution and monocular depth estimation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2148–2157, 2021.

[206] Y. Tang, C. Zhao, J. Wang, C. Zhang, Q. Sun, W. X. Zheng, W. Du, F. Qian, and J. Kurths. Perception and navigation in autonomous systems in the era of learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[207] V. Tankovich, C. Hane, Y. Zhang, A. Kowdle, S. Fanello, and S. Bouaziz. Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching. In *CVPR*, pages 14362–14372, June 2021.

[208] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *CVPR*, pages 6243–6252, 2017.

[209] Z. Teed and J. Deng. Deepv2d: Video to depth with differentiable structure from motion. *ICLR*, 2020.

[210] Z. Teed and J. Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419. Springer, 2020.

[211] Z. Teed and J. Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *NeurIPS*, 34:16558–16569, 2021.

[212] Z. Teed and J. Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *NeurIPS*, 2021.

[213] Z. Teed and J. Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *CVPR*, 2021.

[214] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia. Learning monocular depth estimation infusing traditional stereo knowledge. In *CVPR*, 2019.

[215] F. Tosi, F. Aleotti, P. Z. Ramirez, M. Poggi, S. Salti, L. D. Stefano, and S. Mattoccia. Distilled semantics for comprehensive scene understanding from videos. In *CVPR*, pages 4654–4665, 2020.

[216] F. Tosi, Y. Liao, C. Schmitt, and A. Geiger. Smd-nets: Stereo mixture density networks. In *CVPR*, pages 8942–8952, 2021.

[217] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. In *3DV*, pages 11–20. IEEE, 2017.

[218] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. In *3DV*, pages 11–20. IEEE, 2017.

[219] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool. Sparse and noisy lidar completion with rgb guidance and uncertainty. In *MVA*, pages 1–6. IEEE, 2019.

[220] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[221] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[222] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey. Learning depth from monocular videos using direct methods. In *CVPR*, 2018.

[223] H. Wang, A. Kembhavi, A. Farhadi, A. L. Yuille, and M. Rastegari. Elastic: Improving cnns with dynamic scaling policies. In *CVPR*, pages 2258–2267, 2019.

[224] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition. *PAMI*, 43(10):3349–3364, 2020.

[225] J. Wang, T. Bleja, and L. Agapito. Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. *arXiv preprint arXiv:2206.14735*, 2022.

[226] K. Wang, L. Zhao, J. Zhang, J. Zhang, A. Wang, and H. Bai. Joint depth map super-resolution method via deep hybrid-cross guidance filter. *Pattern Recognition*, 136: 109260, 2023.

[227] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[228] Q. Wang, Z. Wang, K. Genova, P. P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, pages 4690–4699, 2021.

[229] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer. Tartanair: A dataset to push the limits of visual slam. In *IROS*, 2020.

[230] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, pages 568–578, 2021.

[231] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, pages 568–578, 2021.

[232] Y. Wang, A. Ortega, D. Tian, and A. Vetro. A graph-based joint bilateral approach for depth enhancement. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 885–889. IEEE, 2014.

[233] Y. Wang, Z. Lai, G. Huang, B. H. Wang, L. Van Der Maaten, M. Campbell, and K. Q. Weinberger. Anytime stereo image depth estimation on mobile devices. In *ICRA*, pages 5893–5900. IEEE, 2019.

[234] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu. Unos: Unified unsupervised optical-flow and stereo-depth estimation by watching videos. In *CVPR*, pages 8071–8081, 2019.

[235] Y. Wang, J. Yang, and H. Yue. Depth map continuous super-resolution with local implicit guidance function. *Displays*, 78:102418, 2023.

[236] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *TIP*, 13(4):600–612, Apr. 2004.

[237] Z. Wang, X. Ye, B. Sun, J. Yang, R. Xu, and H. Li. Depth upsampling based on deep edge-aware learning. *Pattern Recognition*, 103:107274, 2020.

[238] J. Watson, M. Firman, G. J. Brostow, and D. Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019.

[239] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[240] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *ICRA*, 2019.

[241] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module. In *ECCV*, pages 3–19, 2018.

[242] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *NeurIPS*, 34, 2021.

[243] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *NeurIPS*, 34, 2021.

[244] X. Xiong, H. Xiong, K. Xian, C. Zhao, Z. Cao, and X. Li. Sparse-to-dense depth completion revisited: Sampling strategy and graph construction. In *ECCV*, pages 682–699. Springer, 2020.

[245] G. Xu, J. Cheng, P. Guo, and X. Yang. Attention concatenation volume for accurate and efficient stereo matching. In *CVPR*, pages 12981–12990, 2022.

[246] H. Xu and J. Zhang. Aanet: Adaptive aggregation network for efficient stereo matching. In *CVPR*, pages 1959–1968, 2020.

[247] K. Xu, M. Qin, F. Sun, Y. Wang, Y.-K. Chen, and F. Ren. Learning in the frequency domain. In *CVPR*, pages 1740–1749, 2020.

[248] W. Xu, Y. Xu, T. Chang, and Z. Tu. Co-scale conv-attentional image transformers. In *ICCV*, pages 9981–9990, 2021.

[249] Y. Xu, L. Nan, L. Zhou, J. Wang, and C. C. Wang. Hrbf-fusion: Accurate 3d reconstruction from rgb-d data using on-the-fly implicits. *ACM TOG*, 41(3):1–19, 2022.

[250] Z. Xu, H. Yin, and J. Yao. Deformable spatial propagation networks for depth completion. In *ICIP*, pages 913–917. IEEE, 2020.

[251] J. Yan, H. Zhao, P. Bu, and Y. Jin. Channel-wise attention-based network for self-supervised monocular depth estimation. In *3DV*, pages 464–473. IEEE, 2021.

[252] Z. Yan, K. Wang, X. Li, Z. Zhang, B. Xu, J. Li, and J. Yang. Rignet: Repetitive image guided network for depth completion. *ECCV*, 2022.

[253] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia. Segstereo: Exploiting semantic information for disparity estimation. In *ECCV*, pages 636–651, 2018.

[254] G. Yang, J. Manela, M. Happold, and D. Ramanan. Hierarchical deep stereo matching on high-resolution images. In *CVPR*, June 2019.

[255] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi, and B. Zhou. Drivingstereo: A large-scale dataset for stereo matching in autonomous driving scenarios. In *CVPR*, 2019.

[256] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *CVPR*, pages 1281–1292, 2020.

[257] Q. Yang. A non-local cost aggregation method for stereo matching. In *CVPR*, pages 1402–1409. IEEE, 2012.

[258] Q. Yang, R. Yang, J. Davis, and D. Nistér. Spatial-depth super resolution for range images. In *CVPR*, pages 1–8. IEEE, 2007.

[259] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency, 2017.

[260] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia. Every pixel counts: Unsupervised geometry learning with holistic 3d motion understanding. In *ECCVW*, September 2018.

[261] Z. Yang, P. Wang, W. Yang, W. Xu, and N. Ram. Lego: Learning edge with geometry all at once by watching videos. In *CVPR*, 2018.

[262] X. Ye, B. Sun, Z. Wang, J. Yang, R. Xu, H. Li, and B. Li. Pmbanet: Progressive multi-branch aggregation network for scene depth super-resolution. *TIP*, 29:7427–7442, 2020.

[263] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin. inerf: Inverting neural radiance fields for pose estimation. In *IROS*, pages 1323–1330. IEEE, 2021.

[264] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, pages 1983–1992, 2018.

[265] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *NeurIPS*, 2022.

[266] J. Yuan, H. Jiang, X. Li, J. Qian, J. Y. Li, and J. Yang. Recurrent structure attention guidance for depth super-resolution. In *AAAI*, pages 3331–3339, Jun. 2023.

[267] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao. Multi-stage progressive image restoration. In *CVPR*, pages 14821–14831, 2021.

[268] N. Zenati and N. Zerhouni. Dense stereo matching with application to augmented reality. In *NeurIPS*, pages 1503–1506. IEEE, 2007.

[269] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *CVPR*, 2018.

[270] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, pages 185–194, 2019.

[271] F. Zhang, O. J. Woodford, V. A. Prisacariu, and P. H. Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In *ICCV*, pages 10807–10817, 2021.

[272] K. Zhang, G. Riegler, N. Snavely, and V. Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[273] X. Zhang, S. Bi, K. Sunkavalli, H. Su, and Z. Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In *CVPR*, pages 5449–5458, 2022.

[274] Y. Zhang and T. Funkhouser. Deep depth completion of a single rgb-d image. In *CVPR*, pages 175–185, 2018.

[275] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, pages 286–301, 2018.

[276] Y. Zhang, Y. Chen, X. Bai, S. Yu, K. Yu, Z. Li, and K. Yang. Adaptive unimodal cost volume filtering for deep stereo matching. In *AAAI*, volume 34, pages 12926–12934, 2020.

[277] Y. Zhang, P. Wei, H. Li, and N. Zheng. Multiscale adaptation fusion networks for depth completion. In *IJCNN*, pages 1–7, 2020.

[278] Y. Zhang, M. Poggi, and S. Mattoccia. Temporalstereo: Efficient spatial-temporal stereo matching network. In *IROS*, 2022.

[279] Y. Zhang, X. Guo, M. Poggi, Z. Zhu, G. Huang, and S. Mattoccia. Completionformer: Depth completion with convolutions and vision transformers. In *CVPR*, pages 18527–18536, 2023.

[280] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *ICCV*, pages 3727–3737, 2023.

[281] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, 63(9):1612–1627, 2020.

[282] C. Zhao, G. G. Yen, Q. Sun, C. Zhang, and Y. Tang. Masked gan for unsupervised depth and pose prediction with scale consistency. *IEEE Transactions on Neural Networks and Learning Systems*, 32(12):5392–5403, 2020.

[283] C. Zhao, Y. Tang, and Q. Sun. Unsupervised monocular depth estimation in highly complex environments. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[284] C. Zhao, Y. Zhang, M. Poggi, F. Tosi, X. Guo, Z. Zhu, G. Huang, Y. Tang, and S. Mattoccia. Monovit: Self-supervised monocular depth estimation with a vision transformer. In *3DV*, 2022.

[285] H. Zhao, X. Kong, J. He, Y. Qiao, and C. Dong. Efficient image super-resolution using pixel attention. In *ECCVW*, pages 56–72. Springer, 2020.

[286] S. Zhao, M. Gong, H. Fu, and D. Tao. Adaptive context-aware multi-modal network for depth completion. *TIP*, pages 5264–5276, 2021.

[287] Z. Zhao, J. Zhang, S. Xu, Z. Lin, and H. Pfister. Discrete cosine transform network for guided depth map super-resolution. In *CVPR*, pages 5697–5707, 2022.

[288] Y. Zhong, H. Li, and Y. Dai. Open-world stereo video matching with deep rnn. In *ECCV*, pages 101–116, 2018.

[289] Y. Zhong, C.-Y. Wu, S. You, and U. Neumann. Deep rgb-d canonical correlation analysis for sparse depth completion. *NeurIPS*, 2019.

[290] H. Zhou, B. Ummenhofer, and T. Brox. Deeptam: Deep tracking and mapping. In *ECCV*, pages 822–838, 2018.

[291] H. Zhou, D. Greenwood, S. Taylor, and H. Gong. Constant velocity constraints for self-supervised monocular depth estimation. In *European Conference on Visual Media Production*, pages 1–8, 2020.

[292] H. Zhou, D. Greenwood, and S. Taylor. Self-supervised monocular depth estimation with internal feature fusion. In *BMVC*, 2021.

[293] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, pages 1851–1858, 2017.

[294] Y. Zhu, W. Dong, L. Li, J. Wu, X. Li, and G. Shi. Robust depth completion with uncertainty-driven loss functions. In *AAAI*, 2022.

[295] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *CVPR*, pages 12786–12796, 2022.

[296] Z. Zhu, S. Peng, V. Larsson, Z. Cui, M. R. Oswald, A. Geiger, and M. Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. *arXiv preprint arXiv:2302.03594*, 2023.

[297] X. Zou, F. Xiao, Z. Yu, Y. Li, and Y. J. Lee. Delving deeper into anti-aliasing in convnets. *IJCV*, pages 1–15, 2022.

[298] Y. Zou, Z. Luo, and J.-B. Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV*, pages 1–18, Munich, Germany, 2018. Springer.

[299] Z.-X. Zou, S.-S. Huang, Y.-P. Cao, T.-J. Mu, Y. Shan, and H. Fu. Mononeuralfusion: Online monocular neural 3d reconstruction with geometric priors. *arXiv preprint arXiv:2209.15153*, 2022.

[300] Y. Zuo, Q. Wu, Y. Fang, P. An, L. Huang, and Z. Chen. Multi-scale frequency reconstruction for guided depth map super-resolution via deep residual network. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(2):297–306, 2019.

[301] Y. Zuo, Y. Fang, P. An, X. Shang, and J. Yang. Frequency-dependent depth map enhancement via iterative depth-guided affine transformation and intensity-guided refinement. *IEEE Transactions on Multimedia*, 23:772–783, 2020.