Received December 31, 2021, accepted January 12, 2022, date of publication January 19, 2022, date of current version January 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3144845

SDF-SLAM: A Deep Learning Based Highly Accurate SLAM Using Monocular Camera Aiming at Indoor Map Reconstruction With Semantic and Depth Fusion

CHEN YANG, (Member, IEEE), QI CHEN, YAOYAO YANG, JINGYU ZHANG, MINSHUN WU, (Member, IEEE), AND KUIZHI MEI, (Member, IEEE)
School of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, China

Corresponding author: Chen Yang (chyang00@xjtu.edu.cn)

This work was supported in part by the National Natural Science Foundation (NNSF) of China under Grant 62176206, in part by the Aeronautical Science Foundation of China under Grant 2020Z066070001, and in part by Key-Area Research and Development Program of Guangdong Province under Grant 2019B010154002.

ABSTRACT Simultaneous localization and mapping (SLAM) is considered as a key technique in augmented reality (AR), robotics and unmanned driving. In the field of SLAM, solutions based on monocular sensors have gradually become important due to their ability to recognize more environmental information with simple structures and low costs. Feature-based ORB-SLAM is popular in many applications, but it has many limitations in complex indoor scenes. Firstly, camera pose estimation based on monocular images is greatly affected by the environment; secondly, monocular images lack scale information and cannot be used to obtain image depth information; thirdly, monocular based SLAM builds a fused map of feature points that lacks semantic information, which is incomprehensible for machine. To solve the aforementioned issues, this paper proposes an SDF-SLAM model based on deep learning, which can perform camera pose estimation in a wider indoor environment and can also perform depth estimation and semantic segmentation on monocular images to obtain an understandable three-dimensional map. SDF-SLAM is tested and verified using a CPU platform and two sets of indoor scenes. The results show that the average accuracy of the predicted point cloud coordinates reaches 90%, and the average accuracy of the semantic labels reaches 67%. Moreover, compared with the state-of-the-art SLAM frameworks, such as ORB-SLAM, LSD-SLAM, and CNN-SLAM, the absolute error of the camera trajectory on indoor data with more feature points is reduced from 0.436 m, 0.495 m, and 0.243 m to 0.037 m, respectively. On indoor data with fewer feature points, they decrease from 1.826 m, 1.206 m, and 0.264 m to 0.124 m, respectively.

INDEX TERMS SLAM, deep learning, camera pose estimation, semantic segmentation, three-dimensional map.

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) has become a promising theory for computer vision and mobile robotics. SLAM can use a sensor to estimate its own pose and build a map. SLAM-based intelligent systems, such as Google Driverless Cars and UNIBOT Housekeeper robots, have recently been widely used in indoor scenes. These systems recognize objects and their specific locations by perceiving an indoor environment, thereby creating a map to help humans understand an unknown environment. Initially, the reconstructed maps are mostly raster maps and topological maps, which are used to guide robots to complete tasks such as positioning, navigation, and path planning. However, these maps limit a robot's understanding of the environment and hinder the development of machine intelligence. Now, with the development of intelligent mobile robots, it is very important for a machine to be able to construct an understandable three-dimensional map, which is also called a three-dimensional semantic map, to strengthen human-computer interactions.

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

The application of intelligent systems realized by SLAM technology involves two main solutions: nonvisual solutions based on lidar and visual solutions based on cameras. Cameras, especially monocular cameras, not only have a lower cost but can also obtain more environmental information than radar. Furthermore, deep learning has made tremendous progress in computer vision applications, especially in image feature extraction [5], semantic segmentation [6], and depth estimation [7]. Therefore, monocular SLAM [1], represented by the versatile and accurate monocular SLAM system using ORB (Oriented FAST and Rotated BRIEF) features (named ORB-SLAM) [2] and the large-scale direct monocular SLAM (named LSD-SLAM) [3], which can completely perform real-time camera pose estimation and map reconstruction based on CPU platforms, is increasingly important. However, these frameworks still have three main problems. First, camera pose estimation is easily affected by the shaking of moving bodies, scene brightness, image texture information and other problems, which lead to insufficient stability of system operations. Next, in terms of map reconstruction, the SLAM framework builds maps that are incomprehensible to humans and machines. Thirdly, considering the lack of scale in monocular cameras and that depth images cannot be obtained, absolute pose estimation and real 3D map reconstruction cannot be completed. Therefore, state-of-the-art SLAM frameworks have great limitations in practicality and human-machine interactions.

Motivated by the above challenges, the main idea of this paper is to complete camera pose estimation and three-dimensional semantic map reconstruction, in which camera pose estimation is based on the feature method and deep learning. In this work, we propose and construct a semantic and depth fusion SLAM (SDF-SLAM) framework, which fuses camera pose information and depth and semantic information of each frame to achieve three-dimensional semantic map reconstruction. The main contribution of this paper is as follows:

- We propose the feature point and feature description convolutional neural network (FPFDCNN) for feature point extraction from two different frames of images and generate vectors as descriptors for each feature point. FPFDCNN adopts the pixel shuffle algorithm to carry out superpixel restoration on low-resolution images. Using FPFDCNN can significantly reduce the environmental impact and finally generate high-resolution images.
- We design and implement a semantic and deep fusion convolutional neural network (SDFCNN), which can perform semantic segmentation and depth estimation simultaneously by inferring input color RGB images. It greatly reduces the number of parameters, computation and time required for inferring the two network models.
- We propose a monocular visual SLAM architecture based on deep learning for camera pose estimation and three-dimensional semantic map reconstruction. Moreover, this paper also adds a data correction module to optimize a point cloud map globally to establish a consistent point cloud map and to greatly reduce the inference time and network parameters.

The rest of the paper is organized as follows. Section II briefly presents a review of various SLAM achievements and mainly introduces the related work. Section III provides the research motivation of this paper. In Section IV, we detail the proposed SDF-SLAM method. Subsequently, the experimental results are presented in Section V. Finally, a summary is provided in Section VI. In addition, Appendix A provides a detailed overview of the FPFDCNN structure, and Appendix B provides a detailed overview of the SDFCNN structure.

II. RELATED WORK

Benefitting from SLAM technology in the field of computer vision and robotics, the recent research work has been mainly conducted from the perspectives of camera pose estimation and map reconstruction.

A direct method [9] is a typical way to represent and perform camera pose estimation. The framework of LSD-SLAM [3] falls into this category, which associates data and fuses all pixel data to establish a dense map. The direct method is based on the assumption that the gray level of a spatial point is invariant from various perspectives. In this work, the camera attitude, which includes the rotation matrix and displacement matrix, is initially obtained by minimizing the brightness error of pixels between images in different motion states. Then, all the pixels of each frame in the video image according to the image coordinates of the marked points are converted into coordinate points in the camera coordinate system by the camera internal reference. Then, the camera coordinate system's coordinate points are converted into coordinate points in the world coordinate system by the camera external reference and are denoted as the real coordinates. The world coordinates of all pixels corresponding to world coordinate fusion are used to obtain the whole indoor scenario map.

ORB-SLAM [2] is based on a feature-based method [10] to estimate camera pose and establish a sparse map. It extracts texture information, such as edges and corners, from an image as feature points using the scale invariant feature transform (SIFT) algorithm [12], speeded up robust features (SURF) algorithm [13], oriented fast and rotated BRIEF (ORB) algorithm [14] or other feature extraction algorithms. Furthermore, feature points extracted from two adjacent frames are matched according to their similarity, and feature point matching pairs are obtained using brute force (BF) matchers [15] or fast library for approximate nearest neighbors (Flann)-based matchers [16]. According to a matching pair of feature points, the feature points in the previous frame of an image are projected into the next frame of the image through the change in camera pose. Perspective-n-Point (PnP) [17] was used to minimize the coordinate error between a projected point and a real matching point. PnP transforms the extracted feature points into the world coordinate system through camera pose and fuses all feature points to establish a sparse map.

With the good performance of deep learning in image tasks in recent years, it has become popular to apply deep learning to replace modules in SLAM system. For example, PoseNet [18] uses images as input, FlowNet [19] uses optical flow as input, and DeepVO [20] uses two adjacent frames as input to estimate a camera's six degrees of freedom (DOF) poses through a convolutional neural network. Learning methods have also made remarkable progress in the three-dimensional semantic understanding of images. A three-dimensional semantic map tag can help a machine to quickly understand and perceive the information of the surrounding environment, which enhances human-machine mutual assistance. Convolutional neural networks (CNN), such as the DeepLab [6] network, are used for semantic segmentation of every frame of an image and tag each pixel semantically. Pixel coordinates with semantic information are mapped into world coordinates, and finally, a 3D semantic map is obtained by direct fusion, such as the real-time dense monocular SLAM fused with CNN-predicted depth maps (named CNN-SLAM) [21]. In addition, there are some deep learning-based SLAM systems. Reference [22] proposed DeepSLAM, an unsupervised deep learning-based monocular SLAM system. It takes monocular color images as input and outputs pose trajectory, depth map, and three-dimensional point cloud information simultaneously. In [23], a novel deep learning-based approach that can be used to minimize the semantic SLAM estimation error by identifying different noise patterns and trying to reduce them was proposed.

III. MOTIVATION

First, as shown in Table 1, LSD-SLAM, ORB-SLAM and FlowNet [19] are compared according to the calculation of the camera rotation matrix error and displacement error on the 08, 09, and 10 sequences of the KITTI dataset [29]. The accuracy of camera pose estimation by ORB-SLAM is better than that of the other methods. Therefore, we improve the accuracy of camera pose estimation based on ORB-SLAM. However, the camera pose estimation based on ORB-SLAM is greatly affected by the environment. The accuracy of camera pose estimation based on the feature point method mainly depends on the number of feature points selected and the accuracy of feature point matching. Therefore, when texture features are weak or the environmental features are few, it is impossible to estimate the camera pose, as corner information cannot be extracted. In recent years, deep learning has made tremendous progress in computer vision applications; for example, AlexNet [30] carries out image classification through feature extraction, and the top 1-5 accuracy rate of ImageNet can reach 80.2%. Therefore, this paper designs a feature point extraction algorithm based on a convolutional neural network using an encoder-decoder framework, which improves the accuracy of feature point matching between images and camera pose estimation.

TABLE 1. Comparison of camera pose estimation errors.

No	ORB-SLAM [2]		LSD-SLAM [3]		FlowNet [19]	
	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)
08	5	0.0067	19.39	0.0393	9.98	0.0544
09	1.6	0.0024	9.26	0.0279	12.64	0.0804
10	1.2	0.0018	27.55	0.0409	11.65	0.0728

Trans represents the estimated displacement deviation rate per unit.
Rot represents the estimated rotation angle error per meter.

TABLE 2. Comparison of different types of camera sensors.

Sensor	Depth Camera		Multi-Camera	Monocular Camera
Technology	Structured light	ToF	Stereo matching	Deep learning
Source	Intel [31]	ICAR-I'2015 [25]	StereoLab [32]	TPAMI-I'2020 [24]
Product	D435i	KinectV2	ZED2	CSPN
Resolution	1280×720	512×424	2208×1242	1242×375
Range	10 m	4 m	20 m	unlimited
Error	0.2 m	0.002 m	0.07 m	2.07 m/1 km
Speed	30 fps	30 fps	15 fps	5 fps
Light effect	high	medium	low	low
Hardware Cost	medium	high	low	low
Algorithm Complexity	medium	low	high	high

Next, ORB-SLAM needs to obtain the depth information of the current environment through a visual sensor. Table 2 compares the performance of three visual sensors. Compared with depth cameras and multi-camera, monocular cameras, although their speed is very slow, use a wider environment and have a wider measurement range. At the same time, a monocular camera is less affected by light, and the hardware cost is lower. Therefore, this paper obtains depth information based on a monocular camera. However, depth information on an absolute scale cannot be obtained due to the lack of scale information in monocular images. With the good performance of deep learning in end-to-end image prediction, depth estimation of monocular images based on deep learning can obtain depth images on an absolute scale. For example, the fully convolutional recurrent network (FCRN) was tested on the NYU dataset [36], and the average error of depth estimation in an indoor environment within a range of ten meters reached 0.573.

Finally, a map created by ORB-SLAM is a fused map of feature points that lacks semantic information and cannot be understood by a machine. With the development of machine intelligence, it is very important to establish understandable semantic maps of indoor scenes. As shown in Table 3, three-dimensional semantic maps of indoor scenes play very important roles, such as helping visually impaired people navigate, managing homes to ensure their safety, and instructing robots to help people at work. However, there are still some limitations in three-dimensional map reconstruction using 3D semantic segmentation. For example, Pointnet [33] can generate dense 3D maps and achieves 90% accuracy in

TABLE 3. Three-dimensional semantic map application scenarios.

Work	Source	Scene
[26]	ICBAKE	Help visually impaired people
[27]	ICIIBMS	Housekeeping services
[28]	ICCAS	Warehouse robots

classification on the Modelnet_40 dataset, but it can only identify a single object and has a large number of parameters. When the input data size is only 8192×5, the total number of network parameters is 16.6 M, and the number of calculations can reach 3633 M. In addition, its input point cloud data are unordered data, and there is no correlation between the data and global information. Therefore, we introduce a deep learning method for 2D semantic segmentation in the SDF-SLAM framework and fuse it with semantic images to construct a 3D semantic map.

IV. SYSTEM DESCRIPTION

A. OVERVIEW OF SDF-SLAM

The SDF-SLAM architecture is mainly composed of three modules, as shown in Fig. 1: camera pose estimation, three-dimensional semantic map reconstruction, and data correction. We can use different threads to run each module separately and pass data between each module to implement the entire SDF-SLAM framework. The camera pose estimation module passes pose information and key frame information to the point cloud map module to generate a three-dimensional point cloud map, and then, the three-dimensional semantic map reconstruction module passes depth information to the camera pose module to estimate the camera pose. The data calibration module, which is mainly divided into global optimization and local optimization, generates a consistent three-dimensional semantic point cloud map by integrating camera pose information and point cloud data. Global optimization is similar to the loopback detection method in ORB-SLAM, which randomly selects key frames and performs camera pose estimation on the two key frames to establish a constraint relationship and to further optimize the camera pose information. In addition, partial optimization, in which the feature point world coordinates of the second frame and the first frame are unified according to the feature point matching pairs obtained by FPFDCNN inference, can finally lead to a consistent global point cloud map.

The design flow of SDF-SLAM includes a convolutional neural network, which is used to extract image feature points first, and then the feature points of two adjacent frames of images are matched according to similarity to obtain feature point matching pairs. Next, we apply the method of minimizing projected coordinates and true matching pair coordinates to obtain the camera rotation matrix and displacement matrix. Then, the other convolutional neural network estimates the depth of the image and performs semantic segmentation to simultaneously obtain the depth information and semantic information of an image.

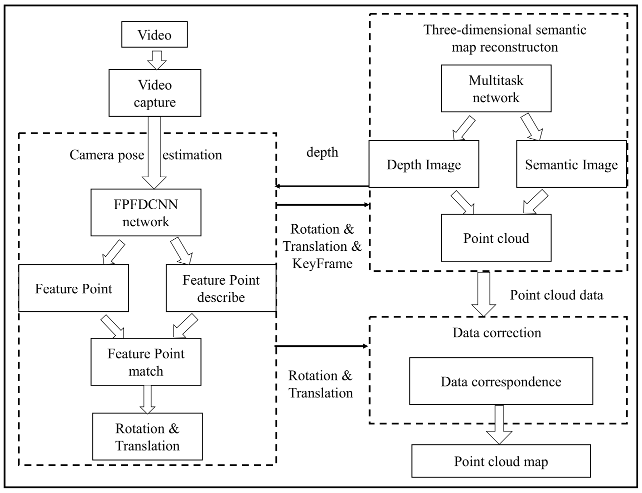

FIGURE 1. SDF-SLAM architecture.

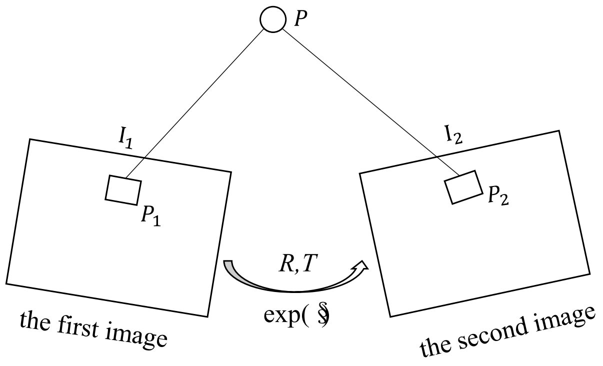

FIGURE 2. The principle of camera pose estimation.

B. CAMERA POSE ESTIMATION

Camera pose estimation is the key to effectively realizing the SLAM system, where the first step is to extract feature points from two different frames of images and generate vectors as descriptors for each feature point. Then, we use the matching pairs to estimate the pose of a camera by the reprojection method. Currently, the state-of-art methods use feature point matching to estimate camera pose based on the PnP method [17], and the calculation principle is shown in Fig. 2, where P is the 3D feature space point, P_1 is the coordinate of P mapped to the first image, P_2 is the coordinate of P mapped to the second image and P_2 is also the matching coordinate point of P_1. We obtain P_1 under the condition that P and Z_1 are determined, where Z_1 represents the camera depth value of point P in the first picture and K represents the camera internal parameter, as shown in (1). The camera change matrix T consists of pose rotation matrix R and displacement matrix t, as shown in (2). Point $P_2^{\wedge}$ is the observation coordinate point of the second image obtained by P through the change in camera pose, as shown in (3). The camera pose change is estimated by minimizing the error between P_2 and $P_2^{\wedge}$, as shown in (4).

$$P_1 = \frac{1}{Z_1} KP \quad (1)$$

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2}$$

$$P_2^{\wedge} = TP \tag{3}$$

$$min\,(T) = arg\,min \frac{1}{2} \sum\nolimits_{i=1}^{n} \left\| P_2 - P_2^{\wedge} \right\| \tag{4}$$

We select the rgbd_dataset_freiburg1_desk2 (f1_desk2) scene from the TUM dataset [35] for feature point extraction named FP dataset. Furthermore, the feature points of the first image are mapped to the second image according to the camera pose estimation from the associated text file, and the feature point matching pairs of the two images are obtained to train the feature point descriptors. If the rotation matrix of the camera at position 1 is $q1$, the displacement matrix is $t1$, and the coordinate of the spatial feature point P of the first image in the camera coordinate system is $p1$. The rotation matrix of the camera at position 2 is $q2$, and the displacement matrix is $t2$, so the coordinate $p2$ of the feature point P in camera coordinates can be obtained. (5) and (6) represent the coordinate information of point P in camera coordinates by changing the camera matrix. (7) represents the position of point P in the second figure and combines $p1$ and $p2$ to produce a matching pair, which is obtained by combining (5) and (6).

$$q1 \times P + t1 = p1 \tag{5}$$

$$q2 \times P + t2 = p2 \tag{6}$$

$$p2 = q2 \times \frac{1}{q1} \times (p1 - t1) + t2 \tag{7}$$

In this work, we propose a convolutional neural network named FPFDCNN for feature point extraction. The structure of FPFDCNN is shown in Fig. 3. The FPFDCNN model consists of five components, including input layer, encoder layers, decoder layers, output layers and concatenate layer. The image of input layer is first extracted by six levels of encoding in the encoder layers. Among these levels, we use small convolution kernels of 3 × 3 and 1 × 1 to extract the feature image of the previous layer without changing the resolution of the feature image by edge filling. The 4 × 4 convolution kernel is used to downsample the upper layer feature image by convolution calculations. Considering that a 256-dimensional vector needs to be obtained for each pixel as a feature point descriptor, we set the number of channels of the last layer of the feature image to 256.

The decoder layers are mainly based on the inverse decoding method to restore the feature image, which is divided into two branches: the first one restores the original feature image by up-sampling, and determines whether each pixel is a feature point. The other one is to get the feature descriptor for each pixel. The pixel shuffle algorithm is used to predict the probability of each pixel as a feature point, which realizes the reaggregation of all points by reordering the feature images according to the square root of the channels for the period. Then, a matrix with the same size as the input image is generated as a result. The probability of each pixel value corresponding to a feature point is calculated. In addition, the bilinear algorithm is used to upsample and restore the feature image, and it can generate a 256-dimensional vector for each pixel as a feature descriptor.

We select N pixel points whose feature point discrimination probability is not 0 as feature points to obtain 2-dimensional feature point coordinates corresponding to their feature point descriptors. According to the correspondence between feature points and descriptors, an $N \times 258$-dimensional matrix is finally generated as the output matrix. For the detailed overview of FPFDCNN structure, please refer to Appendix A.

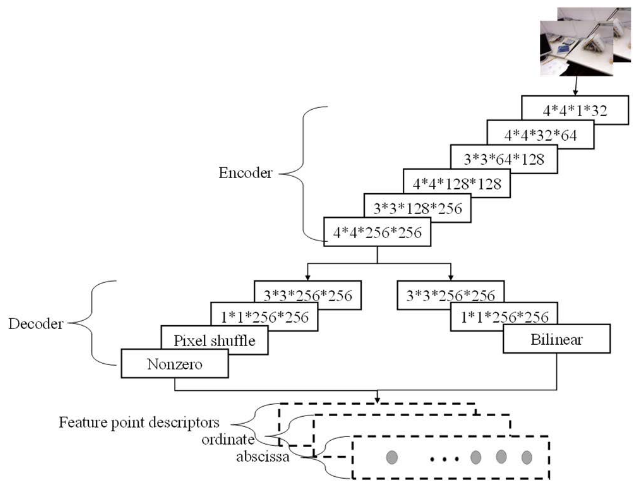

FIGURE 3. **FPFDCNN network structure.**

In addition, a range matcher is designed to find the best matching pair, as shown in Fig. 4. The design flow of the range matcher is as follows: The rectangular area is delineated with position X of the first image feature point as the center and 140 as the side length. Then, the feature points extracted from the second image within the range are selected. A certain feature point, $Out1$, is selected in the first image, and its descriptor includes the third column to the last column, which have scale sizes of 1 × 256. To find n points within the rectangular range of the second image, the descriptor includes the third column to the last column and has a size of n × 256. Then, $Out1$ is copied row by row to obtain an n × 256 matrix, and $Out1$ and $Out2$ are used to calculate the minimum distance between feature points by (8). The coordinates of the two feature points with the shortest distance are connected to obtain a matching pair.

$$distance = min(\sqrt{\sum\nolimits_{i=1}^{256} (Out1_i - Out2_i)^2}) \tag{8}$$

According to the feature point matching pairs obtained by (8) and considering that the feature point dataset contains depth images, we use 3D to 2D minimization projection methods to estimate the camera pose.

As shown in (9), (u, v) represents the 2D coordinates of the feature point with the upper left corner of the image as the origin, and (X, Y, Z) represents the 3D coordinates obtained from the camera coordinate system according to the depth image. Moreover, the rotation matrix R (whose elements are

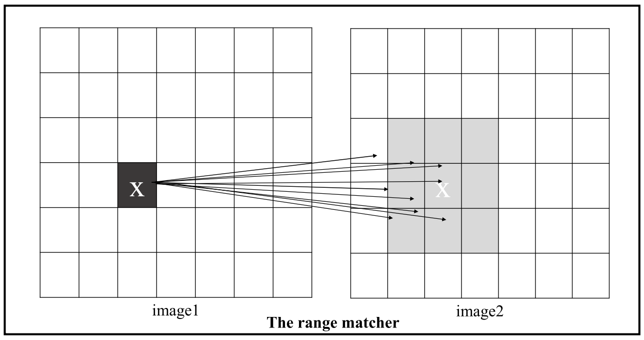

FIGURE 4. Range matcher.

$r_{11} \ldots r_{33}$) and displacement matrix T (whose elements are $t_1 \ldots t_3$) of the camera in world coordinates are obtained according to the random sample consensus (RANSAC) algorithm, which minimizes coordinate errors between coordinates of real matching pairs and (u, v) in the next image and the camera internal parameter matrix ($\mathrm{f_x}$, $\mathrm{f_y}$, $\mathrm{c_x}$, $\mathrm{c_y}$) and the camera distortion parameters, where $\mathrm{f_x}$ and $\mathrm{f_y}$ represent the focal lengths of the x and y axes, respectively, and $\mathrm{c_x}$ and $\mathrm{c_y}$ are the pivot points. Finally, *poses* in world coordinates can be obtained, as shown in (10).

$$\begin{Bmatrix} \mathrm{u} \\ \mathrm{v} \\ 1 \end{Bmatrix} = \begin{bmatrix} \mathrm{f_x} & 0 & \mathrm{c_x} \\ 0 & \mathrm{f_y} & \mathrm{c_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{Bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{Bmatrix} \tag{9}$$

$$pose = -\frac{1}{R} \times T \tag{10}$$

In addition, we define displacement matrix error *errT* and rotation error *errR* by referring to [37], where ΔT represents the change matrix of predicted value pose *PT* and true value pose T, ΔR represents the rotation matrix between predicted value rotation matrix *PR* and true value pose R, and E represents the unit matrix, as shown in (11)-(14).

$$T = \Delta T * PT \tag{11}$$

$$errT = \|\Delta T - E\| \tag{12}$$

$$R = \Delta R * PR \tag{13}$$

$$errR = \|\Delta R - E\| \tag{14}$$

C. THREE-DIMENSIONAL SEMANTIC MAP RECONSTRUCTION

The method of reconstructing a three-dimensional semantic map is divided into three steps. (1) Design a deep semantic fusion convolutional neural network to complete the two tasks of image depth estimation and semantic segmentation. (2) Fuse the depth information and semantic information into three-dimensional semantic point cloud data. (3) Map the point cloud data of each frame to the world coordinate system according to the camera pose to obtain a three-dimensional semantic map. In this paper, we design an SDFCNN model whose structure is shown in Fig. 5. It uses a unified semantic down-sampling layer and an up-sampling layer for feature extraction and restoration and uses different discriminative layers to classify features, probability estimation, and depth regression estimation.

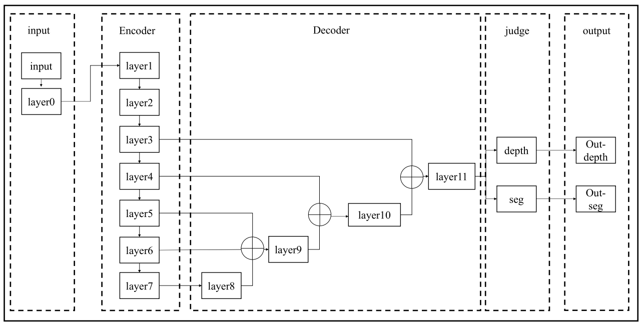

FIGURE 5. SDFCNN network structure.

SDF-SLAM preprocesses an original image in the input layer. According to the statistical characteristics of natural images, in addition to remote sensing images or medical images, the mean value of each channel is (0.485, 0.456, 0.406), and the variance is the value of each channel (0.229, 0.224, 0.225). Therefore, the image is standardized based on (15), and the sample of the feature image conforms to a standard normal distribution with a mean of 0 and a standard deviation of 1. Considering that the resolution of the input image is too large, which affects the number of inference calculations and speed of the SDFCNN model. Therefore, layer 0 is used in the input layer to reduce the resolution of the feature image and the incoming SDFCNN model by a linear interpolation of the preprocessed feature image, which can reduce the number of parameters and shorten the time.

$$\mathrm{img\_out} = \frac{\mathrm{img\_in} \times \mathrm{img\_scale} - \mathrm{img\_mean}}{\mathrm{img\_std}} \tag{15}$$

The encoder layers are mainly based on a convolutional neural network for image down-sampling. We apply the method of operator fusion, so the convolution operation, BatchNorm, and activation function are encapsulated into a convbnrelu module, as shown in the left image of Fig. 6. To prevent the phenomena of gradient explosion and gradient disappearance in the network, the convolution layer uses 1×1 and 3×3 small convolutions for depthwise separable convolution operations. Then, three convbnrelu modules are added into a block module in sequence, as shown in the middle image of Fig. 6. To facilitate network optimization, each layer in the encoder network accumulates the output value of each block with the output value of the previous block through a residual connection to obtain the input value of the next layer. The structure of the layer is shown in the right image of Fig. 6. The numbers of block modules superimposed on each layer are different, and as the number of network layers increases, the number of superimposed block layers increases. In summary, the composition structure of the encoder network is shown in Table 4.

The decoder layers consist of four layers, which refine the feature image through convolution calculations and merge the multiscale feature image with up-sampling. Layer 8 uses

TABLE 4. The encoder layers' composition structure.

Layer	Model	Number
Layer 1	Convbnrelu	1
Layer 2	block	1
Layer 3	block	2
Layer 4	block	3
Layer 5	block	4
Layer 6	block	3
Layer 7	block	4

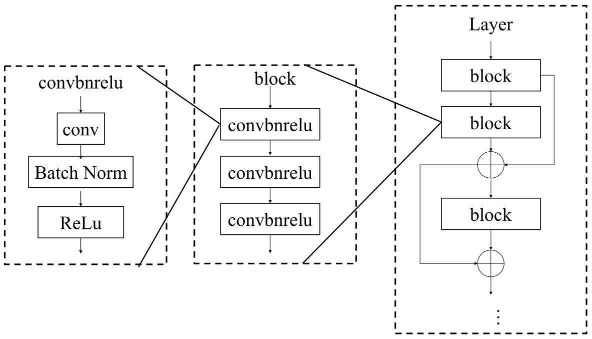

FIGURE 6. The convbnrelu module structure in the left, the block module structure in the middle, and the layer module structure in the right.

1-size convolution kernel and 5-size max pooling to perform operations alternately to obtain a feature image. Then the feature image is restored to the same size as the Layer 6 and Layer 5 through bilinear interpolation up-sampling calculation. Layer 9 is the fusion of three feature images of Layer 5, Layer 6 and Layer 8 through the method of adding corresponding pixels. Layer 10 has the same design idea as Layer 8. The feature image is firstly calculated by bilinear interpolation up-sampling to obtain a feature image with the same size as Layer 4. Then, the feature images of Layer 10 and Layer 4 are fused by adding corresponding pixels. Considering that the size of the feature image restored by Layer 11 is the same as that of Layer 3, that is, the scale is half of the original image, the calculation of max pooling is too much and the time is too long, so Layer 11 selects 1×1 convolution kernel for convolution calculation and then fuses the feature images of Layer 3 and Layer 10 by means of pixel addition.

Next, SDF-SLAM performs semantic segmentation and depth estimation in the judgement layer. Considering that semantic segmentation is performed n times, the semantic segmentation prediction uses a $1 \times 1 \times n$ convolution kernel to calculate the decoder network output feature image and predict the probability of each pixel for classification. Depth prediction uses a 1×1 convolution kernel to perform regression fitting on the depth of field of each pixel and obtains a fitting result. The output layer restores the semantic image and depth image predicted by the discriminant layer to the same image with the original image resolution of 640×480 by bilinear interpolation, which is convenient for the generation of dense point cloud images.

Last, we obtain the image depth information, which helps us acquire the coordinates (X, Y, Z) of all pixels in the camera coordinate system as well as the semantic information that can be obtained for all pixels. Each piece of semantic information corresponds to an RGB value. We integrate depth information and semantic information to generate point cloud data (X, Y, Z, R, B, G) and convert all the point cloud data of each frame image to coordinates in the same world coordinate system through the change in camera pose to obtain a three-dimensional semantic map. For the detailed structure of SDFCNN, please refer to Appendix B.

Additionally, we use MPA and MIoU as semantic image accuracy indexes for comparison, as shown in (16) and (17), where MPA represents the average proportion of the number of correctly classified pixels in each class and MIoU represents the ratio of the intersection and union of the true and predicted values. In addition, RMS and REL are used as performance indexes for the comparison of depth images, as shown in (18) and (19).

$$\mathrm{MPA} = \frac{1}{K+1}\sum\nolimits_{i=0}^{k}\frac{p_{ii}}{\sum_{j=0}^{k}p_{ij}} \tag{16}$$

$$\mathrm{MIoU} = \frac{1}{K+1}\sum\nolimits_{i=0}^{k}\frac{p_{ii}}{p_{ii}+\sum_{j=0}^{k}p_{ji}+\sum_{j=0}^{k}p_{ij}} \tag{17}$$

where p_{ii} denotes the pixel points of class i predicted by class i. p_{ij} denotes the j-th pixel points predicted by class i. p_{ji} denotes the number of class i pixels predicted by class j. K represents the number of categories.

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum\nolimits_{i=1}^{N}\left|D_i - D_i^*\right|^2} \tag{18}$$

$$\mathrm{REL} = \frac{1}{N}\sum\nolimits_{i=1}^{N}\frac{\left|D_i - D_i^*\right|}{D_i^*} \tag{19}$$

$$max\left(\frac{D_i}{D_i^*},\frac{D_i^*}{D_i}\right) < T \tag{20}$$

RMS represents the relative error between the true value D_i and the predicted value D_i^*. REL represents the absolute error between the true value D_i and the predicted value D_i^*, and accuracy refers to the proportion of the correctly predicted pixels out of the total number of pixels, where the correctly predicted pixels are when the ratio between the real value D_i and the predicted value D_i^* is less than the set threshold T, as shown in (20).

V. EXPERIMENTS AND COMPARISON

A. EXPERIMENTAL SETUP

The experimental platform for SDF-SLAM is shown in Table 5. The open source framework PyTorch is mainly used for training and inference of FPFDCNN and SDFCNN, and the functions provided by the torch library for convolution, pooling and other calculations are mainly used. Opencv is mainly used for image processing, such as image reading and the conversion the matrix of camera pose estimation. The PCL library is mainly used for point cloud data processing,

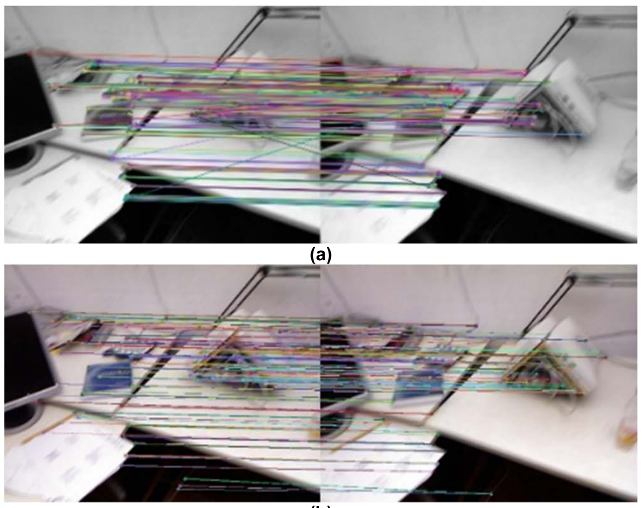

FIGURE 7. Feature point extraction and matching of clear images with ORB and FPFDCNN.

TABLE 5. SDF-SLAM experimental platform.

Platforms	Description
Experimental platform	Intel i7-8750 CPU
Network training platform	RTX2070 GPU
Operating system	Windows
Language	C++ and Python
Open source framework	Pytorch, Opencv. PCL, g2o, eigen, pangolin

such as storage and the conversion of point cloud data, and g2o is mainly used for camera pose optimization. Eigen is mainly used to calculate matrix vectors, and pangolin is mainly used to visualize camera trajectory changes.

B. ACCURACY OF FPFDCNN

Based on the TUM dataset [35], a feature point dataset [34] for feature point extraction and matching is produced. The feature point dataset uses the Harris corner detection algorithm for all image frames to select and map the feature points of the first image to the second image according to the real camera pose estimation; then, the feature point matching pair of the two images is obtained, and then, the feature point matching pair is recorded in the label file.

We extracted feature points from two similar clear images and blurred images using the state-of-the-art ORB method and the proposed FPFDCNN, as shown in Fig. 7 and Fig. 8. According to the probability order of feature points, we select 500 feature points and conduct feature matching for these feature points. Fig. 7 (a) and Fig. 8 (a) represent the ORB method combined with the BF matcher for feature extraction and matching of clear images and blurred images, and Fig. 7 (b) and Fig. 8 (b) represent the FPFDCNN. The FPFDCNN combined with the range matcher can obtain better feature point performance, and the probability of a large error is smaller.

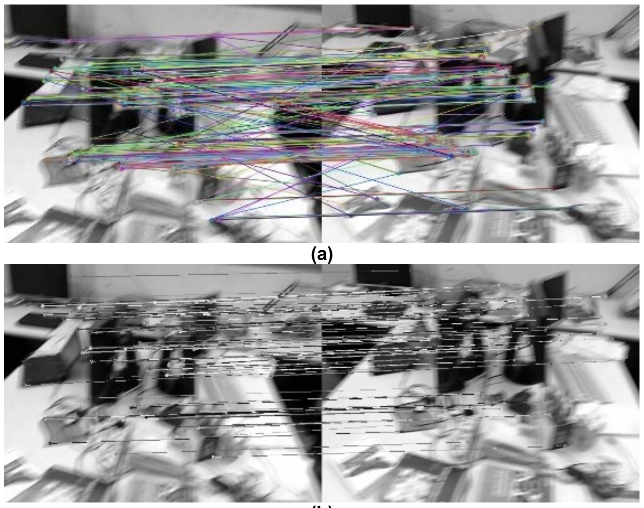

FIGURE 8. Feature point extraction and matching of blurred images with ORB and FPFDCNN.

TABLE 6. Accuracy of matching feature points.

Scenario	Method	Accuracy rate (Correct matching/Match point)
Clear image	ORB+ BF matcher	83.8% (276/329)
	FPFDCNN+ range matcher	90.8% (368/405)
Blurred image	ORB+ BF matcher	65.5% (161/249)
	FPFDCNN+ range matcher	86.0% (263/304)

Using the FPFDCNN to extract and match the feature points of two frames of images, a quantitative analysis of the matching pairs is performed. In the analysis, 100 clear images and 100 blurred images are selected from the feature point dataset as the test set, the feature point matching pairs from the extracted feature points through the camera posture changes are obtained, and the correct matching outcomes within the range of plus and minus five pixels are judged to obtain the feature point matching results. The correct rate is shown in Table 6. It can be seen that in the clear test image, the feature point matching accuracy of the FPFDCNN is improved by 7% compared to that of the ORB method; in the fuzzy test image, the feature point matching accuracy obtained by the FPFDCNN is better than that of the ORB method by 20.5%. The average feature point matching accuracy rate of the entire test set is 14%.

Next, we calculate the camera pose in world coordinates based on ORB method and FPFDCNN model, and compare the errors of rotation and displacement matrices based on (11)-(14). As a result, the average pose error of the TUM (f1_desk2) dataset is shown in Table 7. Using proposed FPFDCNN model, the displacement error is 0.49% less and the rotation error is 0.0008 less than that of ORB method. The camera pose estimation based on FPFDCNN has a large

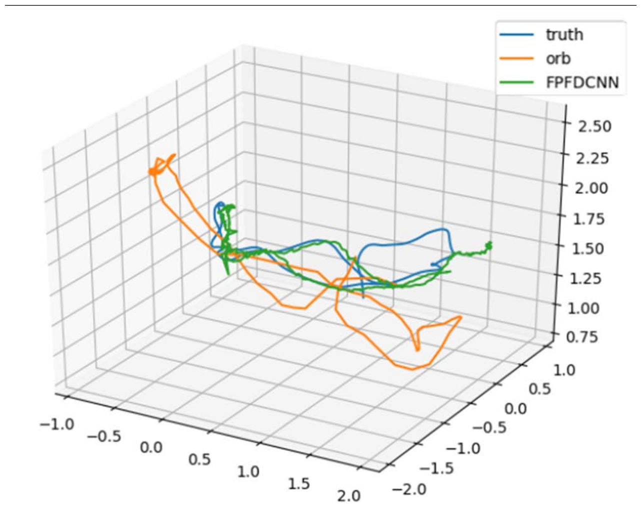

FIGURE 9. Camera trajectory diagram.

TABLE 7. Error in camera pose estimation.

Method	*errT* (%)	*errR* (deg/m)
ORB[2]	4.21	0.0150
FPFDCNN	3.72	0.0142

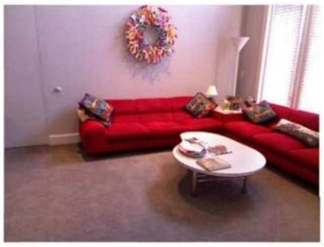

FIGURE 10. The original image.

improvement in displacement accuracy. The camera track is obtained, as shown in Fig. 9, where the blue line represents the real camera pose, the orange line represents the camera pose estimated by the ORB method and the green line represents the camera pose estimated by FPFDCNN. The camera trajectory estimation based on the FPFDCNN model is closer to the real trajectory than that of ORB method.

C. PERFORMANCE OF SDFCNN

SDFCNN is trained for semantic segmentation and depth estimation and requires two loss functions for joint optimization. Semantic segmentation is a classification problem, so the cross-entropy loss function H is selected for training. Depth estimation is a regression problem, so the regression loss function L is used for training. The training of the entire network fuses the two loss functions to obtain the SDFCNN model loss function Loss, as shown in (21), where p is the semantic real value of the pixel, q is the semantic prediction value of the pixel, y is the depth value of the pixel from the real value, and the predicted value of the depth of the pixel λ is between 0 and 1, which means that the trained network is more focused on which task to train. The depth accuracy is required to be high, because depth information is used in

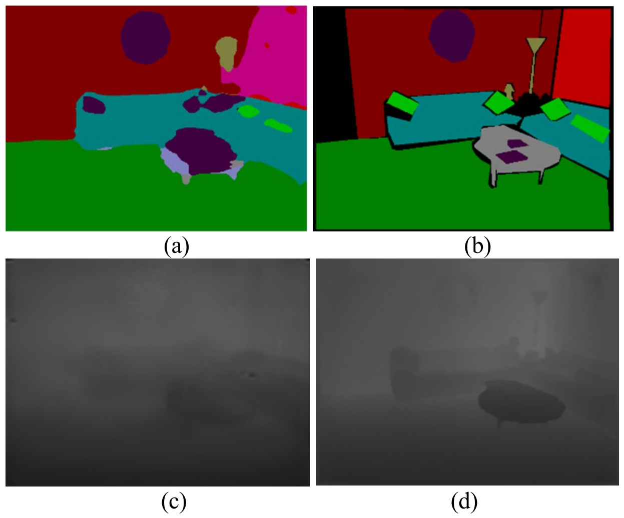

FIGURE 11. Semantic segmentation and depth estimation based on the SDFCNN model.

camera pose estimation. Therefore, model training is more focused on the depth accuracy, so the training value is set to 0.4.

$$\mathrm{Loss}\,(p, q, y, f(x), x) = -\lambda H(p, q) + (1-\lambda) L(y - f(x)) \tag{21}$$

The SDFCNN model is trained to obtain a network model for semantic image segmentation and depth estimation based on the existing NYU indoor dataset [36]. The semantic labels of the NYU dataset have 894 categories, which makes model convergence difficult, so that this work rearranges the labels according to the categories and maps them into 40-class semantic labels. Based on (21), the NYU dataset used for training, twenty images are trained at a time during the training process, the batch size is set to 20 in the hyperparameters, and the epoch is set to 20,000. For the first 20 epochs, the learning rate is 0.1, and then, the learning rate is adjusted to 0.01 for training. We obtain the semantic image and depth image using the SDFCNN model on the CPU platform, as shown in Fig. 11. Fig. 10 shows the original image. The predicted semantic image is shown in Fig. 11 (a), and the semantic label image is shown in Fig. 11 (b). Fig. 11 (c) shows the predicted depth image, and Fig. 11 (d) shows the depth label image. It can be seen that the SDFCNN model can accurately complete semantic segmentation and depth estimation. The overall difference between the predicted semantic image and the real semantic image is small, but the prediction bias appears at the boundary. In addition, the gap between the predicted depth image and the real depth image is also small, especially at close range.

The DeepLab [6] network has achieved good results in semantic image segmentation. We compare the semantic information performance of SDFCNN and DeepLab in terms of accuracy, inference time, network parameters, and calculation, as shown in Table 8. The indexes for the comparison of the semantic image accuracy are mainly MPA and MIoU, as defined in (16) and (17). It can be seen that although

TABLE 8. Performance comparison of semantic segmentation of SDFCNN and DeepLab.

Network	MPA	MIoU	Time	Parameter
DeepLab[6]	0.614	0.354	1.44 s	452.9 M
SDFCNN	0.561	0.319	0.66 s	11.8 M

TABLE 9. Performance comparison of SDFCNN and FCRN in terms of the depth information.

Network	RMS	REL	Accuracy (1.25/1.25^2/ 1.25^3)	Time	Parameter
FCRN[7]	0.590	0.131	0.715/0.864 /0.916	1.38 s	242 M
SDFCNN	0.692	0.149	0.761/0.950 /0.989	0.66 s	11.8 M

the SDFCNN model has a slight decrease in accuracy, i.e., the average correct rate of each class MAP is reduced by 0.053, and the ratio of MIoU at the intersection and union of the true and predicted values is reduced by 0.035. However, the inference time of the DeepLab network is approximately twice slower than that of the SDFCNN model, and the amount of parameters in the DeepLab network is approximately 40 times that of the SDFCNN model.

Considering that the FCRN [7] network has achieved good performance in depth image estimation, we compare the performance between SDFCNN and FCRN, in terms of average variance, absolute difference, accuracy, inference time, network parameters, calculation and so on, as shown in Table 9. In the same CPU platform, the FCRN and SDFCNN models are evaluated on the same test set. Although the SDFCNN model has a slight decrease in accuracy, i.e., the mean square error is reduced by 0.102, and the relative error is reduced by 0.018. At the threshold value of 1.25, the accuracy rate of SDFCNN is increased by 0.046 compared to FCRN; at the threshold value of 1.25^2, the accuracy rate is increased by 0.086; and at the threshold value of 1.25^3, the accuracy rate is increased by 0.073. Moreover, the inference time based on FCRN is approximately twice slower than that of SDFCNN, and the number of parameters in the FCRN network is approximately 20 times that of the SDFCNN model.

D. EVALUATION AND COMPARISON OF SDF-SLAM

On the two datasets TUM (f1_desk2) and TUM (f2_pioneer_360), we use the TUM toolbox EVO to estimate the absolute error ATE of the camera pose obtained by the model in this paper. ATE stands for absolute trajectory error and is used to intuitively respond to the direct error between prediction and reality. ORB-SLAM [2], LSD-SLAM [3] and CNN-SLAM [21] are from relevant papers, and the results are shown in Table 10. We can conclude that our SDF-SLAM based on the deep learning framework designed in this paper reduces the ATE errors from 1.826 m, 1.206 m, and 0.264 m to 0.124 m on the two datasets compared with LSD-SLAM, ORB-SLAM, and CNN-SLAM, respectively, especially on

TABLE 10. Absolute error of camera pose estimation.

Dataset	ATE (m)			
	LSD [3]	ORB [2]	CNN [21]	SDF-SLAM
TUM_f1_desk2	0.436	0.495	0.243	0.037
TUM_f2_pionner_360	1.826	1.206	0.264	0.124

TABLE 11. Performance analysis of point cloud data.

Dataset	wall	window	floor	MPA	Position Accuracy
TUM_f1_desk2	0.942	0.911	n/a	0.641	0.921
TUM_f2_pionner	0.622	n/a	0.972	0.715	0.882

the datasets with few feature textures. Furthermore, the accuracy of camera pose estimation obtained by the feature point method ORB and FPFDCNN is higher than that obtained the direct LSD and CNN methods.

The performance analysis of all point cloud data from the two sets of datasets is shown in Table 11. We select three categories, the wall, window and floor categories, and calculate the class accuracy (CA) of SDF-SLAM and the average accuracy of SDF-SLAM semantic label MPA. It can be seen that the proposed SLAM has a higher accuracy in the semantic segmentation of the five selected categories. On the dataset with more feature textures, the accuracy of the point cloud coordinates reaches 92%, and on the dataset with fewer feature textures, the accuracy of the point cloud coordinates reaches 88%. The average accuracy of the point cloud coordinates on the two sets of data reaches 90%, and the average accuracy rate of the semantic tags reaches 67%.

VI. CONCLUSION

In this paper, we presented SDF-SLAM, a highly accurate SLAM method based on deep learning for indoor camera pose estimation and three-dimensional semantic map reconstruction. The camera pose obtained by SDF-SLAM, which extracts feature points through a CNN, is more accurate than that obtained by ORB-SLAM, which extracts feature points through ORB method. Our system uses a CNN to reconstruct a semantic image and a deep image at the same time. Compared with a single semantic network or deep network, the inference time and network parameters are reduced and the accuracy remains the same.

APPENDIX A
THE STRUCTURE OF FPFDCNN

As shown in Table 12, this appendix describes the structure of FPFDCNN model in detail, including the upper layer model (Pre_Layer), weight, bias, padding, dilation, stride, activation

TABLE 12. The structure of FPFDCNN.

Model	Layer	Pre_Layer	Weight	Bias	Padding	Dilation	Stride	Activate	Feature map	Method
Input	Image	--	--	--	--	--	--	--	640×480×1	--
Encoder	Layer 1	Image	4×4×1×32	32	[1,1]	[1,1]	[2,2]	Elu[1]	240×320×32	--
	Layer 2	Layer 1	4×4×32×64	64	[1,1]	[1,1]	[2,2]	Elu	120×160×64	--
	Layer 3	Layer 2	3×3×64×128	128	[1,1]	[1,1]	[1,1]	Elu	120×160×128	--
	Layer 4	Layer 3	4×4×128×128	128	[1,1]	[1,1]	[2,2]	Elu	60×80×128	--
	Layer 5	Layer 4	3×3×128×256	256	[1,1]	[1,1]	[1,1]	Elu	60×80×256	--
	Layer 6	Layer 5	4×4×256×256	256	[1,1]	[1,1]	[2,2]	Elu	30×40×256	--
Decoder	Layer 7	Layer 6	3×3×256×256	256	[1,1]	[1,1]	[1.1]	Elu	30×40×256	
	Layer 8	Layer 7	1×1×256×256	256	[0,0,]	[1,1]	[1,1]	sigmoid	30×40×256	
	FP	Layer 8	--	--	--	--	--	--	640×480×1	Pixel shuffle
	Layer 9	Layer 6	3×3×256×256	256	[1,1]	[1,1]	[1,1]	Elu	30×40×256	
	Layer 10	Layer 9	1×1×256×256	256	[0,0]	[1,1]	[1,1]	--	30×40×256	
	FD	Layer 10	--	--	--	--	--	--	640×480×256	Bilinear
Output	FP_nonzero	FP	--	--	--	--	--	--	N×2	Nonzero
	FD_FP	FD	--	--	--	--	--	--	N×256	Corresponding with FP
Concatenate	Out	FP_nonzero, FP_FD	--	--	--	--	--	--	N×(256+2)	FP Concatenate FD

[1]$\text{Elu} = \begin{cases} a(e^x - 1) & x < 0 \\ x & x > 0 \end{cases}$

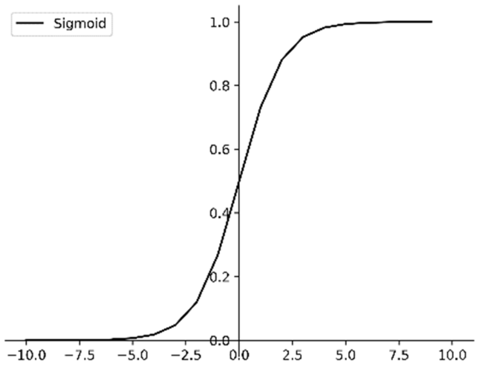

FIGURE 12. The sigmoid function.

function (Activate), input feature image (Feature Map) and the method taken by this layer (Method).

Layer 1 to Layer 6 use convolution kernels of 3 × 3 and 4 × 4 to extract the feature image for image down-sampling. Layer 7 and Layer 8 judge whether the feature map is a feature point, and the output of feature point detection is a dichotomic problem. As shown in Fig. 12, the sigmoid function can map each output value between 0 and 1 to represent the probability of being a feature point, so as to judge whether it is a feature point. In the FP layer, the pixel shuffle algorithm is adopted to perform super-pixel restoration on the low-resolution image and finally generate the high-resolution image. Layer 9 and Layer 10 represent descriptors generated for feature points. Considering that a 256-dimensional vector needs to be obtained for each pixel as a feature point descriptor, we set the number of channels of the last layer of the feature image to 256. The FD layer uses the bilinear algorithm to upsample and restore the feature image. The FP_nonzero layer selects pixel points with non-zero probability as feature points of the input 640 × 480 × 1 feature images, and obtains N feature points, each of which contains two-dimensional coordinate information (x,y) to generate an N × 2 matrix. The FD_FP layer extracts the N × 2 feature point matrix from FP_nonzero layer and the 640 × 480 × 256 feature point descriptors from FD layer according to the coordinate position one by one, and finally generates the N × 256 matrix, denoted as FD. Finally, the Out layer combines the matrix obtained from FP_nonzero layer and FP_FD layer with the abscissa dimension to obtain N×(2+256) as the output.

APPENDIX B THE STRUCTURE OF SDFCNN

As shown in Table 13, this appendix describes the structure of SDFCNN model and the parameters of each layer in detail, including the upper layer (Pre_Layer), the following layer (Post_Layer), the method taken by this layer (Method), weight, bias, padding, stride, group and input feature image (Feature map).

The Layer 0 uses bilinear interpolation to reduce resolution of feature images, parameters and shorten the calculation time.

The encoder layers consist of seven layers, including Layer 1 to Layer 7, which add three convbnrelu modules to a single block module. As the network layer deepens, the number of blocks superimposed on each layer increases. Layer 8 to Layer 11 constitute the decoder layers of SDFCNN. Layers seg and seg1 serve as semantic segmentation of the judgment layer, while layers depth and depth1 serve as depth estimation of the judgment layer. The output layer restores the semantic image and depth image predicted by the judgment layer to an image with the same resolution of 640 × 480 as the original image through the method of bilinear interpolation, which facilitates the generation of dense point cloud image.

TABLE 13. The structure of SDFCNN.

Model	Layer	Pre_Layer	Post_Layer	Method	Weight/Bias	Padding/Stride/Group	Feature map
Input	Layer0	image	Layer0	Bilinear	--	--	400×300×3
	Layer1	Layer0	Layer1	convbnrelu	3×3×3×32/--	1/2/1	200×150×32
	Layer2_1	Layer1		convbnrelu	1×1×32×32/--	0/1/1	200×150×32
	Layer2_2	Layer2_1	Layer2= Layer2_3	convbnrelu	3×3×1×32/--	1/1/32	200×150×32
	Layer2_3	Layer2_2		convbnrelu	1×1×32×16/--	0/1/1	200×150×16
	Layer3_1	Layer2		convbnrelu	1×1×16×96/--	0/1/1	200×150×96
	Layer3_2	Layer3_1	Layer3=	convbnrelu	3×3×1×96/--	1/2/96	100×75×96
	Layer3_3	Layer3_2	Layer3_2+	convbnrelu	1×1×96×24/--	0/1/1	100×75×24
	Layer3_4	Layer3_3	Layer3_6	convbnrelu	1×1×24×144/--	0/1/1	100×75×144
	Layer3_5	Layer3_4		convbnrelu	3×3×1×144/--	1/1/144	100×75×144
	Layer3_6	Layer3_5		convbnrelu	1×1×144×24/--	0/1/1	100×75×24
	Layer4_1	Layer3		convbnrelu	1×1×24×144/--	0/1/1	100×75×144
	Layer4_2	Layer4_1		convbnrelu	3×3×1×144/--	1/2/144	50×38×144
	Layer4_3	Layer4_2	Layer4=	convbnrelu	1×1×144×32/--	0/1/1	50×38×32
	Layer4_4	Layer4_3	Layer4_3+	convbnrelu	1×1×32×192/--	0/1/1	50×38×192
	Layer4_5	Layer4_4	Layer4_6+	convbnrelu	3×3×1×192/--	1/1/192	50×38×192
	Layer4_6	Layer4_5	Layer4_9	convbnrelu	1×1×192×32/--	0/1/1	50×38×32
	Layer4_7	Layer4_6+ Layer4_3		convbnrelu	1×1×32×192/--	0/1/1	50×38×192
	Layer4_8	Layer4_7		convbnrelu	3×3×1×192/--	1/1/192	50×38×192
	Layer4_9	Layer4_8		convbnrelu	1×1×192×32/--	0/1/1	50×38×32
	Layer5_1	Layer4		convbnrelu	1×1×32×192/--	0/1/1	50×38×192
	Layer5_2	Layer5_1		convbnrelu	3×3×1×192/--	1/2/192	25×19×192
Encoder	Layer5_3	Layer5_2	Layer5=	convbnrelu	1×1×192×64/--	0/1/1	25×19×64
	Layer5_4	Layer5_3	Layer5_12+	convbnrelu	1×1×64×384/--	0/1/1	25×19×384
	Layer5_5	Layer5_4	Layer5_9+	convbnrelu	3×3×1×384/--	1/1/384	25×19×384
	Layer5_6	Layer5_5	Layer5_6+	convbnrelu	1×1×384×64/--	0/1/1	25×19×64
	Layer5_7	Layer5_6+ Layer5_3	Layer5_3	convbnrelu	1×1×64×384/--	0/1/1	25×19×384
	Layer5_8	Layer5_7		convbnrelu	3×3×1×384/--	1/1/384	25×19×384
	Layer5_9	Layer5_8		convbnrelu	1×1×64×384/--	0/1/1	25×19×64
	Layer5_10	Layer5_9+ Layer5_6+ Layer5_3		convbnrelu	1×1×64×384/--	0/1/1	25×19×384
	Layer5_11	Layer5_10		convbnrelu	3×3×1×384/--	1/1/384	25×19×384
	Layer5_12	Layer5_11		convbnrelu	1×1×384×64/--	0/1/1	25×19×64
	Layer6_1	Layer5		convbnrelu	1×1×64×384/--	0/1/1	25×19×384
	Layer6_2	Layer6_1		convbnrelu	3×3×1×384/--	1/1/384	25×19×384
	Layer6_3	Layer6_2	Layer6=	convbnrelu	1×1×384×96/--	0/1/1	25×19×96
	Layer6_4	Layer6_3	Layer6_9+	convbnrelu	1×1×96×576/--	0/1/1	25×19×576
	Layer6_5	Layer6_4	Layer6_6+	convbnrelu	3×3×1×576/--	1/1/576	25×19×576
	Layer6_6	Layer6_5	Layer6_3	convbnrelu	1×1×576×96/--	0/1/1	25×19×96
	Layer6_7	Layer6_6+ Layer6_3		convbnrelu	1×1×96×576/--	0/1/1	25×19×576
	Layer6_8	Layer6_7		convbnrelu	3×3×1×576/--	1/1/576	25×19×576
	Layer6_9	Layer6_8		convbnrelu	1×1×576×96/--	0/1/1	25×19×96
	Layer7_1	Layer6		convbnrelu	1×1×96×576/--	0/1/1	25×19×576
	Layer7_2	Layer7_1		convbnrelu	3×3×1×576/--	1/2/576	10×13×576
	Layer7_3	Layer7_2		convbnrelu	1×1×576×160/--	0/1/1	10×13×160
	Layer7_4	Layer7_3	Layer7=	convbnrelu	1×1×160×960/--	0/1/1	10×13×960
	Layer7_5	Layer7_4	Layer7_12+	convbnrelu	3×3×1×960/--	1/1/960	10×13×960
	Layer7_6	Layer7_5	Layer7_9+	convbnrelu	1×1×960×160/--	0/1/1	10×13×160
	Layer7_7	Layer7_6+ Layer7_3	Layer7_6+	convbnrelu	1×1×160×960/--	0/1/1	10×13×960
	Layer7_8	Layer7_7	Layer7_3	convbnrelu	3×3×1×960/--	1/1/960	10×13×960
	Layer7_9	Layer7_8		convbnrelu	1×1×960×160/--	0/1/1	10×13×160
	Layer7_10	Layer7_9+ Layer7_6+ Layer7_3		convbnrelu	1×1×160×960/--	0/1/1	10×13×960
	Layer7_11	Layer7_10		convbnrelu	3×3×1×960/--	1/1/960	10×13×960
	Layer7_12	Layer7_11		convbnrelu	1×1×960×320/--	0/1/1	10×13×320
	Layer7_13	Layer7_10		conv	1×1×320×256/--	0/1/1	10×13×256
	Layer7_14	Layer7_13		conv	1×1×160×256/--	0/1/1	10×13×256
	Layer7_15	Layer7_14+ Layer7_13		Relu	--	--	10×13×256
	Layer7_16	Layer7_15		max_pool	5×5	2/1/--	10×13×256
	Layer7_17	Layer7_16		conv	1×1×256×256/--	0/1/1	10×13×256

TABLE 13. ***(Continued.)*** **The structure of SDFCNN.**

	Layer7_18	Layer7_17		max_pool	5×5	2/1/--	10×13×256
	Layer7_19	Layer7_18	Layer8= Layer7_25	conv	1×1×256×256/--	0/1/1	10×13×256
	Layer7_20	Layer7_19		max_pool	5×5	2/1/--	10×13×256
	Layer7_21	Layer7_20		conv	1×1×256×256/--	0/1/1	10×13×256
	Layer7_22	Layer7_21		max_pool	5×5	2/1/--	10×13×256
	Layer7_23	Layer7_22		conv	1×1×256×256/--	0/1/1	10×13×256
	Layer7_24	Layer7_23+Layer7_20+ Layer7_18+ Layer7_16		conv	1×1×256×256/--	0/1/1	10×13×256
	Layer7_25	Layer7_24		bilinear	--	--	25×19×256
	Layer6_10	Layer6	Layer9= Layer8+ Layer6_10+ Layer5_13	conv	1×1×96×256/--	0/1/1	25×19×256
	Layer5_13	Layer5		conv	1×1×64×256/--	0/1/1	25×19×256
Decoder	Layer9_1	Layer9		max_pool	5×5	2/1/--	25×19×256
	Layer9_2	Layer9_2		conv	1×1×256×256/--	0/1/1	25×19×256
	Layer9_3	Layer9_3		max_pool	5×5	2/1/--	25×19×256
	Layer9_4	Layer9_4		conv	1×1×256×256/--	0/1/1	25×19×256
	Layer9_5	Layer9_5	Layer10= Layer4_10+ Layer9_10	max_pool	5×5	2/1/--	25×19×256
	Layer9_6	Layer9_6		conv	1×1×256×256/--	0/1/1	25×19×256
	Layer9_7	Layer9_7		max_pool	5×5	2/1/--	25×19×256
	Layer9_8	Layer9_8		conv	1×1×256×256/--	0/1/1	25×19×256
	Layer9_9	Layer9_8+ Layer9_6+ Layer9_4+ Layer9_2		conv	1×1×256×256/--	0/1/1	25×19×256
	Layer9_10	Layer9_9		bilinear	--	--	50×38×256
	Layer4_10	Layer4		conv	1×1×64×256/--	0/1/1	50×38×256
	Layer10_1	Layer10		conv	1×1×256×256/--	0/1/1	50×38×256
	Layer10_2	Layer10_2		conv	1×1×256×256/--	0/1/1	50×38×256
	Layer10_3	Layer10_3	Layer11= Layer3_7+ Layer10_6	conv	1×1×256×256/--	0/1/1	50×38×256
	Layer10_4	Layer10_4		conv	1×1×256×256/--	0/1/1	50×38×256
	Layer10_5	Layer10_1+ Layer10_2+ Layer10_3+ Layer10_4		conv	1×1×256×256/--	0/1/1	50×38×256
	Layer10_6	Layer10_5		bilinear	--	--	100×75×256
	Layer3_7	Layer3		conv	1×1×32×256/--	0/1/1	100×75×256
Judge	seg	Layer11	seg	conv+relu	1×1×1×256	0/1/256	100×75×256
	seg1	seg	seg1	conv	1×1×256×40/256	0/1/1	100×75×40
	depth	Layer11	depth	conv+relu	1×1×1×256/--	0/1/256	100×75×256
	depth1	depth	depth1	conv	3×3×256×1/1	1/1/1	100×75×1
Output	out_seg	seg1	out_seg	bilinear	--	--	640×480×40
	out_depth	depth1	out_depth	bilinear	--	--	640×480×1

REFERENCES

[1] H. Zhou and T. Zhang, "The combination of SfM and monocular SLAM," in *Proc. 26th Chin. Control Decis. Conf. (CCDC)*, May 2014, pp. 5282–5286.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[3] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep. 2014, pp. 834–849.

[4] L. Cui and C. Ma, "SDF-SLAM: Semantic depth filter SLAM for dynamic environments," *IEEE Access*, vol. 8, pp. 95301–95311, 2020.

[5] C. Deng, K. Qiu, R. Xiong, and C. Zhou, "Comparative study of deep learning based features in SLAM," in *Proc. 4th Asia–Pacific Conf. Intell. Robot Syst. (ACIRS)*, Jul. 2019, pp. 250–254.

[6] Z. Javed and G.-W. Kim, "A comparative study of recent real time semantic segmentation algorithms for visual semantic SLAM," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2020, pp. 474–476.

[7] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Proc. Int. Conf. 3D Vis.*, Oct. 2016, pp. 239–248.

[8] ICCV Workshop. (2015). *The Future of Real-Time Slam.* [Online]. Available: http://www.computervisionblog.com/2016/01/why-slam-matters-future-of-real-time.html

[9] C. Meng, B. Guo, and X. Liu, "Simultaneous localization and mapping using monocular direct method," in *Proc. 14th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2016, pp. 1–5.

[10] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 225–234.

[11] R. Roberts, H. Nguyen, N. Krishnamurthi, and T. Balch, "Memory-based learning for visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 47–52.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Jan. 2004.

[13] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Berlin, Germany, May 2006, pp. 404–417.

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[15] A. Jakubović and J. Velagić, "Image feature matching and object detection using brute-force matchers," in *Proc. Int. Symp. ELMAR*, Zadar, Croatia, 2018, pp. 83–86.

[16] F. K. Noble, "Comparison of OpenCV's feature detectors and feature matchers," in *Proc. 23rd Int. Conf. Mechatronics Mach. Vis. Pract. (M2VIP)*, Nov. 2016, pp. 1–6.

[17] X. Wenli and Z. Lihua, "Pose estimation problem in computer vision," in *Proc. TENCON IEEE Region Int. Conf. Comput., Commun. Autom.*, Apr. 1993, pp. 1138–1141.
[18] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DOF camera relocalization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2938–2946.
[19] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2758–2766.
[20] V. Mohanty, S. Agrawal, S. Datta, A. Ghosh, V. D. Sharma, and D. Chakravarty, "DeepVO: A deep learning approach for monocular visual odometry," 2016, *arXiv:1611.06069*.
[21] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6565–6574.
[22] R. Li, S. Wang, and D. Gu, "DeepSLAM: A robust monocular SLAM system with unsupervised deep learning," *IEEE Trans. Ind. Electron.*, vol. 68, no. 4, pp. 3577–3587, Apr. 2021.
[23] R. Azzam, T. Taha, S. Huang, and Y. Zweiri, "A deep learning framework for robust semantic SLAM," in *Proc. Adv. Sci. Eng. Technol. Int. Conf. (ASET)*, Feb. 2020, pp. 1–7.
[24] X. Cheng, P. Wang, and R. Yang, "Learning depth with convolutional spatial propagation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2361–2379, Oct. 2020.
[25] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart, "Kinect V2 for mobile robot navigation: Evaluation and modeling," in *Proc. Int. Conf. Adv. Robot. (ICAR)*, Jul. 2015, pp. 388–394.
[26] Y. Endo, K. Sato, A. Yamashita, and K. Matsubayashi, "Indoor positioning and obstacle detection for visually impaired navigation system based on LSD-SLAM," in *Proc. Int. Conf. Biometrics Kansei Eng. (ICBAKE)*, Sep. 2017, pp. 158–162.
[27] S. Wang, H. Zhao, and X. Hao, "Design of an intelligent housekeeping robot based on IoT," in *Proc. Int. Conf. Intell. Informat. Biomed. Sci. (ICIIBMS)*, Nov. 2015, pp. 197–200.
[28] N. Pinkam, F. Bonnet, and N. Y. Chong, "Robot collaboration in warehouse," in *Proc. 16th Int. Conf. Control, Autom. Syst. (ICCAS)*, Oct. 2016, pp. 269–272.
[29] *KITTI Dataset*. [Online]. Available: http://www.cvlibs.net/datasets/kitti/raw_data.php
[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
[31] *Intel, RealSense Depth Camera D435i*. [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/190004/intel-realsense-depth-camera-d435i.html
[32] *Stereolabs, ZED 2 Camera*. [Online]. Available: https://www.stereolabs.com/zed-2/
[33] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.
[34] *FP Dataset*. [Online]. Available: https://github.com/18635729146/FP_DataSet.git
[35] *TUM Dataset*. [Online]. Available: https://vision.in.tum.de/data/datasets/rgbd-dataset
[36] *NYU Dataset*. [Online]. Available: https://cs.nyu.edu/ silberman/datasets/nyu_depth_v2.html
[37] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

CHEN YANG (Member, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2004, and the M.S. and Ph.D. degrees from the Institute of Microelectronics, Tsinghua University, in 2007 and 2016, respectively. He was with VIA Technologies, Inc., Beijing, from 2007 to 2009. He is currently an Associate Professor with the School of Microelectronics, Xi'an Jiaotong University. His current research interests include deep learning, neural network accelerators, reconfigurable computing, and VLSI SoC design.

QI CHEN received the B.S. degree from the Software College of Northeastern University, Shenyang, China, in 2017, and the M.S. degree from the School of Software Engineering, Xi'an Jiaotong University, Xi'an, China, in 2020. Her current research interests include SLAM and artificial neural networks.

YAOYAO YANG received the B.S. degree in software engineering from the Taiyuan University of Technology, Taiyuan, China, in 2015. She is currently pursuing the M.S. degree with the School of Software Engineering, Xi'an Jiaotong University. Her current research interests include deep learning compiler design and convolutional tensor optimization.

JINGYU ZHANG received the B.S. degree from the College of Information Engineering, Shanghai Maritime University, Shanghai, China, in 2015, and the M.S. degree from the School of Software Engineering, Xi'an Jiaotong University, Xi'an, China, in 2020. Her current research interests include on-chip memory management and artificial neural networks.

MINSHUN WU (Member, IEEE) received the B.S. degree in electronics engineering from Hunan Normal University, Changsha, China, in 2001, the M.E. degree in electrical engineering from Hunan University, Changsha, in 2004, and the Ph.D. degree in microelectronics from Xi'an Jiaotong University, Xi'an, China, in 2012. Then, he joined the Faculty of Microelectronics, Xi'an Jiaotong University, where he is currently an Associate Professor. His current research interests include analog and mixed-signal integrated circuit design and testing, with particular emphasis on cost-effective strategies for enhancing performance and reliability while relaxing instrumentation.

KUIZHI MEI (Member, IEEE) received the B.E. and M.S. degrees in electronics engineering and the Ph.D. degree in pattern recognition and intelligence systems from Xi'an Jiaotong University, Xi'an, China, in 1999, 2002, and 2006, respectively. He is currently a Professor with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University. His research interests include image and video process and VLSI and multiprocessor SoC design.