Sensors and Materials, Vol. 37, No. 5 (2025) 2049–2060
MYU Tokyo

S & M 4041

Heterogeneous Sensor Fusion for Obstacle Localization in Mobile Robot Navigation

Chung-Hsun Sun,[1] Ying-Shu Chuang,[1] Yao-Yu Tsai,[2] and Hsiang-Chieh Chen[3*]

[1]Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung City 807618, Taiwan
[2]Department of Electrical Engineering, National United University, Miaoli County 360302, Taiwan
[3]Department of Mechanical Engineering, National Central University, Taoyuan City 320317, Taiwan

(Received November 11, 2024; accepted April 22, 2025)

Keywords: heterogeneous sensor fusion, mobile robot, monocular depth estimation, semantic segmentation

In this study, we focus on the application of heterogeneous sensor fusion technology, which involves the integration of a scanning laser rangefinder (LRF) and a low-cost camera, to detect obstacles in front of a robot. This technology enables real-time obstacle identification and path replanning, allowing the robot to navigate through unknown and cluttered environments efficiently. In indoor navigation, the initial map necessary for path planning is first created using a method related to simultaneous localization and mapping. The two-dimensional location of existing or newly emerged obstacles can be determined by comparing the sensing results of the LRF with the initial map. Furthermore, analyzing captured images using semantic segmentation and monocular depth estimation provides 3D information in the vicinity of a robot. In our work, we leverage the accuracy of an LRF and the high resolution of a visual sensor to effectively integrate heterogeneous sensors, enabling the robot to sense and avoid obstacles. Experimental findings have demonstrated the effectiveness of indoor navigation for a mobile robot in unexplored environments using our proposed sensor fusion technology.

1. Introduction

In recent years, there has been rapid advancement in the sensor industry and computation acceleration, leading to a significant increase in the number of research papers focused on autonomous mobile robots (AMRs). AMRs are typically equipped to perform functions such as positioning, path planning, and movement control. Path planning involves two primary types: creating the initial path based on environmental information and adapting to dynamic obstacles that may appear during navigation. Dynamic obstacles refer to objects that were not originally part of the environmental data and have the potential to obstruct the navigation path of a robot. Addressing the complex issue of detecting and obtaining sufficient information about these obstacles to enable the robot to alter its path is a significant challenge that needs attention.

*Corresponding author: e-mail: hcchen@cc.ncu.edu.tw
https://doi.org/10.18494/SAM5472

ISSN 0914-4935
https://myukk.org/

A laser rangefinder (LRF) is a commonly used sensor in robotic systems to obtain depth information about the surroundings of a robot. Most affordable LRF devices utilize 2D light detection and ranging (LiDAR) scanning, which involves a rotating mechanism to gather environmental data at a fixed height. However, these devices have limitations in perceiving the environment in 3D, potentially leading to collisions with undetected obstacles that do not align with the LiDAR scanning height. To address this issue, some researchers have suggested integrating image sensors with LRFs to enable 3D detection, thereby improving robot navigation safety.[1–3] In this study, we will utilize semantic segmentation and monocular depth estimation technology to generate 3D point clouds of obstacles. By integrating this with the sensing data of an LRF, we will employ a heterogeneous sensor fusion approach to ensure that obstacle information benefits from the accuracy of an LRF and the detailed resolution of an image sensor.

Let us introduce the key vision-based technologies employed in our proposed method, such as semantic segmentation and monocular depth estimation. Semantic segmentation involves assigning pixels to specific categories at the pixel-wise level. Fully convolutional networks[4] replaced the fully connected layer with a convolutional layer to classify all pixels and could be trained in an end-to-end training process. U-Net,[5] a popular semantic segmentation network with an encoder-decoder architecture, enables efficient training with minimal data and yields accurate results. Subsequent enhancements such as U-Net++[6] and U-Net3+[7] have been built upon the U-Net framework. ENet,[8] similar to ResNet,[9] is an efficient real-time semantic segmentation network designed with a main branch and an additional branch using convolution operations. Fast-SCNN[10] adopts a two-branch structure to combine global and local features obtained from more and less convolutional operations, respectively. A model called FasterSeg was proposed in Ref. 11; it utilized neural architecture search (NAS) to design the neural network architecture and was trained using knowledge distillation. Results showed that FasterSeg outperformed the other networks mentioned. Hence, we have opted for FasterSeg, which plays a crucial role in obstacle identification, as the semantic segmentation model in this study.

Our algorithm begins by conducting semantic segmentation and then proceeds to estimate the distance of obstacles using monocular depth estimation. The training methods for monocular depth estimation typically involve supervised or unsupervised learning. PSMNet,[12] a supervised learning network, employs a 3D convolutional neural network (3D-CNN) and pyramid space to address estimation errors in reflective or textureless areas. Supervised learning models usually demand substantial amounts of training data, particularly high-resolution depth maps, which can be difficult and expensive to acquire. Hence, an unsupervised learning-based method is a more favorable choice. Monodepth,[13] an unsupervised learning network, uses a CNN to generate left-view and right-view disparity maps from a single-view image. Further details about monocular depth estimation are outlined in Sect. 2.1. Another model,[14] trained using an unsupervised approach, incorporates a CNN based on Monodepth to predict the parameters of the camera, thereby reducing depth errors. Between the aforementioned two models, the compact Monodepth model with fewer parameters is better suited for our power-limited robot. Consequently, we employed the Monodepth model in this study to predict the disparity map of a single-view image.

The primary objective of this study is to integrate an image-based sensor and an LRF device to generate a point cloud representation of the obstacles in the environment through the utilization of competitive and complementary sensor fusion methodologies. In addition, we performed several experiments related to robot navigation to validate the effectiveness of our method.

2. Proposed Heterogeneous Sensor Fusion

In this study, we focus on integrating two sensor types: the CMOS image sensor and the LRF. By benefiting from the high resolution of an image sensor and the accurate measurement of an LRF, these sensors work in tandem to deliver a more dependable 3D sensing outcome, allowing the robot to detect obstacles in its path within complex environments effectively. More detailed insights into our innovative sensor fusion technology are outlined in the following subsections.

2.1 3D positioning of obstacles by image sensor

The vision-based obstacle detection method proposed in this study encompasses three primary steps, which are described below.

Step 1: Deep-learning-based semantic segmentation. This step identifies potential obstacle pixels in an image using deep-learning-based semantic segmentation and then forms objects through connected component labeling. The main aim of this step is to distinguish obstacle pixels from other non-obstacle pixels. This enables a focused analysis of obstacles, including determining their distances and avoiding them. In this study, we utilize the semantic segmentation network, named FasterSeg,[11] to recognize objects of interest in the surrounding environment, specifically predefined types of obstacles. The fundamental design principles and advantages of FasterSeg are elaborated in subsequent paragraphs.

FasterSeg is a supervised learning network that employs NAS to generate an improved structure for semantic segmentation models automatically. Achieving a balance between high accuracy and low latency can be challenging, as these goals often oppose each other. As a result, FasterSeg is a newly introduced NAS framework designed to achieve fast inference and competitive accuracy. It consists of three main components.

- Efficient search space with multi-resolution branching: This approach involves utilizing a backbone network to downsample with a factor of 8 and then branching out into factors of 8, 16, and 32.
- Regularized latency optimization with finer granularity: This addresses the issue of architecture collapse in latency-constrained searches.
- Teacher–student co-searching for knowledge distillation: This method entails training two networks using knowledge distillation. The first network, called the teacher network, is trained to optimize semantic segmentation accuracy without considering computing speed. The second network, known as the student network, integrates channel parameters learned from the trained teacher network to achieve high segmentation accuracy and fast operation.

The loss function during training in our work is defined as

$$\mathcal{L} = \mathcal{L}_{seg} + ww_2\mathcal{L}_{lat}. \tag{1}$$

Here, $\mathcal{L}_{seg}$ and $\mathcal{L}_{lat}$ are loss functions for segmentation and latency; w_1 and w_2 are their weights, respectively. In our experiments, we set $w_1 = 1$ and $w_2 = 0.1$. Here, $\mathcal{L}_{seg}$ is defined on the basis of cross entropy and online hard example mining.[15] $\mathcal{L}_{lat}$ is defined by operation $\mathcal{O}$, subsampling $\mathcal{S}$, and the channel expansion ratio $\mathcal{X}$, as formulated in Eq. (2).

$$\mathcal{L}_{lat} = w_{21}\mathcal{L}_{\mathcal{O}} + w_{22}\mathcal{L}_{\mathcal{S}} + w_{23}\mathcal{L}_{\mathcal{X}} \tag{2}$$

In our experiments, we set $w_{21} = 0.001$, $w_{22} = 0.997$, and $w_{23} = 0.002$.

Step 2: Monocular depth estimation. This step estimates the depth map for the image captured in the previous Step 1. Once the obstacle pixels are identified, the next step involves determining the depth corresponding to each pixel to obtain the 3D position information of the identified obstacles. In this study, we utilized a method known as monocular depth estimation for the captured image. Initially, we applied a convolutional neural-network-based method called Monodepth[13] to estimate a disparity map. Subsequently, we converted the disparity into a depth map based on camera parameters and geometric relationships. We can leverage our previous works[16,17] to improve Monodepth models if obstacles have been identified.

In this paragraph, the original Monodepth model is introduced and used subsequently. As shown in Fig. 1, the Monodepth model φ aims to predict a disparity image $\hat{D}^L$ from a given left-view image I^L. Here, φ can be achieved through an autoencoder architecture in a self-supervised training manner. Each training sample contains a pair of well-rectified left and right images, I^L and I^R. Learning a disparity image corresponding to the left-view image is regarded as a stereo vision reconstruction problem. Given a left-view image I^L, the model φ first predicts two disparity images, $\hat{D}^L$ and $\hat{D}^R$. For the right image I^R, the left disparity is $\hat{D}^L$ used to reconstruct its opposite left-view corresponding $\hat{I}^L$. Similarly, $\hat{D}^R$ is used for reconstructing $\hat{I}^R$ from I^L. The loss function $\mathcal{L}_{Mono}$ used in the Monodepth model is defined in Eq. (3) as a combination of three critical terms.

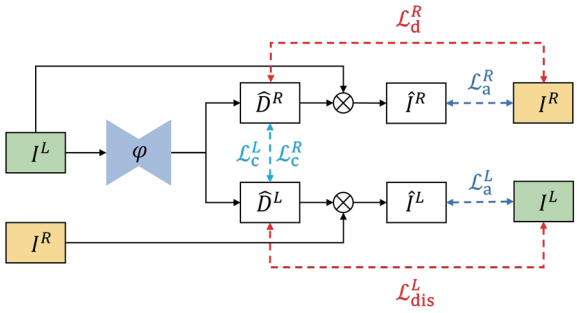

Fig. 1. (Color online) Training scheme of Monodepth model.

$$\mathcal{L}_{Mono} = \alpha_a \left(\mathcal{L}_a^L + \mathcal{L}_a^R \right) + \alpha_d \left(\mathcal{L}_d^L + \mathcal{L}_d^R \right) + \alpha_c \left(\mathcal{L}_c^L + \mathcal{L}_c^R \right) \tag{3}$$

Here, $\mathcal{L}_a$ encourages the reconstructed image to appear similar to the corresponding training input (appearance matching loss), $\mathcal{L}_d$ enforces smooth disparity (disparity smoothness loss), and $\mathcal{L}_c$ prefers the predicted left and right disparities to be consistent (left–right disparity consistency loss). The superscripts L and R represent the left and right views, respectively.

The Monodepth result is a disparity map corresponding to the input image, which necessitates conversion from disparity to depth. In our work, we employed a ZED stereo camera to gather training data comprising pairs of well-rectified left and right images. When converting from disparity to depth, it is crucial to consider the intrinsic parameters of a camera, specifically the focal length along horizontal and vertical axes. The relationship between disparity and depth is not straightforwardly reciprocal, so we introduced the conversion formula as Eq. (4).

$$\mathcal{Z} = \frac{\beta_1}{\beta_2 \mathcal{D} + \beta_3} + \beta_4 \tag{4}$$

Here, $\mathcal{Z}$ denotes depth, $\mathcal{D}$ denotes disparity, and β_1, β_2, β_3, and β_4 are coefficients that can be further determined by training data. In the experiments, only the obstacles within a 6 m range are considered in finding these coefficients because the obstacles beyond 6 m have less impact on navigation safety.

Step 3: Point cloud generation. This step involves the generation of the point cloud consisting of identified obstacles using the results from Steps 1 and 2, thereby finalizing the 3D positioning of the obstacles. The process for generating the point cloud is depicted in Fig. 2. Let us utilize

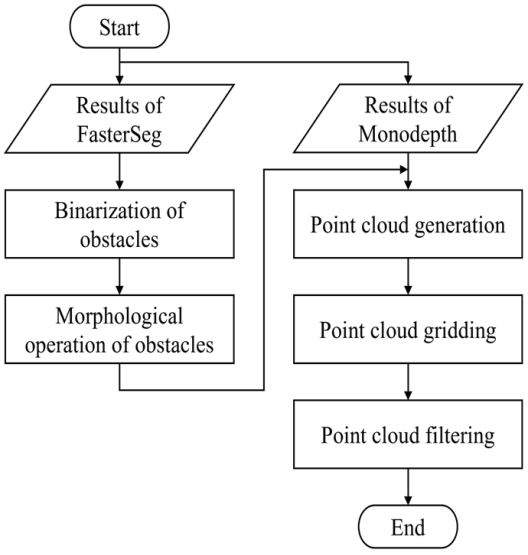

Fig. 2. Point cloud generation process.

Fig. 3 to elucidate the preprocessing outcomes of each step in our point cloud generation process. Figure 3(a) presents the original image, whereas Figs. 3(b) and 3(c) demonstrate the outcomes of depth estimation and segmentation. Subsequently, Fig. 3(d) showcases the binary image representing the obstacle of interest, and Fig. 3(e) exhibits the filtered result achieved through morphological operations, including dilation and erosion. Accordingly, the point cloud can be generated by referencing Figs. 3(b) and 3(e).

We assume that the camera center is the origin of the world coordinate system (WCS), as shown in Fig. 4. A physical 3D object point $h_C = (x_h, y_h, z_h)$ is projected onto the imaging plane at the image point $h_I = (u_h, v_h)$, as calculated using Eq. (5).

$$\begin{cases} u_h = f_u \dfrac{x_h}{z_h} + u_c \\ v_h = f_v \dfrac{y_h}{z_h} + v_c \end{cases} \tag{5}$$

Here, (u_c, v_c) is the image center on the imaging plane, and f_u and f_v are the horizontal and vertical focal lengths, respectively. For a given pixel (u_h, v_h), its reconstructed 3D point (x_h, y_h) at the depth z_h is derived using Eq. (6).

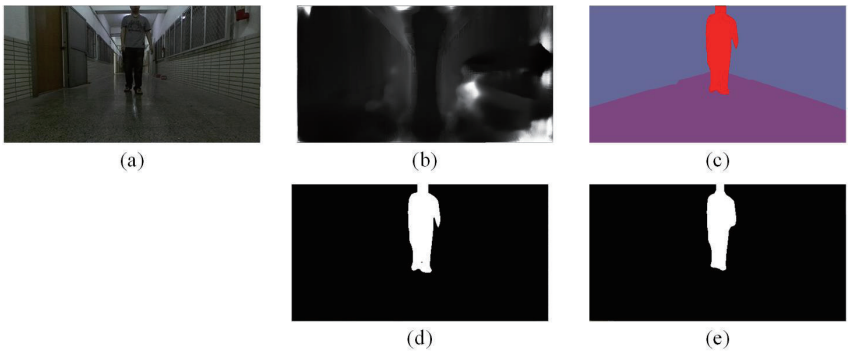

Fig. 3. (Color online) Temporal results during point cloud generation: (a) original image, (b) depth estimation result, (c) segmentation result, (d) obstacle, and (e) filtered result of obstacle.

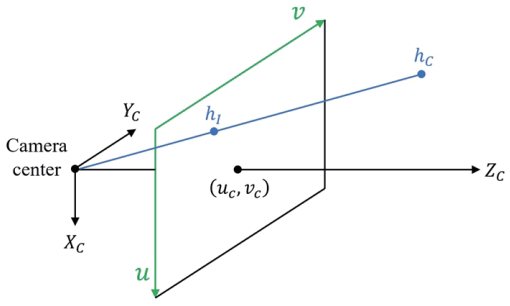

Fig. 4. (Color online) Definition of used camera coordinate system.

$$\begin{cases} x_h = (u_h - u_c) z_h / f_u \\ y_h = (v_h - v_c) z_h / f_v \end{cases} \tag{6}$$

To obtain the point cloud containing obstacles, Eq. (6) is solved on every pixel belonging to the identified obstacles. Figure 5(a) illustrates the point cloud generated with an obstacle, which in this case, is a person. To streamline computation in subsequent steps, the point cloud should be gridded, reducing the total point cloud samples while maintaining the obstacle structure. Figure 5(b) presents the gridded result with a grid size of 4 cm. Following gridding, we filter the point cloud, retaining only the samples where there are more than 15 neighboring points within a radius of 12 cm. The gridded point cloud undergoes further filtration, and only the samples with more than 15 neighboring points within 12 cm are preserved. Figure 5(c) shows the filtered point cloud result. Using only the image sensor, we have reconstructed the 3D structure of the obstacles at this stage.

2.2 Heterogeneous sensor fusion process

The integration of image sensing often results in inaccurate point clouds due to estimation errors from the Monodepth model, which tends to have greater errors for farther obstacles. To address this, our approach involves heterogeneous sensor fusion using competitive and complementary fusion, leveraging the different precision and resolution of the LRF and image sensing. We employ the extended Kalman filter[18] (EKF) as the basis for this sensor fusion, capitalizing on the accuracy of the LRF and the high resolution of the image sensor.

In the competitive fusion step, the distance obtained from the LRF serves as the real state vector, and the EKF is used to predict and update obstacle estimations by considering the distance from image sensing as the measurement. Additionally, a region of interest (ROI) is defined to cover 12 cm above and below the line scanned by the LRF, and the centroid of the point cloud of the obstacle within the ROI is calculated. In the complementary fusion step, the point cloud is corrected on the basis of the Kalman gain and the error between the state and measured vectors obtained from the competitive fusion step. This approach allows for corrected point clouds, providing the high-resolution and accurate 3D structure information of obstacles.

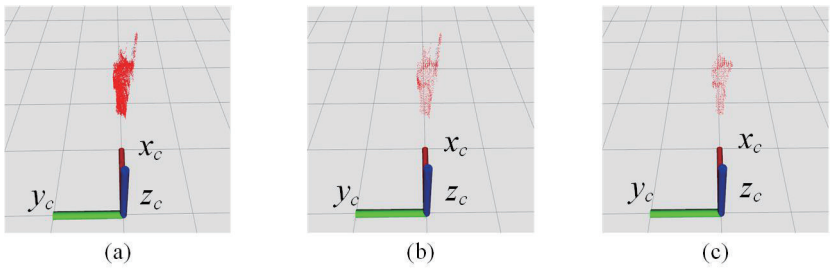

Fig. 5. (Color online) Point cloud of a person: (a) original, (b) gridded result, and (c) filtered result.

3. Experimental Results

In this section, we outline a series of specific experiments conducted to validate the effectiveness of the proposed method for sensor fusion across heterogeneous systems.

3.1 Performance evaluation on FasterSeg

Table 1 presents the relevant parameters that we used in our experiments to train the FasterSeg model. In Fig. 6, columns (a) and (b) depict captured images and their corresponding

Table 1
Hyperparameters for training FasterSeg model.

Parameter Name	Value
Size of input image (pixels)	512 × 288
Image size of input layer (pixels)	512 × 256
Number of training samples	350
Number of validation samples	92
Number of training epochs	500
Batch size	4
Number of layers	8
Learning rate	10^{-2}
Object categories (can be expanded)	Road, wall, obstacle

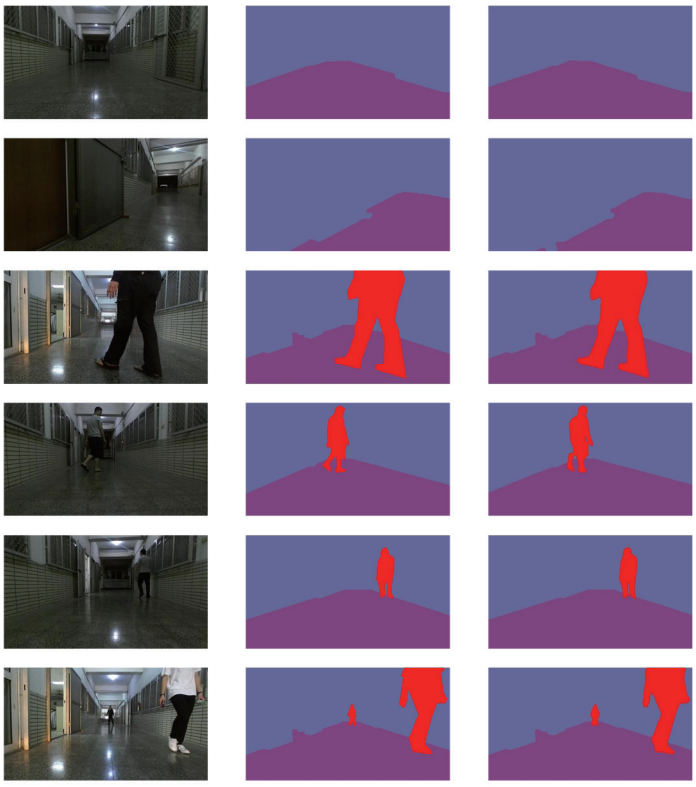

Fig. 6. (Color online) Examples: (a) captured images, (b) labeled ground truth, and (c) results of FasterSeg.

ground truth, while column (c) shows the results produced by FasterSeg. In this figure, the road, wall, and obstacle are represented by purple, gray blue, and red colors, respectively. The segmentation accuracy is summarized in Table 2.

3.2 Performance evaluation on Monodepth model

We utilized the pretrained Monodepth model, as outlined in Ref. 13, which was trained on the KITTI dataset, to estimate the disparity map of a single-view image. Additionally, we gathered 3465 pairs of well-rectified images using the ZED camera and fine-tuned the pretrained model with the parameters specified in Table 3. Our dataset consists of 1995 samples from the testing campus, 849 samples with people as obstacles, and 621 samples with office chairs. After the fine-tuning process, we derived the disparity-to-depth conversion formula, as shown in Eq. (7).

$$\mathcal{Z} = \frac{0.05625}{0.5971\mathcal{D} + 0.000219} - 0.7293 \tag{7}$$

Subsequently, the depth from the well-calibrated ZED sensor was used as the reference ground truth, and the absolute estimation errors from the Monodepth model error were calculated using Eq. (7) and are plotted in Fig. 7. Notably, the Monodepth model error increased with distance, emphasizing the necessity for heterogeneous sensor fusion to reduce it.

3.3 Experiments of heterogeneous sensor fusion

The heterogeneous sensor fusion process addressed in Sect. 2.2 employs the LRF sensed information to rectify the depth estimation errors of the Monodepth model presented in Fig. 7. In this section, the ZED sensed depth data is still employed as the reference ground truth. In Fig. 8, we compare the point clouds when obstacles are located at various distances. The blue and green

Table 2
Performance metrics on segmentation by FasterSeg.

Category	Average Precision (AP) (%)
Wall	95.231
Road	95.875
Obstacle	89.446
Overall	93.513

Table 3
Hyperparameters for Monodepth model fine-tuning.

Parameter Name	Value
Number of samples	3465
Size of input image (pixels)	512 × 256
Number of epochs	25
Batch size	4
Learning rate	10^{-4}

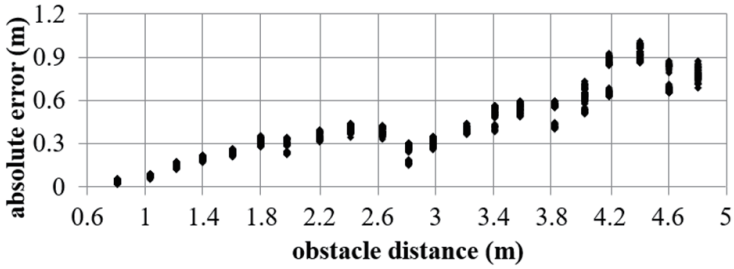

Fig. 7. Monocular depth estimation error.

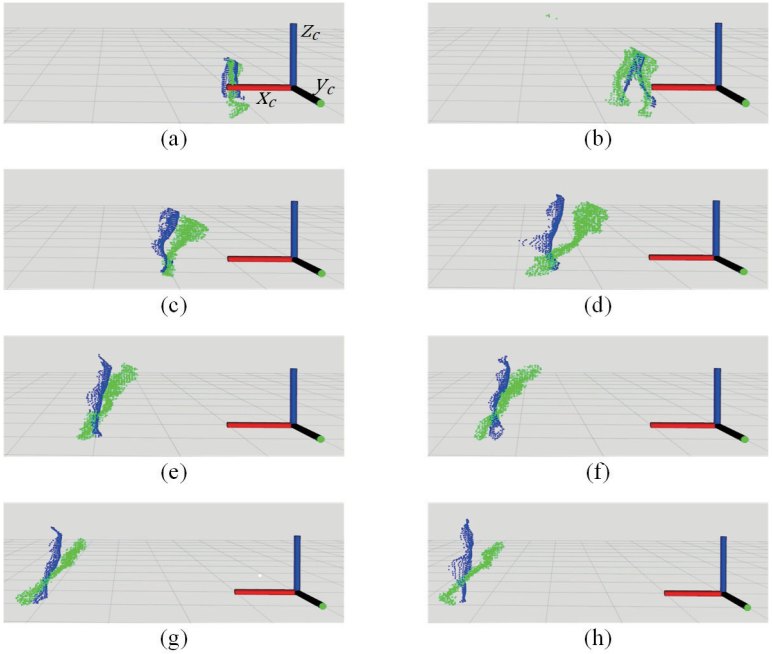

Fig. 8. (Color online) Comparison of point clouds when the obstacle is (a) 1, (b) 1.5, (c) 2, (d) 2.5, (e) 3, (f) 3.2, (g) 4, and (h) 4.2 m away.

points correspond to the point clouds obtained from our sensor fusion and the ZED camera, respectively. Owing to the estimation error of Monodepth,[13,16] the structures are slightly inconsistent with ZED sensed depth information. However, our sensor fusion-based point cloud consistently aligns with the ground truth, regardless of the distance between the obstacle from the camera. The comparative analysis of depth estimation highlights the contributions of this study.

3.4 Navigation test on wheeled robot

We conducted indoor navigation experiments in the corridor of our department building. Initially, we utilized GMapping to construct the map, which was then imported to OpenCV for further processing. This map was gridded, and a generalized Voronoi topological map was built,

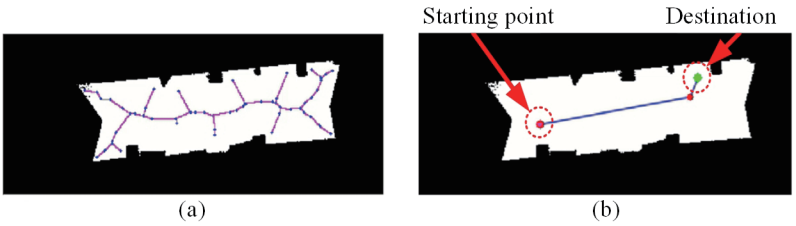

Fig. 9. (Color online) Robot navigation: (a) generalized Voronoi topological map and (b) initially planned path by A* algorithm.

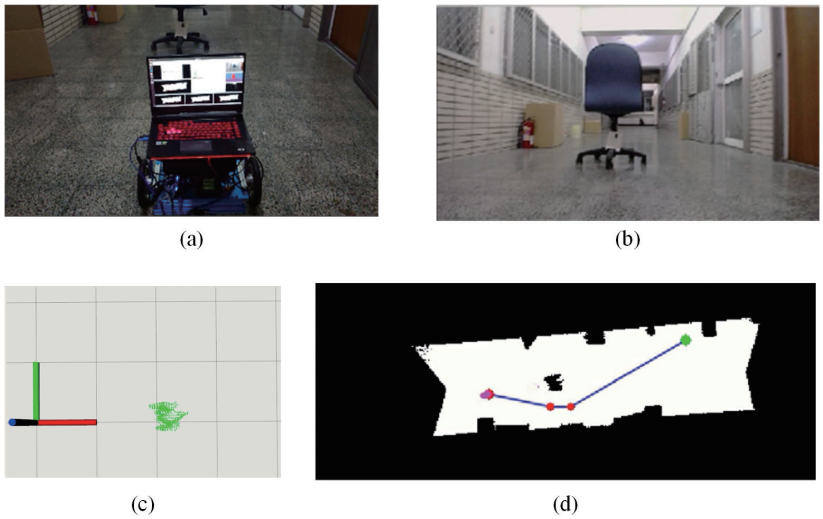

Fig. 10. (Color online) Example of obstacle avoidance: (a) mobile robot, (b) a chair as an obstacle, (c) point cloud of the obstacle, and (d) replanned path for obstacle avoidance.

as shown in Fig. 9(a). Subsequently, the location of the robot was determined using the adaptive Monte Carlo localization method,[19] and the initial path planning was conducted using the A* search algorithm.[20] Figure 9(b) illustrates the path from the robot to its destination at this stage. Plots (a)–(c) in Fig. 10 present an instance of the robot encountering an obstacle within 2 m, resulting in path replanning. Figure 10(d) showcases the replanned path that enabled the robot to bypass the obstacle.

4. Conclusion

In this study, we developed a method of improving robot navigation around obstacles by combining data from different sensors. We used both an image sensor and an LRF sensor to create a more accurate 3D map of the space in front of the robot. During navigation, the camera of the robot captures images, which are then analyzed using FasterSeg to identify obstacles. Simultaneously, the Monodepth model and disparity-to-depth conversion estimate the distance of the obstacles from the camera, creating point clouds relative to the position of the camera. By combining data from both sensors, we can generate more reliable 3D information about

surrounding obstacles, helping the robot navigate through cluttered environments. We also conducted several experiments to validate the effectiveness of our sensor fusion method for practical robot navigation.

Acknowledgments

This research was funded by the National Science and Technology Council, Taiwan, grant number NSTC 112-2221-E-992-064-MY2.

References

1 Z. Wang, Y. Wu, and Q. Niu: IEEE Access **8** (2019) 2847. https://doi.org/10.1109/ACCESS.2019.2962554
2 W. Chen, C. Zhou, G. Shang, X. Wang, Z. Li, C. Xu, and K. Hu: Remote Sens. **14** (2022) 6033. https://doi.org/10.3390/rs14236033
3 Q. Tang, J. Liang, and F. Zhu: Signal Process. **213** (2023) 109165. https://doi.org/10.1016/j.sigpro.2023.109165
4 J. Long, E. Shelhamer, and T. Darrell: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2015) 3431. https://doi.org/10.1109/CVPR.2015.7298965
5 O. Ronneberger, P. Fischer, and T. Brox: 18th Int. Conf. Medical Image Computing and Computer-Assisted Intervention (Springer, 2015) 234. https://doi.org/10.1007/978-3-319-24574-4_28
6 Z. Zhou, Md M. R. Siddiquee, N. Tajbakhsh, and J. Liang: 21st Int. Conf. Medical Image Computing and Computer-Assisted Intervention (Springer, 2018) 3. https://doi.org/10.1007/978-3-030-00889-5_1
7 H. Huang, L. Lin, R. Tong, H. Hu, Q. Zhang, Y. Iwamoto, X. Han, Y. W. Chen, and J. Wu: 2020 IEEE Int. Conf. Acoustics, Speech and Signal Processing (IEEE, 2020) 1055. https://doi.org/10.1109/ICASSP40776.2020.9053405
8 A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello: arXiv:1606.02147v1 (2016). https://doi.org/10.48550/arXiv.1606.02147
9 K. He, X. Zhang, S. Ren, and J. Sun: Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2016) 770. https://doi.org/10.1109/CVPR.2016.90
10 P. K. Poudel, S. Liwicki, and R. Cipolla: arXiv:1902.04502v1 (2019). https://doi.org/10.48550/arXiv.1902.04502
11 W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang: arXiv:1912.10917v2 (2020). https://doi.org/10.48550/arXiv.1912.10917
12 J. R. Chang and Y. S. Chen: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2018) 5410. https://doi.org/10.1109/CVPR.2018.00567
13 C. Godard, O. M. Aodha, and G. J. Brostow: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2017) 270. https://doi.org/10.1109/CVPR.2017.699
14 V. Anisimovskiy, A. Shcherbinin, S. Turko, and I. Kurilin: Canadian Conf. Artificial Intelligence (Springer, 2020) 36. https://doi.org/10.1007/978-3-030-47358-7_4
15 A. Shrivastava, A. Gupta, and R. Girshick: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2016) 761. https://doi.org/10.1109/CVPR.2016.89
16 H. C. Chen: IEEE Access **7** (2019) 117869. https://doi.org/10.1109/ACCESS.2019.2953954
17 H. Chen, H. C. Chen, C. H. Sun, and W. J. Wang: Int. J. Fuzzy Syst. **26** (2024) 1143. https://doi.org/10.1007/s40815-023-01657-0
18 G. Bishop and G. Welch: An Introduction to the Kalman, Proc. 2001 SIGGRAPH (ACM, 2001).
19 X. Wang, C. Li, L. Song, N. Zhang, and H. Fu: Proc. 37th Chinese Control Conf. (IEEE, 2018) 5207. https://doi.org/10.23919/ChiCC.2018.8482698
20 D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan: Procedia Comput. Sci. **179** (2021) 507. https://doi.org/10.1016/j.procs.2021.01.034