IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 November 2022, accepted 25 November 2022, date of publication 6 December 2022, date of current version 9 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3227203

RESEARCH ARTICLE

Street Floor Segmentation for a Wheeled Mobile Robot

JUNHYUK HYUN, SUHAN WOO, AND EUNTAI KIM, (Member, IEEE)
School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea

Corresponding author: Euntai Kim (etkim@yonsei.ac.kr)

This work was supported by the Industry Core Technology Development Project, 20005062, Development of Artificial Intelligence Robot Autonomous Navigation Technology for Agile Movement in Crowded Space, funded by the Ministry of Trade, industry & Energy (MOTIE, Republic of Korea).

ABSTRACT In urban cities, the information about the type or class of street floors enables a wheeled mobile robot to perform many tasks ranging from traversability region identification, localization and the choice of wheel control strategy. In this paper, we considered a new task named as street floor segmentation (SFS) using an RGB camera. The SFS can be considered as the generalized problem of the existing traversability region identification problem in urban situations. Our SFS has two special classes for the possible application to the traversability region identification and they are traversable and non-traversable curbs. The SFS using an RGB camera is implemented using a real-time semantic segmentation (SS) network. A booster module named as Dynamic Context-based Refinement Module (DCRM) was developed to enhance the performance of the SFS. Our network was applied to real-world applications, and its validity is demonstrated through experiment.

INDEX TERMS Real-time, traversability, street floor, segmentation, curb, urban.

I. INTRODUCTION

When a wheeled mobile robot moves in urban cities, the information about the type or the class of the street floor on which the robot moves can be utilized in various ways for the robot's reliable and safe navigation. For example, when a wheeled delivery robot moves in a city, the robot should move only on a sidewalk and crosswalk; it should not move on a driving road. Thus, when it moves from the sidewalk to the grass, the control strategy of the wheels should be changed to ensure that the robot does not slip on the grass. Thus, although the recognition of the street floor type is quite important for robot navigation, it has rarely been studied and only the related problems have been considered. The typical example of the related problems will be to identify the traversable region using various sensors [1], [2]. However, traversable region identification is only a subset of street floor type recognition because if we know the type or the class of the street floor, we can easily identify the traversable region based on the class of the street floor on which a robot moves.

The associate editor coordinating the review of this manuscript and approving it for publication was Essam A. Rashed.

On the other hand, with the development of deep learning, numerous problems in robotics have been tackled using learning-based methods. For example, the classical loop closing is replaced with the learning-based image matching [3], [4], and the classical visual odometry (VO) is being combined with the learning-based VO [5], [6], [7].

In this paper, we considered a new task named as real-time *street floor segmentation*using an RGB camera. SFS is a new task that includes traversable region identification in urban situations as a subset, and produces a per-pixel street floor classification map as shown in Fig. 1.

Here, we use a real-time semantic segmentation (SS) deep learning network to realize SFS. The key feature of SFS is *that we consider two different types of curbs as different classes for the possible application to the traversable region identification* for a wheeled robot. Specifically, curbs can be divided into traversable curb and non-traversable curb in Korea, as shown in Fig. 2. Understandably, distinguishing these two types of curbs is quite critical for safe robot navigation. For example, if a wheeled robot tries to move up a non-traversable curb, the robot is likely to collide with the curb. Conversely, if the wheeled robot tries to move down

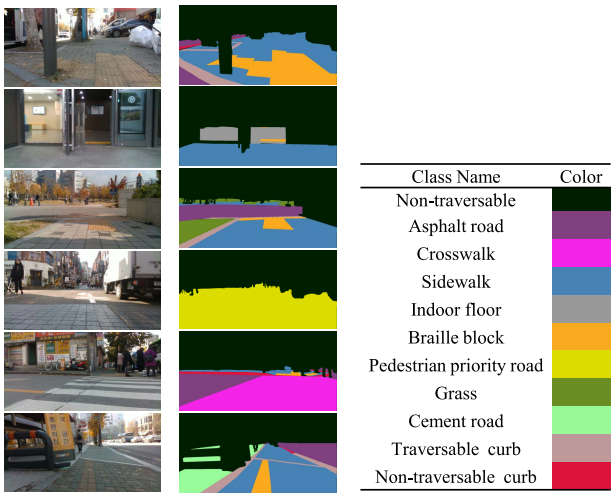


FIGURE 1. Examples of the dataset used in street floor segmentation.

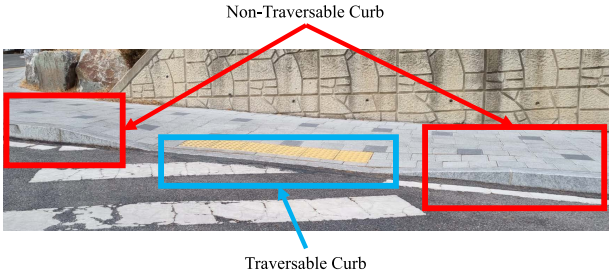


FIGURE 2. Two types of curbs. A wheeled robot can pass the only traversable curbs safely.

a non-traversable curb, the robot is likely to be overturned. To distinguish between the two types of curbs, we can use an additional range sensor such as a LIDAR because two types of curbs have different elevation, but the LIDAR requires the high cost. Some may think that cheap sonar or IR sensors can be used for curb detection. But it is quite difficult to detect curb using sonar or IR sensors, and we have to use multiple sonar sensors [8]. In addition, because the sonar and IR sensors measure semantic information, it should be used together with an RGB camera to fully understand the street floor.

Here, we considered the SFS for a wheeled robot using an RGB camera. One might think that simple supervised learning (=semantic segmentation) will distinguish these two types of curbs, but this is not enough because the two types of curbs look quite similar. To resolve the ambiguity between the two types of curbs, we have to use not only the shape of the curbs but also the context information around the curb. Specifically, the two types of curbs look similar, but they are used under different contexts: Non-traversable curbs are always alone, whereas traversable curbs are always with braille blocks made for blind persons, as shown in the figure. Thus, the problem that we have to tackled in this paper was formulated as follows:

1) SFS should be conducted to assign a class to every pixel, when an RGB image is presented.
2) The classes used in the SFS include the traversable and non-traversable curbs.
3) The network should run fast enough for application to the wheeled mobile robot.

To realize our idea, we proposed a new module named Dynamic Context-based Refinement Module (DCRM) as depicted in Fig. 3. DCRM was applied after the SS network. The preceding SS network used the shape of an image, while the following DCRM focuses on using the context information to realize SFS. The DCRM consists of two parts: Pooling Up-sampling Dynamic Filter (PUDF) and adjustment score normalization (ASN). The PUDF is a dynamic filter that uses the context information of a given pixel. The ASN is a normalization module that stabilizes the function of PUDF by forcing the sum of the adjusting scores to become zero.

The contribution of this paper is twofold:

1) We developed a new task named SFS. The problem SFS considered in this paper is a subset of semantic segmentation (SS) problem, but SFS has two key features. (i) The receptive field of SFS is wider than the that of SS, which will be explained in Section III.B. (ii) The distinction between traversable and non-traversable curbs is very important because the autonomous mobile robot should traverse the curb in the urban street. The problem SFS is also a superset of traversable region identification problem. To our best knowledge, only some studies regarding traversable region identification have been conducted, and none of the study fully consider the characteristics of traversability in urban situations (traversability of curbs). Since SFS is a larger problem than traversable regions detection, the classes of our SFS should include two types of curbs, traversable curb and non-traversable curb.
2) We proposed a new module named DCRM to tackle the SFS. Our module DCRM exploits the context information of pixels. The module aims at improving the segmentation accuracy in real time by looking wide and using the context information around a given pixel. When DCRM is combined with the previous real-time SS networks such as BiSeNet [9], SwiftNet [10], and AFPNet [11], the accuracy is improved. Our SFS network which consists of SS network and DCRM, as shown in Fig. 3, is end-to-end trainable.

The remaining part of the paper is organized as follows: In Section II, related works are discussed. In Section III, the problem SFS considered herein is formulated and the details of our network are explained. Experimental setup and results are presented in Section IV. Finally, some conclusions are drawn in Section V.

II. RELATED WORK

A. TRAVERSABLE REGION IDENTIFICATION USING AN RGB CAMERA

Classical traversable region identification methods [1], [2] mainly use a range sensor such as 3D LIDAR to measure the elevation of the floor. However, they have the shortcomings

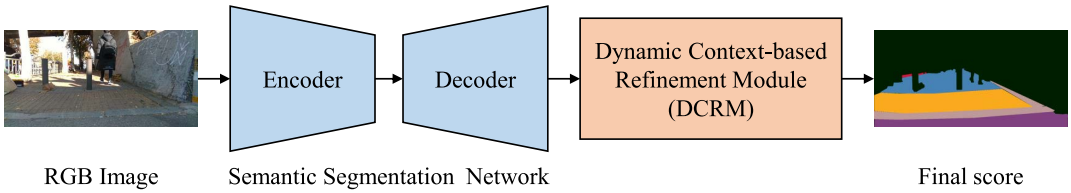


FIGURE 3. Outline of the street floor segmentation network. Our network consists of a semantic segmentation network and DCRM. DCRM refines the street floor segmentation score.

that sematic information of the road is not fully exploited, and the range sensor used in the identification is relatively expensive. Understandably, the traversable region identification using an RGB camera is more attractive than using a range sensor.

In [12], GONet was trained with many positive images of traversable places, but just a small set of negative images depicting blocked and unsafe areas. Positive examples were collected simply by operating a robot through traversable spaces. The GONet used Generative Adversarial Networks (GANs) to predict the traversablity of the given road. In [13], VUNet synthesizes future images for given virtual robot velocity commands using only RGB images at previous and current time steps. Then, VUNet predicts the traversability of the image by applying the synthesized future images to GONet [12]. In [14], the traversability of the area surrounding a mobile robot was estimated by dividing the image into several bins and training the deep learning network to learn the reachable area from each bin. This paper demonstrates good performance when it is applied to mountains, plains, or at least on country roads. However, it cannot be applied to urban streets because it does not consider semantic information. The paper [14] focuses only on the physical traversability without considering semantics of the road. In [15], a semantic segmentation network was used to classify the ground, stairs, slopes of an indoor and it was combined with SLAM, making semantic SLAM. Further, the segmentation result was refined by fusing the semantic segmentation result and the depth. But it cannot be applied in real-time outdoors or in urban situations. This is because range sensors will have more data in a large space. In [16] and [17], the traversability of the road was estimated using semantic segmentation in urban situations. By training the deep learning network, the traversability was estimated from various cameras. However, [15], [16], and [17] did not fully consider the characteristics of traversability in urban situations (traversability of curbs) and the deep learning network structure suitable for traversability estimation.

B. DEEP LEARNING AND REAL-TIME SEMANTIC SEGMENTATION

Deep learning [18], [19], [20] is demonstrating excellent performance in many visual recognition tasks such as image level classification [21], object detection [22], and SS [23]. Among them, SS is a pixel-level classification problem. In order to use SS in mobile robotics, the networks should run in real-time. However, most of popular general SS networks cannot run in real-time due to heavy computation load.

Recently, several real-time SS networks have been proposed. In particular, BiSeNet [9], SwiftNet [10], and AFPNet [11] achieved high performance and high speed among real-time semantic segmentation networks. As the name suggests, BiSeNet uses two paths to extract spatial information and context information, separately. SwiftNet is famous for its simple and high-performance decoder structure, and it also achieves high SS performance and fast processing time. AFPNet proposes the local memory module to up-sample the SS results effectively.

C. DYNAMIC FILTER

Dynamic filter [24] is a content-adaptive convolution filter; and its weights are dynamically generated through a filter generating network conditioned on input images. Here, the key difference between a conventional static filter and a dynamic filter is that the same convolution weights are applied to all the input images in the static filter, whereas the convolution weights change depending on the input image in the dynamic filter. Thus, the dynamic filter enables us to handle a variety of input images in a single and unified framework. The dynamic filter consists of a filter-generating network and a dynamic filtering layer. The dynamic filter achieves high performance in several tasks. For example, high performance is achieved in object detection [25] and semantic segmentation [26] by applying a dynamic filter. But the dynamic filter has the shortcoming that dynamic filter is weak in generalization and has poor robustness [27], [28]. Because of this, dynamic filter sometimes diverges. If divergence occurs, the output from the dynamic model can be extremely large for some inputs.

III. PROPOSED METHOD

A. PROBLEM FORMULATION

Let us suppose that RGB images $\mathcal{I} \in \mathbb{R}^{H\times W\times 3}$ and their ground truths (GTs) annotated at the pixel level as $\mathcal{G}(\mathcal{I}) \in \{0, 1\}^{H\times W\times C}$ are presented, where C denotes the number of the SFS classes. The set of SFS classes are $C_{SFS} = \{$Non-traversable, Asphalt road, Crosswalk, Sidewalk, Indoor floor, Braille block, Pedestrian priority road, Grass, Cement road, Traversable curb, Non-traversable curb$\}$ and their examples are given in Fig. 1, where $C = |C_{SFS}| = 11$. The goal of our

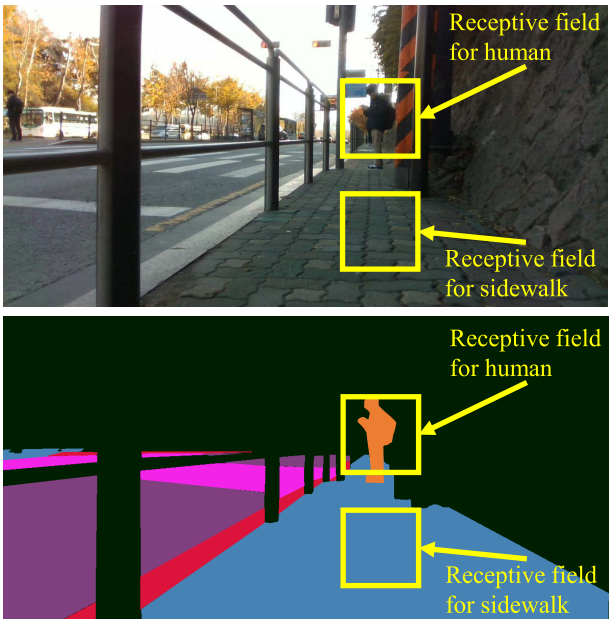


FIGURE 4. Key difference between SFS and the general SS. An instance in SFS is usually much larger than that in SS, so SFS needs larger receptive field than SS.

paper was to train our network $\mathcal{N}(\cdot)$ in end-to-end way so that the loss between the output of the network $\mathcal{N}(\mathcal{I})$ and its GT $\mathcal{G}(\mathcal{I})$ is minimized. As said, the key feature of our problem formulation is that the traversable and non-traversable curbs should be considered as different classes even though they look quite similar.

B. WHY DYNAMIC CONTEXT-BASED REFINEMENT MODULE (DCRM)?

The only difference between our SFS network and existing real-time SS networks is DCRM. That is, SFS network = SS network + DCRM. One might think why we have to use DCRM in SFS. The key difference between SFS and general SS is that an instance in SFS is usually much larger than the one in SS. For example, let us consider an instance labeled as a sidewalk and an instance which might be labeled as a human in the SS, as shown in Fig. 4.

If the receptive field of the convolution filter is as large as indicated using yellow boxes, the convolution filter applied to the instance of the sidewalk does not include context information, whereas the filter applied to the instance of a human includes much context information. This implies that we have to look wider (=widen the receptive field) in SFS than in SS. Obviously, we can widen the receptive field by deepening the network, but we cannot do it because of the real-time requirement. Thus, we developed a shallow module DCRM, which aims to widen the receptive field while spending minimal extra computation.

To increase the performance by adding modules, we need to design the suitable structure for the problem. Simply adding a structure to a network does not always improve the performance [20] and increasing the performance with less computation is even more challenging. We developed a structure that solves the SFS problem well by considering the characteristics of the SFS problem. The SFS problem requires a wide receptive field and can improve recognition performance by adaptively considering the surrounding semantic information. For the design of an adaptive structure, we used a dynamic filter.

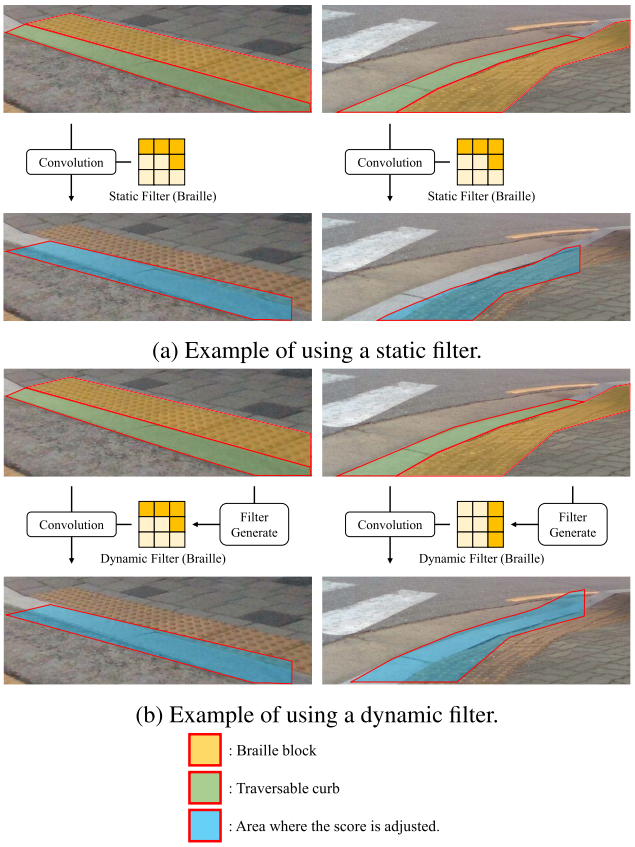
C. WHY DYNAMIC FILTER IN DCRM?

In this subsection, we explain why we use not a normal (=static) convolutional filter but a dynamic convolutional filter [24] in DCRM. As shown in Fig. 3, our DCRM takes the pixel-wise class score map from SS network and outputs its refined version. Fig. 5 shows why we need a dynamic filter. First, let us consider Fig. 5a. In the figures on the left column, a yellow braille block is located on the upper right side of a traversable curb (=colored in green). DCRM can increase (=refine) the score of the traversable curb by considering the braille block located on the upper right side of a traversable curb, or by assigning high values to the upper right weights in the convolution filter. Here, dark yellow denotes the high weights, whereas light yellow denotes the low weights in a convolution filter. If a static filter is used, the same filter should be applied to other situations such as the figures on the right column of Fig. 5a. Here, a yellow braille block is located on the lower right side of a traversable curb (=colored in green). If the same filter which has the high values in the upper right weights as shown in Fig. 5a is applied to the figures on the right column, not the braille block but the road which is located on the upper right side of the traversable curb will affect the traversable curb, degrading the segmentation performance.

To solve the above problem, we have to change the filter weights depending on the input image. That is the reason why we use a dynamic filter in DCRM. The key feature of the dynamic filter is that the filter weights are generated dynamically conditioned on the input image. Let us consider Fig. 5b. The filter on the left column should be generated such that the upper right weights have the high values, whereas the filter on the right column should be generated such that the lower right weights have the high values to refine the score map. Understandably, the reason is that a yellow braille block is located on the upper right side of a traversable curb on the right column, but a yellow braille block is located on the lower right side of the traversable curb in the right column.

D. DYNAMIC CONTEXT-BASED REFINEMENT MODULE

The architecture of DCRM is given in Fig. 6. As shown in the figure, it has three paths from input to output. The top path is a filter generating network of the dynamic filter, and it outputs the filter weight. The filter generating network CONV_1 was implemented using very shallow convolution networks. When a per-pixel classification result $\mathcal{N}_{SS}(\mathcal{I})$ returned from the SS network is presented, the tensor $\mathcal{N}_{SS}(\mathcal{I})$ goes through a shallow network CONV_1 on the top path, and the shallow



(a) Example of using a static filter.

(b) Example of using a dynamic filter.

FIGURE 5. Static filter vs. dynamic filter. The dynamic filter enables content-adaptive learning.

network outputs a pixel-wise filter weight $\mathcal{W}$ with a size of $H \times W \times C \times C$, where $H \times W$ denotes the size of the output of the SS network. Thus, for each pixel $(h, w) \in H \times W$, the filter generating network CONV_1 outputs a fully connected filter with the size of $C \times C$, where, $C \times C$ means the score refinement from C classes to C classes. Obviously, the fully connected filter was implemented using 1×1 convolution.

The path in the middle is actually a main convolution filter and its weights are provided by the filter generating network on the top path. The key feature of the path is pooling followed by upsampling: Pooling downsamples the input $H \times W$ into $\frac{H}{S} \times \frac{W}{S}$ by max pooling and upsampling restores the reduced tensor $\frac{H}{S} \times \frac{W}{S}$ to the original size $H \times W$, when S is size and stride of pooling. One might think that pooling (=downsampling) followed by upsampling means "doing nothing" and why we have to do null operations. The reason for them is that they widen the receptive field, enabling DCRM to use the context information from the nearby pixels. One also might think that one can use a deep convolution layer to widen the receptive field, but it is not suitable for real-time applications. The goal of our pooling followed by upsampling is to widen the receptive field while consuming minimal additional time. The top two paths come from the dynamic filter and this part is named as Pooling Up-sampling Dynamic Filter (PUDF)

The path at the bottom is the skip connection (=identity mapping) and it is motivated by the residual module in ResNet [20]. The path ensures at least the performance obtained when the DCRM is not used at all.

At the end of the path in the middle, adjustment score normalization (ASN) is added to prevent PUDF from changing the sum of total score. Specifically, it is known that the dynamic model has a risk of divergence when the generalization performance is degraded [27], [28]. If divergence occurs, the output from the dynamic model can be extremely large for some inputs. PUDF is also the case. To solve the problem, we used an adjustment score normalization defined by

$$\sigma(T)_{h,w,c} = T_{h,w,c} - \frac{1}{C}\sum_{c=1}^{C} T_{h,w,c} \tag{1}$$

where $\sigma(\cdot)$ is output from the ASN, T is the output of PUDF, C is the number of types of floors, and h, w, c is the vertical, horizontal, and depth position of the score, respectively. The ASN forces the sum of the adjustment scores to become zero and prevents PUDF from changing the sum of the total score

IV. EXPERIMENT

A. DATASET

To implement our SFS, we built our dataset using our wheeled robot platform. Our wheeled robot platform is a Jackal UGV robot and Intel RealSense Depth Camera D435 is installed on the robot as shown in Fig. 7. Using the platform, we collected 1,251 training and 355 validation images, respectively. As explained in Section III-A, 11 classes were defined, and the collected images are annotated manually at the pixel level. Examples of the collected images and their GT images are given in Fig. 1. Each image has a size of $1{,}280 \times 640$.

B. TRAINING DETAILS

Our SFS network consists of an SS network and DCRM. We use three popular real-time SS networks for the application to a wheeled mobile robot, and they are BiSeNet [9] SwiftNet [10], AFPNet [11]. The three SS networks are initially pretrained on ImageNet [21]. Our SFS network consisting of SS network and DCRM is trained end-to-end. We use 20 as the value of S which is size and stride of pooling in PUDF. We train our SFS networks on the training set in Section IV.1 using the following settings: online hard example mining, the auxiliary loss function, a polynomial schedule with learning rate of 0.01, weight decay of 0.0005, and coefficient of 0.9. Training images were cropped to 640×640 with a scale range [0.5 2.0] and horizontal flipping with a probability of 0.5 was applied to the training images. Our SFS networks were trained using 12 mini-batches for 50,000 epochs, with the first 1,000 epochs reserved for warning-up. In all experiments, we performed five independent runs and compute the average of mIoUs over the five runs. Here, we used a single TitanXP with CUDA 11.0, CUDNN 8.0, and PyTorch 1.4.0 for training and evaluation.

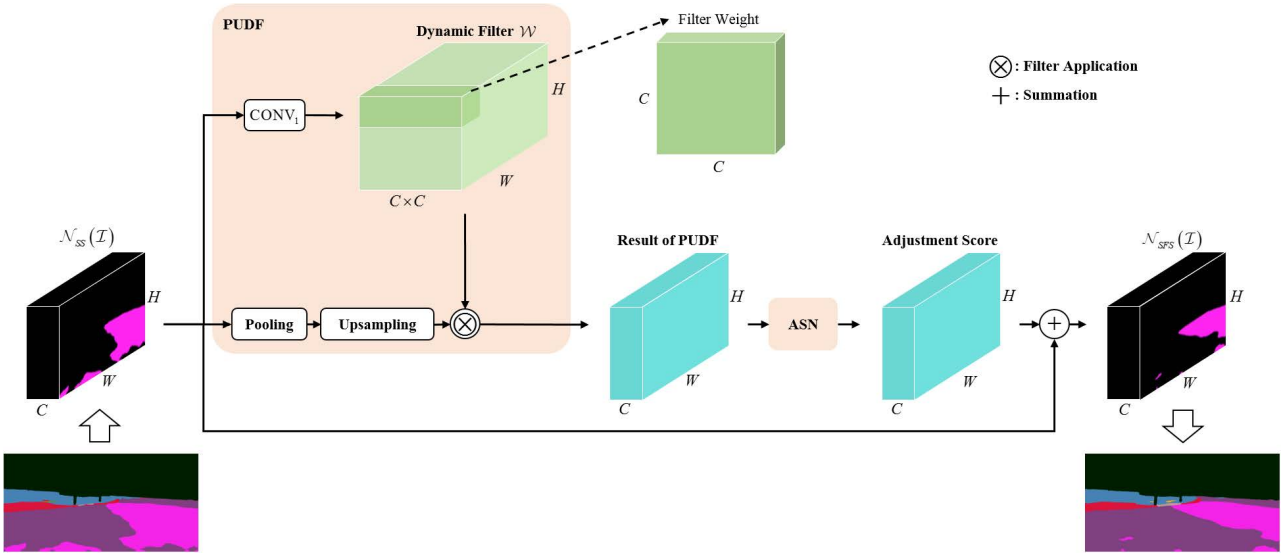


FIGURE 6. Architecture of the DCRM. The DCRM consists of Pooling Up-sampling Dynamic Filter (PUDF) and Adjustment Score Normalization (ASN). ⊗ is filter application operation. The DCRM adjusts scores using nearby semantic information.

FIGURE 7. Robot and camera.

C. DCRM

To see the effectiveness of the DCRM, we compared the SFS performance in two cases of using DCRM and not using DCRM. The SFS performances measured in mIOU are summarized in Table 1 for the two cases. The results in the table clearly show that the SFS performance was improved for the three different SS networks when the proposed DCRM was used after the SS networks. Specifically, the accuracy of cement road increased significantly, and that for the two types of curbs are also increased in all networks. The reason for it might be that the traversable curbs were always next to braille blocks, as explained in Section III. Further, as the instances of cement roads usually lack local features, so large receptive field of DCRM might help to recognize the cement roads. Some classes had the mixed results. That is, the accuracy of the remaining classes is increased by DCRM in some networks, whereas the accuracy was decreased after applying DCRM in the other network. For example, the IoU of crosswalk is increased in BiSeNet and AFPNet, whereas it was decreased in SwiftNet. Overall, however, the number of classes with increased performance was larger than that of the classes with decreased performance. Further, the amount of increase was higher than that of the decrease, increasing the overall accuracy of SFS networks.

In addition, since the additional computation of the DCRM is small, our SFS networks also run in real-time even if the DCRM is added. The exact run speed of our SFS network is given in the third row of Table 1.

Examples of our SFS are given in Fig. 8. From the figure, we can see the DCRM worked well for the curbs and the instances which need wide receptive field. As explained, DCRM has two features: (1) It leverages the context information and (2) it widens the receptive field. Understandably, the two types of curbs have different context neighbors, and their accuracy were improved by DCRM, as shown in the first two rows of Fig. 8. Further, instances of Asphalt road, Crosswalk, Sidewalk need wide receptive field and their accuracy is also enhanced by DCRM, as shown in the last three rows of Fig. 8. A full video clip in which our SFS network is applied to the test set is also submitted as a supplemental file.

D. ABLATION STUDY

In this subsection, we conducted ablation experiments to see the effects of the two internal operations within the proposed DCRM: PUDF and ASN. The results of the ablation study are summarized in Table 2. First, let us considered the running speed of our SFS network including the DCRM, which is given in the rightmost column of the table. When BiseNet was used as an SS network, its FPS is 65.80 on Titan XP. When our DCRM was added to the BiseNet, the FPS is slightly decreased from 65.80 to 61.33 on Titan XP, and this implies that our DCRM improved the performance while consuming minimal computation. In table 2, the first row is the vanilla BiSeNet and it is used as a baseline. As said before, when DCRM was added to the BiseNet, the accuracy was improved by almost 0.8% while consuming minimal time. To see the effects of the two internal operations within the DCRM, we removed them one by one. First, we replaced the

TABLE 1. Experiment results of our SFS networks.

	Network	SwiftNet		BiSeNet		AFPNet	
		Vanilla	with DCSRM	Vanilla	with DCSRM	Vanilla	with DCSRM
	mIoU (%)	71.68	**72.31**	72.42	**73.18**	71.93	**73.07**
	FPS	**75.55**	70.93	**65.80**	61.33	**72.89**	67.90
IoU (%)	Non-traversable	97.25	**97.29**	97.20	**97.20**	97.17	**97.29**
	Asphalt road	**81.20**	81.04	81.59	**81.72**	**81.52**	82.24
	Crosswalk	**75.31**	75.16	74.86	**76.22**	75.19	**76.02**
	Block sidewalk	89.44	**89.69**	89.00	**89.39**	89.59	**89.97**
	Indoor floor	86.24	**87.40**	91.88	**92.45**	89.40	**90.15**
	Braille block	82.24	**82.46**	81.18	**81.87**	**81.56**	81.48
	Pedestrian priority road	**81.27**	80.05	82.37	**82.87**	81.35	**82.20**
	Grass	**61.67**	61.47	**64.60**	64.10	61.21	**62.90**
	Cement road	37.72	**44.04**	41.08	**44.17**	40.28	**45.33**
	Traversable curb	47.16	**47.50**	46.17	**48.07**	46.47	**47.45**
	Non-traversable curb	48.97	**49.35**	46.66	**46.94**	47.49	**48.75**

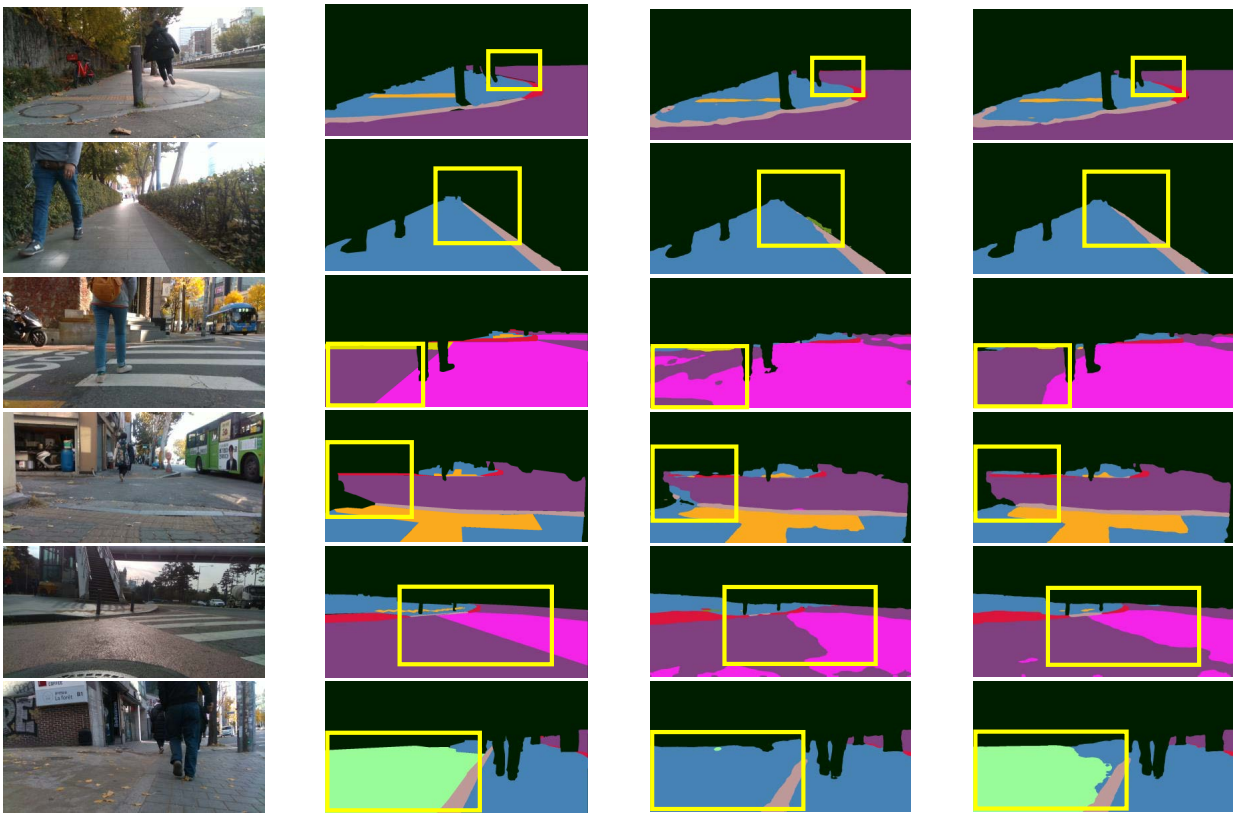


FIGURE 8. SFS results using BiSeNet and proposed DCRM. (a) Image, (b) Ground Truth, (c) Output of BiSeNet, (d) Output of BiSeNet+DCRM. The yellow boxes indicate the regions in which the DCRM improved the SFS.

PUDF with a simple convolution. When PUDF was not used, the accuracy decreased. Second, we removed the ASN in the DCRM. In that case, the accuracy also decreased. From the ablation study, we can see that our DCRM worked effectively when the two internal modules were used together.

E. COMPARISON WITH OTHER MODELS

To show the validity of the proposed DCRM in the real-time SFS problem, we compare our network with the previous popular SS networks. The compared networks are SegNet [29], Inception-v3 [15], [30] and Seq ERF-PSPNet [16], [17]. SegNet is one of the most well-known SS network; Inception-v3 [15], [30] and Seq ERF-PSPNet [16], [17] are the recently reported networks which deal with the traversable regions detection. In the experiment, our method used BiSeNet [9] as a backbone, and we added DCRM module. Experimental results are given in Table 3

From the table, we can see that our model achieves the best accuracy among all the compared models. In particular, our model demonstrates significant improvement in traversable

TABLE 2. Experiment results of our SFS networks.

	Module for extract adjustment score	ASN	mIoU (%)	FPS
BiSeNet	-	-	72.42	**65.80**
BiSeNet + DCRM	PUDF	w. ASN	**73.18**	61.33
BiSeNet + Refinement module with static filter	Static filter	w. ASN	73.02	62.40
BiSeNet + PUDF without ASN	PUDF	w.o. ASN	73.01	62.30

TABLE 3. Comparison of the proposed network with SegNet, Inception-v3, and Seq ERF-PSPNet.

Network		Proposed Method (w/ BiSeNet)	SegNet [29]	Inception-v3 used in [15], [30]	Seq ERF-PSPNet used in [16], [17]
	mIoU (%)	**73.18**	58.33	63.72	64.61
	FPS	61.33	11.24	35.98	**109.14**
IoU (%)	Non-traversable	**97.20**	96.36	95.22	96.20
	Asphalt road	**81.72**	69.52	76.68	75.49
	Crosswalk	**76.22**	42.53	70.00	72.93
	Block sidewalk	**89.39**	81.49	85.05	85.63
	Indoor floor	**92.45**	58.87	83.54	84.91
	Braille block	**81.87**	78.99	65.95	75.11
	Pedestrian priority road	**82.87**	68.82	82.16	78.26
	Grass	**64.10**	55.41	50.52	52.07
	Cement road	**44.17**	12.65	35.74	22.07
	Traversable curb	**48.07**	39.97	26.71	33.81
	Non-traversable curb	**46.94**	37.00	29.31	34.24

TABLE 4. Network performance when hyper-parameters are changed.

Parameter		mIoU (%)
Initial learning rate	1.00E-01	71.06
	1.00E-02	**73.18**
	1.00E-03	71.41
Weight decay	5.00E-05	69.76
	5.00E-04	**73.18**
	5.00E-03	72.87
Power of learning rate decay	0.85	72.8
	0.9	**73.18**
	0.95	72.78

and non-traversable curbs. The proposed model is not the fastest among the compared models, but it is fast enough for real-time SFS.

F. HYPER-PARAMETERS

In the additional experiments, we repeated the same experiment as in Section IV-B while varying three parameters in Table 4. The three parameters are an initial learning rate η, weight decay λ used in L2 regularization and the power p of a polynomial function which controls the decay of the learning rate. First, let us consider the initial learning rate η. We conducted the same experiments as in Section IV-B but changed the initial learning rate η from 0.001 to 0.1 by increasing one order of magnitude. The results are given below.

The best mIoU is achieved when the initial learning rate is 0.01. It seems that too small learning rate makes the training much delayed and take long time. It is also likely that the too small learning rate increases the possibility that the training is trapped in local minima. On the contrary, too large learning rate prevents us from exploiting the optimal solution around near-optimal solutions, degrading the performance.

In the same way, we repeated the same experiment while varying weight decay λ from 5×10^{-5} to 5×10^{-3} by increasing one order of magnitude and varying the power p of a polynomial function from 0.85 to 0.95 in increments of 0.05. When we use $\eta = 0.01$, $\lambda = 5 \times 10^{-4}$ and $p = 0.90$, we achieved the best mIoU.

V. CONCLUSION

In this paper, we developed a new problem named SFS and proposed a solution to the problem using deep learning networks. SFS is motivated by SS and can be applied to various mobile robot applications including traversable region identification. To the end, the classes of our SFS include traversable and non- traversable curbs, separately. Our solution to this problem consisted of an SS network and DCRM. The DCRM exploited the context information of the pixels to improve SFS performance. Our networks were applied to real world applications and our networks demonstrated the attractive results.

REFERENCES

[1] J. Sock, J. Kim, J. Min, and K. Kwak, "Probabilistic traversability map generation using 3D-LIDAR and camera," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 5631–5637.

[2] Y. Zhao, P. Liu, W. Xue, R. Miao, Z. Gong, and R. Ying, "Semantic probabilistic traversable map generation for robot path planning," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2019, pp. 2576–2582.

[3] S. Arshad and G.-W. Kim, "Role of deep learning in loop closure detection for visual and LiDAR SLAM: A survey," *Sensors*, vol. 21, no. 4, p. 1243, Feb. 2021.

[4] W. Chen, Y. Liu, W. Wang, E. Bakker, T. Georgiou, P. Fieguth, L. Liu, and M. S. Lew, "Deep learning for instance retrieval: A survey," 2021, *arXiv:2101.11282*.
[5] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2043–2050.
[6] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. M. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 340–349.
[7] S. Li, X. Wu, Y. Cao, and H. Zha, "Generalizing to the open world: Deep visual odometry with online adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13184–13193.
[8] J. Rhee and J. Seo, "Low-cost curb detection and localization system using multiple ultrasonic sensors," *Sensors*, vol. 19, no. 6, p. 1389, Mar. 2019.
[9] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 325–341.
[10] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained ImageNet architectures for real-time semantic segmentation of road-driving images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12607–12616.
[11] J. Hyun, H. Seong, S. Kim, and E. Kim, "Adjacent feature propagation network (AFPNet) for real-time semantic segmentation," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 9, pp. 5877–5888, Sep. 2022.
[12] N. Hirose, A. Sadeghian, M. Vazquez, P. Goebel, and S. Savarese, "GONet: A semi-supervised deep learning approach for traversability estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3044–3051.
[13] N. Hirose, A. Sadeghian, F. Xia, R. Martin-Martin, and S. Savarese, "VUNet: Dynamic scene view synthesis for traversability estimation using an RGB camera," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2062–2069, Apr. 2019.
[14] S. Palazzo, D. C. Guastella, L. Cantelli, P. Spadaro, F. Rundo, G. Muscato, D. Giordano, and C. Spampinato, "Domain adaptation for outdoor robot traversability estimation from RGB data with safety-preserving loss," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10014–10021.
[15] W. Deng, K. Huang, X. Chen, Z. Zhou, C. Shi, R. Guo, and H. Zhang, "Semantic RGB-D SLAM for rescue robot navigation," *IEEE Access*, vol. 8, pp. 221320–221329, 2020.
[16] K. Yang, L. M. Bergasa, E. Romera, and K. Wang, "Robustifying semantic cognition of traversability across wearable RGB-depth cameras," *Appl. Opt.*, vol. 58, no. 12, pp. 3141–3155, 2019.
[17] K. Yang, L. M. Bergasa, E. Romera, D. Sun, K. Wang, and R. Barea, "Semantic perception of curbs beyond traversability for real-world navigation assistance systems," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Sep. 2018, pp. 1–7.
[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.
[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and A. C. Berg, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.
[23] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.
[24] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.
[25] C. Chen and Q. Ling, "Adaptive convolution for object detection," *IEEE Trans. Multimedia*, vol. 21, no. 12, pp. 3205–3217, Dec. 2019.
[26] J. He, Z. Deng, and Y. Qiao, "Dynamic multi-scale filters for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3562–3572.
[27] J. Zhou, V. Jampani, Z. Pi, Q. Liu, and M.-H. Yang, "Decoupled dynamic filter networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6647–6656.
[28] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7436–7456, Nov. 2022.
[29] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Jan. 2017.
[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

JUNHYUK HYUN received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2014, where he is currently pursuing the combined master's and doctoral degrees. He has studied computer vision, machine learning, and deep learning.

SUHAN WOO received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2017, where he is currently pursuing the master's and doctoral degrees. He has studied computer vision, machine learning, and deep learning.

EUNTAI KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Yonsei University, South Korea, in 1992, 1994, and 1999, respectively. From 1999 to 2002, he was a Full-Time Lecturer at the Department of Control and Instrumentation Engineering, Hankyong National University, South Korea. Since 2002, he has been a Faculty Member with the School of Electrical and Electronic Engineering, Yonsei University, where he is currently a Professor. He was a Visiting Researcher with the Berkeley Initiative in Soft Computing, University of California, USA, in 2008. He was also a Visiting Researcher with Korea Institute of Science and Technology (KIST), Korea, in 2018. His current research interests include computational intelligence, statistical machine learning and deep learning and their application to intelligent robotics, autonomous vehicles, and robot vision.