IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 16, 2020, accepted September 13, 2020, date of publication September 18, 2020,
date of current version September 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3024650

FMPN: Fusing Multiple Progressive CNNs for Depth Map Super-Resolution

SHUAIHAO LI[1,2], BIN ZHANG[3], (Member, IEEE), WEIPING ZHU[4], (Member, IEEE), AND XINFENG YANG[4]

[1]Research Center for International Business and Economy, Sichuan International Studies University, Chongqing 400031, China
[2]International Business School, Sichuan International Studies University, Chongqing 400031, China
[3]Department of Computer Science, City University of Hong Kong, Hong Kong 999077, China
[4]School of Computer Science, Wuhan University, Wuhan 430072, China

Corresponding author: Shuaihao Li (lishuaihao@whu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFC1604000 and Grant 2016YFB1101702, in part by the Nature Science Foundation of China under Grant 61772573, and in part by the National Mapping and Geographic Information Bureau of China under Grant SIDT20170101.

ABSTRACT The Convolution Neural Network (CNN) is widely used in the super-resolution task of depth map. However, the ones with simple architecture and high efficiency generally lack accuracy, while the ones with high accuracy demonstrate low efficiency and training difficulties due to their over-deep level and complex architecture. We propose a depth map super-resolution fusion framework. This framework fuses multiple Progressive Convolution Neural Networks (PCNNs) with different architectures by a pixel-wise Partial Differential Equation (PDE). Each individual PCNN uses progressive learning and deep supervising to construct a mapping from low resolution space to high resolution space. The PDE model automatically classifies and processes the high-resolution depth maps with different feature output by fusing multiple PCNNs. The fusion term in PDE is used to preserve or integrate the complementary features of the depth maps, and the divergence term in PDE is used to remove noise to improve the spatial accuracy and visual effect of the final output depth map. This method enables simple structured Neural Networks with high accuracy, high efficiency and relatively simple network training for depth map super-resolution.

INDEX TERMS Depth map super-resolution, progressive convolution neural network, partial differential equation, fusion network.

I. INTRODUCTION

The Time of Flight (ToF) sensor has a very low spatial resolution due to its large size of single-pixel photosensitive element, which makes it difficult to meet the requirements of high-resolution depth map for practical applications such as three-dimensional reconstruction [1]. Therefore, since the advent of ToF sensor, the research on super-resolution of depth map acquired by ToF sensor has become a research hotspot in the field of computer vision, and various algorithms have emerged. These algorithms can be roughly classified into five categories. The first is filter-based algorithm [2]–[6], which usually has the advantages of speed and efficiency, but there are often some bad phenomena such as edge blurring or loss of structural details caused by over-smoothing in the super-resolution results. The second type is based on the sparse representation [7]–[9], which evidences high accuracy, but its calculation speed is usually slow, and a large training sample set is needed. Therefore, it is more suitable for off-line image preprocessing without real-time requirements. The third type is based on deep fusion [10]–[13]. This kind of algorithm can preserve the detailed information of depth map very well, but it needs to obtain motion estimation and displacement labeling of multi-frame depth maps first. Due to its high time complexity with difficulties to meet real-time requirements, it is limited for super-resolution multiples. The fourth type is based on the optimization [14]–[19]. This kind of algorithm ensures high-accuracy but results in some shortcomings such as large amount of calculation and poor real-time performance. The fifth type is based on deep learning [20]–[23]. This kind of algorithm is the mainstream image super-resolution

The associate editor coordinating the review of this manuscript and approving it for publication was Haiyong Zheng.

algorithm at present, and its reconstruction effect is generally superior to other types of algorithms.

The Convolution Neural Network (CNN) is the most advanced network architecture in deep learning. In recent years, many CNN-based neural networks [22]–[26] have been used for image super-resolution tasks. Some of them adopt simple network architecture and few layers, such as SRCNN [22] only utilizing three Convolution layers and obtaining learn rich image features. Such networks are relatively efficient and easy to train. But the accuracy of these networks is lower. Some neural networks [27] adopt hundreds or even thousands of layers and complex structures to obtain high accuracy, but it is difficult to fine-tune and optimize network parameters in training, resulting in low efficiency which is difficult to improve under the huge network scale. In order to solve the above problems, the Progressive Convolution Neural Network (PCNN) [28] is proposed, which fuses multiple convolution neural networks for image super-resolution. Compared with a single CNN, under the condition that the number of layers is roughly the same, PCNN has been improved in various evaluation metrics to a certain extent. However, for the deeper individual CNN, PCNN does not have performance advantages.

In this paper, we propose a fusion framework for depth map super-resolution by fusing multiple progressive convolution neural networks (FMPN). The FMPN fuses multiple PCNNs with different architectures by pixel-wise Partial Differential Equation (PDE) model. The PDE model enables the neural networks with simple structure and fewer layers to outperform deeper neural networks in accuracy and efficiency of super-resolution. Each individual PCNN constructs a mapping from low resolution to high resolution space. Since the depth map output from each PCNN might contain some random features such as noises and some complementary features relative to other depth maps, the PDE model are applied to filter noise, reserve complementary features or integrate them into the depth map final output. The experimental results show that, for the fusion scheme with relatively simple structure and less layers, the proposed fusion network is superior to many other state-of-the-art super-resolution algorithms and networks in terms of objective evaluation and subjective visual quality.

II. RELATED WORK

A. PROGRESSIVE CONVOLUTION NEURAL NETWORK (PCNN)

As shown in Figure 1, PCNN fuses two individual neural networks in turn, in which the output of the first network acts as the input of the second network and the output of the second network is the final output of PCNN. Each individual network consists of three parts: feature extraction and presentation layer, non-linear mapping layer and super-resolution reconstruction layer.

In theory, each PCNN can fuse multiple individual networks and progressively learn the high-frequency features output from each individual network. Thus, the similarity between the output of the last individual network and the Ground Truth will become the highest, so as to achieve the super-resolution reconstruction of low-resolution depth map. The corresponding PCNN model can be expressed by equation (1).

$$\hat{\beta}_i = F_{\{S_1,S_2,\cdots,S_M\}}(\alpha_i) = s_M(\cdots s_2(s_1(\alpha_i))), \quad (1)$$

where, i denotes the number of training samples $\{i = 1,\ldots,N\}$, M denotes the number of individual networks in PCNN, α denotes the input low-resolution depth map, F denotes the output of PCNN, and $\hat{\beta}_i$ denotes the predicted value output from the *ith* low-resolution depth map α_i.

However, in practice, if the weights of some individual networks are not frozen, PCNN is equal to a deeper individual network with the same number of layers as the sum of the individual networks. Therefore, PCNN usually consists of only two individual networks, and the weights of one of the individual networks needs to be frozen. All learnable parameters in PCNN depend on the individual network that is not frozen. As shown in Figure 1, the weights of the orange individual network are frozen, while the weights of the green individual network can be learned and optimized in training.

B. PARTIAL DIFFERENTIAL EQUATION (PDE)

The fusion of depth maps based on Partial Differential Equation (PDE) [29] is a frame-by-frame evolution process of the input multi-frame depth maps by the fusion term of PDE. The related features in the current depth map are transferred to the next depth map through the fusion term of PDE, and the complementary features in each depth map are gradually superimposed into the final depth map. The continuous evolution equation for all depth maps can be expressed as

$$\frac{\partial U_i}{\partial t} = \mathrm{div}(\mathrm{D}_i\nabla U_i) - \beta_i \mathrm{div}(g_F(|\nabla U|_{\max})\nabla U_{\max}) + \gamma\,\mathrm{div}(g_R(\nabla U_i, U_k^{t=0})\nabla U_i), \quad (2)$$

where, the three terms on the right of the equation represent the diffusion term, the fusion term and the regularization term in turn. The diffusion term is used to suppress noise, and div() is the divergence operator. D is the gradient matrix; i represents the current depth map; U represents the gray levels of the pixel. The fusion term, i.e. inverse diffusion term, is used to inject the relevant features into the current depth map from the other ones. The relevant features are provided by the depth map corresponding to the maximum absolute value of the gradient. g_F is a non-incremental function of the absolute gradient value, which is used to modulate the fusion quantity. Here, $g_F(|\nabla U|_{\max}) = 1$ is a constant positive function that provides an isotropic fusion. $|\nabla U|_{\max}$ represents the depth map corresponding to the maximum absolute value of the gradient. β_i denotes a positive weight parameter, which is used to set the important degree of the fusion term relative

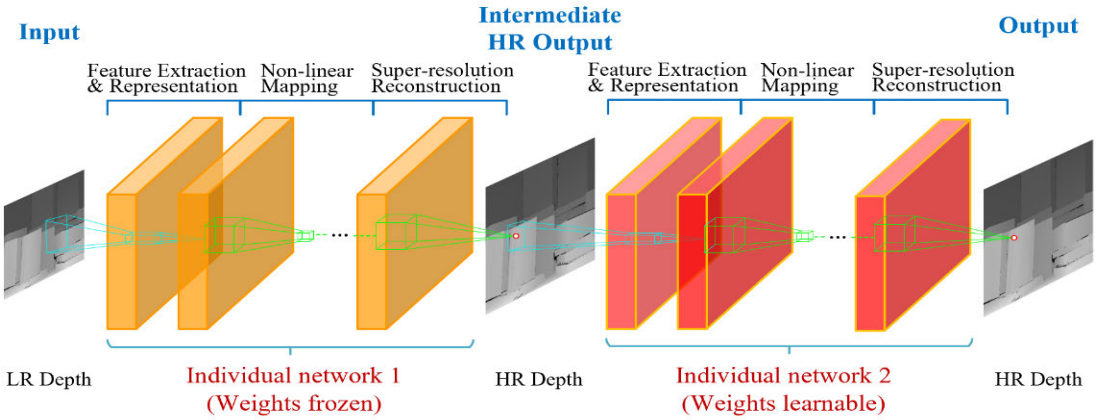


FIGURE 1. Basic framework of Progressive Convolution Neural Network (PCNN).

to the diffusion term. β_i can be expressed as

$$\beta_i = \begin{cases} 0, & \text{if } i = \max \\ \beta \in [0, 1], & otherwise, \end{cases} \tag{3}$$

In the inverse diffusion, the maximum gradient of each pixel is automatically detected. If the maximum absolute value of the gradient exists in the current depth map, the current pixel value will not be updated, $\beta_i = 0$. Otherwise, if the maximum exists in another depth map, the gradient information is injected into the current depth map through the inverse diffusion process.

The third term of (2) is a regularization term, which is used to limit the degree of inverse diffusion to control oscillation. γ represents a positive weight parameter, and g_R is a piecewise linear function. when $g_R = 1$, g_R is used to constrain the extreme value of the gradient by limiting the oscillation of the pixel gray value in each depth map to the range between the minimum value and the maximum value.

Each evolution process produces one output. In fact, the fusion process is the continuous convergence of the root mean square error (RMSE) among the output values. The termination of the convergence process can be achieved by setting the number of iterations or calculating the distance between the output values and comparing it with the preset threshold.

Before using the diffusion equation in a discrete image domain, the numerical value needs to be discretized first. The discretization process can be carried out by calculating the gradient first and then the divergence.

In the fusion term of (2), the discretization form used to calculate the maximum absolute value of the gradient for the nearest neighborhood can be expressed as

$$\frac{\partial U_i}{\partial t} = -\beta_i \left[D_x^+(U_{\max}) - D_x^-(U_{\max}) \right], \tag{4}$$

where

$$D_x^{\pm}(U) = \pm \frac{U(x \pm dx) - U(x)}{dx}. \tag{5}$$

In the regularization term of (2), the discretization form of g_R can be expressed as

$$g_R(D_x^+(U_i), D_x^+(U_k^{t=0})) = \begin{cases} \dfrac{D_x^+(U_i) - \min_k \left[D_x^+(U_k^{t=0}), 0 \right]}{D_x^+(U_i)}, \\ \quad \text{if } D_x^+(U_i) < \min_k \left[D_x^+(U_k^{t=0}), 0 \right] \\ \dfrac{D_x^+(U_i) - \max_k \left[D_x^+(U_k^{t=0}), 0 \right]}{D_x^+(U_i)}, \\ \quad \text{if } D_x^+(U_i) > \max_k \left[D_x^+(U_k^{t=0}), 0 \right] \\ 0, \quad \text{otherwise.} \end{cases} \tag{6}$$

$$\max_x^{\pm} = \arg\max_j \left(\left| D_x^{\pm} \left[U_j(x) \right] \right| \right) \tag{7}$$

where $\max_x^{\pm}$ represents the depth map where the forward and backward maximum difference absolute values of the current pixel gradient are located.

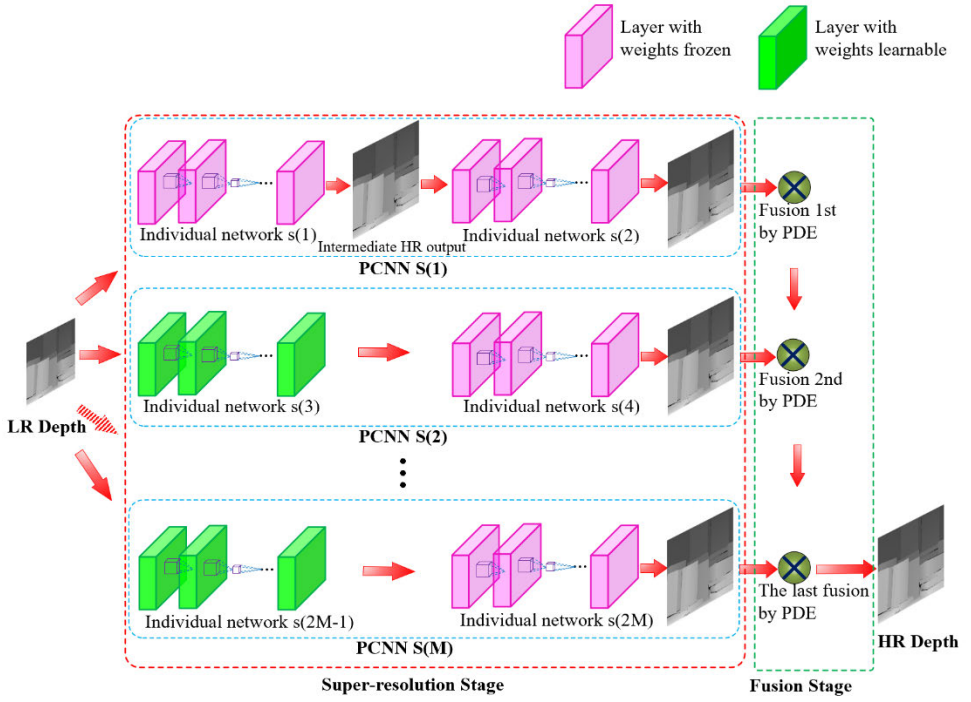
$$\beta_{ix}^{\pm} = \begin{cases} 0 & i = \max_x^{\pm} \\ \beta \in [0, 1] & \text{otherwise} \end{cases} \tag{8}$$

$D_x^+(U_{\max}) - D_x^-(U_{\max})$ is the discretization form of $\text{div}(\nabla U_{\max})$ in (4). In the discretization process, the forward and backward maximum difference absolute values $\max_x^+$ and $\max_x^-$ of the current pixel are detected first. If $\max_x^+$ or $\max_x^-$ exists in the current depth map, then $\beta_{ix}^+ = 0$ or $\beta_{ix}^- = 0$. Otherwise, $\beta_{ix}^+ \in [0, 1]$ or $\beta_{ix}^- \in [0, 1]$. Since $\max_x^+$ and $\max_x^-$ may exist in different depth maps, β_i may have two values at the same time. The divergence can be obtained by calculating the difference between the two values of β_i.

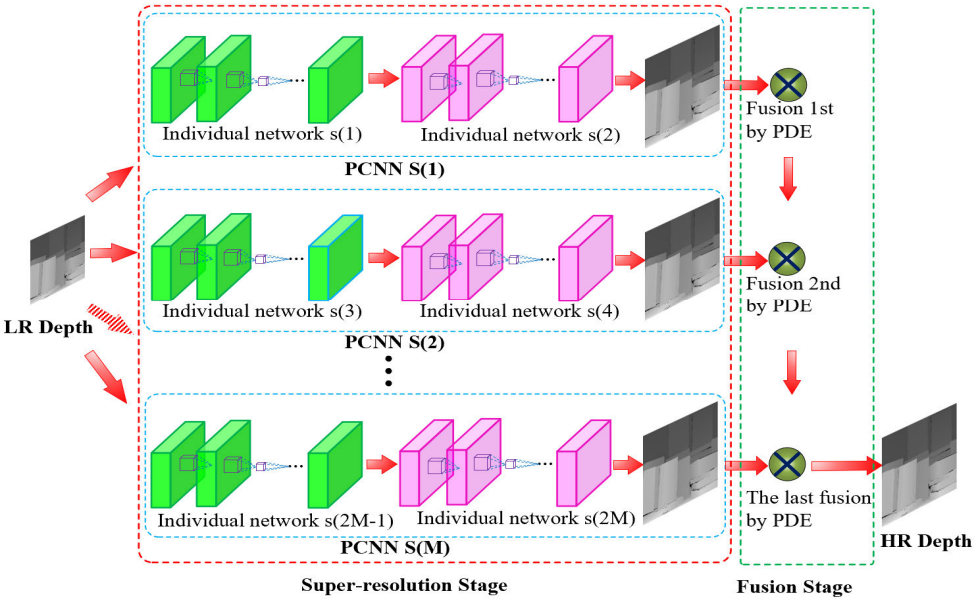
III. PROPOSED METHOD

A. FUSING MULTIPLE PCNNs FOR DEPTH MAP SUPER-RESOLUTION

The fusion model of FMPN is shown in Figure 2. The FMPN model consists of two steps: 1) the super-resolution of multiple PCNNs based on deep learning; 2) the fusion based on PDE. In the process of super-resolution, we construct a network S, and then use these PCNNs to predict the



(a). The weights of first individual network in some PCNNs are frozen.



(b). The weights of first individual network in each PCNN are not frozen.

FIGURE 2. FMPN model with M individual PCNNs.

high-resolution results of the low-resolution depth map. In the fusion stage, PDE is used to continuously and frame by frame evolve all high-resolution depth maps output by PCNNs, and finally generate high-precision and high-resolution output.

The Adoption of Individual Network: In general, the more layers of CNN, the higher the complexity of the network, the better its performance, and the higher the super-resolution accuracy, but the more network parameters need to be trained, the more difficult training, and the higher the computation and time complexity. However, the advent of PCNN [28] proves that CNN can still achieve the state-of-the-art performance through the fusion of SRCNN [22] with only three layers and simple parameters. SRCNN can be trained on different large training sets with random weight initialization and fixed learning rate. Therefore, we adopt SRCNN as the individual network in FMPN. In addition, to prove the

performance of the proposed FMPN, we only use SRCNNs with a fewer layer because it can achieve a high performance.

It should be noted that the inputs are the same depth map for all PCNNs, so there is no displacement in the output of multiple PCNNs. Therefore, it is not necessary to register multiple depth maps before using the PDE model; therefore, reduces the time complexity and avoids the error of registration process.

The PDE model of FMPN can be expressed as

$$\mathbb{N}(\alpha_i) = \frac{\partial U_p}{\partial t} = \mathrm{div}(\mathrm{D}_i \nabla U_i) - \beta_i \mathrm{div}(g_F(|\nabla U|_{\max})\nabla U_{\max}) + \gamma \mathrm{div}(g_R(\nabla U_i, U_k^{t=0})\nabla U_i). \quad (9)$$

It can be seen that (9) is an expanded version of (2). Here, The training samples are $\{(\alpha_i, \beta_i), i = 1, \ldots, N\}$, α denotes the low-resolution input, β denotes the high-resolution output, $\hat{\beta}_i$ denote the output of all PCNNs for the *ith* low-resolution depth map α_i, and $\mathbb{N}$ denotes the whole FMPN.

Suppose that the number of PCNNNs is M, which is denoted as $\{S_j, j = 1, \ldots, M\}$. $\frac{\partial U_p}{\partial t}$ denotes the evolution by PDE for depth maps output from all PCNNs, P is the output of each PCNN, which is expressed as

$$P = S_j(\alpha_i) = s_{j2}\left(s_{j1}(\alpha_i)\right), \quad (10)$$

where s_{j1} and s_{j2} denote two individual networks of the *jth* PCNN. Since there are multiple PCNNs in FMPN, and each PCNN may consist of individual networks with different layers and structures, the number of different individual networks may be multiple and can be expressed as $\{s_k, k = 1, \ldots, L\}$.

The training process minimizes the mean absolute error of loss function L on the sample set:

$$L(S) = \frac{1}{N}\sum_{i=1}^{N}\left|\beta_i - \hat{\beta}_i\right|, \quad (11)$$

where N denotes the total number of samples in training set.

In the training of FMPN, the weights of PCNNs could be either frozen or not. If the weights of all individual networks in FMPN are frozen, there will be no possibility and necessity of parameter tuning and training for each PCNN. Therefore, freezing the weights of FMPN can only choose some of PCNNs. As described in the last paragraph of Section 2.2, in general, the weights of one of the individual networks that construct PCNN is frozen. So, freezing or not of PCNN in FMPN refers to the operation to the learnable individual network in PCNN. The performance changes of FMPN caused by the freezing of weights will be analyzed and evaluated in the experiments.

For the freezing of weights, Figure 2 (a) and Figure 2 (b) show two FMPNs constructed by M PCNNs, respectively. Figure 2 (a) shows the partial freezing form of the weights of FMPN, and Figure 2 (b) is the non-freezing form. In the training of FMPN, only the weights of the unfrozen individual networks in each PCNN are tuned. For the frozen PCNNs, the S_j in equation (10) will not be updated during the FMPN training. The FMPN in Figure 2 (b) fine-tunes both the unfrozen PCNNs and unfrozen individual networks. This will result in different s_k and S_j after the FMPN training. In Figure 2 (a), since the first individual network s (1) of PCNN S (1) is frozen, the weights of s (1) is constant in the FMPN training. As the output of s (1), the high-resolution depth map output by PCNN S (1) is also constant.

B. TRAINING INDIVIDUAL NETWORK

We adopt the Back Propagation (BP) algorithm [30] to train the network.

We use Middlebury [31] and NYU v2 datasets [32] for training and testing. We choose 117 depth maps from these two datasets as the training sample set. Among them, there are 100 training samples and 17 test samples. In order to make the training dataset more efficient, we expand the total number of training samples: The initial 100 depth maps were rotated by 90°, 180° and 270° respectively to obtain 400 samples. Then, image blocks were extracted from these depth maps with a step size of 14, and 894,400 image blocks were finally obtained as the initial input of SRCNN.

We first train a three-layer SRCNN as the benchmark. The purpose of training is to obtain the optimal parameter $\theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$, here W_1 and B_1 denote the filters and biases respectively. The network structure follows the SRCNN 9-5-5 in [22]. Where, the convolution kernel in the first layer is 9×9, and the convolution kernel dimension and the number of output feature maps are both 64. The feature map is (33-9)/1 + 1 = 25. The second and third layers have the same convolution kernel size and dimension, which are 5×5 and 32, respectively. Each of these two layers outputs 32 feature maps. The feature map output from the second layer is (25-5)/1 + 1 = 21, and the feature map obtained from the third layer is (21-5)/1 + 1 = 17. The final feature map obtained by training is 17×17, so the 17×17 block of the image center is used as the label data. We extend SRCNN to a deeper network by inheriting the existing weights. For example, when inserting three new layers with 32 3×3 filters into the SRCNN 9-5-5, we only randomly initialize the new 3×3 filter, while the existing 9×9 and 5×5 layers are obtained by inheriting the existing weights. In this way, we can quickly get a new 6-layer individual network SRCNN 9-5-3-3-3-5. Of course, other deeper SRCNNs can also be generated in this way. Figure 3 gives an example of trained deeper SRCNN.

Each inserted convolution kernel is Zero padded so that the output feature maps have the same size. To reduce the time consumption of training, the new extended SRCNN is only trained to 20 epochs. Initialization of the weights is obtained randomly from Gaussian distribution with zero mean and standard deviation 10^{-3}. Since the network structure is

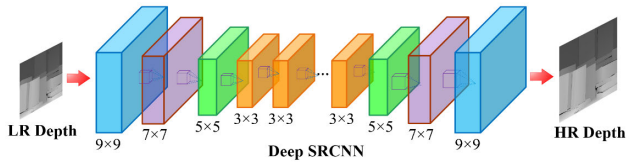


FIGURE 3. An extended deep-layer SRCNN.

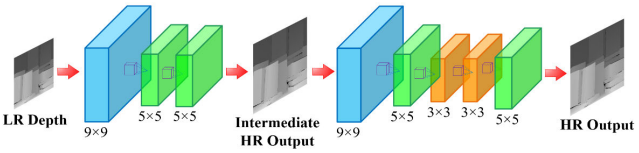


FIGURE 4. PCNN with network structure of 3 + 5.

relatively simple and the number of layers is small, the learning rate is limited at 10^{-4}.

C. TRAINING PCNN

For PCNN training, the training algorithm and sample set are the same as individual networks.

We construct PCNN by combining and arranging two trained SRCNNs in a chain order. For example, we construct four PCNNs with network structures of 3 + 3, 3 + 5, 5 + 3 and 5 + 5, respectively. Take 3 + 5 as an example, 3 and 5 represents the layers number of the first SRCNN and the second SRCNN, respectively. The feature map output from the first SRCNN is used as the input of the second SRCNN. The learning rate and weights initialization of the second SRCNN is the same as those of the first SRCNN. The weights of the second SRCNN is frozen, so all learnable parameters in a PCNN depend on the first SRCNN. Initialization of the PCNN is obtained randomly from Gaussian distribution with zero mean and standard deviation 10^{-3}. We use Mean Absolute Error (MAE) between the second SRCNN output and Ground Truth as the loss function. The training of each PCNN will proceed 15 epochs. Figure 4 shows a trained PCNN with network structure of 3 + 5, that is, PCNN (3 + 5).

IV. EXPERIMENTAL RESULTS

A. EXPERIMENT SETUP

In this section, the performance of the improved method (FMPN) is evaluated both quantitatively and qualitatively with respective benchmark and state-of-the-art super-resolution algorithms.

1) BASELINE METHODS

We compare our results with the following *four* categories of the methods. *1) Interpolation-based methods*: Bicubic Interpolation (Bicubic) and Moving Least Squares Filter (MLS) [33]; *2) Single depth map-based methods:* Cross-based Local Multipoint Filtering (CLMF) [34] and Unified Multi-lateral Filter for RGB-D Sensors (UMLC) [35];

TABLE 1. Quantitative results (PSNR(dB)/SSIM) of FMPN with different PCNNs at 4× upsampling.

Networ	Layer	Free	Art	Books	Laundry
SRCN	3	-	32.923/0.91	31.812/0.88	32.844/0.91
SRCN	5	-	33.125/0.91	32.032/0.88	33.047/0.91
SRCN	7	-	33.162/0.91	32.085/0.88	33.110/0.91
PCNN	3+3	y	33.157/0.91	32.076/0.88	33.097/0.91
PCNN	3+5	y	33.169/0.91	32.096/0.89	33.121/0.92
PCNN	5+3	y	33.175/0.91	32.112/0.89	33.129/0.92
PCNN	5+7	y	33.181/0.91	32.272/0.89	33.135/0.92
PCNN	7+9	y	33.205/0.91	32.303/0.89	33.171/0.92
FMPN	(3+5) +(5+7)	n	33.376/0.92	32.575/0.89	33.291/0.92
FMPN	(3+3) +(3+5)	y	33.667/0.93	32.866/0.91	33.493/0.94
FMPN	(3+3) +(3+5)	y	33.671/0.93	32.869/0.91	33.495/0.94
FMPN	(3+3) +(3+5)	n	33.772/0.93	32.971/0.91	33.598/0.94
FMPN	(5+7) +(7+9)	n	33.779/0.93	32.980/0.91	33.606/0.94
FMPN	(3+5) +(5+7)	y	33.927/0.94	33.126/0.91	33.757/0.94
FMPN	(3+5) +(5+7)	n	34.066/0.94	33.269/0.92	33.902/0.95
FMPN	(7+9) +(9+11)	n	**34.214/0.95**	**33.394/0.92**	**34.109/0.95**

3) Color image guided methods: Joint Geodesic Filter (JGF) [3], Edge-adaptive Non-local Means Filter (Edge) [14], Auto Regressive Model (AR) [16], Guided Image Filtering (Guided) [36], and Unified Multi-lateral Filter (UML) [37]; *4) Deep learning-based methods* including Super-resolution Using Very Deep Convolution Networks (VDSR) [38] and Deeply Recursive Convolution Network (DRCN) [39]. The results of the comparison algorithms are generated by the authors' codes.

2) EXPERIMENTAL DATASETS

In the quantitative analysis, we conducted experiments on the public dataset *Art, Books, Dolls, Laundry, Moebius,* and *Reindeer* from the Middlebury stereo datasets [31], and *Books, Shark, Devil* from ToF-Mark datasets [15]. In the qualitative analysis, we expand the datasets by using *Dolls* and *Laundry* from the Middlebury 2005 datasets [31], *Devil* and *Shark* from ToF-Mark datasets [15], and *Playtable*, *Jadeplant* from Middlebury 2014 datasets [40] to evaluate the visual effects of various methods.

3) EVALUATION METRICS

To increase the diversity and objectivity of evaluation, PSNR and SSIM are used to evaluate the performance of various fusion schemes of FMPN in Table 1, and MAE is used to analyze the performance of different methods in Table 2 -Table 3.

4) EXPERIMENTAL SCHEME

First, the high-resolution depth maps from Middlebury datasets are carried out bi-cubic interpolation by 1/2, 1/4, 1/8, and 1/16 to obtain the low-resolution depth maps as the input of FMPN. Then, Gaussian noise with depth information is added to the down-sampled low-resolution depth maps to simulate the actual acquisition process of the depth maps. Finally, we use a trained-well FMPN to up-sample the low-resolution depth maps by 2×, 4×, 8× and 16×, respectively. It should be noted that for the ToF-Mark datasets with a resolution of only 120 × 160 [15], our up-sampling factor is consistent with the author's 6.25×.

TABLE 2. Quantitative upsampling results (in MAE) on noise-free middlebury datasets. Red indicates the best performance, green indicates the second best, and blue indicates the third best.

Datasets	Art				Reindeer				Dolls				Books				Laundry				Moebius				Average			
Methods	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16
Bicubic	0.48	0.97	1.85	3.59	0.30	0.55	0.99	1.88	0.20	0.36	0.66	1.18	0.13	0.29	0.59	1.15	0.28	0.54	1.04	1.95	0.13	0.30	0.59	1.13	0.25	0.50	0.95	1.81
UMLC [35]	0.29	0.50	1.15	2.56	0.20	0.30	0.64	1.60	0.17	0.25	0.51	1.07	0.14	0.22	0.47	0.97	0.19	0.30	0.67	1.45	0.14	0.23	0.47	0.98	0.19	0.30	0.65	1.44
Edge [14]	0.41	0.65	1.03	2.11	0.20	0.37	0.63	1.28	0.16	0.31	0.56	1.05	0.17	0.30	0.56	1.03	0.17	0.32	0.54	1.14	0.18	0.29	0.51	1.10	0.22	0.37	0.64	1.29
Guided [36]	0.63	1.01	1.70	3.46	0.42	0.53	0.88	1.80	0.28	0.35	0.56	1.13	0.22	0.35	0.58	1.14	0.38	0.52	0.95	1.90	0.23	0.37	0.59	1.16	0.36	0.52	0.88	1.77
UML [37]	0.30	0.53	1.23	2.61	0.20	0.34	0.71	1.63	0.18	0.29	0.56	1.11	0.14	0.25	0.52	1.00	0.19	0.33	0.73	1.48	0.14	0.25	0.51	1.01	0.19	0.33	0.71	1.47
AR [16]	0.18	0.49	0.64	2.01	0.22	0.40	0.58	1.00	0.21	0.34	0.50	0.82	0.12	0.22	0.37	0.77	0.20	0.34	0.53	1.12	0.10	0.20	0.40	0.79	0.17	0.33	0.50	1.09
CLMF [34]	0.43	0.74	1.37	2.95	0.32	0.51	0.84	1.51	0.24	0.34	0.66	1.02	0.14	0.28	0.51	1.06	0.30	0.50	0.82	1.66	0.15	0.29	0.52	1.01	0.26	0.44	0.79	1.54
JGF [3]	0.29	0.47	0.78	1.57	0.23	0.38	0.64	1.09	0.19	0.33	0.59	1.06	0.15	0.24	0.43	0.81	0.21	0.36	0.64	1.20	0.15	0.25	0.46	0.80	0.20	0.34	0.59	1.08
MLS [33]	0.27	0.68	1.40	2.20	0.32	0.64	0.74	1.43	0.24	0.36	0.61	0.98	0.16	0.26	0.48	1.16	0.23	0.39	0.81	1.53	0.15	0.25	0.49	0.93	0.23	0.43	0.70	1.37
VDSR [38]	0.21	0.49	0.68	1.60	0.19	0.29	0.57	0.99	0.18	0.37	0.49	0.85	0.16	0.16	0.36	0.73	0.23	0.38	0.58	1.07	0.13	0.22	0.36	0.75	0.19	0.32	0.51	1.00
DRCN [39]	0.17	0.46	0.67	1.55	0.16	0.30	0.55	0.96	0.15	0.26	0.53	0.83	0.15	0.23	0.35	0.76	0.19	0.28	0.51	1.11	0.12	0.25	0.39	0.78	0.18	0.34	0.50	1.00
FMPN (ours)	0.16	0.45	0.63	1.53	0.17	0.27	0.53	0.97	0.14	0.24	0.47	0.79	0.11	0.15	0.33	0.71	0.16	0.29	0.49	1.09	0.10	0.19	0.35	0.72	0.14	0.27	0.47	0.97

TABLE 3. Quantitative upsampling results (in MAE) on noisy middlebury datasets. Red indicates the best performance, green indicates the second best, and blue indicates the third best.

Datasets	Art				Reindeer				Dolls				Books				Laundry				Moebius				Average			
Methods	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16
Bicubic	3.52	3.84	4.47	5.72	3.39	3.52	3.82	4.45	3.28	3.34	3.47	3.72	3.30	3.37	3.51	3.82	3.35	3.49	3.77	4.35	3.28	3.36	3.50	3.80	3.35	3.49	3.76	4.31
UMLC [35]	1.31	2.07	2.94	4.23	0.95	1.62	2.36	3.34	0.92	1.57	2.25	3.05	0.90	1.57	2.27	3.02	0.98	1.67	2.43	3.41	0.94	1.59	2.30	3.11	1.00	1.68	2.43	3.36
Edge [14]	1.69	2.40	3.60	5.75	1.20	1.60	2.40	3.97	1.14	1.54	2.07	3.02	1.12	1.44	1.81	2.59	1.28	1.63	2.20	3.34	1.13	1.45	1.95	2.91	1.26	1.68	2.34	3.60
Guided [36]	1.49	1.97	3.00	4.91	1.29	1.99	2.99	4.14	1.19	1.94	2.80	3.50	0.80	1.22	1.95	3.04	1.28	2.05	3.04	4.10	1.18	1.90	2.77	3.55	1.21	1.85	2.76	3.87
UML [37]	1.28	2.13	3.10	4.56	1.00	1.86	2.76	3.72	0.96	1.81	2.65	3.40	0.94	1.80	2.65	3.40	1.03	1.91	2.79	3.82	0.96	1.81	2.67	3.45	1.03	1.89	2.77	3.73
AR [16]	0.76	1.01	1.70	3.05	0.48	0.80	1.29	2.02	0.59	0.91	1.32	2.08	0.47	0.70	1.15	1.81	0.51	0.85	1.30	2.24	0.46	0.72	1.15	1.92	0.55	0.83	1.32	2.19
CLMF [34]	1.19	1.77	2.59	4.91	0.96	1.56	2.54	3.85	0.96	1.54	2.37	3.25	0.90	1.48	2.38	3.36	0.94	1.55	2.50	3.81	0.87	1.44	2.32	3.30	0.97	1.56	2.45	3.75
JGF [3]	2.36	2.74	3.64	5.46	2.18	2.40	2.89	3.94	2.09	2.22	2.49	3.25	2.12	2.25	2.49	3.25	2.16	2.37	2.85	3.90	2.09	2.24	2.56	3.28	2.17	2.37	2.82	3.85
MLS [33]	1.43	1.95	3.37	4.67	0.92	1.49	2.76	3.53	0.81	1.34	2.57	3.09	0.81	1.39	2.68	3.21	0.94	1.53	2.83	3.58	0.87	1.40	2.65	3.16	0.96	1.52	2.83	3.54
VDSR [38]	0.81	1.02	1.65	2.96	0.53	0.78	1.31	1.97	0.63	0.90	1.35	2.06	0.45	0.68	1.13	1.80	0.49	0.90	1.29	2.29	0.44	0.70	1.13	1.89	0.56	0.83	1.31	2.16
DRCN [39]	0.75	1.00	1.64	2.99	0.46	0.79	1.25	1.99	0.66	0.88	1.26	1.99	0.46	0.70	1.11	1.77	0.46	0.83	1.25	2.23	0.45	0.67	1.11	1.86	0.54	0.81	1.27	2.14
FMPN (ours)	0.74	0.99	1.62	2.92	0.45	0.76	1.22	1.93	0.61	0.87	1.22	1.97	0.44	0.66	1.09	1.72	0.45	0.81	1.28	2.17	0.41	0.69	1.06	1.81	0.52	0.80	1.25	2.09

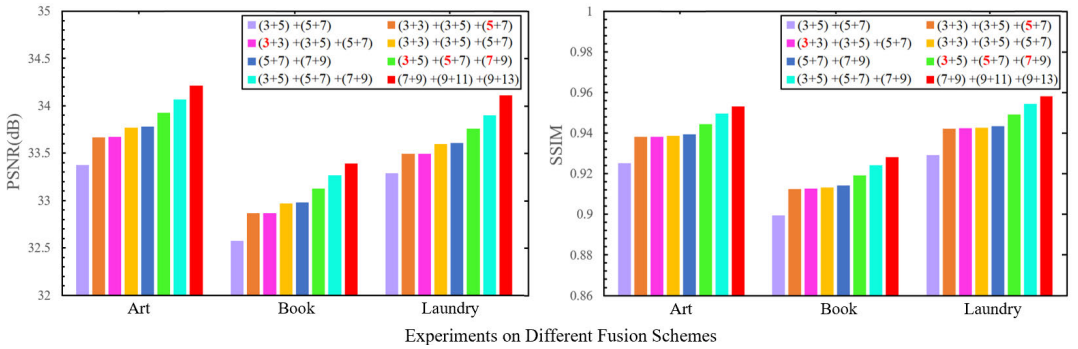


FIGURE 5. Quantitative results (PSNR(dB)/SSIM) of FMPN with different PCNNs at 4× upsampling.

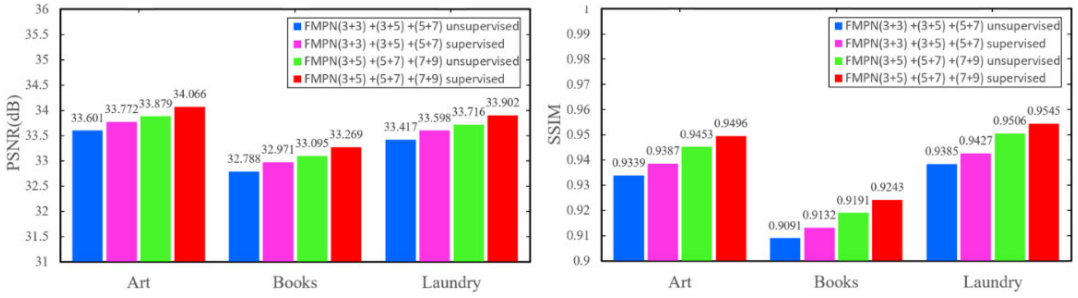


FIGURE 6. Quantitative results (PSNR(dB)/SSIM) of FMPNs with different weights initialization at 4× upsampling.

5) EXPERIMENTAL PLATFORM

The experiments are carried out on Windows 10 64-bit operating system, using a laptop with a quad-core 2.2 GHz Intel(R) i7-4770HQ CPU, 16 GB RAM, Intel (R) Iris (TM) Pro Graphics 5200. The proposed method is implemented based on MATLAB R2018a and Caffe [41]. MATLAB is used to

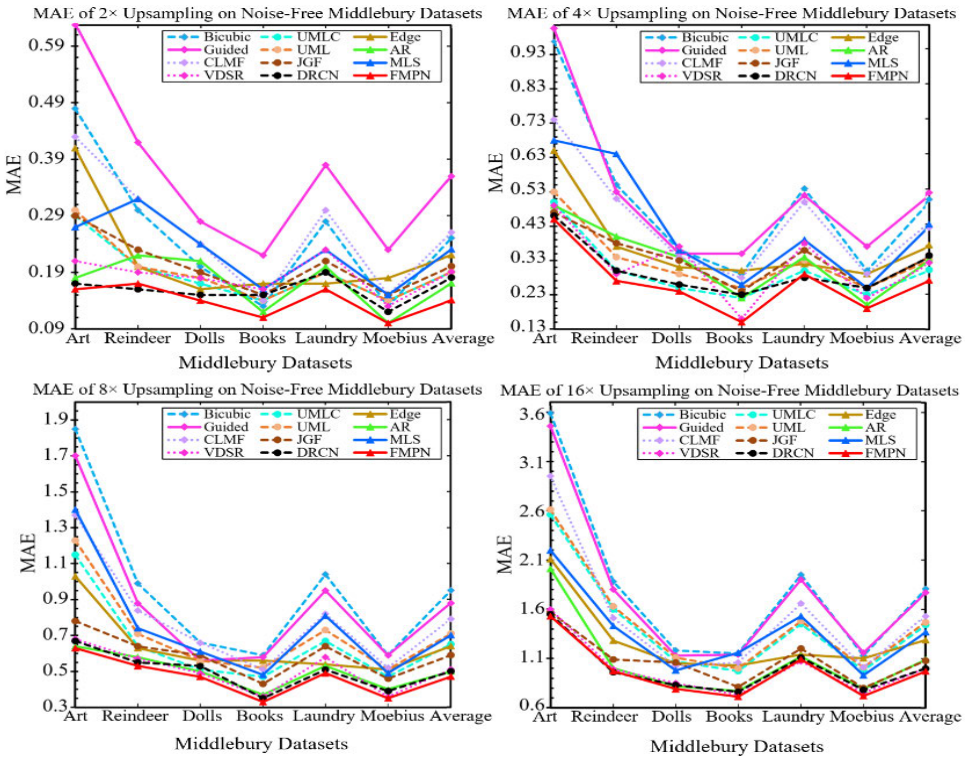


FIGURE 7. Quantitative upsampling results (in MAE) on noise-free middlebury datasets at 4 upsampling factors.

evaluate the quality of super-resolution depth maps. Caffe is used for the construction, training and test of the neural network.

B. QUANTITATIVE RESULTS

1) EXPERIMENTS ON DIFFERENT FUSION SCHEMES

We first evaluate the FMPNs constructed by different. fusion schemes. Table 1 shows the comparison results of FMPNs with different fusion schemes for upsampling with scale 4×. The fusion scheme includes whether the weights of the SRCNNs are frozen or not, and different network structures. For network structure, FMPN consists of SRCNNs with different fusion layers, and PCNNs constructed by different SRCNNs and their multi-permutations. It should be noted that in almost all cases, the second SRCNN of PCNN is frozen. The "freezing" in FMPN refers specifically to the freezing of the first SRCNNs of each PCNNs. In Table 1, the SRCNNs with frozen weight are marked in red. For example, (**5** + 7) refers to freezing the weight of the first SRCNN (with 5 layers) in PCNN. Generally, as the weight of the second SRCNN is frozen by default, the second SRCNN (with 7 layers) is not marked in red.

In Table 1, we notice that the performance of the FMPNs with unfrozen weight are better than that of the FMPNs with frozen weight under the same network structure. For the FMPNs with frozen weight, (**3** + 3) + (3 + 5) + (5 + 7) perform better compared to (3 + 3) + (3 + 5) + (**5** + 7). This is because the SRCNN with more layers has more weights, so its performance is more affected by the freezing.

For different network structures, it can be seen that the accuracy of. SRCNN increases by using more layers. The accuracy of individual SRCNN with more layers is higher compared to PCNN with fewer layers. For example, the accuracy of individual 7-layer SRCNN is higher than that of PCNN 3 + 3. This is the motivation of PCNN fusion in this paper. It could be seen that FMPN ((3 + 5) + (5 + 7) + (7 + 9)) and FMPN ((5 + 7) + (7 + 9)) perform better compared to individual PCNN (7 + 9). FMPN ((3 + 3) + (3 + 5) + (5 + 7)) and FMPN ((3 + 5) + (5 + 7) + (7 + 9)) perform still better compared to FMPN ((3 + 5) + (5 + 7)) and FMPN ((5 + 7) + (7 + 9)), respectively. This indicates that using more PCNNs might increase the accuracy of FMPN. We also find that the accuracy of FMPN ((3 + 3) + (3 + 5) + (5 + 7)) is lower than FMPN ((3 + 5) + (5 + 7) + (7 + 9)), and the accuracy of FMPN ((3 + 5) + (5 + 7) + (7 + 9)) is lower than FMPN ((7 + 9) + (9 + 11) + (9 + 13)). This implies that the deeper layers of PCNN we use for FMPN, the better accuracy we may get.

By comparing FMPN ((**3** + 5) + (**5** + 7) + (**7** + 9)) and FMPN ((3 + 5) + (5 + 7) + (7 + 9)), it can be seen that learning by completely freezing the weights of PCNNs may improve the PSNR 0.15dB and SSIM 0.005 for unfrozen FMPN. If we fine-tune the whole network without freezing the weights of the first SRCNNs of the PCNNs, the gain

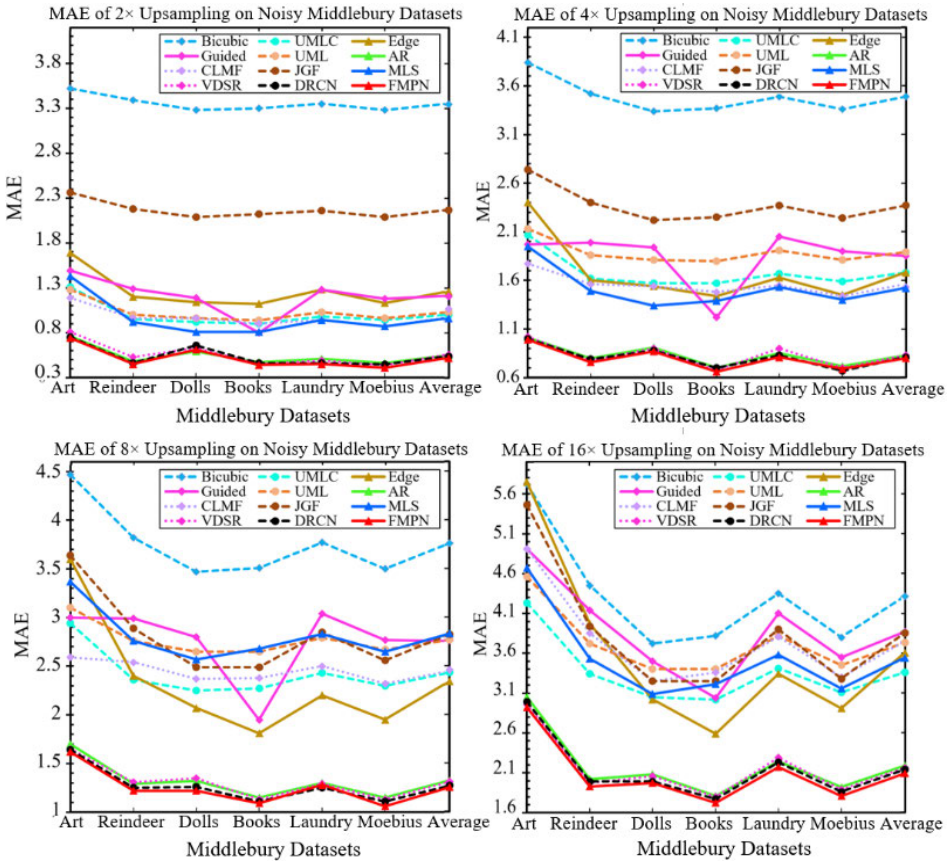


FIGURE 8. Quantitative upsampling results (in MAE) on noisy middlebury datasets at 4 upsampling factors.

will increase to more than 0.3dB PSNR and 0.01 SSIM. This shows the effectiveness of FMPN fusion.

Figure 5 intuitively shows the comparison results of FMPNs with different fusion schemes with 4× upsampling factor.

Moreover, we test the FMPNs with different weight initialization. In Figure 6, it could be seen that both of the FMPN ((3 + 3) + (3 + 5) + (5 + 7)) and the FMPN ((3 + 5) + (5 + 7) + (7 + 9)) trained from unsupervised. weights perform much worse compared to the corresponding FMPNs fine-tuned from existing networks. Due to the unsupervised weights initialization, for FMPNs, the convergence will be more difficult than the way of inheriting the weights from individual PCNNs.

2) COMPARISON TO THE STATE-OF-THE-ART

a: EXPERIMENTS ON MIDDLEBURY DATABASES [31]

In Table 2 and Table 3, we compare the FMPN with other 11 kinds of the state-of-the-art methods by upsampling at four different factors (2×, 4×, 8×, and 16×) on noise-free and noisy Middlebury databases [31], respectively. Red indicates the best performance, green indicates the second best, and blue indicates the third best. In the two tables, the FMPN is constructed by three PCNNs whose layers are (3 + 5), (5 + 7) and (7 + 9) respectively.

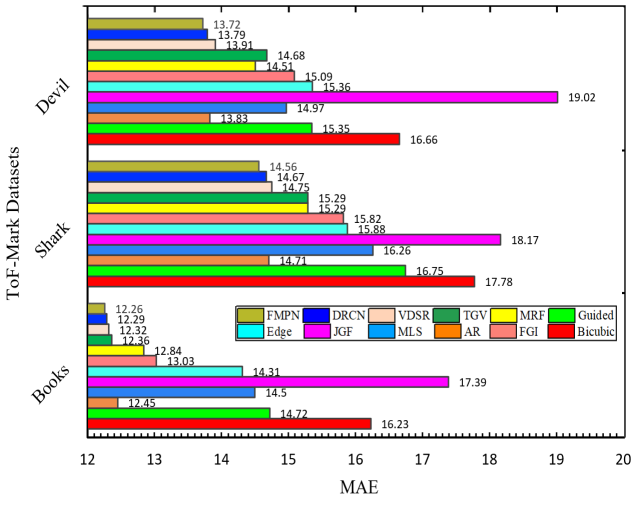


FIGURE 9. Quantitative upsampling results (in MAE) with scale 6.25× on the ToF-Mark datasets [15].

From the two tables, we can see that the accuracy of FMPN ((3 + 5) + (5 + 7) + (7 + 9)) outperforms the other methods in most cases at four upsampling factors. However, in a few cases, the results of FMPN are slightly worse than other methods such as DRCN [39]. For example, in Table 2, the MAE results of FMPN at the 2× scale on *Reindeer* and at the

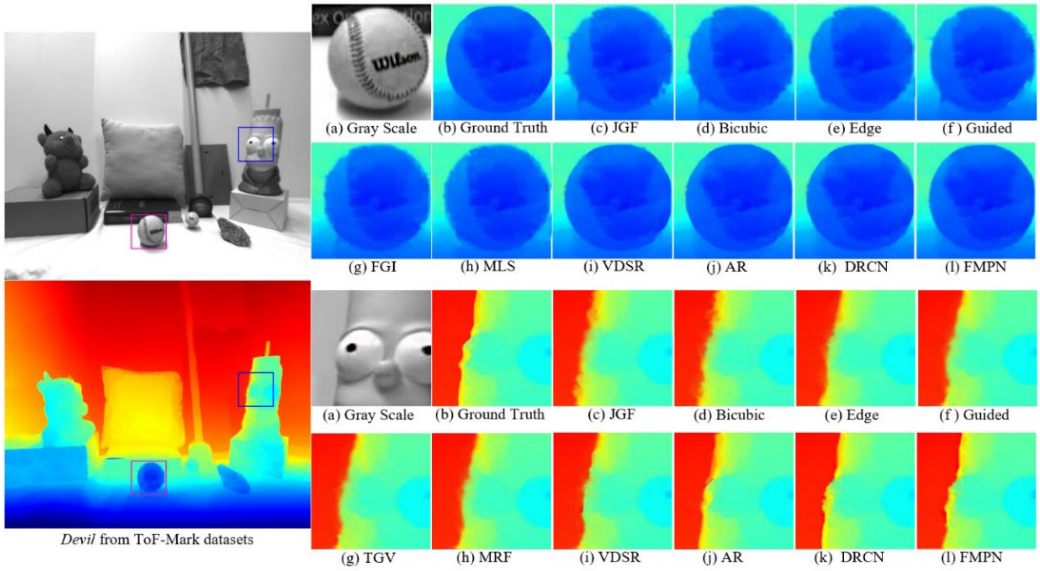


FIGURE 10. Visual comparison of upsampling noise-free data set Devil (scaling factor = 6.25).

4× scale on *Laundry* are 0.17 and 0.29 respectively, slightly higher than the 0.16 and 0.28 of DRCN [39]. In Table 3, the MAE results of FMPN at the 8× scale on *Laundry* and at the 4× scale on *Moebius* are 1.28 and 0.69 respectively, slightly higher than the 1.25 and 0.67 of DRCN [39]. This is because the number of Convolution layers used in DRCN [39] is much higher than that of the FMPN. But for the average of the six datasets tested, the FMPN achieved the lowest MAE in all cases and performed best. This proves the effectiveness of the FMPN in the super-resolution of depth map and its good performance in noise suppression.

As summary, compared with other deeper networks, the network size of the FMPN is relatively small, showing better performance in various metrics. The FMPN ((3 + 5) + (5 + 7) + (7 + 9)) consists of 172,320 unfrozen parameters, which is still smaller than VDSR [38] (20 layers, 650K + parameters) and DRCN [39]. On single Iris Pro Graphics 5200 GPU, it takes about 0.05-0.09ms per image in average, which is less than VDSR [38] and DRCN [39].

Figure 7 and Figure 8 show the MAE results corresponding to Table 2 and Table 3 more intuitively.

b: EXPERIMENTS ON ToF-MARK DATASETS [15]

In addition, the FMPN is assessed on TOF-Mark dataset [15], which contained three data sets, *Books, Shark* and *Devil*, captured by the ToF sensors. The resolution of the original depth maps is 120 × 160. The upsampling factor is approximately 6.25×. The FMPN is compared with eleven state-of-the-art methods: DRCN [39], VDSR [38], TGV [15], Edge [14], JGF [3], MLS [33], AR [16], Bicubic, and Markov Random Field (MRF) [12], The Generalized Variation (TGV) [15], Fast Interpolation (FGI) [19]. Figure 9 shows the results of quantitative comparison. It can be observed that the FMPN achieves the minimum MAE results in all cases, and its overall performance is the best.

C. QUALITATIVE RESULTS

We evaluate the FMPN visually by upsampling the noise-free and noisy datasets at three upsampling factors (i.e., 6.25×, 8×, 16×). The comparison results are shown in Figure10-Figure15. These datasets include *Dolls* and *Laundry* in the Middlebury 2005 datasets [31], *Playtable* and *Jadeplant* in the Middlebury 2014 datasets [40], and *Devil* and *Shark* in the ToF-Mark datasets [15].

1) EXPERIMENTS UNDER NOISE-FREE CONDITIONS

Figure 10-Figure12 show the results of the super-resolution on the noise-free datasets.

As can be seen from Figure 10, the results of JGF [3] have the most serious blurring depth edges, which indicates that its edge-preserving performance is the lowest. The results of Bicubic, Edge [14], FGI [19] and TGV [15], also present certain level of blurring depth edges and texture-copy artifacts. The results of MLS [33] and MRF [12] are slightly more desirable than the afore mentioned ones while the results of AR [16], VDSR [38], DRCN [39] and FMPN are most appealing. Especially, in edge areas, FMPN demonstrated the closest to Ground Truth and therefore proves its effectiveness in depth map super-resolution.

In Figure 11, the visual effects generated by all methods decreases, when the up-sampling multiple rises from 6.25× to 8×, Bicubic shows the worst visual effect. Guided suffers remarkable blurring depth edges and texture-copy artifacts. In addition, we noted that artifacts exist in MLS [33], CLMF [34], UML [37], UMLC [35] and JGF [3] to some degree. Edge [14] displays less artifacts than VDSR [38]. Both AR [16] and DRCN [39] show sharp edges yet with a

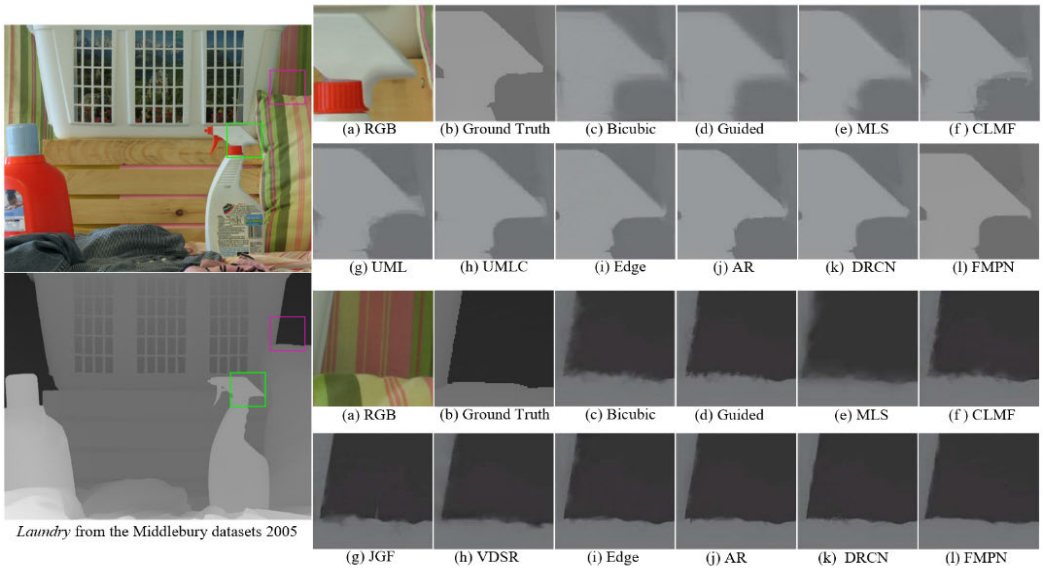


FIGURE 11. Visual comparison of upsampling noise-free data set Laundry (scaling factor = 8).

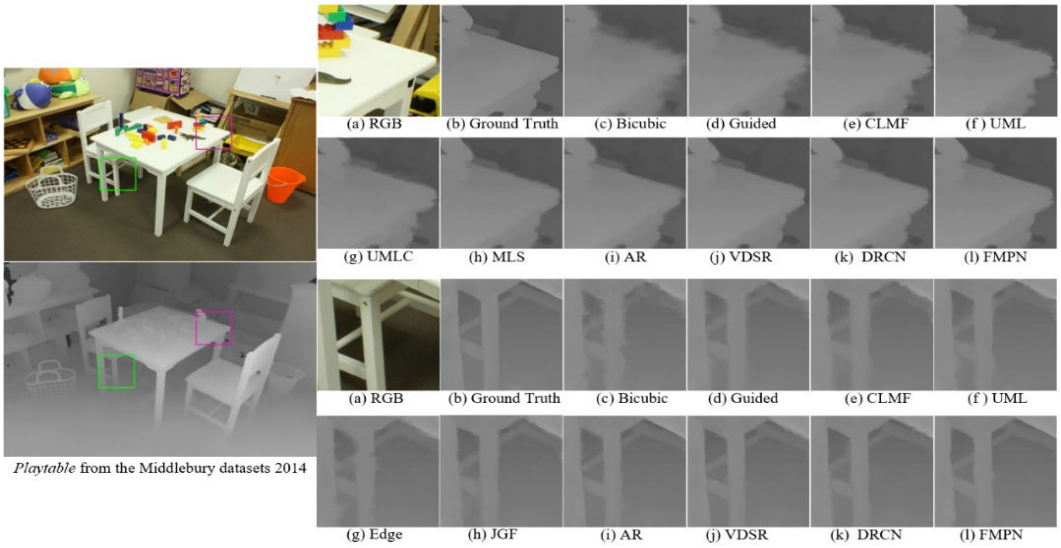


FIGURE 12. Visual comparison of upsampling noise-free data set Playtable (scaling factor = 16).

few artifacts. However, the boundaries in FMPN are generally smoother and sharper along the edge direction resulted from its PDE fusion process.

When the upsampling multiple increases to 16×, the super-resolution process for depth maps usually becomes difficult, because the super-resolution of this multiple has to expand from 1 pixel to 256 pixels (i.e., $16 \times 16 = 256$). However, the experimental results in Figure 12 demonstrate that despite the apparent edge blurring of other methods FMPN displays relatively clear edges and rich details. For example, the four methods (i.e. Bicubic, Guided [36], CLMF [34] and UML [37]) all indicate obvious texture-copy artifacts and notable jagging artifacts. UMLC [35], MLS [33], Edge [14] and JGF [3] suppress texture copying but still suffer from blurry edge. While VDSR [38], AR [16] and DRCN [39] generate more visual appealing results than their peers, FMPN demonstrated its capacity of producing the sharpest result without jagging artifacts and preserving the rich details of the scene in regions with fine structures.

2) EXPERIMENTS UNDER NOISY CONDITIONS

To further demonstrate the effectiveness of the FMPN, we also carried out experiments on noisy data sets, and the visualization results are shown in Figure13-Figure15. The three data sets *Shark*, *Dolls*, and *Jadeplant* used in the test were all added with depth-based Gaussian noise. The noise

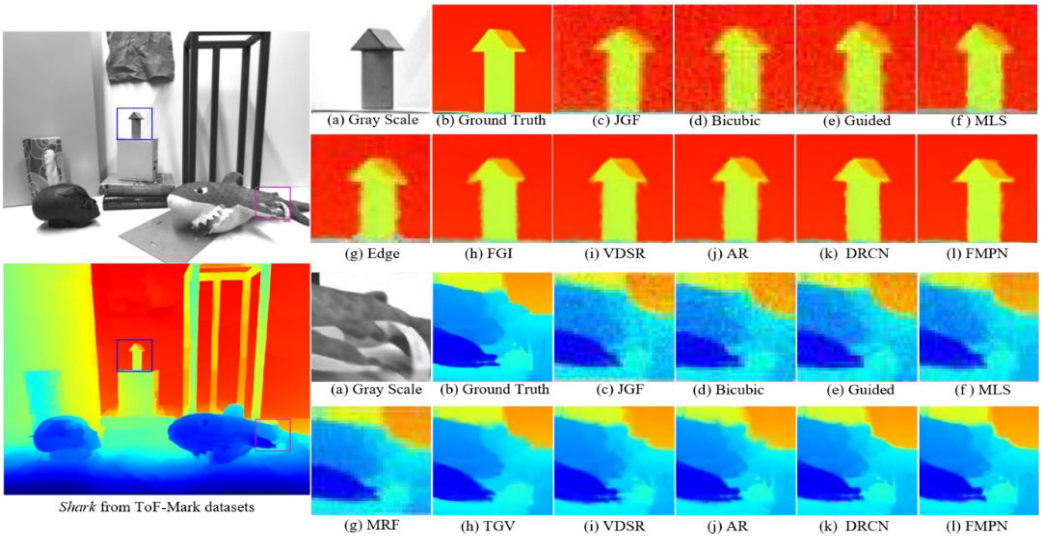


FIGURE 13. Visual comparison of upsampling noisy data set Shark (scaling factor = 6.25).

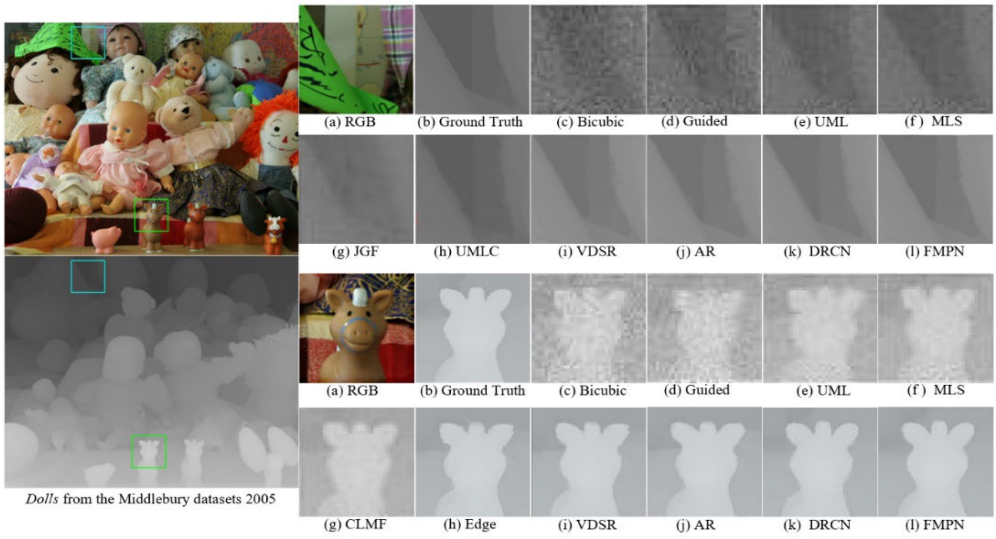


FIGURE 14. Visual comparison of upsampling noisy data set Dolls (scaling factor = 8).

adding process adopts the method provided by [13]: $\eta(x) = \mathbb{N}\left(0, \sigma\ (x)^{-1}\right)$, where $\sigma = 651$, and x is the depth value of each voxel in the low-resolution depth map used for testing.

Figure13 reveals that there are still much noises in the results of JGF [3], Bicubic, Guided [36], MLS [33], MRF [12] and Edge [14], evidenced with blurring edges, texture-copy artifacts or jagging artifacts to some degree. It indicates that the denoising capacity of these methods are limited FGI [19] and TGV [15] produce clearer results, but with some obvious edge discontinuities. VDSR [38], AR [16] and DRCN [39] provide comparable results to FMPN in denoising, but cannot eliminate texture-copy artifacts and blurring depth edges either. In contrast, the FMPN achieves the best visual result without visible noise.

For data set *Dolls* that has complex edge structure and rich texture details, with strong depth discontinuity in the edge region, it can be challenging for super-resolution. Figure14 presents visual comparison results for depth map "*Dolls*". The results of Bicubic, MLS [33], and JGF [3] contain visible noise due to their limited denoising capacity. Edge [14] performs poorly in edge-preserving, evidenced by the serious blurry artifacts shown on the top of the pony's ear highlighted in the green square. VDSR [38], AR [16] and DRCN [39] demonstrated their denoise capacity similar to FMPN, but they are incapable eliminating artifacts and abolishing discontinuities. In contrast, the FMPN can effectively remove noise, generate complex details, and achieve the best visual result closest to the Ground Truth.

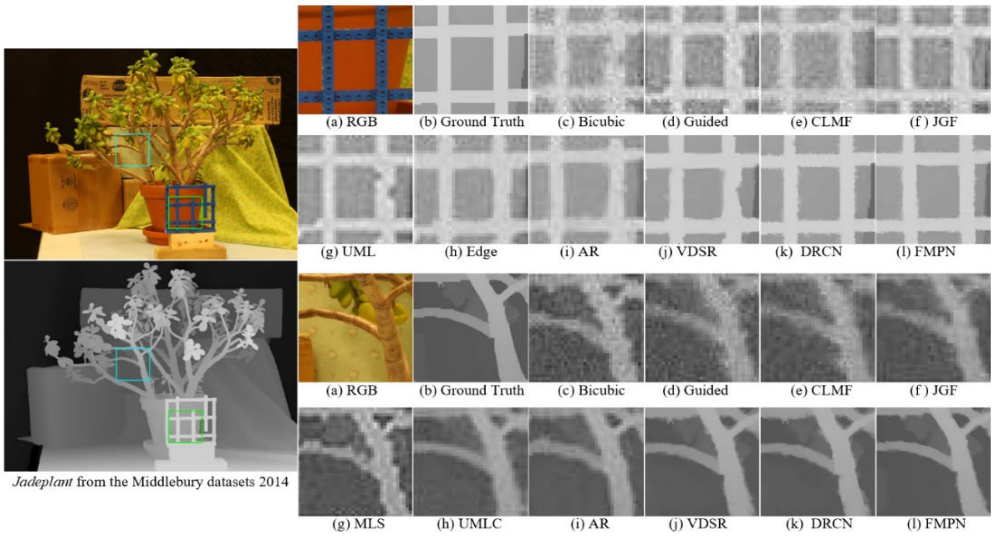


FIGURE 15. Visual comparison of upsampling noisy data set Jadeplant (scaling factor = 16).

TABLE 4. Average running time comparison (seconds).

Methods / Datasets	Bicubic	Guided [36]	AR [16]	Edge [14]	TGV [15]	VDSR [38]	DRCN [39]	FMPN(Ours)
Middlebury	0.01	25.79	301.62	189.67	4377.91	671.55	15105.21	201.59
ToF-Mark	0.01	19.33	161.16	88.91	1765.27	307.93	9211.76	97.01

For larger super-resolution ratios, such as 16×, it is often a challenge to recover the high-resolution details, especially for noisy depth maps as mentioned before. We present the visual comparison results for this case in Figure15 which shows that only VDSR [38] and DRCN [39] demonstrated competitive visual effects as well as FMPN. Bicubic, CLMF [34], JGF [3], UML [37], and MLS [33] still contain a considerable amount of noise, severe texture-copy artifacts, and blurring depth edges. FMPN achieves the best visual result by providing the best performance in mitigating texture-copy artifacts and preserving depth edges. This benefits from its PDE-based fusion process Firstly, a part of the noise is removed by super-resolution, then the remaining noise is effectively eliminated through the diffusion process in PDE, and the detailed features of depth maps output by all PCNNs are further integrated through the fusion process of PDE.

D. AVERAGE RUNNING TIME COMPARISON

We perform our experiments on a laptop proposed in Part A of Section IV. The proposed method is implemented based on Matlab and C++ code.

Table 4 shows the average running time in seconds of different methods with 2× upsampling factor on the Middlebury and the ToF-Mark datasets. TGV [15] and DRCN [39] take more than 4,000 seconds and 15,000 seconds respectively. Guided [36] takes the least time except for Bicubic. The average running time of FMPN is between Edge [14] and AR [16]. Although the performance of VDSR [38] and FMPN is similar, the time consumption of VDSR [38] is much more than that of FMPN.

V. CONCLUSION

In this paper, we present a novel depth map super-resolution method based on fusion of multiple Convolution neural networks. The key contributions are two-folds. The first one is to adopt the progressive learning and deep supervision to efficiently carry out the super-resolution of the noisy depth map at the large upsampling factors. The second one is to use the PDE-based fusion process integrating all output details of the progressive Convolution neural networks and further removing the noise, so as to realizing the high-precision output of the network with simple structure. To verify the proposed method, enough experiments on the Middlebury and ToF-Mark datasets for depth map super-resolution are carried out. We discussed various fusion schemes of the proposed method and compared it with the SRCNNs, PCNNs within different network structures, benchmark and the state-of-the-art methods. The experimental results prove the proposed network fusion schemes are obviously superior to other individual networks in PSNR and SSIM and can achieve the best subjective and objective results by comparing it with state-of-the-art methods.

REFERENCES

[1] S. Foix, G. Alenya, and C. Torras, "Lock-in Time-of-Flight (ToF) cameras: A survey," *IEEE Sensors J.*, vol. 11, no. 9, pp. 1917–1926, Sep. 2011.

[2] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 1–8, 2007.

[3] M.-Y. Liu, O. Tuzel, and Y. Taguchi, "Joint geodesic upsampling of depth images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 169–176.

[4] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon, "High quality depth map upsampling for 3D-TOF cameras," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1623–1630.

[5] F. Kou, W. Chen, C. Wen, and Z. Li, "Gradient domain guided image filtering," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4528–4539, Nov. 2015.

[6] J. T. Barron and B. Poole, "The fast-bilateral solver," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 617–632.

[7] H. Kwon, Y.-W. Tai, and S. Lin, "Data-driven depth map refinement via multi-scale sparse representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 159–167.

[8] M. Kiechle, S. Hawe, and M. Kleinsteuber, "A joint intensity and depth co-sparse analysis model for depth map super-resolution," in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 1545–1552.

[9] T. W. Hui, C. C. Loy, and X. Tang, "Depth map super-resolution by deep multi-scale guidance," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 353–369.

[10] S. Lu, X. Ren, and F. Liu, "Depth enhancement via low-rank matrix completion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3390–3397.

[11] X. Shen, C. Zhou, L. Xu, and J. Jia, "Mutual-structure for joint filtering," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3406–3414.

[12] J. Yang, X. Ye, K. Li, and C. Hou, "Depth recovery using an adaptive color-guided auto-regressive model," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 158–171.

[13] S. Schuon, C. Theobalt, J. Davis, and S. Thrun, "LidarBoost: Depth superresolution for ToF 3D shape scanning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 343–350.

[14] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. S. Kweon, "High-quality depth map upsampling and completion for RGB-D cameras," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5559–5572, Dec. 2014.

[15] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof, "Image guided depth upsampling using anisotropic total generalized variation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 993–1000.

[16] J. Yang, X. Ye, K. Li, C. Hou, and Y. Wang, "Color-guided depth recovery from RGB-D data using an adaptive autoregressive model," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3443–3458, Aug. 2014.

[17] Y. Zuo, Q. Wu, J. Zhang, and P. An, "Explicit edge inconsistency evaluation model for color-guided depth map enhancement," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 2, pp. 439–453, Feb. 2018.

[18] X. Song, Y. Dai, and X. Qin, "Deep depth super-resolution: Learning depth super-resolution using deep Convolution neural network," in *Proc. Asian Conf. Comput. Vis.*, 2016, pp. 360–376.

[19] Y. Li, D. Min, M. N. Do, and J. Lu, "Fast guided global interpolation for depth and motion," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 717–733.

[20] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2024–2039, Oct. 2016.

[21] A. Roy and S. Todorovic, "Monocular depth estimation using neural regression forest," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5506–5514.

[22] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolution network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.

[23] G. Riegler, M. Räther, and H. Bischof, "Atgv-net: Accurate depth super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 268–284.

[24] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690.

[25] Y. Wen, B. Sheng, P. Li, W. Lin, and D. D. Feng, "Deep color guided Coarse-to-Fine convolutional network cascade for depth image super-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 994–1006, Feb. 2019.

[26] D. Ferstl, M. Ruther, and H. Bischof, "Variational depth superresolution using example-based edge representations," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 513–521.

[27] N. K. Bose and N. A. Ahuja, "Superresolution and noise filtering using moving least squares," *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2239–2248, Aug. 2006.

[28] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 1–14.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[30] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016, *arXiv:1606.04671*. [Online]. Available: http://arxiv.org/abs/1606.04671

[31] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

[32] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 111–124, 2009.

[33] K. Karsch, C. Liu, and S. B. Kang, "Depth transfer: Depth extraction from video using non-parametric sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2144–2158, Nov. 2014.

[34] J. Lu, K. Shi, D. Min, L. Lin, and M. N. Do, "Cross-based local multipoint filtering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 430–437.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[36] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5162–5170.

[37] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*. [Online]. Available: http://arxiv.org/abs/1212.5701

[38] H. Ren, M. El-Khamy, and J. Lee, "Image super resolution based on fusing multiple convolution neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 54–61.

[39] F. Garcia, D. Aouada, B. Ottersten, B. Mirbach, and T. Solignac, "Real-time depth enhancement by fusion for RGB-D cameras," *IET Comput. Vis.*, vol. 7, no. 5, pp. 335–345, Oct. 2013.

[40] Y. Zhang and T. Funkhouser, "Deep depth completion of a single RGB-D image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 175–185.

[41] F. Garcia, D. Aouada, B. Mirbach, T. Solignac, and B. Ottersten, "Unified multi-lateral filter for real-time depth map enhancement," *Image Vis. Comput.*, vol. 41, pp. 26–41, Sep. 2015.

SHUAIHAO LI received the Ph.D. degree from the School of Computer Science, Wuhan University, in 2019. He is currently with the International Business School, Sichuan International Studies University. He has published seven articles in major international journals indexed by Science Citation Index Expanded. His research interests include computer vision, 3D reconstruction, image processing, and information retrieval.

BIN ZHANG (Member, IEEE) received the M.S. degree in computer technology from Wuhan University, in 2012. He is currently pursuing the Ph.D. degree in computer with Wuhan University and the City University of Hong Kong. He is the author of a computer book and more than ten articles. His research interests include artificial intelligence and machine learning. He is a member of the China Technology Economics Association/China Computer Federation (CCF)/ACM.

WEIPING ZHU (Member, IEEE) received the Ph.D. degree from The Hong Kong Polytechnic University, Hong Kong, in 2013. He was with The Hong Kong Polytechnic University and The University of Hong Kong. He was also visited IRISA-INRIA, France, from 2011 to 2012. He is currently an Associate Professor with the School of Computer Science, Wuhan University. He is also a Chutian Young Scholar and a Luojia Young Scholar. He has published over 40 articles in major international journals and conference proceedings. His research interests include RFID, WSN, distributed computing, and pervasive computing.

XINFENG YANG was born in Henan, China, in 1979. He is currently pursuing the Ph.D. degree in software engineering with the Computer School, Wuhan University. He is also an Associate Professor with the Nanyang Institute of Technology. His research interests include image processing, cloud computing, and 3D reconstruction.