EFFICIENTLY FUSING SPARSE LIDAR FOR ENHANCED SELF-SUPERVISED MONOCULAR DEPTH ESTIMATION

Yue Wang[1,2,*], *Mingrong Gong*[1,*], *Lei Xia*[1], *Qieshi Zhang*[1,†], *Jun Cheng*[1]
[1]Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems,
Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China.
[2]Department of Computer Science, University College London, Gower Street, WC1E 6BT, UK.
[*]Co-first authors, contributed equally to this work
[†]Corresponding authors, email: qs.zhang@siat.ac.cn

ABSTRACT

Monocular self-supervised depth estimation with a low-cost sensor is the mainstream solution to gathering dense depth maps for robots and autonomous driving. In this paper, based on the philosophy *"less is more"* (i.e., focusing only on valid pixels in sparse LiDAR), we propose a novel framework, Efficient Sparse Depth (EffisDepth), for predicting dense depth. The Sparse Feature Extractor (SFE) embedded in the proposed framework effectively handles sparse LiDAR by forming sparse tensors. The Slender Group Block (SGB) is the main building block in SFE, which extracts features from sparse tensors via a structure of two branches. Extensive experiments show that our method achieves state-of-the-art performance on the KITTI benchmark, demonstrating the effectiveness of each proposed component and the self-supervised learning framework.

Index Terms— Self-supervised depth estimation, Sparse LiDAR, Sparse tensor

1. INTRODUCTION

Dense depth estimation is an essential task for autonomous driving and robotics, which enables the agent to have the ability to perceive the 3D world. Over the last years, the learning-based self-supervised monocular depth prediction has become the mainstream of research [1, 2]. Yet, limited by the inherent ill-posed problem of a monocular camera, these models always cannot predict accurate depth under the failure of photometric consistency caused by occlusion and texture sparsity. To break this limitation, extra sensors can compensate for the scale ambiguity with enough spatial information. For example, the dense LiDAR (e.g., 32- or 64- beam) can serve as the source of supervision to produce scale-aware estimates [3, 4]. While these methods are still unable to meet the demand for automotive-grade configurations due to the high price of these sensors. Therefore, leveraging a type of low-cost sensors in monocular depth estimation has received an increasing amount of attention [4–11]. DynaDepth [5] integrates information from monocular sequences and IMU motion dynamics; the work of Chawla et al. [11] propose a dynamically-weighted loss from the ubiquitously copresent Global Positioning Systems (GPS) to complement the appearance-based losses; FusionDepth [4] and PL++ [6] exploit the sparse LiDAR for monocular estimations.

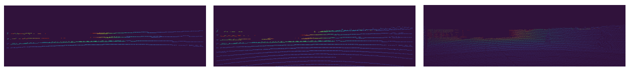
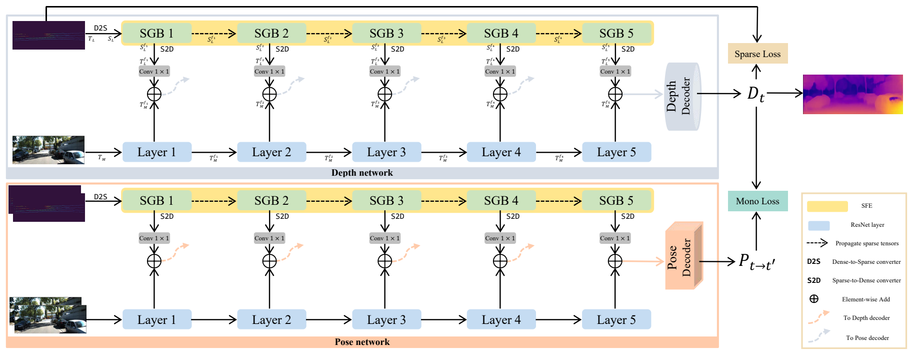

(a) 4-beam (b) 16-beam (c) 64-beam

Fig. 1: **Different LiDAR densities.** A 4-beam depth map contains less than 1% of the number of valid pixels.

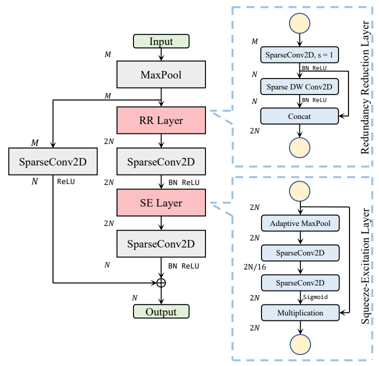
However, the IMU- and GPS-aided methods do not show an obvious improvement over the monocular methods. While methods using sparse LiDAR reduce the errors, they have several fatal defects: (1) the sparse LiDAR provides low-dense information (e.g. extremely sparse signals from 4-beam LiDAR) that contains surge empty-value space. These methods fail to consider the fact that the uninformative content not only causes an enormous waste of computational power also interferes with the correct learning of spatial information of the network. (2) Although applying some extra techniques to the predicted initial depth can improve the accuracy, such as correcting the errors using an additional refinement network with knowledge distillation and GDC [4] post-processing module, it enormously raises the cost of the whole network and takes exceptional time-consuming that is hard for real-time application. (3) Either the image-based vanilla 2D CNN or "Pseudo-LiDAR" based 3D CNN they used will significantly increase the parameters burden of the original monocular network.

In summary, these methods do not allow cheap sensors to be utilized in a "cheap" way, which is unfit for automotive applications. Therefore, motivated by the sparse characteristics, we design a Sparse Feature Extractor (SFE) module, consisting of sparse convolutions, tailored to these shortcomings in our EffisDepth. Instead of interpolation algorithms, adding extra confidence channels or other data prepossessing for sparse LiDAR signals [4, 6–10], EffisDepth, the end-to-end self-supervised framework, with the input of monocular images and sparse LiDAR points can directly utilize the raw

WE ACKNOWLEDGE THE SUPPORT OF THE NATIONAL NATURAL SCIENCE FOUNDATION OF CHINA (NOS. U21A20487, U1913202); THE NATIONAL NATURAL SCIENCE FOUNDATION OF GUANGDONG PROVINCE (NO. 2022A1515140119)



(a) Framework Overview



(b) SGB: Slender Group Block

Fig. 2: **EffisDepth**. **(a) Framework Overview:** Our proposed end-to-end framework take the monocular images and the matching sparse LiDAR points as input. It jointly trains the Depth network and the Pose network to predict accurate dense depth. Note that the yellow area is our SFE module and the dashed arrows represent propagating sparse tensor. **(b) SGB:** The block details of SGB, a two-branch design with an RR layer and SE layer.

sensor data to predict dense depth. The SFE module is different from the vanilla convolution modules as it is only concerned with the valid points extracted from the low-density information. Within this module, all feature propagation is based on the compact sparse tensors. Note that we specially focus on the minimal sparse LiDAR (e.g., 4-beam LiDAR) as the low-cost sensors in our proposed method. Our EffisDepth substantially outperforms the state-of-the-art methods that employ or not sparse LiDAR. We summarize our contributions as follows:

1) We propose a novel self-supervised framework, termed EffisDepth, to predict dense depth by fusing the features from extremely sparse LiDAR points and monocular images in an end-to-end manner, which significantly boosts performance in a cost-effective manner.
2) For the purpose of exploring a more suitable environment for sparse LiDAR signals, we propose a special lightweight "plugin" module Sparse Feature Extractor (SFE) to efficiently handle sparse LiDAR points. It does so by only processing the valid pixels which are saved in the sparse tensor instead of tensor. An elaborate Slender Group Block (SGB) is designed as the layer of SFE to extract features from the sparse tensor.
3) We conduct extensive experiments to evaluate our method on the KITTI dataset, the results demonstrates our EffisDepth outperforms the current state-of-art methods.

2. METHODOLOGY

We describe the overview of our framework in Sec. 2.1. Then, we introduce our proposed Sparse Feature Extractor module (SFE) in Sec. 2.2. We especially demonstrate the innovative block Slender Group Block (SGB) in Sec. 2.3. The loss function is introduced in Sec. 2.4.

2.1. Framework Overview

We propose a self-supervised framework for dense depth estimation by taking monocular image $I^{H\times W\times 3}$ and the sparse depth map $D^{H\times W\times 1}$ projected from sparse LiDAR points as input, as illustrated in Fig. 2(a). Following the classical self-supervised monocular prediction works [1, 2], the framework comprises depth and pose networks. Additionally, we deploy a parallel-encoder structure in both two networks to respectively perform feature extraction from the two-sensors input. Unlike most previous approaches using two-ResNet design to extract features from monocular images and another modality separately, we only employ ResNet for the camera input and propose a Sparse Feature Extractor (SFE) for the sparse LiDAR input considering the LiDAR signals show huge differences with dense images data. The sparse LiDAR generates points cloud $P^{N\times 3}$ corresponding to each monocular image $I^{H\times W\times 3}$, where N is the number of the points and each point $p \in P$ contains location coordinates, X, Y, and Z in 3D space. The matched sparse depth map $D^{H\times W\times 1}$ can be obtained by projecting points P in 3D space onto an image plane. However, most space in the sparse depth map from sparse LiDAR is empty, and only 1% of valid pixels are contained. These dominant hole regions in the sparse depth map will lead to inefficient extracting features and massive memory consumption when using vanilla convolutions. Our proposed FSE, which will be introduced in the next section, can avoid being fed unnecessary information and overcome the above problems. Finally, the fused multi-scale features are propagated into the depth and pose decoders to predict dense D_t and the pose $P_{t\to t'}$ of camera ego-motion, where t and t' represent consecutive two frames of monocular sequences.

2.2. Sparse Feature Extractor (SFE)

As described above, SFE is built on the sparse tensor. The sparse tensor has the high expressiveness for data where most of the space is empty. It can save both the memory and computation of our network using sparse LiDAR. As shown in Fig. 3(b), a sparse tensor $S = \{C^{N\times 3}, F^{N\times M}\}$ contains two matrices, $C^{N\times 3}$ and $F^{N\times M}$, where $C^{N\times 3}$ stores the location information of those N valid pixels and $F^{N\times M}$ stores the corresponding feature vectors $\mathbf{f}$ of those ones, M is the number of feature maps, can be formulated as:

$$C = \begin{bmatrix} x_1 & y_1 & b_1 \\ x_2 & y_2 & b_2 \\ & \vdots & \\ x_N & y_N & b_N \end{bmatrix}, \quad F = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_N \end{bmatrix}, \tag{1}$$

where (x_n, y_n) represents the coordinate of the n-th valid pixel ($n \in \{1, 2, ..., N\}$), b_n is the batch index of the sparse depth map where the n-th valid point located. $\mathbf{f}_n \in \mathbb{R}^M$ denotes the feature vector of the n-th valid pixel, where M indicates the number of feature maps. For simplicity and without sacrificing generality, we assume batch size as 1 and disregard the batch index in the formulas below.

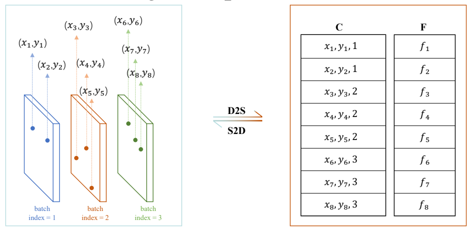
However, the tensor still permeates the whole framework outside of SFE. In order to embed SFE smoothly, we place two types of data converters in our framework, the Dense-to-Sparse (D2S) converter and the Sparse-to-Dense (S2D) converter, used for mutual conversion between the tensor and the sparse tensor, an example of which is shown in Fig. 3. The D2S converter changes the data representation of tensor $T^{H \times W \times M}$ to prepare a sparse tensor:

$$\begin{aligned} &\forall x_n, y_n \in T \wedge T(x_n, y_n) > 0, \\ &D2S(T) = \{(x_n, y_n), T(x_n, y_n)\}, \end{aligned} \tag{2}$$

It only picks up the valid pixels and restores them in a portable form of the sparse tensor. Similarly, a sparse tensor $S^{N \times M} = \{C^{N \times 3}, F^{N \times M}\}$ can be recovered to a tensor $T'^{H \times W \times M}$ through the S2D converter. It does so in a reverse way of padding zeros in empty pixels:

$$\begin{aligned} S2D(S) &= T'(x_n, y_n) \\ &= \begin{cases} \mathbf{f}_n \in F, & if(x_n, y_n) \in C, \\ 0, & otherwise. \end{cases} \end{aligned} \tag{3}$$

Next, we detail the pipeline of how to handle the sparse LiDAR within SFE. As shown in Fig. 2(a), in the depth network, the tensor T_L of the LiDAR projecting sparse depth map is first converted into a sparse tensor S_L through an S2D converter, and then sent into a series of Slender Group Blocks (SGBs) for features extraction, that we will elaborate in the next section. The sparse tensor $S_L^{f_i}$ is the output features from the i-th SGB, we arrange it to pass a D2S converter to recover the tensor $T_L^{f_i}$ for fusing with the corresponding features $T_M^{f_i}$ from monocular images later. In the pose network, except for the additional need to concatenate two frames as input, SFE works the same as it does in the depth network. For simplicity, we only mark the data format of features propagation in the depth network in the figure.



(a) tensor (b) sparse tensor

Fig. 3: **An example of the tensor, sparse tensor, and their transformation (D2S&S2D). (a) Tensor:** The shape of the tensor is $3 \times H \times W \times C$, where 3 indicates the batch size. In this batch, each one contains a few valid pixels, and its rest region contains 0. **(b) Sparse tensor:** The sparse tensor only stores the valid pixels from the tensor.

2.3. Slender Group Block (SGB)

As described above, we re-purpose some architectural innovations of vanilla convolutions to design SGB consisting of sparse convolutions. As shown in Fig. 2(b), it comprises the main and residual branches. On the top of the main branch, we first place a Redundancy Reduction (RR) layer inspired by the fancy intrinsic feature maps exploration [12], it increases the number of feature maps in a "cheap" way that contains a series of linear transformations. Furthermore, a Squeeze-Excitation [13] (SE) layer is added to more powerfully capture the most salient information from the sparse map. The output of the previous SGB is directly taken as input to the next SGB. We highlight that the entire pipeline of SFE is in the sparse form, thus can realize the efficiency with high sparsity input. In our work, we implement sparse convolutions in Minkowski engine [14].

2.4. Loss Functions

Our loss function is composed of two components, an appearance-based Mono loss L_{mono} and a LiDAR-aided Sparse Loss L_{sparse}. Following [2], the predicted depth map $D^{H \times W \times 1}$ and 6-DOF vector $\mathbf{T}_{t \to t'}$ can be used for view synthesis to calculate the Mono Loss L_{mono}. We utilize the Scale Invariant Logarithmic loss $(SILog)$ for the Sparse Loss L_{sparse}:

$$L_{sparse} = \lambda * \sqrt{\eta Si}, \tag{4}$$

$$Si = D\left(y_{sparse}, \hat{y}\right) = \frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}\left(\sum_i d_i\right)^2, \tag{5}$$

where y^{sparse} and $\hat{y}$ denote the depth from the 4-beam LiDAR and the predicted depth respectively, $d_i = \log y_i^{sparse} - \log \hat{y}_i$ is the difference between them at pixel i, and n is the number of valid pixels (we filter out the invalid pixels for optimization). The final loss is the sum of the Mono Loss L_{mono} and the Sparse Loss L_{sparse}:

$$L = L_{mono} + L_{sparse}. \tag{6}$$

3. EXPERIMENTS

3.1. Dataset and Implementation Details

We use the KITTI original dataset with the data split of Eigen [15]. This results in 39810 images for training, 4424 for validation and 697 for evaluation. The 4-beam data is obtained through sampling from the 64-beam LiDAR point cloud of the KITTI dataset same as Pseudo LiDAR++ [4]. The proposed framework is implemented using PyTorch, and the models are trained 20 epochs with Adam optimizer, a batch size of 8. All experiments are performed on 640 × 192 resolution images on a single Nvidia Titan X GPU. We take ResNet pretrained on ImageNet as the encoder for features extraction of monocular images.

Table 1: **Depth estimation results on KITTI original Dataset with the KITTI Eigen [15] test split**: The best and the second best results are shown in **bold** and underlined. All methods use a resolution of 640×192 pixels. M, S, and L, respectively, refer to Monocular, Stereo, and Sparse LiDAR data. D refers to supervised training. Metrics indicated by red: *lower is better*, Metrics indicated by yellow: *higher is better*.

Methods	Train	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
LEGO [1]	M	0.162	1.352	6.276	0.252	0.783	0.921	0.969
MonoDepth2 [2]	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981
PackNet-SfM [16]	M	0.111	0.785	4.601	0.189	0.878	0.960	0.982
RM-Depth [17]	M	0.108	0.71	4.513	0.183	0.884	0.964	0.983
MonoFormer [18]	M	0.108	0.806	4.594	0.184	0.884	0.963	0.983
SD-SSMDE R50 [19]	M	0.100	0.661	4.264	0.172	0.896	0.967	0.985
MonoDepth2 [2]	M+S	0.106	0.818	4.750	0.196	0.874	0.957	0.979
PLADE-Net [20]	S	0.092	0.626	4.046	0.175	0.896	0.965	0.984
Dorn [21]	M+D	0.099	0.593	3.714	0.161	0.897	0.966	0.986
BTS [22]	M+D	0.091	0.555	4.033	0.174	0.904	0.967	0.984
Guizilini *et al.* [3]	M+L	0.082	**0.424**	3.73	**0.131**	0.917	-	-
FusionDepth (Initial Depth) [6]	M+L	0.078	0.515	3.67	0.154	0.935	0.973	0.986
DynaDepth R50 (rescaled) [5]	M+IMU	0.108	0.761	4.608	0.187	0.883	0.962	0.982
Chawla *et al.* [11]	M+GPS	0.112	0.894	4.852	0.192	0.877	0.958	0.981
EffisDepth	M+L	0.073	0.477	**3.512**	0.146	0.937	**0.975**	**0.988**
EffisDepth-large	M+L	**0.071**	0.480	3.604	0.147	**0.941**	**0.975**	0.987

3.2. Experiment Results and Analyses

Comparison with State-of-the-arts. We present the result of our EffisDepth with the normal SFE that has almost negligible parameters. For fair comparison, we also present the result of EffisDepth-large with the SFE-large that has similar size as ResNet50 and extracts 4 times features maps than SFE. As shown in Table 1, our method achieves leading scores compared to the current state-of-art methods. We compare our method against camera-only methods and methods using sparse sensor signals as guidance. As unfolded, the self-supervised monocular (M) [1, 2, 16–18] methods usually fail to reduce Abs Rel under 0.1 due to the inherent scale ambiguous problem of monocular cameras. Although the stereo methods [20] can alleviate this absence, they also perform less satisfactorily, as the re-project photo-metric loss of camera-only methods has limitations. Besides, the camera-fusion methods [3, 6, 11] also do not yield high performance. Conversely, our proposed EffisDepth and EffisDepth-large thoroughly exploits the low-cost 4-beam LiDAR to achieve Abs Rel of 0.073 and 0.071, significantly exceeding the previous excellent 4-beam method [3, 6] and setting state of the art with self-supervised mono-fusion methods. The visualization results of EffisDepth-large in Fig. 4 illustrate that our method has a marked improvement in predicting close and long shots.

Table 2: **Ablation.** Results for ablating variant components of our model. M and L, respectively, indicate monocular images and sparse LiDAR signals. R50 and R18, respectively, ResNet50 and ResNet18.

Modality	Feature Extractor		Params	Params Growth	Sparse Loss	Abs Rel
	Depth	Pose				
M	R50	R50	25.6×2M	-	✗	0.110
M + L	R50	R50	25.6×2M	-	✓	0.105
M + L	R50, SFE	R50	(25.6×2+1.4)M	2.73%	✓	0.076
M + L	R50, SFE	R50, SFE	(25.6×2+1.4×2)M	5.47%	✓	0.073
M + L	R18, SFE	R18, SFE	(11.7×2+1.4×2)M	-48.83%	✓	0.075
M + L	R50, SFE-large	R50, SFE-large	(25.6×2+22.7×2)M	88.67%	✓	0.071

Ablation Study. We perform a group of experiments to evaluate our proposed model. We take out SFE from the architecture, where the results are shown in the first two lines of Table 2. Compared with the single modality method, directly using the Sparse Loss from 4-beam LiDAR marginally improves performance by 4.5%. It is mainly because the 4-beam LiDAR points are too sparse to provide the supervised signals. Next, we only add an SFE to the depth network, and the performance is significantly improved by about 30%. It is worth pointing out that the parameter of an SFE is only 1.4M (5.5% of the size of ResNet50). SFE only increases 2.7% parameters of the monocular network, unlike [5, 6] using ResNet as the encoder of the additional modality such that doubling the parameters of the monocular network. To understand the capability of the whole framework, We test the two other combinations: SFE with ResNet18 and SFE-large with ResNet50. As shown in the 4-6 th lines in Table 2, the first combination also outperforms all the previous works listed in Table 1. Meanwhile, its parameters decrease approximately 50% of the single modality method. Moreover, the second combination arrives the best performance among results in Table 2. Overall, the experiments prove the effectiveness of each components of our method. Finally, we choose the combination of R50 and SFE for EffisDepth due to the high cost performance.

Table 3: Results for our model with different density LiDAR as input on the KITTI dataset.

	1-beam	2-beam	3-beam	4-beam	16-beam
Abs Rel	0.091	0.087	0.084	0.073	0.064

Analysis of the capability. To explore the capability of our EffisDepth to handle variant densities of LiDAR, we take 1-,2-,3- and 16-beam LiDAR to replace the 4-beam LiDAR inputs in our EffisDepth. As shown in Table 3, even if the model is fed into 1-beam LiDAR points, it still outperforms the latest SOTA camera-only methods [17, 18, 20] including methods using monocular and stereo cameras. When taking the input of 16-beam LiDAR, our model obviously exceeds the latest supervised methods [23] (CVPR 2022, Abs Rel of 0.071) using the ground truth data. The results demonstrate that our method has the capability of processing the LiDAR inputs with different density by a few parameters, even the 16-beam and 1-beam LiDAR.

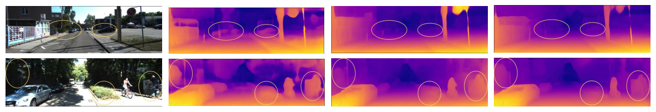

(a) Input (b) ours (c) FusionDepth (d) Monodepth2

Fig. 4: Visualization of depth predictions.

4. CONCLUSIONS

We propose an end-to-end self-supervised framework, EffisDepth, that efficiently processes sparse LiDAR to fuse with monocular images in an innovative way. Experimental results prove that our methods achieve state-of-the-art performance.

5. REFERENCES

[1] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia, "LEGO: learning edge with geometry all at once by watching videos," in *CVPR*, 2018, pp. 225–234.

[2] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow, "Digging into self-supervised monocular depth estimation," in *ICCV*, 2019, pp. 3827–3837.

[3] Vitor Guizilini, Jie Li, Rares Ambrus, Sudeep Pillai, and Adrien Gaidon, "Robust semi-supervised monocular depth estimation with reprojected distances," in *CoRL*, 2019, pp. 503–512.

[4] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark E. Campbell, and Kilian Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," in *ICLR*, 2020.

[5] Sen Zhang, Jing Zhang, and Dacheng Tao, "Towards scale-aware, robust, and generalizable unsupervised monocular depth estimation by integrating imu motion dynamics," *arXiv preprint arXiv:2207.04680*, 2022.

[6] Ziyue Feng, Longlong Jing, Peng Yin, Yingli Tian, and Bing Li, "Advancing self-supervised monocular depth learning with sparse lidar," in *CoRL*, 2022, pp. 685–694.

[7] Jaesung Choe, Kyungdon Joo, Tooba Imtiaz, and In So Kweon, "Volumetric propagation network: Stereo-lidar fusion for long-range depth estimation," *IEEE RA-L*, vol. 6, no. 3, pp. 4672–4679, 2021.

[8] Guangyao Xu, Junfeng Fan, En Li, Xiaoyu Long, and Rui Guo, "Robust and accurate depth estimation by fusing lidar and stereo," *arXiv preprint arXiv:2207.06139*, 2022.

[9] Yu-Kai Huang, Yueh-Cheng Liu, Tsung-Han Wu, Hung-Ting Su, Yu-Cheng Chang, Tsung-Lin Tsou, Yu-An Wang, and Winston H. Hsu, "S3: learnable sparse signal superdensity for guided depth estimation," in *CVPR*, 2021, pp. 16706–16716.

[10] Nguyen Anh Minh Mai, Pierre Duthon, Louahdi Khoudour, Alain Crouzil, and Sergio A Velastin, "Sparse lidar and stereo fusion (sls-fusion) for depth estimationand 3d object detection," *arXiv preprint arXiv:2103.03977*, 2021.

[11] Hemang Chawla, Arnav Varma, Elahe Arani, and Bahram Zonooz, "Multimodal scale consistency and awareness for monocular self-supervised depth estimation," in *ICRA*, 2021, pp. 5140–5146.

[12] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu, "Ghostnet: More features from cheap operations," in *CVPR*, 2020, pp. 1577–1586.

[13] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018, pp. 7132–7141.

[14] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *CVPR*, 2019, pp. 3075–3084.

[15] David Eigen and Rob Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015, pp. 2650–2658.

[16] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon, "3d packing for self-supervised monocular depth estimation," in *CVPR*, 2020, pp. 2482–2491.

[17] Tak-Wai Hui, "Rm-depth: Unsupervised learning of recurrent monocular depth in dynamic scenes," in *CVPR*, 2022, pp. 1665–1674.

[18] Jinwoo Bae, Sungho Moon, and Sunghoon Im, "Monoformer: Towards generalization of self-supervised monocular depth estimation with transformers," *arXiv preprint arXiv:2205.11083*, 2022.

[19] Andra Petrovai and Sergiu Nedevschi, "Exploiting pseudo labels in a self-supervised learning framework for improved monocular depth estimation," in *CVPR*, 2022, pp. 1578–1588.

[20] Juan Luis Gonzalez and Munchurl Kim, "Plade-net: Towards pixel-level accuracy for self-supervised single-view depth estimation with neural positional encoding and distilled matting loss," in *CVPR*, 2021, pp. 6851–6860.

[21] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao, "Deep ordinal regression network for monocular depth estimation," in *CVPR*, 2018, pp. 2002–2011.

[22] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[23] Vaishakh Patil, Christos Sakaridis, Alexander Liniger, and Luc Van Gool, "P3depth: Monocular depth estimation with a piecewise planarity prior," in *CVPR*, 2022, pp. 1600–1611.