



StairNetV3: depth-aware stair modeling using deep learning

Chen Wang¹ · Zhongcai Pei¹ · Shuang Qiu¹ · Yachun Wang¹ · Zhiyong Tang¹

Accepted: 4 January 2024

© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2024

Abstract

Vision-based stair modeling can help autonomous mobile robots deal with the challenge of climbing stairs, especially in unfamiliar environments. To address the problem that current monocular methods are difficult to model stairs accurately without depth information in scenes with fuzzy visual cues, this paper proposes a depth-aware stair modeling method for monocular vision. Specifically, we take the prediction of depth images and the extraction of stair geometric features as joint tasks in a convolutional neural network, with the designed information propagation architecture, we can achieve effective supervision for stair geometric feature learning by depth features. In addition, to complete the stair modeling, we take the convex lines, concave lines, tread surfaces and riser surfaces as stair geometric features and apply Gaussian kernels to enable StairNetV3 to predict contextual information within the stair lines. Combined with the depth information obtained by depth sensors, we propose a point cloud reconstruction method that can quickly segment point clouds of stair step surfaces. The experiments show that the proposed method has a significant improvement over the previous best monocular vision method, with an intersection over union increase of 3.4%, and the lightweight version has a fast detection speed and can meet the requirements of most real-time applications.

Keywords Stair modeling · Depth-awareness · Point cloud reconstruction · Deep learning

1 Introduction

In recent years, with the development of autonomous humanoid robots, the perception and feature representation of the surrounding environments, especially various obstacles, have gradually become a research hotspot. In urban environments, stairs are common indoor and outdoor build-

ing structures which are difficult for robots to pass through. Because stairs are artificially constructed building structures with obvious geometric features, most stair detection methods rely on the extraction of some stair geometric features. For example, line-based extraction methods [1–3] abstract stair geometric features as a set of lines continuously distributed in an image, and potential stair lines are extracted in RGB or depth images through Canny edge detection [4], Sobel filtering, Hough transform [5], etc. Plane-based extraction methods [6–8] abstract stair geometric features as a set of planes continuously distributed in space, and potential stair surfaces in point cloud data are extracted through algorithms such as random sample consensus (RANSAC) [9] and super-voxel clustering [10].

The works mentioned above extract stair features using manually designed feature extraction rules, and these methods may fail to some extent in complex scenes. For the line extraction methods, there have been several deep learning based works for semantic line detection task [11, 12], wire-frame parsing task [13–16], and lane line detection task [17, 18]. For example, Reference [12] translates the line detection task from the image domain to the parametric domain in CNN. Reference [13] proposes an end-to-end architec-

Zhongcai Pei, Shuang Qiu and Yachun Wang have contributed equally to this work.

✉ Zhiyong Tang
zyt_76@buaa.edu.cn

Chen Wang
venus@buaa.edu.cn

Zhongcai Pei
peizc@buaa.edu.cn

Shuang Qiu
zb2003108@buaa.edu.cn

Yachun Wang
yachun@buaa.edu.cn

¹ School of Automation Science and Electrical Engineering, Beihang University, Xueyuan Road, Beijing 100191, Beijing, China

ture to output junctions and lines for wireframe parsing. Reference [18] proposes a CNN with asymmetric kernel convolution based on lane line detection and departure estimation features to deal with complex traffic scenes. To deal with complex and challenging scenes, there have been also some machine learning techniques applied to stair detection task. For visually impaired people, it is only necessary to determine if there are stairs around. Support vector machines (SVM) [19] can be used to determine if there are stairs in an image [20, 21]. For autonomous robots, the stair modeling is needed to pass through stairs successfully. Some works first apply object detection algorithms to locate the region of interest (ROI) which contains stairs, for example, you only look once (YOLO) [22], and then extract stair features within the ROI through traditional image processing algorithms [23]. The real-time performance of these methods with two steps is often poor. To achieve end-to-end stair feature extraction, StairNet [24] proposes a novel stair feature representation method which is conducive to neural network learning, and uses a simple fully convolutional network to extract the stair line features. StairNet does not need manually designed rules and shows strong robustness and adaptability to complex scenes, and end-to-end detection has good real-time performance. To improve the performance of StairNet in scenes with extremely fuzzy visual clues, StairNetV2 [25] adds depth input to the network to explore the complementary relationship between RGB and depth images. However, StairNetV2 relies on dense depth images generated by binocular vision. Without binocular sensors, StairNetV2 will not be applicable and its applicability is inferior to monocular vision methods.

To solve the above problems, we propose depth-aware StairNetV3 to learn depth features through depth supervision under the monocular vision, which can retain the wide applicability of StairNet while achieve detection performance close to StairNetV2. In addition, we improve the stair feature representation method for the StairNet series by adding Gaussian kernels, making the network can perceive stair line endpoints. To achieve complete modeling of stair building structures, we add a semantic segmentation branch to extract stair step surfaces while extracting stair lines. Combined with point clouds from depth sensors, stair point cloud segmentation can be achieved at a very fast speed. The overall flow of our method is shown in Fig. 1.

The remainder of this paper is organized as follows: Sect. 2 summarizes the related works on stair detection, Sect. 3 discusses the details of the methods, including the improved stair feature representation method, the network architecture, the stair point cloud reconstruction method and the loss function. Section 4 discusses the experiments conducted, including dataset and metrics, implementation details, performance experiments, comparison experiments and ablation experiments. Section 5 discusses the StairNet series and some

limitations, and looks forward to future work. Section 6 summarizes the whole paper.

2 Related works

Stairs are a common architectural structure, with different materials, structures, and design styles. Stair detection in different environments is a challenging task. Some works are dedicated to detecting stairs in monocular vision. For example, reference [26] applies Gabor filters on grayscale RGB images to extract stair edges and judges whether the scene contains stairs through the projection-histogram of edge pixel values in the horizontal and vertical directions. Reference [27] regards a stair line as a periodic signal in the spatial domain. The input RGB image is processed through Canny edge detection to obtain a binary image, which is then processed through 2D fast Fourier transform to the frequency domain, resulting in an image containing almost only stair edges. Reference [28] regards the stair line endpoints as an intersection of three line segments and searches for such points in the edge image after Gabor filtering to extract potential stair lines. With the development of machine learning and deep learning, stair detection has gradually shifted from traditional image processing dominated methods to deep learning dominated methods. For examples, Reference [29] proposes a lightweight Look-Behind fully convolutional network to determine whether there are stairs in an image. Reference [30] applies Adaboost classifier and Haar features to detect stairs in an image, and uses ground plane estimation and temporal consistency validation to reduce the false positives. Reference [31] first applies YOLOv5s [32] to locate the box containing stairs, and then applies U-Net [33] with ResNet-34 [34] backbone to segment stair lines within the box. This research achieves pixel-level stair line location through deep learning methods, but the multi-stage execution of the method leads to the poor real-time performance. Reference [24] achieves end-to-end stair line detection through a fully convolutional neural network by designing a novel stair feature representation method.

In recent years, with the decline in the prices of binocular sensors and lidars, more researches have been devoted to detecting stairs through binocular vision and stereo vision. Most binocular vision detection methods are based on monocular detection methods, using depth features from depth images to remove interfering textures. For example, References [35, 36] first detect stair edges in RGB images through traditional image processing methods, and then extract corresponding one-dimensional depth features in depth images to distinguish between stairs and pedestrian crosswalks. Reference [37] applies Sobel filtering and Hough transform to detect potential lines, and then extracts a 480-dimensional feature vector in the depth image and feeds the

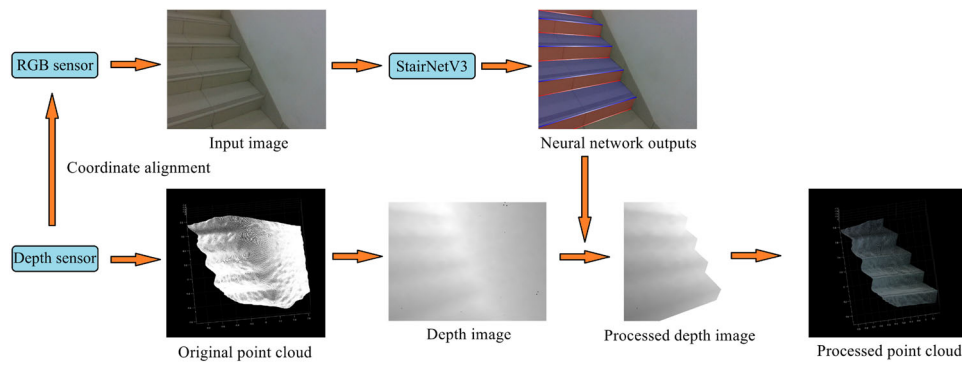


Fig. 1 The overall flow of our method. After the RGB image captured by the monocular camera is processed by StairNetV3, the stair modeling results are obtained, including convex lines (blue lines in the figure), concave lines (red lines in the figure), tread surfaces (blue masks in the figure), riser surfaces (red masks in the figure) and stair line endpoints.

vector into SVM to determine whether the scene is ascending stairs, descending stairs or other situations. These methods combine multi-modal features through manually designed rules, and their adaptability in complex scenes is difficult to guarantee. To explore the potential complementary relationship between RGB and depth images, StairNetV2 [25] adds a selective module to the network to obtain effective multi-modal combination automatically through learning, greatly improving the detection reliability in complex scenes, especially at night. For stereo vision-based methods, the aim is to extract the point clouds of stair step surfaces in three-dimensional space. For example, Reference [38] first uses RANSAC to search for planes, then uses principal component analysis (PCA) to estimate the normal and surface curvature of the planes and connect adjacent planes. After distinguishing obstacles in the scene based on the proportion of inliers, the riser surfaces and tread surfaces are determined by the angle between the candidate plane normals and the ground plane normal. Reference [39] uses region growing clustering to cluster point clouds obtained from depth cameras, effectively distinguishing stair step surfaces and walls, and avoiding interference from obstacles through histograms of the inlier number on planes at different heights. Reference [40] uses PointNet [41] to process point clouds obtained from depth cameras and classifies the scene into ascending stairs, descending stairs or other situations. Reference [42] first uses Depth-Cut-based Ground Detection (DCGD) [43] to detect the ground, then each point cloud segment is fed into PointNet to perform object classification. Stairs have two different semantic labels including stairs leading to upstairs and stairs leading to downstairs, which visually impaired people can understand by touch.

As mentioned above, the monocular methods have the advantages of fast speed and wide applicability, but often fail in some challenge scenes with fuzzy visual cues. While the binocular and stereo methods have the advantages of robust-

ness to stair textures and high precision, but often rely on depth sensors and have poor real-time performance. In this paper, we are committed to combining the advantages of both monocular methods and binocular methods and overcoming the shortcomings.

3 Methods

In this section, we describe the methods in detail, including the improved stair feature representation method, the detailed network architecture, the post-processing algorithms for network outputs, and the loss function.

3.1 Stair feature representation

Different from the approach of only focusing on line extraction in StairNet and StairNetV2, StairNetV3 is dedicated to achieving complete modeling of stair structures. We use a simple CNN to simultaneously conduct line extraction and surface extraction, distinguishing between convex lines/concave lines and riser surfaces/tread surfaces. For stair lines, endpoint information is crucial in the post-processing of network results. We hope that StairNetV3 can directly learn the stair line endpoints, so we introduce Gaussian kernels.

The network input is a full-color RGB image, denoted as $I \in [0, 255]^{H \times W \times 3}$, and its height and width are H and W , respectively. Our goal is to obtain a heatmap that reflects the stair line positions, denoted as $O \in [0, 1]^{\frac{H}{S_H} \times \frac{W}{S_W} \times 1}$, where S_H and S_W are the output strides in the height and width directions, respectively, determining the heatmap size. To express the endpoint information of stair lines, we hope that the response of stair line endpoints in the heatmap is 1, while the response of the rest of the stair line decreases

as the distance from the endpoint increases. Therefore, we apply Gaussian kernels to determine the response value of each pixel in the heatmap, as shown in Eq. 1.

$$Y = \begin{cases} e^{-\frac{(x-p_x)^2+(y-p_y)^2}{2\sigma^2}}, & \text{if } (x, y) \in L \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where (x, y) represents the pixel coordinates in the original image, Y represents the pixel value at the corresponding pixel coordinates $\left(\left\lfloor \frac{x}{S_W} \right\rfloor, \left\lfloor \frac{y}{S_H} \right\rfloor\right)$ in the heatmap, (p_x, p_y) represents the pixel coordinates of the closer endpoint of stair line L to (x, y) in the original image, and σ is the standard deviation and can be adjusted according to demand [44]. According to Eq. 1, it can be seen that when (x, y) is the midpoint of stair line L , Y takes the minimum value. Since the threshold for distinguishing positive and negative samples is usually set to 0.5, we hope that the Y value at the midpoint is greater than 0.5. Therefore, let $\sigma = \frac{1}{2}\|L\|_2 = \frac{1}{2}\sqrt{(p_{x1} - p_{x2})^2 + (p_{y1} - p_{y2})^2}$, where (p_{x1}, p_{y1}) and (p_{x2}, p_{y2}) represent the left and right endpoints of stair line L in the original image, respectively. So when $x = \frac{p_{x1} + p_{x2}}{2}$ and $y = \frac{p_{y1} + p_{y2}}{2}$, $Y_{min} = e^{-\frac{1}{2}} \approx 0.6$. For the prediction of stair step surfaces, we add a branch to the network for generating semantic segmentation masks. The branch has four upsampling operations to obtain tensors that match the input image size, and the segmentation results have three classifications, including riser surface, tread surface, and background. The above process is shown in Fig. 2.

3.2 Network architecture

Different from the network architecture with two input branches in StairNetV2, StairNetV3 only has an input branch for RGB image, and the branch for depth image is set in the outputs. To achieve effective depth-awareness, the network contains two parallel backbones, which are stacked with several squeeze-and-excitation (SE)-ResNeXt [45, 46] blocks with dilated convolution. Both backbones follow an encoder-decoder structure, where one backbone is used for depth-awareness and the other backbone is used for obtaining stair modeling features, and there are two different ways of interactions between the backbones. Through the shared input branch, the stair modeling backbone get the implicit supervision from depth features by the way of gradient back propagation. In the decoder of the depth-awareness backbone, a depth-aware block (DAB) is applied to aggregate depth features from different layers then directly pass it to the encoder of the stair modeling backbone, and the feature fusion is achieved through a selective module [25] to provide direct supervision for the learning of stair modeling. For the details of selective module, first, the RGB and depth feature

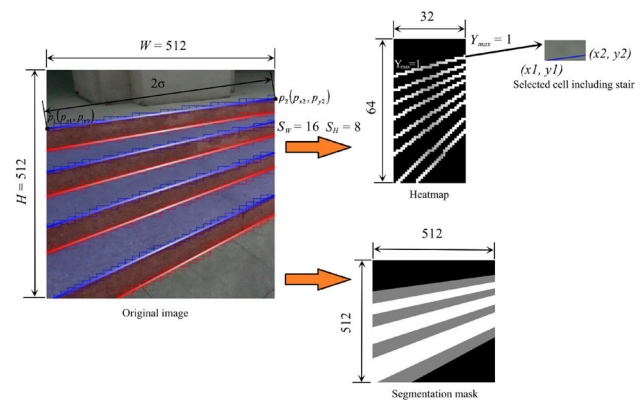


Fig. 2 Illustration of stair feature representation. The line features include convex lines (blue lines in the original image) and concave lines (red lines in the original image), and the surface features include tread surfaces (blue masks in the original image) and riser surfaces (red masks in the original image). To obtain the stair line endpoints, Gaussian kernels are applied to make the two stair line endpoints obtain the maximum response of 1 and the midpoint of stair line obtain the minimum response, approximately 0.6. The color of each stair line in the original image gradually fades from both ends to the middle, reflecting the working mode of the Gaussian kernels. For cells containing stair lines, the normalized coordinates (x_1, y_1) and (x_2, y_2) of the stair line segment relative to the upper left corner of the cell are regressed. The stair step surfaces are predicted through an output segmentation mask with the same size as the original image

maps are added element by element, and a $1 \times 1 \times c$ vector is obtained through global average pooling. After that, the softmax activation function is applied to the vector to obtain two feature descriptors that restrict each other. The original feature maps are multiplied by the feature descriptors element by element and then added to obtain the mixed feature maps. The network architecture is shown in Fig. 3.

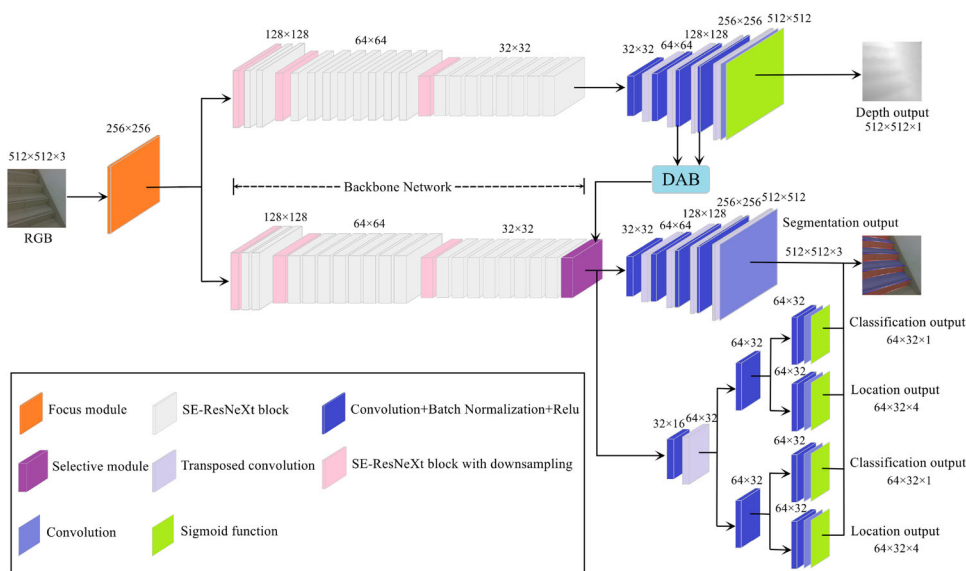
3.2.1 Multi-task learning

The proposed network is a typical multi-task architecture which contains line extraction task (LST), semantic segmentation task (SST) and depth estimation task (DET). For the stair modeling backbone, the LST has four branches, including two heatmap branches with a size of $64 \times 32 \times 1$ and two location branches with a size of $64 \times 32 \times 4$. The heatmap branch outputs the confidence of containing stair lines in the cell, and the location branch outputs the normalized coordinates of the contained stair lines. The SST only has a semantic segmentation branch with a size of $512 \times 512 \times 3$ for classifying stair step surfaces. For the depth-awareness backbone, it only outputs an estimated depth image with a size of $512 \times 512 \times 1$.

3.2.2 Implicit depth supervision

In StairNetV3, the depth-aware backbone and the stair modeling backbone share parameters through the common focus

Fig. 3 Network architecture. The network contains two parallel backbones, and the upper backbone is used to output depth images and the lower backbone is used to output stair modeling results. The depth features from different feature layers are fused and passed to the stair modeling backbone through the DAB, and the feature fusion is achieved through a selective module, enabling direct supervision of the stair modeling results by the depth-aware features



module [32]. The focus module is implemented through a tensor slicing operation and a point convolution, and the feature maps obtained by slicing and stacking are adjusted in channel number and fused across channels through point convolution. Specifically, given an input image $I \in [0, 255]^{H \times W \times 3}$ with height and width of H and W respectively. After the focus module, we can get $f_\theta(I')$, where f_θ represents the weights of point convolution and $I' \in [-1, 1]^{\frac{H}{2} \times \frac{W}{2} \times 12}$, then the entire learning process can be considered as the following optimization problem, as show in Eq. 2.

$$\min_{\theta} N_D(f_\theta(I'); D^*) + \min_{\theta} N_M(f_\theta(I'); M^*) \quad (2)$$

where N_D and N_M represent data fitting terms, D^* represents the ground-truth of depth image and M^* represents the ground-truth of stair modeling, including the classification ground-truth of stair lines, location ground-truth of stair lines and the segmentation ground-truth of stair steps. Through sharing the weights f_θ of point convolution in focus module and gradient back propagation, the point convolution can generate weights that are beneficial for obtaining depth images, and the depth images are enlightening for the stair modeling, so the learning process of stair modeling is optimized in an implicit depth supervision manner.

3.2.3 Depth-aware block

In StairNetV2, the RGB and depth features are fused trough a selective module, and the input of depth images provides direct depth supervision for network learning, and we hope to apply this supervision method in StairNetV3. As the optimization goal of the depth-aware backbone is to regress the depth image aligned to the input RGB image, we take the feature maps obtained after the second and third upsampling

in the decoder of the depth-aware backbone and fuse them through the DAB, then the fused depth feature maps are sent to the selective module to implement direct depth supervision for the learning of stair modeling backbone, further improving the stair modeling performance. The DAB is shown in Fig. 4.

3.3 Post-processing algorithms

3.3.1 Line segments linking method

For line extraction based stair detection algorithms, the outputs are often a set of discrete stair line segments. To obtain the stair line equations, these discrete line segments need to be linked and fitted. The traditional image processing based line extraction method often use edge connection algorithms to connect adjacent line segments [20, 21, 28]. For the structured stair line information output by CNN, StairNetV2 proposes a line clustering method based on least squares [25], which can quickly complete clustering using adjacent information and geometric information between stair line segments. However, repeated least squares calculations still requires some processing time in real-time inference. To make the post-processing of line linking faster, we assign higher-level semantic information to each cell containing stair lines through Gaussian kernels. Each cell can not only obtain whether it contains a stair line, but also obtain its relative distance from the stair endpoint (which can be understood as the confidence of containing a stair line endpoint in the cell). Therefore, the line segments linking (LSL) method can directly obtain the cells containing stair line endpoints which are more critical to the LSL, and the detailed steps are as follows:

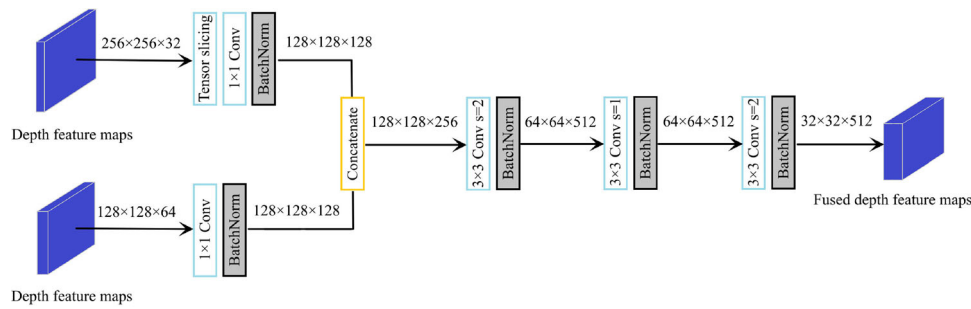


Fig. 4 The structure of DAB. The DAB first fuses the feature maps from the second and third upsampling layers of the decoder in the depth backbone through a 1×1 convolution and tensor slicing operation. The fused feature maps are then processed through three 3×3 convolutions

1. The cells with confidence scores greater than a certain threshold (set to 0.75 in the research) are selected and sorted according to their confidence scores.
2. After the cells with high confidence scores are selected, the top 50 cells with highest confidence scores are selected and the adjacent cells are grouped together.
3. Groups with only one cell are removed.
4. For each groups, the stair lines are fitted using least squares method to get the line equation $y = kx + b$.
5. For each two groups, their line equations are checked. If the equations intersect in the range of $[0, 512]$, or if their left endpoints are close or their right endpoints are close, then the two groups are merged into a new group. After that, the slope-intercept equations of the lines are recalculated.

Overall, LSL method can obtain the stair line equations through information from only 50 cells and 2 rounds of least squares calculation, and its usage time can be ignored, as shown in Fig. 5.

3.3.2 Stair point cloud reconstruction method

After obtaining the stair line equations, to achieve a complete stair modeling, it is necessary to extract the point clouds of the stair step surfaces. Most plane-based methods apply point cloud segmentation algorithms to obtain the point clouds of stair step surfaces. Due to the huge amount of point cloud data, the real-time performance is often poor. To achieve fast stair surface segmentation, we transform the segmentation operation from three-dimensional space to two-dimensional image. Through the segmentation output branch, we directly obtain the stair step surfaces in the images and apply the segmentation results to the depth map. Then, combined with the camera intrinsics, we segment the point clouds of stair step surfaces in a point cloud reconstruction manner. Specifically, given a point cloud data $P \in R^{X,Y,Z}$ from a depth

sensor, the distance $Z \in [dmin, dmax]^{H \times W}$ in the forward direction can be obtained, namely the depth map, where H and W are the height and width of the aligned RGB image. The segmentation output is denoted as $S \in [0, 1]^{H \times W \times 3}$, and the third dimension of S predicts the one-hot encoding (c_1, c_2, c_3) for three classifications, then the maximum value is retained and set to 1 and other values are set to 0. After the above transformation, each element in S can be described as the following Eq. 3.

$$S_{h,w,i}^* = \begin{cases} 1, & \text{if } S_{h,w,i} \in c_i \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $h \in [0, H)$, $w \in [0, W)$, $i \in [0, 3)$, for each classification i , the corresponding 0–1 matrix S_i^* is obtained, and through dot product, the corresponding depth map $Z_i^* = Z * S_i^*$ is obtained. For each element $Z_{h,w,i}^*$ in Z_i^* , the corresponding coordinates in camera coordinate system can be obtained through Eq. 4.

$$P_{h,w,i}^* = \begin{cases} K^{-1} Z_{h,w,i}^* p_{h,w}, & \text{if } Z_{h,w,i}^* \neq 0 \\ (0, 0, 0), & \text{otherwise} \end{cases} \quad (4)$$

where $P_{h,w,i}^*$ are the coordinates in camera coordinate system, K is the camera intrinsic matrix, $p_{h,w}$ are the coordinates in pixel coordinate system, and h , w and i have the same meanings as in Eq. 3. After the above process, the point cloud P_i^* belonging to the classification i can be obtained, as shown in Fig. 6.

3.4 Loss function

StairNetV3 has multiple output branches, which can be divided into branches for classification and branches for location. For the classification task, we use the Binary Cross Entropy (BCE) loss function with a sigmoid function. For the regression task, we use the Mean Squared Error (MSE)

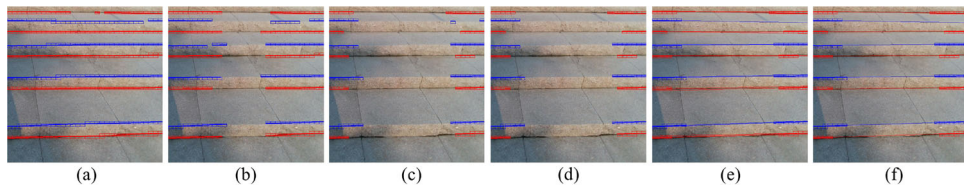
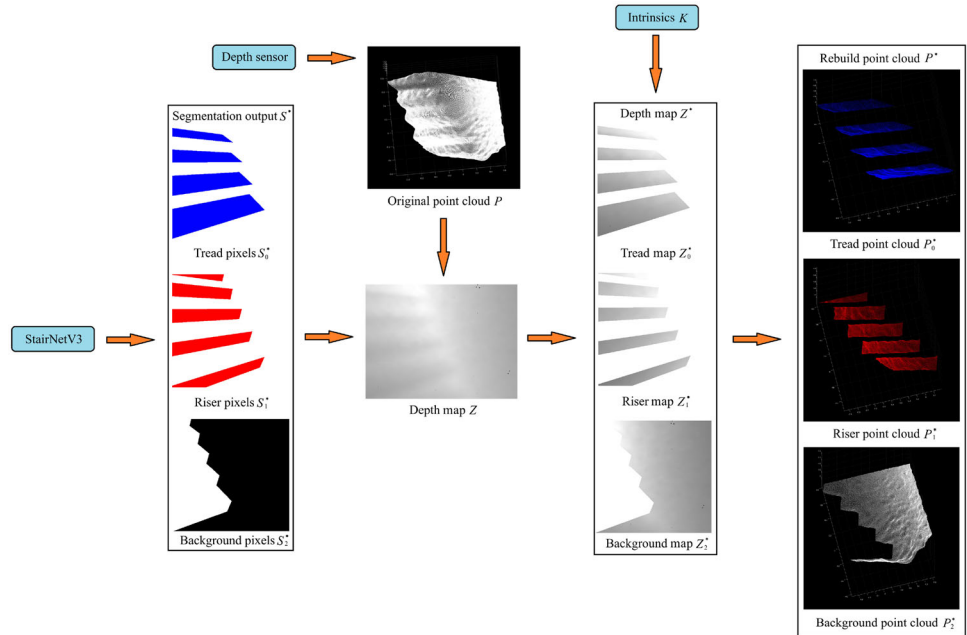


Fig. 5 Illustration of LSL method. **a** shows the original stair line detection results output by StairNetV3; **b** shows the results after filtering with a threshold of 0.75; **c** shows the top 50 cells with the highest confidence scores and the stair lines they contain; **d** shows the results after

filtering isolated cells; **e** shows the initial fitted stair lines through the least squares method; **f** shows the final fitted results after the merging of groups

Fig. 6 Illustration of stair point cloud reconstruction. After obtaining the point cloud data from depth sensor, the depth map Z is generated. After obtaining the semantic segmentation results of StairNetV3, the 0–1 matrix S_i^* can be obtained for each classification. Through dot product, the depth map Z_i^* for each classification is obtained. Finally, combined with the camera intrinsics K , the point cloud P_i^* corresponding to each classification is obtained through point cloud reconstruction



loss function. For the LET, we regard it as a regression problem, and the loss function is described as Eq. 5.

$$L_l = \frac{1}{MN} \sum_i^M \sum_j^N (\alpha_1 h_{ij} L_{MSE}(p_{ij}, p_{ij}^*) + \alpha_2 (1 - h_{ij}) L_{MSE}(p_{ij}, p_{ij}^*) + h_{ij} L_{MSE}(t_{ij}, t_{ij}^*)) \quad (5)$$

where M and N represent the height and width of the output feature maps, namely, 64 and 32, and i and j represent the position of the cell in the feature maps. L_{MSE} represents MSE loss function, p_{ij} represents the confidence predicted by each cell, and p_{ij}^* is the corresponding ground truth. h_{ij} represents whether there is a stair line in a cell and its values are 1 and 0, which represent containing a stair line and without a stair line, respectively. t_{ij} represents the normalized coordinates of endpoints in each cell, and t_{ij}^* is the corresponding ground-truth. For the confidence loss, we give different weights to positive and negative samples to adjust the imbalance between them, and α_1 and α_2 represent the weight coefficients of positive

samples and negative samples, respectively. In our research, $\alpha_1=15$, $\alpha_2=5$.

For location loss, we inherit the loss function with dynamic weights [25] from StairNetV2, as shown in equation 6.

$$L_{MSE}(t_{ij}, t_{ij}^*) = \lambda_1 L_{MSE}(x_{ij}, x_{ij}^*) + \lambda_2 L_{MSE}(y_{ij}, y_{ij}^*) \quad (6)$$

where x_{ij} represents the predicted normalized coordinates in the horizontal direction, and x_{ij}^* is the corresponding ground truth. y_{ij} represents the predicted normalized coordinates in the vertical direction, and y_{ij}^* is the corresponding ground truth. λ_1 and λ_2 represent the corresponding dynamic weights, respectively. In initial epoch, $\lambda_1=\lambda_2=10$, After each epoch, the dynamic weights are adjusted automatically according to a certain evaluation index. The rest parameters have the same meanings as Eq. 5.

For the SST, its essence is a classification task. We apply BCE loss and its loss function is shown in Eq. 7.

$$L_s = \frac{1}{N^2 C} \sum_i^N \sum_j^N \sum_k^C L_{BCE}(m_{ij}, m_{ij}^*) \quad (7)$$

where N represents the width or height of the output feature maps, namely, 512. C represents the number of classifications, which is 3, i and j represent the position of a pixel in the feature maps, and k represents the classification. L_{BCE} represents the BCE loss function. m_{ij} represents the prediction probability of a pixel belonging to a certain classification, and m_{ij}^* is the corresponding ground-truth.

For the DET, it is essentially a regression task, and we still use the MSE loss, and the loss function is shown in Eq. 8.

$$L_d = \psi \frac{1}{N^2} \sum_i^N \sum_j^N L_{MSE}(d_{ij}, d_{ij}^*) \quad (8)$$

where N , i and j have the same meanings as in Eq. 7, L_{MSE} represents the MSE loss function, d_{ij} represents the predicted normalized pixel value, and d_{ij}^* is its corresponding ground-truth. ψ is a weight coefficient, and in our research, $\psi=10$. Overall, the loss function L of StairNetV3 can be represented as Eq. 9.

$$L = L_l^b + L_l^r + L_s + L_d \quad (9)$$

where L_l^b and L_l^r represent the losses of convex lines and concave lines, respectively.

4 Experiments

4.1 Dataset and metrics

Based on the Stair dataset with depth maps [47], we remove the image pairs with totally black RGB images, and finally retain 2276 RGB-D image pairs as the training set and 556 RGB-D image pairs as the validation set. To evaluate the point cloud reconstruction, we use the Realsense D435i depth camera [48] to collect 154 RGB-D image pairs with a resolution of 640×480 as the test set. During the collection, we record the pose of camera and the linear transformation relationship of depth image generation for point cloud reconstruction. To implement the semantic segmentation of stair step surfaces, we make pixel-level segmentation labels for the training set and the validation set, and the classifications include stair riser surface, stair tread surface and background. The organized dataset is named as the RGB-D stair dataset [49].

To accurately evaluate our method, we apply several normal evaluation metrics. For stair lines, we use precision,

recall, and IOU with a confidence level of 0.5 for evaluation. This is consistent with the evaluation methods of StairNetV2. For stair step surfaces, we use pixel accuracy (PA) [50] and mean pixel accuracy (MPA) [50] for evaluation.

4.2 Implementation details

We implement StairNetV3 using PyTorch 1.7.0, and take training and validation on a desktop platform with an i9 13900 CPU and an RTX 3090 GPU. During training, we use Adam optimizer [51] with weight decay of 10^{-6} to optimize the network. The data loader has a batch size of 4. The input images have a size of 512×512 , and the random flip with a 0.5 probability in the horizontal direction is applied for data augmentation.

For the training scheme, we try two kinds of schemes including simultaneous training and progressive training, and Table 1 shows the results. For the first two steps of progressive scheme and the simultaneous scheme, the learning rate is initialized to 2.5×10^{-4} and halves every 50 epochs. For the last step of progressive scheme, the learning rate is initialized to 1.0×10^{-5} to fine tune the model. We train all the steps with 300 epochs and end the training after 80 consecutive epochs without any improvement in accuracy.

Table 1 shows that the simultaneous training scheme has better accuracy than the progressive training scheme. It is worth noting that the model get a certain degree of accuracy after the first step of the progressive scheme, which can prove the implicit supervision effect of depth-aware backbone on the stair modeling backbone.

To verify the role of the DAB and explore its optimal combination with implicit depth supervision, we conduct some experiments of parameter tuning. We take a model without DAB as the baseline, and other models place the DAB after the second downsampling, the third downsampling and the fourth downsampling for comparison. As the DAB is applied together with selective module, we conduct another three comparison experiments with simple summation instead of the selective module to explore more details, and Table 2 shows the results.

Table 2 shows that the position of DAB can impact on network performance. When the DAB is placed after the second downsampling, the model performance is even lower than the model without DAB. When the DAB is placed after the third downsampling and fourth downsampling, the model performance is significantly improved. From the results of feature fusion method with summation, it can be seen that the selective module can not take effect after the second downsampling, which cause the lower performance than model without DAB. In the shallow layers of stair modeling backbone, due to the implicit deep supervision of focus module, the feature maps have more depth features than the deep layers. And the design of selective module makes it more

Table 1 Results of different training schemes

Training scheme	Step	Precision (%)	Recall (%)	IOU (%)	PA (%)	MPA (%)
Progressive	Training of depth-aware backbone	73.29	76.06	59.55	38.88	29.88
	Training of stair modeling backbone	76.82	80.84	64.99	96.01	80.76
	Joint training	76.36	81.50	65.08	96.10	81.11
Simultaneous	Joint training	77.25	81.80	65.92	96.30	81.27

Table 2 The parameter tuning experimental results of DAB

Feature fusion method	The position of DAB	Precision (%)	Recall (%)	IOU (%)	PA (%)	MPA (%)
Without fusion	–	77.00	81.27	65.40	96.03	81.02
Selective module	After the second downsampling	76.99	81.22	65.36	96.09	81.08
	After the third downsampling	77.16	81.86	65.89	96.31	81.25
	After the fourth downsampling	77.25	81.80	65.92	96.30	81.27
Summation	After the second downsampling	76.99	81.65	65.63	96.14	81.10
	After the third downsampling	77.24	81.68	65.83	96.13	81.04
	After the fourth downsampling	76.97	81.66	65.63	96.13	81.08

suitable for the fusion of different modal features, so the DAB with selective module performs better in the deep layers.

4.3 Performance experiments

In this section, we conduct several experiments to test the performance of the models, mainly including the accuracy on the validation set, the accuracy of the post-processing algorithms and the inference speed. To meet various application scenes, we give three versions of StairNetV3, including StairNetV3-large, StairNetV3-medium, and StairNetV3-small, which can be adjusted using a width factor. To prevent the group convolution in the residual block from becoming depthwise separable convolution [52], the width factor not only affects the channel number in each layer of the network, but also the group number in each group convolution to ensure that the number of filters in each group remains unchanged. For the models of StairNetV3, we use the precision, recall, and IOU at a confidence of 0.5, as well as PA and MPA as evaluation indicators. For LSL method, we define true positive (TP) as a linked line which can be matched to a ground-truth stair line, false positive (FP) as a linked line which can not be matched to any ground-truth stair line and false negative (FN) as a ground-truth stair line which can not be matched with any linked line. So the precision, recall and IOU of LSL method can be obtained. For the stair point cloud reconstruction method, since the pixels of the image and the points of the point cloud correspond one-to-one, PA and MPA can be considered as the segmentation precision of the stair point cloud on the validation set. And Table 3 shows the results.

For the network inference speed test, we use a desktop platform described in Sect. 4.2, and a mobile platform with

an Ryzen 7 7735H CPU and an RTX 4060 laptop GPU for test. To reflect the inference speed under different workloads, we test the model inference time in two scenes, which have a batch size of 1 and a batch size of the maximum value corresponding to the GPU memory, respectively. The former corresponds to online image or video stream applications and the latter corresponds to offline image or video stream applications. Table 4 shows the results.

As we can see from Tables 3 and 4, as the model becomes lighter, the model accuracy gradually decreases, the speed gradually increases, and the performance fluctuation of the models is stable. When batch size is 1, the difference in inference time between the three models is not significant, allowing for flexible selection based on accuracy demands, real-time demands, and computational resource allocation. When batch size is the maximum value, there is a significant difference in the model inference speeds. The large and medium models are suitable for tasks that prioritize accuracy, while the small model is suitable for tasks that prioritize speed. Besides, for the post-processing algorithms, the accuracy of LSL is independent of the models and the accuracy of stair point cloud reconstruction changes with the sizes of the models. Some visualization results of StairNetV3-large are shown in Fig. 7, which intuitively demonstrates the adaptability of StairNetV3 to various stairs and shooting conditions. Particularly, even without depth images as input, StairNetV3 can still show stable performance in scenes with fuzzy visual clues, such as night and descending stairs.

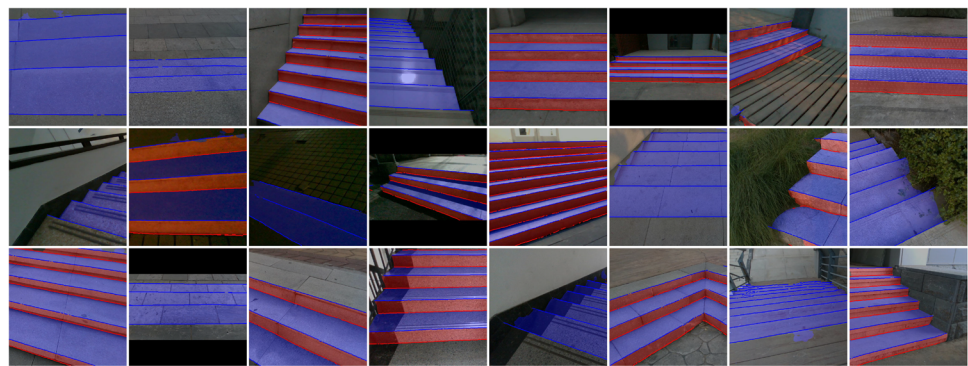
After StairNetV3 obtains the segmentation results of the stair step surfaces, the point cloud of each stair step surface can be obtained through point cloud reconstruction. Because the images of the validation set are padded and

Table 3 The results of StairNetV3 accuracy experiments

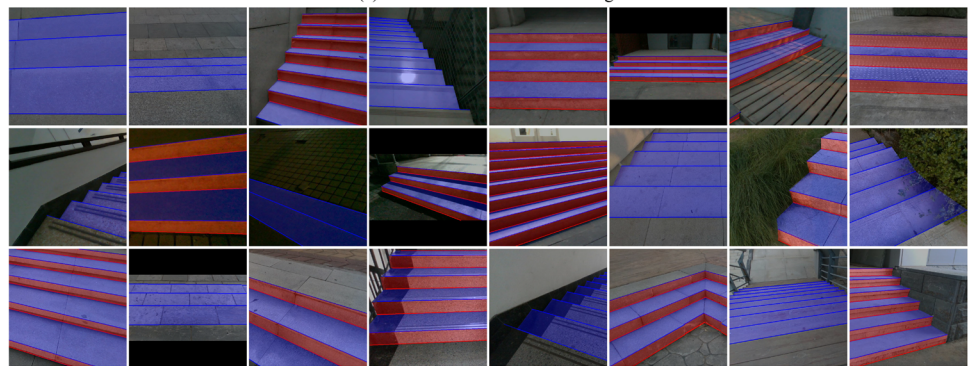
Model	Width factor	Precision (%)	Recall (%)	IOU (%)	Precision (LSL) (%)	Recall (LSL) (%)	IOU (LSL) (%)	PA (%)	MPA (%)
StairNetV3-large	1.0	77.3	81.8	65.9	97.7	73.8	72.5	96.3	81.3
StairNetV3-medium	0.75	75.5	81.8	64.6	97.1	73.3	71.6	96.0	80.9
StairNetV3-small	0.5	74.1	81.0	63.1	98.1	73.8	72.7	95.9	80.7

Table 4 The results of StarNetV3 inference speed experiments

Model	Width-factor	#Params	GFlops	Runtime (ms) Batch size = 1		Runtime (ms) Batch size = maximum	
				Desktop-platform	Mobile-platform	Desktop-platform	Mobile-platform
StairNetV3-large	1.0	3.0×10^7	62.3	27.1	80.2	15.5	62.2
StairNetV3-medium	0.75	1.7×10^7	35.4	24.1	65.4	11.4	43.0
StairNetV3-small	0.5	7.8×10^6	16.1	19.6	57.2	7.0	26.3

Fig. 7 Some visualization results of StairNetV3-large

(a) The results of StairNetV3-large



(b) Ground-truth

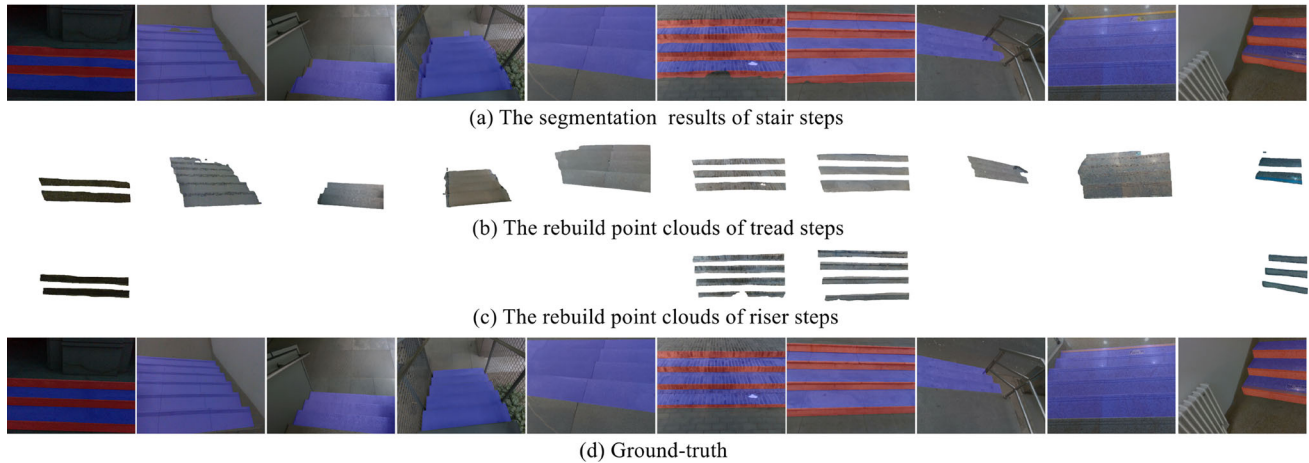
scaled and the linear transformation relationships used to generate the depth maps are not saved, these images cannot be truly used for point cloud reconstruction. We test point cloud reconstruction on the test set, and Table 5 shows the results. Some visualization results of the stair step surface segmentation are shown in Fig. 8. Besides, we implement a classical point cloud segmentation method based on point cloud cluster for comparison. The steps of classical method mainly include point cloud downsampling, normal estimation and point cloud cluster. In this paper, we apply Open3D [53] to implement the voxel downsampling and the normal estimation, and we apply K -means cluster of Scikit-learning

[54] to implement the point cloud cluster. Because the stair structures may be lost after the voxel size exceeds 0.02, we only test three values of the voxel size, which are 0, 0.01, and 0.02, and 0 represents no downsampling. Table 6 shows the results.

As can be seen from Tables 5 and 6, StairNetV3 with the point cloud reconstruction algorithm has better real-time performance on the two kinds of platforms than the classical point cloud segmentation method. The classical method is still inferior to our method after downsampling. This is benefit from our approach of implementing point cloud segmentation by combining image segmentation with point

Table 5 Speed test results of point cloud reconstruction

Process	Runtime (ms) (batch size = 1)	
	Desktop-platform	Mobile-platform
StairNetV3-large	27.1	80.2
Post-processing of CNN outputs	2.1	3.1
Processing of depth image	6.9	10.4
Rebuilding point cloud	3.5	5.3
Whole process	39.6	99.0


Fig. 8 Some visualization results of stair step surface segmentation and point cloud reconstruction

cloud reconstruction, which avoids the slow speed and large data processing drawbacks of direct point cloud segmentation.

4.4 Comparison experiments

In this section, we compare StairNetV3 with some previous methods, including traditional image processing method, combined method of deep learning and traditional image processing, deep learning method DHT [12] for semantic line detection, deep learning method L-CNN [13] for wireframe parsing, deep learning method StairNetV1 and binocular deep learning method StairNetV2. We implement the traditional method using Gabor filters, Canny edge detection and Hough transform, and then we combine them with YOLOv5 [32] as a combined method of deep learning and traditional

image processing. All the deep learning methods are trained and validated on the RGB-D stair dataset. The outputs of all the methods are adjusted to be consistent with StairNetV3, and all the methods are carried out on the desktop platform described in Sect. 4.2, and Table 7 shows the results.

It can be seen that deep learning methods have better accuracy than traditional methods. Because traditional methods have heavy reliance on manually designed rules and the selection of thresholds, it is difficult to perform well in large dataset containing complex and variable scenes. While, deep learning methods extract features by learning, which can greatly improve adaptability to different scenes compared to traditional methods. Among the deep learning methods, DHT and L-CNN have lower accuracy than the StairNet series, this is because different deep learning tasks need different feature representation methods. DHT is designed for detect-

Table 6 Speed test results of classical point cloud segmentation

Process	Runtime (ms) (batch size = 1)					
	Desktop-platform			Mobile-platform		
Voxel size	0	0.01	0.02	0	0.01	0.02
Downsampling	0.0	23.9	14.8	0.0	32.0	21.4
Normal estimation	207.9	66.7	23.6	296.5	90.9	33.8
K-means cluster	576.7	162.2	60.5	870.5	244.5	91.6
Whole time	784.6	252.8	98.9	1167.0	367.4	146.8

Table 7 Experimental results of comparison experiments

Method	Precision (%)	Recall (%)	IOU (%)	GFlops	Runtime (ms)
Gabor + Canny + Hough (Our implementation)	18.9	19.8	10.7	–	4.0
YOLOv5 + Gabor + Canny + Hough (Our implementation)	28.5	22.5	14.3	13.2	14.9
DHT	34.6	27.8	18.2	–	15.9
L-CNN	30.5	52.7	24.0	–	23.1
StairNet 0.25 ×	74.8	77.7	61.6	2.8	2.6
StairNet 0.5 ×	76.3	77.3	62.4	10.4	4.7
StairNet 1 ×	77.4	76.4	62.5	39.6	9.6
StairNetV2 0.25 × (Binocular)	73.7	85.5	65.5	0.6	1.3
StairNetV2 0.5 × (Binocular)	77.0	85.7	68.2	2.3	2.4
StairNetV2 1 × (Binocular)	79.8	86.0	70.6	8.7	5.3
StairNetV3-small (Ours)	74.1	81.0	63.1	16.1	7.0
StairNetV3-medium (Ours)	75.5	81.8	64.6	35.4	11.4
StairNetV3-large (Ours)	77.3	81.8	65.9	62.3	15.5

The bold values indicate the best performance of each column

ing meaningful semantic lines while L-CNN is designed for wireframe parsing, both of them are not suitable for the stair line detection.

For the StairNet series, in fact, the feature representation method with Gaussian kernels used by StairNetV3 is not conducive to improving accuracy. This is because the confidences of some positive samples are weakened, especially the positive samples in the middle of the stair lines have a confidence of only 0.6. For the feature representation method of StairNet and StairNetV2, the confidences of positive samples are all 1, so there is a strong distinction between the background and foreground. Even under these circumstances, because of the awareness of depth information, StairNetV3 still has better performance than StairNet. Furthermore, StairNetV3-small with less parameters and faster speed still surpasses StairNet 1 × in accuracy, which proves the effectiveness of the proposed method. However, StairNetV3 can not surpass the binocular method StairNetV2, the depth inputs have strong inspiration for detecting stair lines, which make StairNetV2 can even work in extremely dark scenes, and the monocular methods are still limited to some extent by lighting condition.

Some visualization results of the comparison experiments are shown in Fig. 9. We can see that traditional method has a certain degree of false detection and missed detection in different scenes due to their sensitivity to manually designed rules and thresholds. The combined method of deep learning and traditional image processing can eliminate false detection to some extent, but the missed detection is still serious. DHT detects some structure lines and misses some stair lines while L-CNN has some false detection. The StairNet series show great adaptability to various complex scenes, and StairNetV3 has similar performance to StairNetV2 in environments with fuzzy visual cues, such as night and

descending stairs, which is better than StairNet. In addition, when there is ground with similar texture to stairs, StairNetV3 has less false detection compared to StairNet.

4.5 Ablation study

The proposed pipeline contains three tasks including LET, SST and DET, and these tasks can affect each other, we conduct some ablation experiments to analyze the details. As the DAB must be applied with the DET, we only need to conduct 9 sets of ablation experiments. Besides, to explore the affection of model composition on the post-processing algorithms, we also test the accuracy of the LSL and the stair point cloud reconstruction methods, and Table 8 shows the results.

The results show that both SST and DET can improve the accuracy of LET, but DET can not improve the accuracy of SST. DAB can improve the effectiveness of DET and LET can also improve the accuracy of SST. For LSL, SST, DET with DAB can improve the accuracy compared to the only LET, but LET combined with DET can not improve the accuracy. For the stair point cloud reconstruction method, as the pixels of the image and the points of the point cloud correspond one to one, the PA and MPA of the model are the accuracy of stair point cloud reconstruction method. In conclusion, LET and SST can supervise each other while DET can only supervise LET, and DAB can make DET more effective.

5 Discussion

In this paper, we propose a monocular method StairNetV3 using deep learning for stair modeling, including the novel depth-aware architecture with depth supervision in both

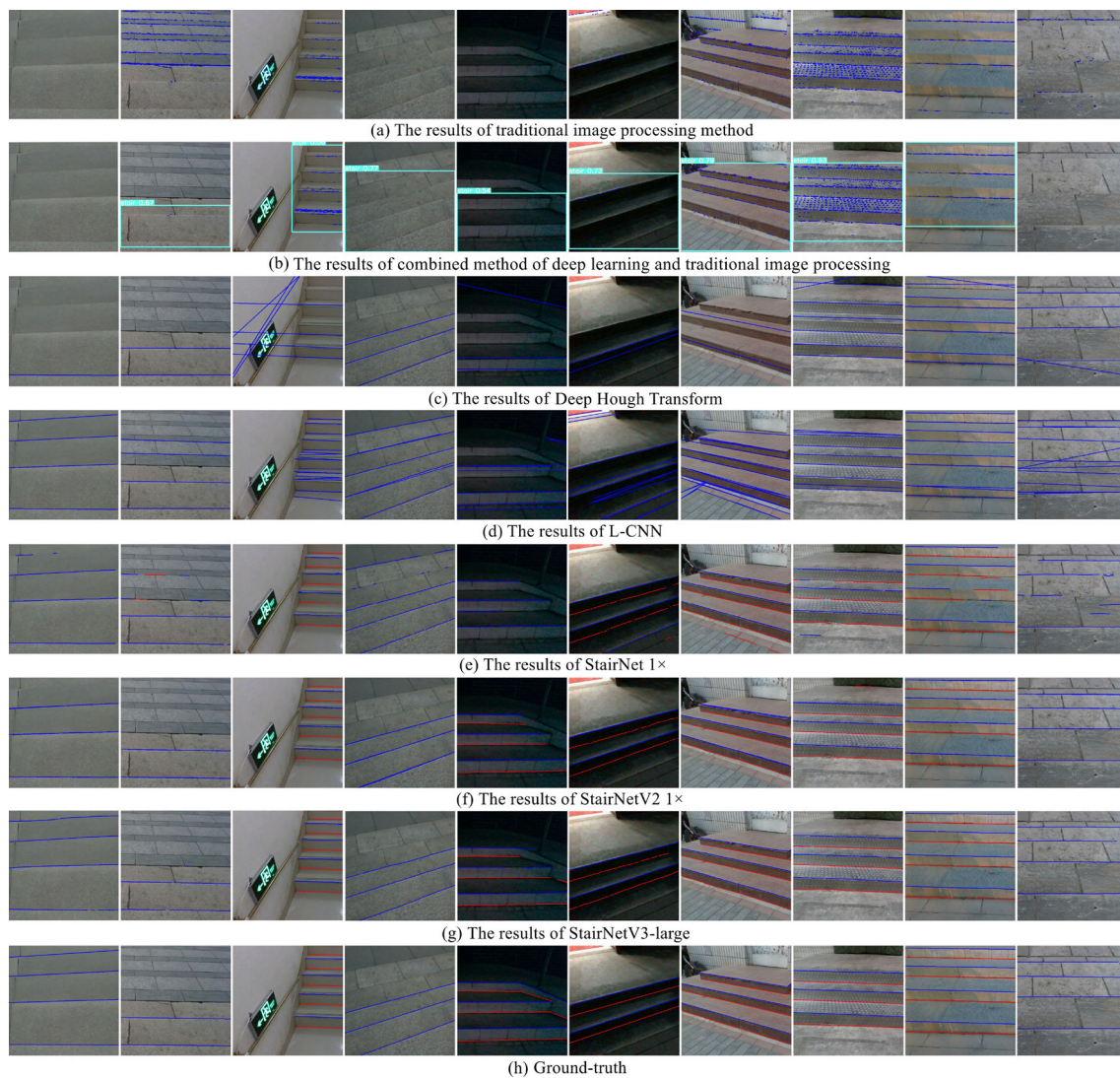


Fig. 9 Some visualization results of comparison experiments

implicit and direct manners, the improved stair feature representation method, the new post-processing algorithm of line segments linking and the new stair point cloud segmentation method with semantic segmentation and point cloud reconstruction. Among the StairNet series, StairNet solves the low adaptability and robustness of traditional methods [1–5] in complex scenes and brings deep learning to stair line detection task in an end-to-end manner. However, the limitation of monocular vision prevents StairNet from performing well in scenes with fuzzy visual clues, such as night scene, scene containing objects with similar stair textures and when descending stairs. To solve this problems, StairNetV2 with multi-modal inputs is proposed. StairNetV2 follows the feature representation method of StairNet and applies a novel selective module to fuse the RGB and depth feature maps in the network. Although StairNetV2 reaches a high level for stair line detection task, it has some limitation for fur-

ther application. The multi-modal inputs need dense depth maps generated by depth camera and the line extraction based method can not model stairs totally. So, our original intention in proposing StairNetV3 is enabling the network to have multi-modal awareness ability while expanding its applicability.

Stair detection can be regarded as a kind of line detection task, and for the line detection task, there have been several deep learning methods. Some deep learning based works have achieved success in semantic line detection and wireframe parsing tasks [11–16]. In our comparison experiments, we choose DHT and L-CNN as representatives for semantic line detection method and wireframe parsing method, respectively. And the results show that these methods are not suitable for detecting stair lines. Not like the traditional methods, the success of deep learning methods often depends on reasonable feature representation methods. StairNetV3 finds

Table 8 Experimental results of ablation experiments

Backbone	Precision (%)	Recall (%)	IOU (%)	Precision (LSL) (%)	Recall (LSL) (%)	IOU (LSL) (%)	PA (%)	MPA (%)
LET	76.66	80.64	64.75	97.12	72.59	71.03	—	—
SST	—	—	—	—	—	—	95.88	80.85
LET + SST	76.66	81.56	65.34	97.84	73.73	72.54	96.28	81.25
LET + DET	77.14	80.87	65.24	97.22	71.68	70.20	—	—
SST + DET	—	—	—	—	—	—	95.81	80.83
LET + SST + DET	77.00	81.27	65.40	97.44	74.01	72.58	96.03	81.02
DAB + LET + DET	77.47	80.85	65.46	97.35	72.77	71.33	—	—
DAB + SST + DET	—	—	—	—	—	—	95.83	80.82
DAB + LET + SST + DET	77.25	81.80	65.92	97.73	73.78	72.52	96.30	81.27

a suitable feature representation method and realizes a reasonable detection pipeline for stair detection.

As mentioned above, StairNetV3 combines the multi-modal awareness ability of StairNetV2 and the applicability of StairNet, and can model stairs with the added semantic segmentation branch. However, StairNetV3 still has room for improvement. The depth-aware backbone and the stair modeling backbone both follow the design of StairNet, which leads to StairNetV3 having the models with the largest number of parameters in the StairNet series. In the future work, we will optimize the backbones to get a better real-time performance. In the comparison experiments, the gap between StairNetV2 and StairNetV3 can not be ignored. Although StairNetV2 can perform in extremely dark night scenes with multi-modal inputs, which monocular methods can never achieve, we will try to optimize the depth-aware mechanism to fully utilize depth features.

In the StairNet series, all of them are dedicated to obtaining stair geometric features in 2D images. It still need some post-processing methods and the distance sensors (depth camera, lidar, ultrasonic sensors, etc) to estimate the stair geometric parameters. We will study the estimation process through CNN and take the stair modeling pipeline in an end-to-end manner in the future.

6 Conclusion

We propose a depth-aware stair modeling convolutional neural network that can simultaneously extract stair lines and stair step surfaces under monocular vision, overcoming the poor performance in environments with fuzzy visual cues, such as night and descending stairs. To learn more abundant stair line information, including stair line endpoints, we propose a new stair line feature representation method with Gaussian kernels. To achieve fast extraction of stair step surfaces, we propose a point cloud reconstruction method for the segmentation of the point clouds belonging to stair step surfaces in an image. Experiments conducted on the RGB-D stair dataset demonstrate the effectiveness, efficiency and usability of our method. In future work, we will study the extraction of stair geometric features in the geodetic coordinate system to further explore the potential of monocular methods.

Author Contributions CW, ZP, SQ and YW made the dataset. CW and SQ made all the figures used in the paper. CW designed the software architecture and wrote the paper. CW, YW and ZT conceived the experiments and conducted the experiments. YW, SQ and ZP analysed the results. All authors reviewed the manuscript.

Funding No funding was received to assist with the preparation of this manuscript.

Data availability Our dataset is available at <https://data.mendeley.com/datasets/6kffmj7g2/1>.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

- Krausz, N.E., Hargrove, L.J.: Recognition of ascending stairs from 2d images for control of powered lower limb prostheses. In: 2015 7th International IEEE/EMBS Conference on Neural Engineering (NER), pp. 615–618 (2015). <https://doi.org/10.1109/NER.2015.7146698>
- Murakami, S., Shimakawa, M., Kivota, K., Kato, T.: Study on stairs detection using RGB-depth images. In: 2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS), pp. 1186–1191 (2014). <https://doi.org/10.1109/SCIS-ISIS.2014.7044705>
- Shahrabadi, S., Rodrigues, J.M., Buf, J.: Detection of indoor and outdoor stairs. In: Iberian Conference on Pattern Recognition & Image Analysis, pp. 847–854 (2013). https://doi.org/10.1007/978-3-642-38628-2_100
- Canny, J.: A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 679–698 (1986)
- Hough, P.V.C.: Method and Means for Recognizing Complex Patterns (1962)
- Westfechtel, T., Ohno, K., Mertsching, B., Nickchen, D., Kojima, S., Tadokoro, S.: 3d graph based stairway detection and localization for mobile robots. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 473–479 (2016). <https://doi.org/10.1109/IROS.2016.7759096>
- Perez-Yus, A., Gutierrez-Gomez, D., Lopez-Nicolas, G., Guerrero, J.J.: Stairs detection with odometry-aided traversal from a wearable RGB-D camera. *Comput. Vis. Image Underst.* **154**, 192–205 (2017). <https://doi.org/10.1016/j.cviu.2016.04.007>
- Zhao, X., Chen, W., Yan, X., Wang, J., Wu, X.: Real-time stairs geometric parameters estimation for lower limb rehabilitation exoskeleton. In: 2018 Chinese Control And Decision Conference (CCDC), pp. 5018–5023 (2018). <https://doi.org/10.1109/CCDC.2018.8408001>
- Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**(6), 381–395 (1981). <https://doi.org/10.1145/358669.358692>
- Oh, K.W., Choi, K.S.: Supervoxel-based staircase detection from range data. *IEIE Trans. Smart Process. Comput.* **4**(6), 403–406 (2015). <https://doi.org/10.5573/IEIESPC.2015.4.6.403>
- Lee, J.-T., Kim, H.-U., Lee, C., Kim, C.-S.: Semantic line detection and its applications. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 3249–3257 (2017). <https://doi.org/10.1109/ICCV.2017.350>
- Zhao, K., Han, Q., Zhang, C.-B., Xu, J., Cheng, M.-M.: Deep hough transform for semantic line detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**(9), 4793–4806 (2022). <https://doi.org/10.1109/TPAMI.2021.3077129>
- Zhou, Y., Qi, H., Ma, Y.: End-to-end wireframe parsing. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 962–971 (2019). <https://doi.org/10.1109/ICCV.2019.00105>
- Xue, N., Wu, T., Bai, S., Wang, F., Xia, G.-S., Zhang, L., Torr, P.H.S.: Holistically-attracted wireframe parsing. In: 2020

- IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2785–2794 (2020). <https://doi.org/10.1109/CVPR42600.2020.00286>
15. Zhang, H., Luo, Y., Qin, F., He, Y., Liu, X.: Elsd: efficient line segment detector and descriptor. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2949–2958 (2021). <https://doi.org/10.1109/ICCV48922.2021.00296>
 16. Dai, X., Gong, H., Wu, S., Yuan, X., Yi, M.: Fully convolutional line parsing. *Neurocomputing* **506**, 1–11 (2022). <https://doi.org/10.1016/j.neucom.2022.07.026>
 17. Qin, Z., Wang, H., Li, X.: Ultra fast structure-aware deep lane detection. In: *Computer Vision—ECCV 2020*, pp. 276–291. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58586-0_17
 18. Haris, M., Hou, J., Wang, X.: Lane line detection and departure estimation in a complex environment by using an asymmetric kernel convolution algorithm. *Vis. Comput.* **39**, 519–538 (2023). <https://doi.org/10.1007/s00371-021-02353-6>
 19. Platt, J.C.: Sequential minimal optimization: a fast algorithm for training support vector machines. In: *Advances in Kernel Methods—Support Vector Learning*, vol. 208 (1998)
 20. Khaliluzzaman, M., Yakub, M., Chakraborty, N.: Comparative analysis of stairways detection based on RGB and RGB-D image. In: 2018 International Conference on Innovations in Science, Engineering and Technology (ICISSET), pp. 519–524 (2018). <https://doi.org/10.1109/ICISSET.2018.8745624>
 21. Khaliluzzaman, M., Deb, K., Jo, K.-H.: Geometrical feature based stairways detection and recognition using depth sensor. In: *IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society*, pp. 3250–3255 (2018). <https://doi.org/10.1109/IECON.2018.8591340>
 22. Redmon, J., Farhadi, A.: YOLOv3: An Incremental Improvement (2018)
 23. Patil, U., Gujarathi, A., Kulkarni, A., Jain, A., Malke, L., Tekade, R., Paigwar, K., Chaturvedi, P.: Deep learning based stair detection and statistical image filtering for autonomous stair climbing. In: 2019 Third IEEE International Conference on Robotic Computing (IRC), pp. 159–166 (2019). <https://doi.org/10.1109/IRC.2019.00031>
 24. Wang, C., Pei, Z., Shuang, Q., Tang, Z.: Deep leaning-based ultra-fast stair detection. *Sci. Rep.* **12**, 16124 (2022). <https://doi.org/10.1038/s41598-022-20667-w>
 25. Wang, C., Pei, Z., Qiu, S., Tang, Z.: RGB-D-based stair detection and estimation using deep learning. *Sensors* (2023). <https://doi.org/10.3390/s23042175>
 26. Huang, X., Tang, Z.: Staircase detection algorithm based on projection-histogram. In: 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp. 1130–1133 (2018). Doi: <https://doi.org/10.1109/IMCEC.2018.8469186>
 27. Carbonara, S., Guaragnella, C.: Efficient stairs detection algorithm assisted navigation for vision impaired people. In: 2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings, pp. 313–318 (2014). <https://doi.org/10.1109/INISTA.2014.6873637>
 28. Khaliluzzaman, M., Deb, K., Jo, K.-H.: Stairways detection and distance estimation approach based on three connected point and triangular similarity. In: 2016 9th International Conference on Human System Interactions (HSI), pp. 330–336 (2016). <https://doi.org/10.1109/HSI.2016.7529653>
 29. Diamantis, D.E., Koutsiou, D.C.C., Iakovidis, D.K.: Staircase detection using a lightweight look-behind fully convolutional neural network. In: Macintyre, J., Iliadis, L., Maglogiannis, I., Jayne, C. (eds.) *Engineering Applications of Neural Networks*, pp. 522–532. Springer, Cham (2019)
 30. Lee, Y.H., Leung, T.-S., Medioni, G.: Real-time staircase detection from a wearable stereo system. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 3770–3773 (2012)
 31. Rekhawar, N., Govindani, Y., Rao, N.: Deep learning based detection, segmentation and vision based pose estimation of staircase. In: 2022 1st International Conference on the Paradigm Shifts in Communication, Embedded Systems, Machine Learning and Signal Processing (PCEMS), pp. 78–83 (2022). <https://doi.org/10.1109/PCEMS55161.2022.9807915>
 32. Glenn, J.: yolov5 (2019). <https://github.com/ultralytics/yolov5>
 33. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
 34. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). <https://doi.org/10.1109/CVPR.2016.90>
 35. Wang, S., Pan, H., Zhang, C., Tian, Y.: RGB-D image-based detection of stairs, pedestrian crosswalks and traffic signs. *J. Vis. Commun. Image Represent.* **25**(2), 263–272 (2014). <https://doi.org/10.1016/j.jvcir.2013.11.005>
 36. Wang, S., Tian, Y.: Detecting stairs and pedestrian crosswalks for the blind by RGBD camera. In: 2012 IEEE International Conference on Bioinformatics and Biomedicine Workshops, pp. 732–739 (2012). <https://doi.org/10.1109/BIBMW.2012.6470227>
 37. Munoz, R., Rong, X., Tian, Y.: Depth-aware indoor staircase detection and recognition for the visually impaired. In: 2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), pp. 1–6 (2016). <https://doi.org/10.1109/ICMEW.2016.7574706>
 38. Pérez-Yus, A., López-Nicolás, G., Guerrero, J.J.: Detection and modelling of staircases using a wearable depth sensor. In: Agapito, L., Bronstein, M.M., Rother, C. (eds.) *Computer Vision—ECCV 2014 Workshops*, pp. 449–463. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16199-0_32
 39. Yifei, Y., Jianzhong, W.: Stair area recognition in complex environment based on point cloud. *J. Electron. Meas. Instrum.* **34**(4), 124–133 (2020). <https://doi.org/10.13382/j.jemi.B1902746>
 40. Matsumura, H., Premachandra, C.: Deep-learning-based stair detection using 3d point cloud data for preventing walking accidents of the visually impaired. *IEEE Access* **10**, 56249–56255 (2022). <https://doi.org/10.1109/ACCESS.2022.3178154>
 41. Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J.: Pointnet: deep learning on point sets for 3d classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 77–85 (2017). <https://doi.org/10.1109/CVPR.2017.16>
 42. Zatout, C., Larabi, S.: Semantic scene synthesis: application to assistive systems. *Vis. Comput.* **38**, 2691–2705 (2022). <https://doi.org/10.1007/s00371-021-02147-w>
 43. Zatout, C., Larabi, S., Mendili, I., Barnabé, S.A.E.: Ego-semantic labeling of scene from depth image for visually impaired and blind people. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 4376–4384 (2019). <https://doi.org/10.1109/ICCVW.2019.00538>
 44. Zhou, X., Wang, D., Krähenbühl, P.: Objects as Points (2019)
 45. Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **42**(8), 2011–2023 (2020). <https://doi.org/10.1109/TPAMI.2019.2913372>
 46. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5987–5995 (2017). <https://doi.org/10.1109/CVPR.2017.634>

47. Wang, C., Pei, Z., Qiu, S., Tang, Z.: Stair dataset with depth maps. Mendeley Data (2023). <https://doi.org/10.17632/p28ncjnvvgk.2>
48. Intel: Depth Camera D435i. <https://www.intelrealsense.com/depth-camera-d435i/>
49. Wang, C., Pei, Z., Qiu, S., Wang, Y., Tang, Z.: RGB-D stair dataset. Mendeley Data (2023). <https://doi.org/10.17632/6kffmj7g2.1>
50. Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Garcia-Rodriguez, J.: A review on deep learning techniques. *Appl. Semant. Segm.* **9**, 1–7 (2017)
51. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2017)
52. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications (2017)
53. Zhou, Q.-Y., Park, J., Koltun, V.: Open3D: a modern library for 3D data processing. *arXiv:1801.09847* (2018)
54. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Shuang Qiu received the M.S. degree in aeronautical engineering from Nanjing University of Science and Technology, Nanjing, China. He is currently working toward the Ph.D. degree in School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His current research interests include wearable robotics systems, human-robot interaction and deep learning.



Yachun Wang received B.E degree in College of Information and Electrical Engineering of China Agricultural University, Beijing, China. She is currently pursuing the M.S. degree in School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. Her research interests include computer vision, deep learning and human keypoints detection.



Zhiyong Tang received B.E degree and Ph.D. degree in Beihang University, Beijing, China. He is now a associate professor in School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include electromechanical hydraulic servo control, network control and intelligent robot system.



Chen Wang received B.E degree in School of Mechanical Engineering, Tianjin University, Tianjin, China. He received M.E. degree in School of Mechanical Engineering & Automation, Beihang University, Beijing, China. He is currently pursuing the Ph.D. degree in School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include computer vision, deep learning and environmental perception.



Zhongcai Pei received B.E degree, M.E. degree and Ph.D. degree in School of Mechatronics Engineering, Harbin Institute of Technology, Harbin, China. He is now a professor in School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include electromechanical hydraulic control systems, digital hydraulic technology, parallel mechanism and robot technology, medical equipment and exoskeleton assisted support robots and micro

air vehicles.