Depth-Independent Depth Completion via Least Square Estimation

Xianze Fang, Yunkai Wang, Zexi Chen, Yue Wang, Rong Xiong
Zhejiang University
{fangxzzju,wangyunkai,chenzexi,ywang24,rxiong}@zju.edu.cn

Abstract—The depth completion task[8] aims to complete a per-pixel dense depth map from a sparse depth map. In this paper, we propose an efficient least square based depth-independent method to complete the sparse depth map utilizing the RGB image and the sparse depth map in two independent stages. In this way we can decouple the neural network and the sparse depth input, so that when some features of the sparse depth map change, such as the sparsity, our method can still produce a promising result. Moreover, due to the positional encoding and linear procession in our pipeline, we can easily produce a super-resolution dense depth map of high quality. We also test the generalization of our method on different datasets compared to some outstanding algorithms. Experiments on the benchmark show that our method produces competitive performance.

I. INTRODUCTION

Depth estimation [16] is an important task in many areas, such as robotics and virtual reality. Although existing depth sensors contribute a lot to 3D perception tasks, they still have limitations [9, 27] in directly obtaining a reliable dense depth map in both indoor and outdoor environments. For example, LiDARs can produce accurate distance measurements but their measurements are sparse, while structured light sensors can produce dense depth maps but are constrained to indoor environments. The depth completion task aims at predicting a reliable dense depth map utilizing sparse depth measurements, which can be easily obtained by the existing sensors.

The main challenge of the depth completion task is the extreme sparsity of the depth measurements (e.g. less than 1% pixels have depth measurements). Recently neural network has achieved great success in many image processing areas including the depth completion task. In all these learning-based depth completion algorithms, we can divide them into two categories which are shown in Fig. 1. The first class is a depth-dependent neural network method [15, 21, 10, 14]. The pixels without valid depth values are set to a constant number such as 0. The RGB image is concatenated with the sparse depth map as the input of the deep neural network. These methods can be used in an end-to-end manner, but the neural networks need to resolve the ambiguity between valid and invalid depth measurements, and to learn the features of the sparse depth map such as the sparsity, the noise of the sparse measurements and the distributions of the sparse points. These factors make themselves less flexible for the changing of the sparse depth map. Moreover, the learning space of their neural networks should be large to cover the sparse depth map inputs, which not only increases the parameters of their networks but also limits the generalization abilities.

Another kind of algorithm uses depth-independent neural networks which do not directly use the sparse depth map as the input of the neural networks [5, 6]. Liu. et al. [11] are one of the pioneers who propose a depth-independent Learning Steering Kernel (LSK) method to complete the sparse depth. However, the regression kernels they use are inflexible and too simple to model the complicated images. Therefore, they need to use a depth-dependent CNN to finetune the predicted depth map in addition. Followed LSK, our proposed method is shown in Fig. 1 (b). We use the neural network only to extract the features of the guided RGB image, and then use a analytical solver to fit the sparse depth measurements. In this way when the sparsity or other features of the sparse depth map change, the output of the neural network will not be impacted at all. In addition, to improve the performance and be able to upsample the dense depth map, we add positional encoding layers to the RGB features. Compared with kernels in LSK, our base functions are more flexible. The parameter number of our neural network is small because the network is only used for RGB image feature extraction. We evaluate our method on public datasets compared with other methods. Experiments on benchmarks show that our method outperforms some previous works in accuracy and runtime. In conclusion, our contributions are summarized as follows:

1) The neural network we use is depth-independent which makes our model more flexible for the feature changing
of the sparse depth map input. We propose a base function representation for images which has greater expressiveness than regression kernels.

2) With the extra positional information added, we can get super-resolution dense depth maps of high quality.

3) Our method is light-weighted and fast compared with other methods. With about 1% number of other network parameters, we achieve competitive results on generalization experiments.

We organize our paper as follows: Sec. II will review the related works of the related areas. We will explain our method in detail in Sec. III. Sec. IV will present our experiment results in detail. Finally, we will conclude our paper in Sec. V.

II. RELATED WORKS

A. Monocular Depth Estimation

Many previous works have been done on predicting the depth map from a monocular RGB image. Early works [22, 23] rely on hand-crafted features and probabilistic graphical models. Recently many methods based on supervised models are proposed and make good use of convolutional neural networks. These models are always in end-to-end manner. In addition, some other works [26] use multi-task models to synchronously predict the dense depth maps and segmentation results, which can benefit the performance of each other. Some methods also predict the uncertainty map at the same time, which is used to finetune the predicted depth map. Bhutani et al. [1] proposes unsupervised methods which also demonstrate promising results.

Although many monocular depth estimation methods reach promising results, the task that predicting the dense depth map only from a monocular RGB image is highly ill-posed. Their generalization abilities are limited.

B. Depth Completion

With an extra sparse depth map known, the depth completion task can predict more reliable dense depth maps. The sparse depth measurements can be obtained from visual-SLAM algorithms like ORB-SLAM [17] and Vins-MONO [20], or directly obtained from the distance sensors like LiDARs and structured light sensors. Many methods use end-to-end neural networks to complete the depth map from sparse to dense. Ma et al. [15] get a promising result using an end-to-end convolutional neural network to proceed with the concatenation of RGB image and the sparse depth map. As a depth-dependent method it runs fast but when the features of the input sparse depth change, the performance is not good. Tang et al. [25] prove that depth map can be represented using a series of basis. Takenda et al. [24] propose a non-parametric statistics estimation method to interpolate the sparse depth map, but the steering kernels it use is hand-crafted and inflexible. Liu et al. [11] promote the algorithm using learned steering kernels to replace the hand-crafted steering kernels, which significantly improves the performance. Spatial propagation is a traditional interpolated algorithm. Liu S. et al. [13] used a neural network to predict the affinity matrix of the image and use the affinity matrix to iteratively finetune the output which can be used in many vision tasks. Cheng et al. [4] applied the spatial propagation algorithm in the depth completion task. Park et al. [19] promoted the convolutional spatial propagation network by using a flexible set of neighbor points to replace the fixed set. By doing this, it can learn the affinities of non-local points of the image. The spatial propagation network methods need an iterative regression process and their networks are large, making them inefficient.

In this paper, we propose a differentiable depth-independent network which decouples the RGB image and the sparse depth map. Our model is flexible for the changing of the sparse depth map, and we can upsample a super-resolution dense depth map of high quality.

A. Motivation

Our goal is to predict a dense depth map utilizing the RGB image and the sparse depth map in a decoupling way. [21, 25] have proven that depth map can be represented using a series of basis, which we call base functions. Following this thought, we use neural network to estimate the base functions of the RGB image. Then, like the final layer of a normal neural network, we regard the dense depth map as the linear combination of the base functions. Therefore we propose a least square estimation method to fit the base functions and the sparse depth map. We will describe each part in detail.

B. Neural Network Architecture

Feature Extraction. Traditional CNN detector shows excellent performance in detecting the image regions which
have rich object context information. But in some ill-posed regions which have few textures, it’s difficult to detect the features. We follow the idea in [2] by extracting the features from the RGB image with a pyramid-like CNN. We use 4 different size average pooling blocks in the network to extract different level features. Then we concatenate them and output the feature map of the RGB image. We can describe this part as follows:

\[ h = \phi(c) \]  

(1)

where \( \phi \) refers to the feature extraction network, and \( c \) refers to the RGB value of each pixel.

**Positional Encoding.** Traditionally, we focus on the color information of the pixel but ignore the coordinate information. According to [12], adding coordinate layers to the feature map can improve the performance in many applications including depth prediction. Moreover, according to [7], mapping the coordinate to high dimensions contributes to representing the high-frequency features of the image. These techniques enable better fitting of the depth image. The encoding function we used is as follows:

\[
\gamma(p) = [x, y, y, \sin(2 \pi x), \cos(2 \pi x), \\
\sin(2 \pi y), \cos(2 \pi y), \ldots, \sin(2^{E-2} \pi x), \\
\cos(2^{E-2} \pi x), \sin(2^{E-2} \pi y), \cos(2^{E-2} \pi y)]
\]  

(2)

where \( E \) is the number of dimensions of positional encoding, \( p = (x, y) \) denotes pixel position, \( x \) denotes coordinates along the width, and \( y \) along the height. The output of the positional encoding \( \gamma(p) \) is then concatenated with the feature map as the input of the base functions estimation part.

**Base Functions Estimation.** We concatenate the feature map and the encoded positional map as the input of a fully-connected neural network (FCNN). The outputs of the FCNN are vectors that are one-to-one corresponding to the pixels. We call these vectors the base functions. For each pixel, we have:

\[
f_k = \psi(\phi(c), \gamma(p))
\]  

(3)

where \( k \) refers to the index of each pixel, \( \psi \) refers to the FCNN, and \( f_k \) is a \( N \times 1 \) vector.

**C. Least Square Optimization and IRLS Algorithm**

**Solving the Least Square Problem.** After feature extraction and base function estimation, we get a base vector \( f_k \in \mathbb{C}^{N \times 1} \) of each pixel, which is called base function. Assuming there are \( m \) pixels in the RGB picture, of which \( n \) pixels have a known depth value. The known sparse depth is \( \mathbf{D} \in \mathbb{C}^{m \times 1} \).

For the whole picture, we use \( \mathbf{F}_d \in \mathbb{C}^{m \times (N+1)}, m > (N+1) \) to represent the base functions of all the pixels and \( \mathbf{F}_s \in \mathbb{C}^{m \times (N+1)}, n > (N+1) \) for depth-known pixels, while \( \mathbf{D}_d \in \mathbb{C}^{m \times 1} \) for depth values for all the pixels and \( \mathbf{D}_s \in \mathbb{C}^{m \times 1} \) for known depth values. We have the equations below:

\[
\mathbf{D}_d = \mathbf{F}_d \cdot \mathbf{W}^T
\]  

(6)

\[
\mathbf{D}_s = \mathbf{F}_s \cdot \mathbf{W}^T
\]  

(7)

As described in Sec. III-B, we already have the \( \mathbf{F} \), which is sampled from \( \mathbf{F}_d \) according to the known depth pixels. So now our task is to solve Eqn. 7 which is an over-defined matrix equation. This equation can also be written as a least square estimation problem as follows:

\[
\mathbf{argmin}_W \| \mathbf{D}_s - (\mathbf{F}_s \cdot \mathbf{W}^T) \|.
\]  

(8)

**Algorithm 1 SVD-decomposition Solving Algorithm**

**Require:** \( \mathbf{D}_s \in \mathbb{C}^{m \times 1}, \mathbf{F}_s \in \mathbb{C}^{m \times (N+1)} \)

**Ensure:** \( \mathbf{argmin}_W \| \mathbf{D}_s - (\mathbf{F}_s \cdot \mathbf{W}^T) \| :\

1: if rank(\( \mathbf{F}_s \)) = r = (N+1) < n then

2: \( \mathbf{SVD}(\mathbf{F}_s) = (U_1, U_2) \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{V}^H \)

3: \( \mathbf{W} = \mathbf{V} \Sigma_1^{-1} U_1^H \mathbf{D}_s \)

4: else\( \mathbf{rank}(\mathbf{F}_s)) = r < (N+1) < n \) then

5: \( \mathbf{SVD}(\mathbf{F}_s) = (U_1, U_2) \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{V}_1^H \mathbf{V}_2^H \)

6: \( \mathbf{W} = \mathbf{V}_1 \Sigma_1^{-1} U_1^H \mathbf{D}_s \)

7: end if

**Algorithm 2 Iterative Re-weighted Least Square Algorithm**

**Require:** \( \mathbf{D}_s \in \mathbb{C}^{m \times 1}, \mathbf{F}_s \in \mathbb{C}^{m \times (N+1)} \)

**Ensure:** \( \mathbf{W} = \mathbb{W}_{\text{init}}[w_i], (i = 0, 1, 2, ..., N) ;

1: \( \mathbf{W} = \mathbb{W}_{\text{init}} = (\mathbf{F}_s \cdot \mathbf{F}_s^T)^{-1} \cdot (\mathbf{F}_s \cdot \mathbf{D}_s) \)

2: while not meet the stopping criterion do

3: \( e = \text{abs}(\mathbf{D}_s - \mathbf{F}_s \cdot \mathbf{W}) \)

4: \( e' = (\max(\text{threshold}, e))^{-1} \)

5: \( \text{Weight} = \text{diag}(e') \)

6: \( \mathbf{W} = (\text{Weight} \cdot \mathbf{F}_s \cdot \mathbf{F}_s^T)^{-1} \cdot (\text{Weight} \cdot \mathbf{F}_s \cdot \mathbf{D}_s) \)

7: end while

We can use Moore-Penrose inverse of matrix \( \mathbf{F}_s \) to solve the least square regression problem. According to Moore-Penrose inverse theory, we have:

\[
\mathbf{W}^T = \mathbf{F}_s^T \cdot \mathbf{D}_s = (\mathbf{F}_s \cdot \mathbf{F}_s^T)^{-1} \cdot (\mathbf{F}_s^T \cdot \mathbf{D}_s)
\]  

(9)

In the training process, we should keep the least square estimation differentiable so we can train the model end-to-end. But in the practical try, we find that the inverse function in Pytorch may easily reach a numerical instability condition because the dimension of the matrix \( \mathbf{F}_s \cdot \mathbf{F}_s^T \) is too big for the algorithm to converge so the matrix may be treated as a singular matrix. So we use the SVD-decomposition algorithm to solve the least square estimation problem which is more numerically stable in our training process. The process is shown in Alg. 1.
IRLS Algorithm. In real-world applications, there might be some noise in the sparse depth measurements. In our pipeline, we can easily apply an iterative re-weighted least square algorithm (IRLS) to reduce the influence of the noise. The algorithm is shown in Alg. 2. The IRLS algorithm is optional in the whole pipeline.

D. Loss Functions

Our training loss consists of three parts. We will explain each part in detail next.

1) **Loss on predicted depth.** We directly estimate the error between the predicted depth and ground truth depth:

\[ \ell_{direct} = ||D_{gt} - D_{predict}||_t \]

When \( t = 1 \) it represents \( \ell_1 \) loss and \( t = 2 \) represents \( \ell_2 \) loss. We use \( \ell_1 \) loss in our method in our training process.

2) **Loss on depth gradient.** Previous work has shown that applying depth gradient loss in training can significantly improve the performance. The gradient of the depth map contains information about the edges of objects. Therefore this loss supervises the boundaries in predictions and makes them sharper.

\[ \ell_{grad} = ||\nabla D_{gt} - \nabla D_{predict}||_1 \]

3) **Loss on regularization** We regard the depth value of each pixel as a weighted sum of base functions. To make the most use of each base function and avoid over-fitting, we apply a regularization loss function in training.

\[ \ell_{regular} = \| \sum_{i=0}^{N} w_i \|_1 \]

The total loss is the weighted combination of the above loss functions:

\[ \mathcal{L} = \alpha \ell_{direct} + \beta \ell_{grad} + \gamma \ell_{regular} \]

where \( \alpha, \beta, \gamma \) are the hyper-parameters set manually.

IV. EXPERIMENTS

In this section, we will first introduce our results on the NYUDepthv2 benchmark compared with other methods. Secondly, we will show the ablation studies. Thirdly we will show our ability to super-resolution the dense depth map. Last but not least, we will compare the generalization ability of our method with other algorithms.

A. Experiments on Benchmark

**NYUDepthv2 Dataset.** The NYUDepthv2 dataset [18] contains about 407k RGB frames with depth maps in 464 different indoor scenes. There are 1449 labeled frames in it whose depth is labeled densely. A small set including 654 images is officially chosen for the benchmark. The resolution of the raw image in NYUDepthv2 is 480 × 640. We first downsample the image to half the resolution and then center-crop them to 228 × 304 in training and evaluation.

**Experiment Setup.** We implement our method using PyTorch 1.8 with CUDA 10.2. We train our model on the NYUDepthv2 dataset using an Nvidia 2080Ti GPU. The weights of the network are initialized randomly. We use a small batch size 2 and train the model for 100 epochs. We set the learning rate to 1 × 10^-3 for the first 30 epochs and then reduce it to 1e-4 for the next 30 epochs and then reduce it to 1e-5. We set the weight decay to 5e-4 for regularization.

**Evaluation Metrics.** We now adopt the standard evaluation metrics. Assuming that the ground truth depth value at position \( x_i \) is \( \hat{d}_i = D_i(x_i) \), the predicted depth value at position \( x_i \) is \( \hat{d}_i = \hat{D}_i(x_i) \). The evaluation metrics are specified as follows:

1) **Root Mean Squared Error (RMSE):**

\[ \text{RMSE} = \sqrt{\frac{1}{M} \sum_i (\hat{d}_i - d_i)^2} \]

2) **Mean Absolute Relative Error (REL):**

\[ \text{REL} = \frac{1}{M} \sum_i \frac{|\hat{d}_i - d_i|}{|d_i|} \]

3) **Threshold (\( \delta \in [1.25, 1.25^2, 1.25^3] \)):**

Percentage of \( \hat{d}_i \) s.t. \( \max(\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}) < \delta \)

**Benchmark Evaluation.** We evaluate our model on the NYUDepthv2 benchmark. Tab. 1 shows the comparison results of accuracy. Our model outperforms Sparse-to-Dense.
(S2D), the interpolated result of Steering Kernels (SK) and the interpolated result of Learning Steering Kernels (LSK).

B. Ablation Study

Different Dimensions of Base Functions. We test different dimensions of the base functions \( f_k \) in Eqn. 3. We set the dimension number \( N \) from 16 to 512 while keeping other settings the same. Our batch size to 2 and learning rate is \( 1e-4 \). We only add 2 raw position layers in these experiments for comparison. Each epoch contains \( 6k \) randomly chosen pictures in the NYUDepthv2 dataset. The evaluation results after each epoch are shown in Fig. 7. The bigger the number \( N \) is, the faster and better the model converges. When \( N = 512 \), the model performs the best. But for two reasons we can’t increase the \( N \) infinitely. Firstly the dimension number of the base functions can not exceed the number of known sparse depth points, otherwise the Eqn. 4 can’t be solved. Secondly, as the dimension number of \( f_k \) increases, the least square function becomes harder to be solved, and it is more likely to meet numerical error. As a trade-off, we choose \( N = 128 \) in the rest of our experiments.

Positional Encoding. To prove that the positional encoding part contributes to the result, we conduct an ablation study on this part. We change the parameters \( E \) in Eqn. 2. We set the batch size to 2, learning rate to \( 1e-4 \) and \( N = 128 \). The results are shown in Fig. 5. No Position means no positional information is used. Only position means we only use the two positional layers which are the coordinates of the pixels. Position encoding \( x \) means we use positional encoding and set \( E = x \). As we can see in the figure, with positional encoding layers, the model converges faster and better. But there is no big difference between different positional encoding layers. To make our model more efficient, we choose \( E = 5 \) in the rest of the experiments.

C. Change of the Sparse Depth Maps

Different Numbers of Known Depth Measurements. In our method, the sparse depth measurements are used for least square estimation after all the neural network processes. So when we change the number of known depth measurements, the performance of the neural networks will not change at all. It will only influence the least square estimation process. With more known depth measurements, the final result ought to be better. We change the numbers of sampled points of known depth measurements and evaluate the models on the NYUDepthv2. The RMSE and REL results are shown in Fig. 7. As the numbers of known depth measurements increase, the results of S2D don’t change much, while LSK’s and our method’s results increase obviously.

Sparse Points with Noise. In real-world applications, the sparse depth measurements may have noise. Because our method is based on the least square algorithm, it is built to reduce the impact of the noise. We can use IRLS algorithm to decrease the influence of the noise of the sparse depth input. To prove that, we randomly choose 300 points of the total 1000 known depth points and add noise that is uniformly distributed in \([0, 1]\). Only can our method apply IRLS algorithm. Results are shown in Tab. II. Noise obviously impacts the results of all these methods, and our method is affected the most. With IRLS algorithm applied, we can reduce the impact of the noise.

| Method | Error(\%) | Change |
|--------|-----------|--------|
|        | RMSE | REL | RMSE change | REL change |
| S2D    | 0.201 | 0.045 | - | - |
| S2D+Noise | 0.346 | 0.107 | +0.145 | +6.20% |
| LSK    | 0.169 | 0.031 | - | - |
| LSK+Noise | 0.358 | 0.114 | +0.189 | +8.30% |
| Ours   | 0.129 | 0.027 | - | - |
| Ours+Noise | 0.366 | 0.120 | +0.237 | +9.33% |
| Ours+Noise+IRLS | 0.331 | 0.089 | +0.203 | +6.19% |

D. Super-Resolution Experiments

Kernel Visualization Comparison. In Learning Steering Kernels (LSK) algorithm [11], which is also depth-independent, each pixel has a regression kernel predicted from the RGB image. This kernel measures the affinity between the chosen pixel and the surrounding pixels. In our
method, we have the base function of each pixel estimated from the RGB image. We calculate the correlation coefficient between the exact pixels and other pixels by using dot product to draw kernels as well. The result is shown in Fig. 6. The kernels in our method are more flexible and can cover the whole image while the kernels in the LSK can only be oval-like and only concern the local feature.

**Super-Resolution Results Comparison.** With the help of the two-stage strategy in our method, we can upsample the dense depth map in a reliable way. After feature extraction process, we get a feature map of the RGB image $\Phi(c)$. We first interpolate the feature map to our target resolution and then add the positional encoding layers which are rebuilt according to the new resolution into the feature map. Then we follow the normal step of our method. In the experiment, we first downsample the RGB image to $120 \times 160$ and get its associated sparse depth map as the input. Then we upsample the dense depth map to $360 \times 480$. For S2D and LSK algorithm, we simply upsample their output dense depth map using bi-linear interpolation. Besides RMSE and REL error, we also evaluate the Soft Edge Error (SEE) metric on pixels belonging to object boundaries, defined as the minimum absolute error between the predicted depth map and the ground truth, according to [3]. The visualization is shown in Fig. 8 and the evaluation results are shown in Tab. III. The edges of upsampled dense depth maps predicted by our method are more clear, and the errors are smaller.

| Method | Error(\%) | Accuracy(\%) |
|--------|-----------|--------------|
|        | RMSE      | REL          |
| S2D    | 0.487     | 1.367        |
| LSK$^2$| 0.059     | 0.251        |
| Ours   | 0.046     | 0.215        |

$^1$ All take 1000 sampled points.
$^2$ The result of Learning Steering Kernel part.

**E. Generalization wrt. Different Datasets**

We validate the model trained on the NYUDepthv2 dataset directly on a new dataset DIODE (Dense Indoor and Outdoor DEpth). DIODE is a dataset contains diverse high-resolution color images with accurate, dense, and far-range depth measurements. There are 325 images in the indoor validation subset and 446 images in the outdoor validation subset. In this part, we also test the SOTA algorithm CSPN method [4].

The evaluation data is shown in Tab. IV, and the visualization result is shown in Fig. 9.

Our model contains a convolutional neural network and a fully connected neural network. The structures of both are simple and straight. We compare the parameter numbers of models of different methods. We also compare the runtimes of all these methods, using the same resolution $228 \times 304$. Comparison result is shown in Tab. V. Our method is the most light-weighted and fast.

**TABLE III**

EVALUATION ON UPSAMPLE EXPERIMENT

| Method | Error(\%) | Accuracy(\%) |
|--------|-----------|--------------|
|        | RMSE      | REL          |
| S2D    | 0.487     | 1.367        |
| LSK$^2$| 0.059     | 0.251        |
| Ours   | 0.046     | 0.215        |

$^1$ All take 1000 sampled points.
$^2$ The result of Learning Steering Kernel part.

**V. CONCLUSIONS**

In this paper, we propose a depth-independent least square estimation network to complete the sparse depth map to a dense one. We decouple the RGB image and the sparse depth map because the neural network in our model only extract the features of the RGB image. With experiments on the public datasets, we show that the performance of our method is promising both in accuracy and runtime. Besides, our model has a good performance when some features of the sparse depth map change. Moreover, our model can generalize well compared with other methods.
REFERENCES

[1] Vishal Bhutani et al. “Unsupervised Depth and Confidence Prediction from Monocular Images using Bayesian Inference”. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2020, pp. 10108–10115.

[2] Jia-Ren Chang and Yong-Sheng Chen. “Pyramid stereo matching network”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 5410–5418.

[3] Chuangrong Chen, Xiaozhi Chen, and Hui Cheng. “On the Over-Smoothing Problem of CNN Based Disparity Estimation”. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). Oct. 2019.

[4] Xinjing Cheng, Peng Wang, and Ruigang Yang. “Depth estimation via affinity learned with convolutional spatial propagation network”. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, pp. 103–119.

[5] Xinjing Cheng, Peng Wang, and Ruigang Yang. “Learning Depth with Convolutional Spatial Propagation Network”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 42.10 (2020), pp. 2361–2379. DOI: 10.1109/TPAMI.2019.2947374.

[6] Xinjing Cheng et al. “Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion”. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. 07, 2020, pp. 10615–10622.

[7] Jan Czarnowski et al. “Deepfactors: Real-time probabilistic dense monocular slam”. In: IEEE Robotics and Automation Letters 5.2 (2020), pp. 721–728.

[8] Junjie Hu et al. “Deep Depth Completion: A Survey”. In: arXiv preprint arXiv:2205.05335 (2022).

[9] Xizuan Huang et al. “Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion”. In: IEEE Transactions on Image Processing 29 (2019), pp. 3429–3441.

[10] Maximilian Jaritz et al. “Sparse and dense data with cnns: Depth completion and semantic segmentation”. In: 2018 International Conference on 3D Vision (3DV). IEEE. 2018, pp. 52–60.

[11] Lina Liu et al. “Learning steering kernels for guided depth completion”. In: IEEE Transactions on Image Processing 30 (2021), pp. 2850–2861.

[12] Rosanne Liu et al. “An intriguing failing of convolutional neural networks and the coordconv solution”. In: arXiv preprint arXiv:1807.03247 (2018).

[13] Sifei Liu et al. “Learning affinity via spatial propagation networks”. In: Advances in Neural Information Processing Systems 30 (2017).

[14] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. “Self-Supervised Sparse-to-Dense: Self-Supervised Depth Completion from LiDAR and Monocular Camera”. In: 2019 International Conference on Robotics and Automation (ICRA). 2019, pp. 3288–3295. DOI: 10.1109/ICRA.2019.8793637.

[15] Fangchang Ma and Sertac Karaman. “Sparse-to-dense: Depth prediction from sparse depth samples and a single image”. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE. 2018, pp. 4796–4803.

[16] Yue Ming et al. “Deep learning for monocular depth estimation: A review”. In: Neurocomputing 438 (2021), pp. 14–33.

[17] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: IEEE transactions on robotics 31.5 (2015), pp. 1147–1163.

[18] Pushmeet Kohli Nathan Silberman Derek Hoiem and Rob Fergus. “Indoor Segmentation and Support Inference from RGBD Images”. In: ECCV. 2012.

[19] Jinsun Park et al. “Non-local spatial propagation network for depth completion”. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16. Springer. 2020, pp. 120–136.

[20] Tong Qin, Peiliang Li, and Shaojie Shen. “Vins-mono: A robust and versatile monocular visual-inertial state estimator”. In: IEEE Transactions on Robotics 34.4 (2018), pp. 1004–1020.

[21] Chao Qu, Ty Nguyen, and Camillo Taylor. “Depth Completion via Deep Basis Fitting”. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). Mar. 2020.

[22] Ashutosh Saxena, Sung H Chung, Andrew Y Ng, et al. “Learning depth from single monocular images”. In: NIPS. Vol. 18. 2005, pp. 1–8.

[23] Ashutosh Saxena, Min Sun, and Andrew Y Ng. “Make3d: Learning 3d scene structure from a single still image”. In: IEEE transactions on pattern analysis and machine intelligence 31.5 (2008), pp. 824–840.

[24] Hiroyuki Takeda, Sina Farsi, and Peyman Milanfar. “Kernel regression for image processing and reconstruction”. In: IEEE Transactions on Image processing 16.2 (2007), pp. 349–366.

[25] Chengzhou Tang and Ping Tan. “Ba-net: Dense bundle adjustment network”. In: arXiv preprint arXiv:1806.04807 (2018).

[26] Simon Vandenhende, Stamatiou Georgoulis, and Luc Van Gool. “Mti-net: Multi-scale task interaction networks for multi-task learning”. In: European Conference on Computer Vision. Springer. 2020, pp. 527–543.

[27] Yinda Zhang and Thomas Funkhouser. “Deep depth completion of a single rgb-d image”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 175–185.