Sensors and Materials, Vol. 35, No. 7 (2023) 2549–2566
MYU Tokyo

S & M 3340

Construction and Development of Visual Simultaneous Localization and Mapping System for Multi-person Dynamic Spaces

Neng-Sheng Pai, Xi-Zhi Huang, Pi-Yun Chen,* Zi-Heng Zhong, and Wei-Zhe Huang

Department of Mechanical Engineering, National Chin-Yi University of Technology, Taichung 411030, Taiwan, R.O.C

(Received December 28, 2022; accepted June 28, 2023)

Keywords: edge computing, vision system, simultaneous localization and mapping, semantic segmentation

In this study, we developed a visual simultaneous localization and mapping (SLAM) dynamic object filtering system applicable to multi-person dynamic spaces. When a visual mobile vehicle with an edge computing platform is used in the dynamic living space, the visual SLAM system will likely be disturbed by dynamic persons, leading to poor mapping and localization. We also implemented the dynamic object filtering function by preprocessing with a semantic segmentation neural network and designed a high-performance neural network architecture for the edge computing platform. The proposed design is capable of real-time operation to improve the effect of dynamic objects on the visual SLAM system.

1. Introduction

Owing to the COVID-19 pandemic, interpersonal contact decreased and labor shortages in factories emerged. Driven by the smart production of Industry 4.0, autonomous mobile robots (AMRs) have become a popular development project. Moreover, people's average life span has been prolonged by enhancements in medical systems. As a result, the demand for long-term care grew and nursing personnel became insufficient. The development of service AMRs may be one of the irreversible development processes in the future.[1–3]

An AMR remedies the defect in an automated guided vehicle (AGV).[4,5] Moreover, it can evade obstacles autonomously and keep working efficiently while preventing collisions.[6] Service systems have paid closer attention to users' privacy and independence. The integration of edge computing into AMR devices provides marked scalability, complemented by its lightweight design and real-time capabilities. AMR devices can process and analyze data locally, reducing their reliance on remote servers, through the utilization of edge computing. By adopting this architectural approach, AMR devices can efficiently perform more complex tasks, providing increased flexibility and expandability. Consequently, they become the central focus of global industries' development endeavors. With AI's rapid and flourishing development in

*Corresponding author: e-mail: chenby@ncut.edu.tw
https://doi.org/10.18494/SAM4295

ISSN 0914-4935

recent years, the system with edge computing and vision sensors can perform tasks such as semantic segmentation, object detection, and visual simultaneous localization and mapping (SLAM) at a lower cost.

However, one of the important abilities of the AMR system is SLAM. This function is often assumed to work depending on a static environment. Therefore, when the object serviced by the AMR is a person and the activity space is in a multi-person dynamic living environment, the AMR may fail to obtain the optimal mapping quality and positioning accuracy owing to dynamic pedestrians. In this study, we used the semantic segmentation technique to build a filtering system for this problem and to complete the preprocessing task of visual SLAM systems, reducing the interference of dynamic pedestrians.

The semantic segmentation technique is one of the key tasks in the vision domain. The technique completes the pixel-scale classification tasks. It enables the system to segment and comprehend the meaning of scenes. The objects can be filtered or segmented out. A high-accuracy segmentation neural network architecture, such as HRNet[7] or U-Net,[8] exhibits superior performance in terms of semantic segmentation accuracy compared with MobileNet[9] and EfficientNet.[10] Deploying neural network architectures, such as HRNet, on edge computing platforms for real-time execution poses challenges. Although these architectures may improve semantic segmentation accuracy, their suitability for dynamic living environments that demand immediate responses is limited on edge computing platforms. Hence, their deployment is not a good choice for the environment's changeful living space. In this study, we designed an applicable neural network architecture according to the optimization and efficiency of edge computing platforms to find a balancing point between accuracy and instruction cycle.

2. Related Work

The literature review comprises four parts: edge computing platform, sensor, SLAM, and semantic segmentation.

2.1 Edge computing platform

The edge computing platform can be the core computing power of AMRs. The service private system should protect privacy and prevent private data from leaking out. In this study, we used NVIDIA Jetson AGX Xavier[11] as the core of the operation, supporting the TensorRT Software Development Kit (SDK) released by NVIDIA.[12] The SDK can optimize the operating efficiency according to hardware. The sufficient hardware configuration of Jetson AGX can complete most edge computing tasks instead of uploading data to the network.

2.2 Sensor

The 3D LiDAR is commonly used as the main sensor of conventional autonomous vehicles or AMRs.[13–15] However, with the development of hardware and deep learning, cameras for implementing pure vision have become a low-cost multifunction choice. For example, Tesla

changed the vehicle autopilot assistance system into a pure vision version in 2022.[16] A subsidiary company of Toyota in charge of developing autopilot technology also declared that it would develop a pure vision autopilot technology centered on camera lenses. The edge computing platform vehicle constructed for the system in this study used a binocular camera as the sensor, and the function in the study was completed by vision.

2.3 SLAM

SLAM can be divided into visual SLAM and laser SLAM according to the sensors used. The fundamental purpose is to construct a global static map and localize its position. The former has abundant texture information, whereas the latter has high accuracy and relatively simple architecture. In addition to visual SLAM, we employed real-time appearance-based mapping (RTAB-Map)[17] as an algorithm. This algorithm has mapping, localization, infinite loop detection, and memory management functions. The overall system is relatively complete and practical.

2.4 Semantic segmentation

Semantic segmentation classifies all pixels in the image. Each classified object is provided with the spatial position information in the image. Deep learning is mainly used for image segmentation. It is often used in autopilot to provide the cognition of the external environment and to assist doctors in finishing medical decision-making.[18,19] Semantic segmentation was preferentially used for the object filtering function of SLAM systems. Considering the overall performance of edge computing platforms, the semantic segmentation task in this study mainly referred to recently released real-time architectures, such as BiSeNet v2,[20] STDC,[21] and DeepLab v3+.[22] These architectures were optimized and designed for platforms.

3. Methodology

This study is divided into three major parts: real-time semantic segmentation neural network, target mask image expansion, and filtering the target out of the depth map. The visual SLAM architecture with a preprocessing filtering system is shown in Fig. 1.

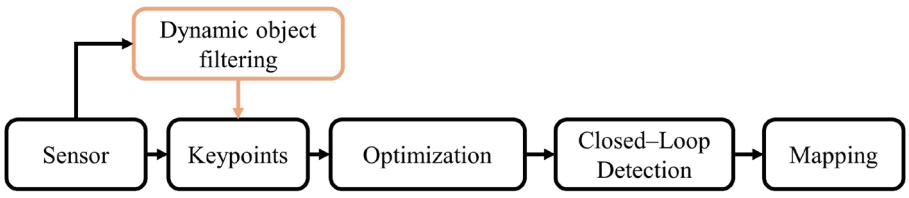


Fig. 1. (Color online) Visual SLAM system architecture with dynamic object filtering.

The visual SLAM is divided into four parts: keypoints, optimization, closed-loop detection, and mapping, as shown in Fig. 1. In the keypoint stage, the image feature extraction, description, and matching were performed to estimate the motion and position of the sensor. The mismatching of features was reduced by optimizing the process to correct the direction of motion of the sensor. The displacement and angular errors accumulated in the system were eliminated by closed-loop detection and the mapping was gradually completed.

The dynamic object filtering with the semantic segmentation neural network of this system is shown in the orange part in Fig. 1. It filtered the keypoints in the architecture, as shown in the schematic diagram in Fig. 2. The green points in Fig. 2(a) are the original keypoints described by the scale-invariant feature transform (SIFT). Figure 2(b) shows the dynamic pedestrians being used as static background keypoints after filtering the targets.

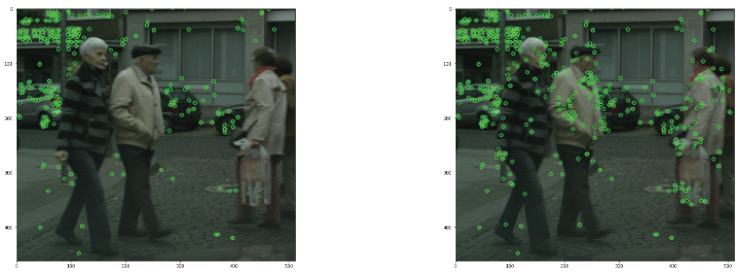
The architecture of dynamic object filtering operated by this system is shown in Fig. 3. It is approximately divided into three steps. A color image was used as a semantic segmentation neural network input to obtain the target mask image. The target mask image was expanded before the depth image was filtered according to the mask image. The related keypoints were removed on the basis of the filtered depth image.

The keypoints formed between the dynamic object and the static background were regarded as dynamic keypoints. The mask image was expanded to some extent using a mean filter to moderately enlarge the filter range.

3.1 Real-time semantic segmentation neural network

The main architecture of the semantic segmentation neural network used in this system is shown in Fig. 4. The green blocks show the downsampling process. S_i represents the No. i downsampling layer, AT is the attention block, Fusion is the feature fusion block, 0.5× represents double downsampling, and 4× represents quadruple upsampling.

The detailed structure is shown in Table 1, taking an input 384 × 640 × 3 color image as an example. The convolutional batch normalization ReLU6 (CBR6) is the convolutional and normalized basic block through an activation function. The Short-Term Dense Concatenate Lite (STDCL) is the variation version of the Short-Term Dense Concatenate (STDC) proposed in Ref. 21, k is the convolution kernel size, c is the number of channels, s is the stride, and r is the number of replications of this layer.



(a) (b)

Fig. 2. (Color online) Schematic diagram of filtering of dynamic object keypoints. (a) Original keypoints and (b) filtered keypoints.

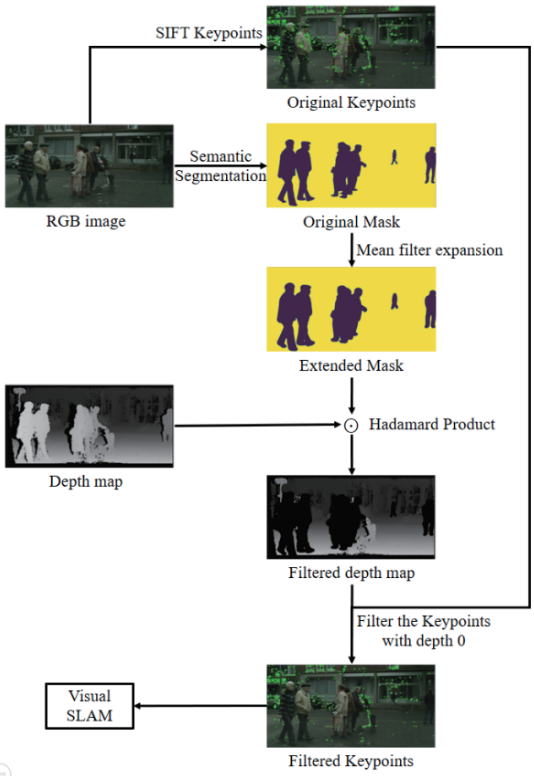


Fig. 3. (Color online) Schematic diagram of dynamic object.

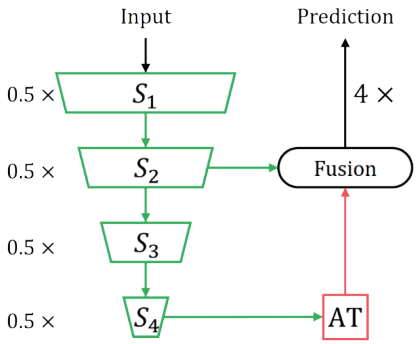


Fig. 4. (Color online) Main architecture of semantic segmentation neural network.

Table 1
Semantic segmentation neural network architecture.

Stage		k	c	s	r	Output size
Input						384 × 640 × 3
S_1	CBR6	3	32	2	1	192 × 320 × 32
	CBR6	3	32	1	1	
S_2	CBR6	3	64	2	1	96 × 160 × 64
	CBR6	3	64	1	1	
S_3	STDCL	3	128	2	1	48 × 80 × 128
	STDCL	3	128	1	1	
S_4	STDCL	3	256	2	1	24 × 40 × 256
	STDCL	3	256	1	1	
AT					1	24 × 40 × 256
Fusion			64		1	90 × 160 × class

The overall structure is similar to the classic encoding/decoding structure of UNET. The long skip connections used in the encoding and decoding processes have been proven effective.[23] For a higher computational efficiency, according to the ideas of detail and semantic branches proposed in Ref. 18, the spatial information and feature details were maintained at S_2 in Fig. 4. The receptive field (RF) of the network increased rapidly. The advanced semantic features were obtained at S_4. A novel encoding/decoding scheme was introduced, incorporating a novel

approach inspired by the STDC module presented by Feng *et al.* [21] and the DeepLab framework proposed by Chen *et al.*[22] This scheme utilizes a reduced number of long skip connections, facilitating the rapid fusion of two types of feature information.

The STDCL block is shown in Fig. 5. It comprises three CBR6 blocks, with s being the stride and c denoting the number of channels from the layer. The STDCL block has rich features of different RFs. Maintaining the features of different RFs positively benefits the semantic segmentation tasks.

Blocks are mainly referred to as STDC blocks in Ref. 19. The blocks used more channels to encode the detailed feature information of small RFs, whereas the semantic feature of large RFs paid closer attention to the mode of expression of information. Excessive channels might induce information redundancy. The STDCL proposed in this study removed the Pointwise Convolution of Layer 1 of STDC. The downsampling process was completed directly in Layer 1 CBR6 so that the structure could be more compact and simple, thereby increasing the computational efficiency, as shown in Fig. 6.

The overall block maintained three layers of convolution operation with the convolution kernel size of 3 × 3. It is the same as the main structure of STDC blocks. As a result, there were similar RFs, as shown in Table 2.

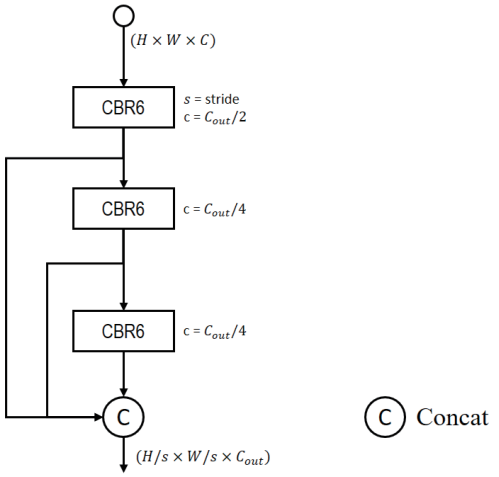


Fig. 5. STDCL block.

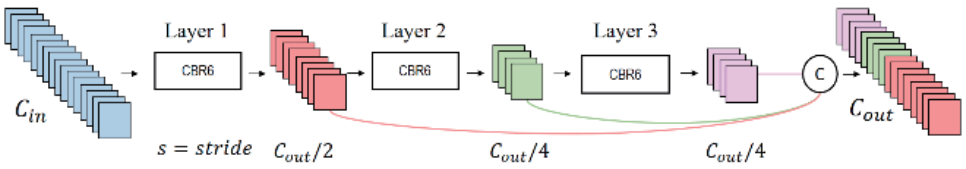


Fig. 6. (Color online) Schematic diagram of STDCL block.

Table 2
RFs of STDCL block.

	Layer 1	Layer 3	Layer 2	Fusion
RF stride = 1	3 × 3	5 × 5	7 × 7	3 × 3, 5 × 5, 7 × 7
RF stride = 2	3 × 3	7 × 7	11 × 11	3 × 3, 7 × 7, 11 × 11

Pointwise Convolution could control the number of channels and the total number of parameters. However, the test result shows that when the model is lightweight and the hardware computation capability is strong, the Pointwise Convolution slowed down operations and the accuracy did not increase significantly.

The AT block was similar to the Squeeze-and-Excitation structure proposed by SENet.[24] The information relationship between channel features was learned on the basis of neural networks and used as weighting weights. This highlighted more important feature information. A better network effect could be obtained with a lower amount of computation. It is used in object detection and semantic segmentation domains to enhance and guide the semantic feature to respond to detail features, achieve a better feature fusion effect, and enhance the capability for semantic classification and detail segmentation. The residual AT block of one fully connected layer of the semantic segmentation neural network in this study is shown in Fig. 7, where GAPooling is Global Average Pooling, FC is Fully Connected Layer, BN is Batch Normalization, and Sigmoid is the sigmoid activation function.

A fully connected layer learned the weighted value of primary features after pooling, and a sigmoid activation function normalized it. The weighted feature and primary feature were residually fused after weighing the primary feature to maintain more useful feature information.

The feature fusion block is shown in Fig. 8. In reference to the fusion method of Bilateral Guided Aggregation in BiseNetV2, the architecture guided two features on different scales to effectively communicate the feature information of different scales. The utilization of features was higher than by the simple fusion method. Compared to simple fusion methods, this architecture places greater emphasis on the complementarity and correlation between features at different scales, thereby enhancing the utilization of features.

S2 and AT in Fig. 8 are the downsampling Layer 2 in Fig. 4, in charge of the spatial and detail information and the semantic information of the large RF of the attention block. Two features on different scales were fused first in the fusion block. The sigmoid function normalized the semantic feature of the large RF. Element multiplication was used for weighted fusion. As the design objective was a lightweight architecture, the final feature channel was relatively narrow and the number of features was small. Fusing the features by concatenation could maintain complete information for the subsequent classification.

This system gave priority to the dynamic object filtering function and filtered persons. To achieve the best segmentation capability, the training set labels only included the background and pedestrians. The general pedestrian classification labels are shown in Fig. 9, in which (a) shows the original image from the Cityscapes dataset,[25] (b) is the schematic diagram of labels of eight classes, and (c) is the schematic diagram only with the person and background classes. When there were considerable classes, an individual class would not have a large proportion.

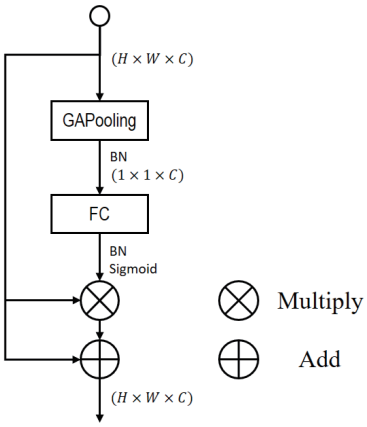


Fig. 7. Residual AT block.

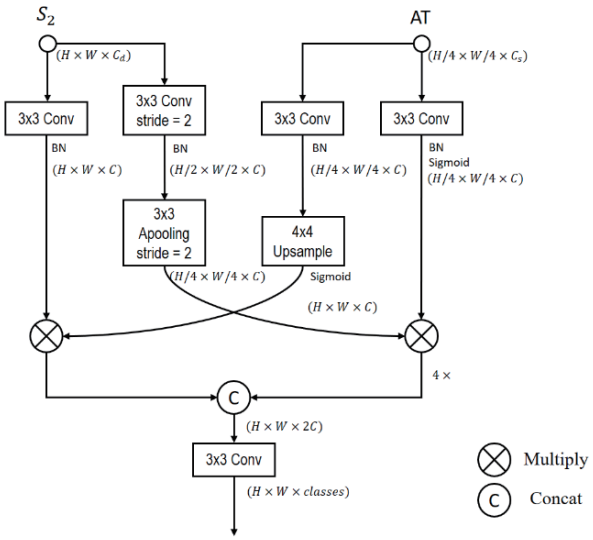
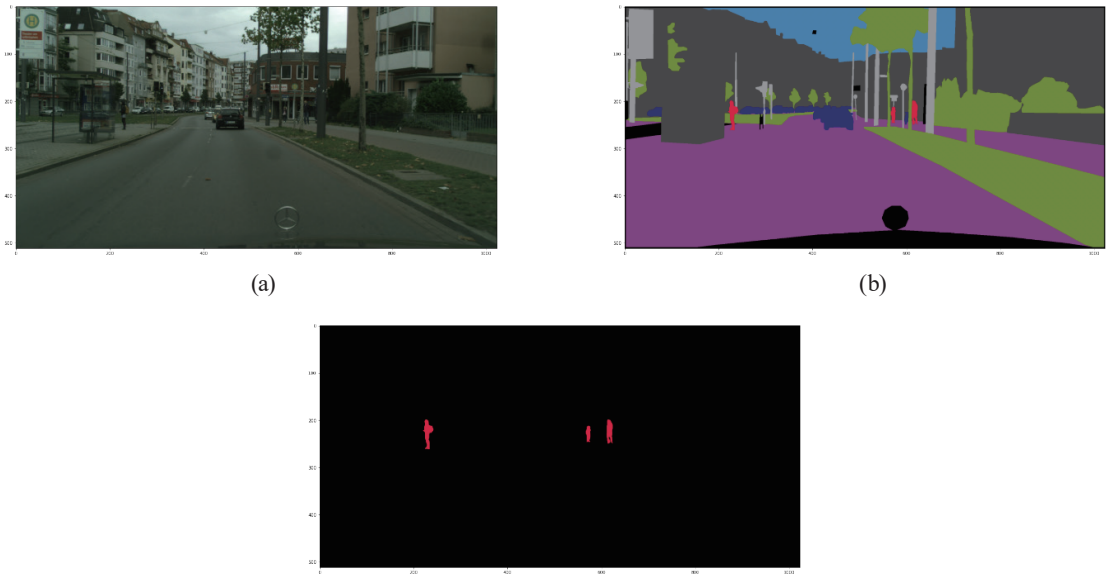


Fig. 8. Feature fusion block.



(a) (b) (c)

Fig. 9. (Color online) Schematic diagram of label proportion unbalance. (a) Original image, (b) eight classes, and (c) two classes.

However, when the target of classification was only pedestrians, the proportion of background to persons was unbalanced. This resulted in the neural network classifying the background first and obtaining relatively lower loss values. Hence, the desired segmentation effect could not be achieved. For the class proportion unbalanced problem, this system used the focal loss proposed in Ref. 26 as the loss function.

The focal loss is expected to focus on difficult training samples in the training process. In the standard cross entropy, as shown in Eq. (1), the loss value given by simple or difficult samples is

very large. The neural network only concentrates on the total loss in the training process. Therefore, the total loss of simple samples (background) is likely larger than the loss of difficult samples (pedestrian). The parameter update of a neural network is almost in the learning background.

$$CE(p_t) = -\log(p_t), p_t = \begin{cases} y', \text{ if } y = 1 \\ 1 - y', \text{ if } y = 0 \end{cases} \tag{1}$$

Here, y is the true value of the label, y' is the predicted value using the sigmoid activation function, $y = 1$ is the classified target, and $y = 0$ is the background. For the classified target, the higher the probability of the prediction value, the lower the loss, and vice versa for the background. The focal loss was provided with a δ weight based on Eq. (1), as expressed in Eq. (2).

$$\delta CE(p_t) = -\delta \log(p_t) \tag{2}$$

The δ weight was exclusively adjusted to scale and minimize the loss value, thereby affecting the parameters. The label proportion unbalanced problem could not be solved. Therefore, the regulatory factor γ was provided, as expressed in Eq. (3).

$$FL(p_t) = -\delta(1 - p_t)^{\gamma} \log(p_t) \tag{3}$$

The focal loss, as defined in Eq. (3), reduces the loss for simple training samples and increases the loss for difficult training samples. This is achieved by taking into account the lower confidence (p_t) typically associated with difficult samples, resulting in a higher focal loss. On the other hand, simple samples with higher confidence exhibit a decreased focal loss. By adjusting the loss based on sample confidence, the focal loss effectively addresses the varying difficulty levels encountered in training samples. The neural network can concentrate on learning the difficult samples to reduce the total loss.

3.2 Target mask image expansion

The visual SLAM operated by this system was a characteristic method, and the movement variation of hardware was calculated by feature matching. According to the test results of different feature algorithms, the juncture of the filtered target and background was also one of the regions where the keypoints were formed. Figure 10 shows that varying degrees of keypoints were formed in the juncture of ORiented Brief (ORB) and SIFT keypoints. Therefore, after obtaining the target mask by semantic segmentation, the mask was expanded, and the juncture was regarded as one of the filter ranges.

In terms of expansion, Eq. (4) utilizes a mean filter on the original mask image, $mask_I$. The purpose of this mean filter is to smoothen and extend the features present in the mask image, ultimately enhancing its comprehensive representation.

(a) (b)

Fig. 10. (Color online) Schematic diagram of keypoints formed between filter target and background. (a) ORB keypoints and (b) SIFT keypoints.

$$mask_f(x,y) = \frac{1}{k_w k_h} \sum_{(s,t)\in S_{xy}} mask_I(s,t) \tag{4}$$

Here, $mask_f(x, y)$ is the pixel value after the mean filter, $mask_I(s,t)$ is the pixel value of the original mask image, and k_w and k_h are the kernel width and height of the filter, respectively. S_{xy} is the coordinate set using xy as center size $k_w \times k_h$, wherein k_w and k_h should be adjusted according to the mask image resolution. The filtered mask image $mask_f$ was binarized using Eq. (5).

$$mask_g(x,y) = \begin{cases} 1, mask_f(x,y) \geq \beta_{th} \\ 0, mask_f(x,y) < \beta_{th} \end{cases} \tag{5}$$

Here, $mask_g$ is the expanded mask image and β_{th} is the binary threshold. The mask image was used to filter the depth image. For the subsequent operation, the data were reversely processed when processing the mask image, exactly contrary to the training set label, with the 0 value in the mask image being the filtered target and 1 being the background. The overall process is shown in Fig. 11, including the original mask image, mean filter expansion processing, and expanded mask image after binarization from top to bottom, where the yellow part is 1 and the purple part is 0.

3.3 Filtering target out of depth map

After expanding the target mask image, the expanded mask image $mask_g$ and the original *depth* map depth were computed by Hadamard product matrix element multiplication, as expressed in Eq. (6).

$$depth_f = depth \odot mask_g \tag{6}$$

Here, $depth_f$ is the filtered depth image, as shown in Fig. 12. After the calculation using Eq. (6), the pixel position depth of the filter target would be 0. The keypoints lacking depth would be removed from the operation of visual SLAM.

Fig. 11. (Color online) Schematic diagram of target mask image expansion process. (a) Original mask image, (b) mean filter processing, and (c) binarization.

Fig. 12. Filterd depth image

4. Experiments

4.1 Experimental procedure and objective

In this study, we used a ZED binocular camera as the vision sensor and constructed the system on the NVIDIA Jetson AGX Xavier edge computing platform. First, different semantic segmentation neural network architectures were built using the PC terminal for training. The trained network was moved to the AGX edge computing platform and optimized by TensorRT. This can help test its accuracy and operating speed, and can guarantee the operating efficiency of the network architecture on the edge computing platform. Afterward, the filtering system proposed in this study was used in visual SLAM. The filtering effect and the quality of the constructed map were tested.

4.2 Semantic segmentation neural network architecture

In the semantic segmentation neural network architecture design, the use of the two-branch and single-branch architectures similar to BiSeNetv2 was attempted, as shown in Fig. 13. The main purpose is to test the operating efficiency of the two architectures on the edge computing platform.

In Fig. 13, the blue blocks are CBR6 blocks and the green blocks are STDC blocks.[21] Fusion is the detailed design of the Bilateral Guided Aggregation Layer proposed by BiSeNet v2.[20] The two architectures were built using the Cityscapes dataset, with the resolution of 384 × 384, 34 classes, Adam optimization, classification cross-entropy loss, and data obtained by training 80 Epochs, as shown in Table 3. fps_{32} and fps_{16} represent the numbers of inference frames built into float 32 and float 16 models on the AGX platform in TensorRT, respectively.

The two-branch architecture could effectively control the number of parameters,[20] but there was no obvious advantage on the AGX edge computing platform. Therefore, following the single-branch architecture in Fig. 13, the Fusion structure was modified. With the fusion architecture proposed in BiSeNet v2 as the control group, the depthwise separable convolution + Pointwise Convolution (DWConv) was reduced to 3 × 3 standard convolution (Conv). The feature fusion was changed from feature addition to the union to keep more features. The results of the resolution of 384 × 640, eight classes, Adam optimization, classification cross-entropy loss, and training 80 Epochs from the Cityscapes dataset are shown in Table 4.

The data in the table are the results of the Cityscapes dataset, with a resolution of 384 × 640, eight classes, Adam optimization, classification cross-entropy loss, and training 80 Epochs. The accuracy could be increased by about 1% after two modifications. However, note that increasing

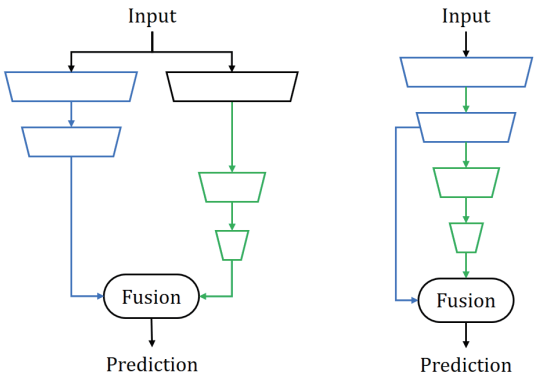


Fig. 13. (Color online) Two-branch and single-branch architectures.

Table 3
Two-branch and single-branch test data.

Architecture		MIoU	fps32	fps16
Bilateral segmentation-STDC	Preserve 1/4 and 1/16 features for fusion	0.3498	144	189
Single stream-STDC		0.3607	150	200

the number of parameters compromises speed owing to the inherent trade-off between model capacity and computational efficiency. As the number of parameters increases, the model gains better capability to represent intricate patterns, but at the cost of higher computational resource requirements, resulting in lower processing speed. The effect of replacing STDC blocks with STDCL blocks is shown in Table 5.

The STDCL simplifies the structure of STDC by reducing one layer of point-wise convolution while increasing the depth of each feature map. Although this leads to an increase in the number of parameters, it results in improved accuracy and computational speed. Finally, the AT block was added, and the effect is shown in Table 6.

After adding the AT block, some speed was sacrificed, but the accuracy could be further increased. The comparison results are shown in Table 7.

In the resolution of 512 × 1024, the MIoU was not the highest, but it had the optimal equilibrium for the AGX platform.

Table 4
Fusion block test data.

Architecture		MIoU	Params	fps32	fps16
Single stream, STDC	Control	0.6274	599720	144	231
	DWConv→Conv	0.6301	759080	138	234
	DWConv→Conv Add→Concate	0.6371	763688	133	222

Table 5
STDC and STDCL test data.

Architecture		MIoU	Params	fps32	fps16
Single stream, Fusion	STDC	0.6371	763688	133	222
	STDCL	0.6507	1114280	134	236

Table 6
Test data after AT block is added.

Architecture		MIoU	Params	fps32	fps16
Single stream, Fusion	Control	0.6507	1114280	134	236
	Add AT block	0.6718	1346088	121	222

Table 7
Data of comparison with other neural network architectures.

Resolution	Model	MIoU	Params	fps32	fps16
512 × 1024	BiSeNet v1(Xception)	0.6761	26.4M	5	15
	BiSeNet v2	0.6659	2.2M	32	54
	DeepLab v3+(resnet50)	0.7345	40.4M	2	9
	DeepLab v3+(MobileNet v2)	0.6672	4.6M	6	15
	UNET	0.7084	7.7M	7	18
	Our architecture	0.6907	1.3M	63	111
384 × 640		0.6718		121	222
320 × 512		0.6502		174	295
240 × 480		0.6279		254	451

4.3 Person segmentation and depth map filtering

There were 67096 training data and 3772 test data derived from the COCO dataset,[27] Supervisely dataset,[28] VOC2012 dataset,[29] and self-made dataset. The labels included background and person. The neural network architecture in this study and the focal loss were used for training. The results of training 600 Epochs are shown in Figs. 14–16, with the orange line being the training set and the blue line being the training result of the test set. The curves were smoothed.

The real-time segmentation effects of AGX and ZED are shown in Fig. 17. The scene was the laboratory, and the person was moving inside the laboratory. For better visualization effects, the background was removed. As the subsequent filtering function was contrary to visual SLAM, the target to be filtered was removed, leaving the static background only.

The neural network can be trained for different classification and segmentation effects. The filtering process is identical to that described in Sect. 3.3. By performing elementwise multiplication between the mask generated from semantic segmentation and the depth map, the final filtered image is obtained.

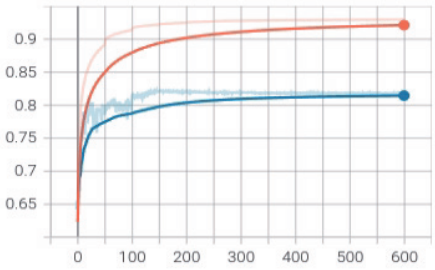


Fig. 14. (Color online) MIoU, optimal result training set: 0.9295 and test set: 0.8178.

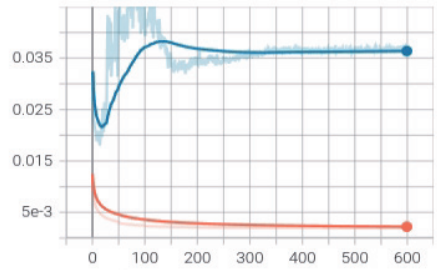


Fig. 15. (Color online) Focal loss, optimal result training set: 0.0019 and test set: 0.0374.

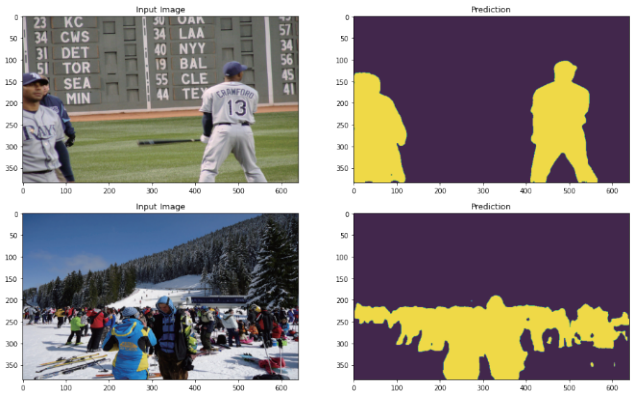


Fig. 16. (Color online) Test set segmentation effect.

4.4 Visual SLAM and dynamic person filtering

Finally, the filtered depth map was put in the visual SLAM. The keypoints filtering effect is shown in Fig. 18. The image was formed by fusing the input image with a depth map. The gray part represents depth 0. The circles representing the keypoint positions are calculated by the system. In the expansion of the dynamic person filtering mask image, the size of the mean filter is 25 × 25. Figures 18(a) and 18(b) show the conditions without and with filtering, respectively.

The actual test was performed according to location 1 in Fig. 19. The person moved in the laboratory in the same way. Moreover, the visual SLAM was used for mapping and the filtering effect was determined. The map constructed without filtering is shown in Fig. 20. The filtered image is shown in Fig. 21.

The actual measurement location 2 is shown in Fig. 22. The vehicle following the user was simulated in the corridor. There were consecutive figures before filtering, as shown in Fig. 23. In

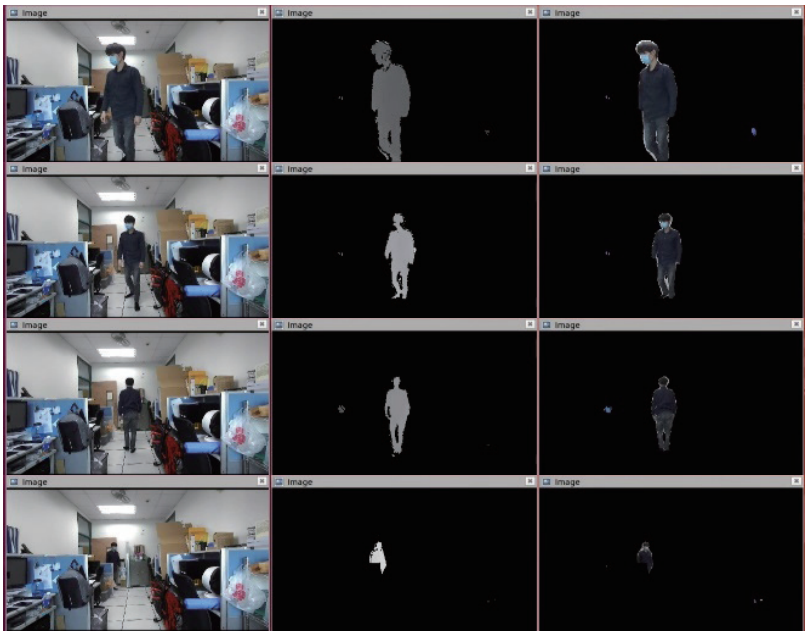
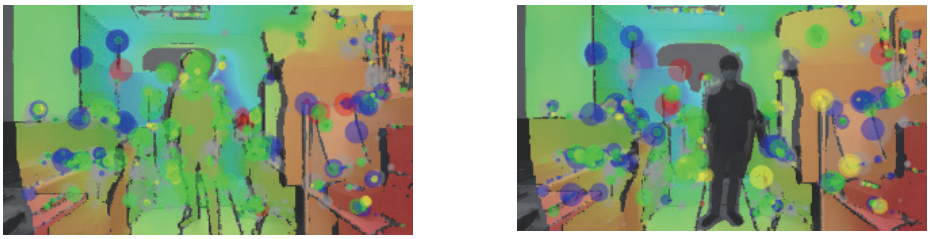


Fig. 17. (Color online) Real-time segmentation visualization effect of edge computing platform.



(a) (b)

Fig. 18. (Color online) Schematic diagram of keypoints. (a) Without filtering and (b) with filtering.

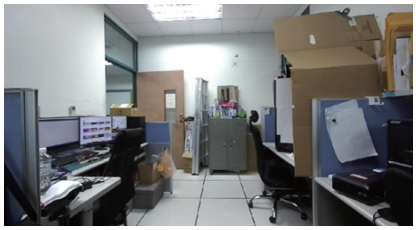


Fig. 19. (Color online) Photo of actual measurement location 1.

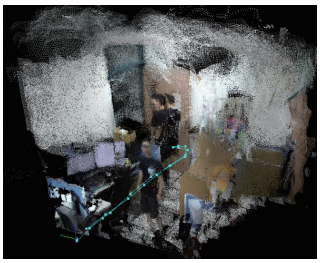


Fig. 20. (Color online) Mapping of actual measurement location 1 without filtering.

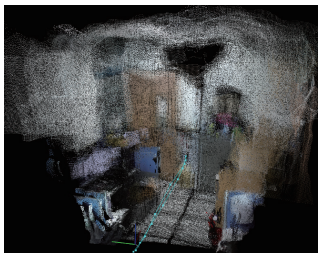
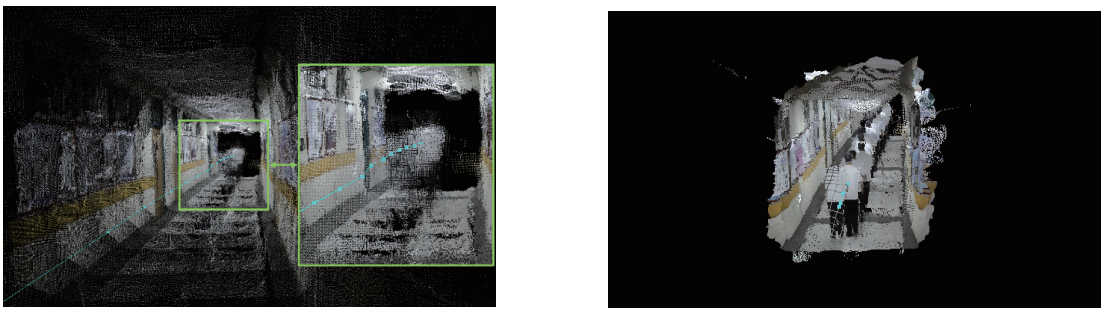


Fig. 21. (Color online) Mapping of actual measurement location 1 with filtering.

Fig. 22. (Color online) Schematic diagram of continuous movement of a person in front of the sensor.

the case of a short distance, the visual point cloud was diluted, as shown in Fig. 23(a). However, when the picture was zoomed out, the consecutive figures were more obvious, as shown in Fig. 23(b). When filtering was activated, even if there were persons remaining in front of the sensor, a clean static map could be constructed, as shown in Fig. 24.

The overall system operating speed is one of the requirements for mapping. To meet the requirement for speed, several frames may have a few speckles due to the poor effect of semantic segmentation. This is illustrated by the speckles on the arm and body edges in the green frame of Fig. 24. Because the quantity of speckles was small, the subsequent task would not be affected directly.



(a) (b)

Fig. 23. (Color online) Corridor following test before activation of the filtering function. (a) Consecutive figures and (b) visualization effect of zooming out.

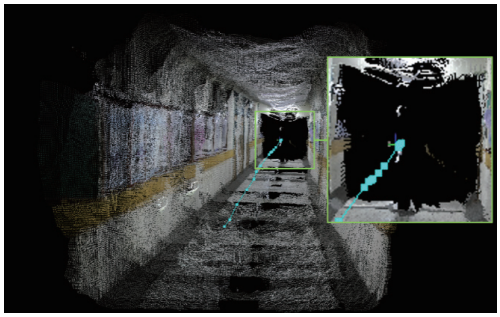


Fig. 24. (Color online) Corridor following test after filtering function is activated.

5. Conclusions

For the service of autonomous mobile vehicles using an edge computing platform, in this study, we completed a dynamic pedestrian filtering system of visual SLAM applicable to multi-person dynamic spaces. The dynamic targets could be filtered effectively in real time. The visual service vehicle could be effectively used in the living space.

The object filtering of visual SLAM was implemented by using a semantic segmentation neural network. A high-efficiency neural network was designed for edge computing platforms to achieve a real-time low-load computation capability. The filtering system filtered the depth image according to the prediction result of semantic segmentation, and relevant pixels or keypoints were removed from the visual SLAM.

Acknowledgments

This work was supported by the National Science and Technology Council, Taiwan, under contract number MOST 111-2221-E-167-031 (duration: August 1, 2022-July 31, 2023).

References

1 What Are Autonomous Robots? 8 Applications for Today's AMRs: https://locusrobotics.com/what-are-autonomous-robots/ (accessed June 2022).
2 E. A. Oyekanlu, A. C. Smith, W. P. Thomas, G. Mulroy, D. Hitesh, D. J. Kuhn, J. D. Mcghinnis, S. C. Buonavita, N. A. Looper, M. Ng, A. Ng'oma, W. Liu, P. G. Mcbride, M. G. Shultz, C. Cerasi, and D. Sun: IEEE Access **8** (2020) 202312. https://doi.org/10.1109/ACCESS.2020.3035729
3 F. Ingrand and M. Ghallab: Artif Intell. **247** (2017) 10. https://doi.org/10.1016/j.artint.2014.11.003
4 Automated Guided Vehicle: https://en.wikipedia.org/wiki/Automated_guided_vehicle (accessed June 2022).
5 M. O. Shneier and R. V. Bostelman: NISTIR **8022** (2015) 1. https://doi.org/10.6028/NIST.IR.8022
6 S. G. Tzafestas: J Intell Robotic Syst. **91** (2018) 35. https://doi.org/10.1007/s10846-018-0805-9
7 J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao: IEEE Trans Pattern Anal Mach Intell. **43** (2020) 3349. https://doi.org/10.48550/arXiv.1908.07919
8 N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni: IEEE Access **9** (2021) 82031. https://doi.org/10.1109/ACCESS.2021.3086020
9 MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications: https://arxiv.org/abs/1704.04861 (accessed June 2022).
10 M. Tan and Q. V. Le: 36th Int. Conf. Machine Learning (PMLR, 2019) 6105–6114.
11 NVIDIA Jetson AGX Xavier: https://www.nvidia.com/zh-tw/autonomous-machines/jetson-agx-xavier/ (accessed June 2022).
12 TensorRT SDK-NVIDIA Developer: https://developer.nvidia.com/tensorrt (accessed June 2022).
13 Waymo Driver: https://waymo.com/intl/zh-tw/waymo-driver/ (accessed June 2022).
14 I. S. Weon, S. G. Lee, and J.K. Ryu: IEEE Access **8** (2020) 65599. https://doi.org/10.1109/ACCESS.2020.2982681
15 K. Kim, J. Im, and G. Jee: IEEE Trans Intell Transport Syst. **23** (2022) 17575. https://doi.org/10.1109/TITS.2022.3160235
16 Transitioning to Tesla Vision: https://www.tesla.com/support/transitioning-tesla-vision (accessed June 2022).
17 RTAB-Map: http://introlab.github.io/rtabmap/ (accessed June 2022).
18 N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni: IEEE Access **9** (2021) 82031. https://doi.org/10.1109/ACCESS.2021.3086020
19 Q. Ha, K. Watanabe, T. Karasawa, Y. Ushiku, and T. Harada: 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS) (IEEE, 2017) 5108–5115.
20 C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang: Int. J. Comput. Vis. **129** (2021) 3051. https://doi.org/10.1007/s11263-021-01515-2.
21 M. Fan, S. Lai, J. Huang, X. Wie, Z. Chai, J. Luo, and X. Wei: 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR) (IEEE, 2021) 9711–9720.
22 L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam: Computer Vision-ECCV 2018 (ECCV, 2018) 83.
23 M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal: Deep Learning and Data Labeling for Medical Applications (DLMIA, 2016) 179.
24 J. Hu, L. Shen, and G. Sun: 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition (IEEE, 2018) 7132–7141.
25 Cityscapes Dataset: https://www.cityscapes-dataset.com/ (accessed June 2022).
26 T. Y. Lin, P. Goyal, R. Girshick, H. He, and P. Dollár: 2017 IEEE Int. Conf. Computer Vision (ICCV) (IEEE, 2017) 2999–3007.
27 COCO dataset: https://cocodataset.org/#home (accessed June 2022).
28 Dataset management in Supervisely: https://supervise.ly/data-and-users/data-management (accessed June 2022).
29 The PASCAL Visual Object Classes Challenge 2012 (VOC2012): http://host.robots.ox.ac.uk/pascal/VOC/voc2012/ (accessed June 2022).