2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Real-time Bird's-Eye-View Panoptic Segmentation for Monocular-based Indoor Navigation

Dawit Kim*[1], Jungmo Koo*[1], Jongseob Yun[1], and Soonyong Park[1]

Abstract— **Bird's-Eye-View (BEV) segmentation is a essential technology for safe and efficient navigation. This is even more necessary in indoor driving, where there are dynamic and unstructured objects such as people and robot. However, since there is no way to generate training data, most of the researches have been conducted mainly in outdoor environments. In this paper, we propose an innovative approach to address this challenge. We automatically generate BEV training data for indoor environments based on the physics engine of a simulator. This eliminates the needs for tons of real data and virtual environments. We also propose a lightweight network architecture capable of BEV panoptic segmentation in real-time. Based on this network and simple image processing, our framework shows fast but also robust performance in real-world environments with no gap between simulation and reality. The network can inference at a speed of about 61 FPS on AGX Orin in FP16 mode. Furthermore, it outperforms existing algorithms in the semantic segmentation task, achieving 14% higher mIoU.**

I. INTRODUCTION

The driving environment of a mobile robot or car is usually complex, as it is surrounded by a large number of obstacles. To ensure safe and efficient navigation, acquiring precise information about the spatial location and volume of these obstacles is imperative. A conservative approach prioritizing safety can prevent accidents but may suffer from accuracy degradation. On the other hand, approach focusing on accuracy rather than safety can be expensive due to the needs of more sensors and high computational power. Therefore, dealing with the trade-off between safety and efficiency is an essential part of autonomous driving perception.

LiDAR-based navigation is a representative example of focusing on safety. LiDAR can accurately measure the distance to obstacles based on physical signals. As this is important for recognizing the driving environment, many perception-related studies have used LiDAR-based approaches and shown good performance in tasks such as 3D object detection [1]–[7]. However, LiDAR requires high initial cost, and may provide less accurate measurements depending on the material of the object. It also demands higher power consumption compared to other sensors.

Meanwhile, vision-based navigation is being actively studied as a cost-effective alternative to LiDAR. In particular, 3D object detection [8]–[10] and occupancy prediction [11], [12] can be good substitutes of LiDAR because they include prediction of physical distance information. These studies showed promising performance using large-scale open datasets such as nuScene [13] and KITTI [14]. However, 3D object detection only predict instances and does not encompass regions such as drivable area. Although occupancy prediction densely represents the environment, it requires a lot of computational power and is too slow for real-time applications.

Bird's-Eye-View (BEV) segmentation [15]–[18] can be a way to overcome the problems mentioned above. By predicting a top-view image of the scene, it effectively represents the surroundings. This method can also predict areas occluded by obstacles, providing assistance for efficient navigation. However, in the view of train data, BEV images must be paired with front-view images, which is challenging to achieve in real-world. To solve this problem, satellite or aerial images must be used, but it is unrealistic to completely match them due to pair them with corresponding front-view images due to unsynchronized time stamps. Even this can only be used in outdoor environments. There is no way to obtain data for indoor environments blocked by a ceiling.

In this paper, we present significant contributions that advance the field of indoor BEV segmentation. Our work focuses on addressing key challenges related to data generation and real-world applicability. Our main contributions are as follows:

- We present an automatic method to generate indoor BEV data and a pipeline that bridges the sim-to-real gap (the gap between simulation and real world) for real-world inference.
- We propose a lightweight BEV panoptic segmentation network called PeneBEV (**Pene**trating **B**ird's-**E**ye-**V**iew). This network provides a map like a bird sees through a indoor space from the air. It achieved 61fps when running in FP16 mode on AGX Orin.
- Our model achieved 77.25% mIoU performance and about 6.6 cm distance error on our test set when trained solely on automatically generated data.

II. RELATED WORK

A. BEV segmentation based on RGB image

Pioneering researches [10], [15]–[21] predict a BEV map directly from front-facing camera RGB image in an end-to-end manner. The algorithm requires a BEV segmentation ground truth that accurately corresponds to the frontal image. For outdoor environments, BEV data can be obtained by various methods, including satellite imagery, maps, LiDAR, and GPS. However, collecting BEV images in indoor

*These two authors contributed equally.

[1]NAVER LABS, Seongnam-si, Gyeonggi-do, 13561, South Korea. email: {dawit.kim, jungmo.koo, jseob.y, soonyong.park}@naverlabs.com

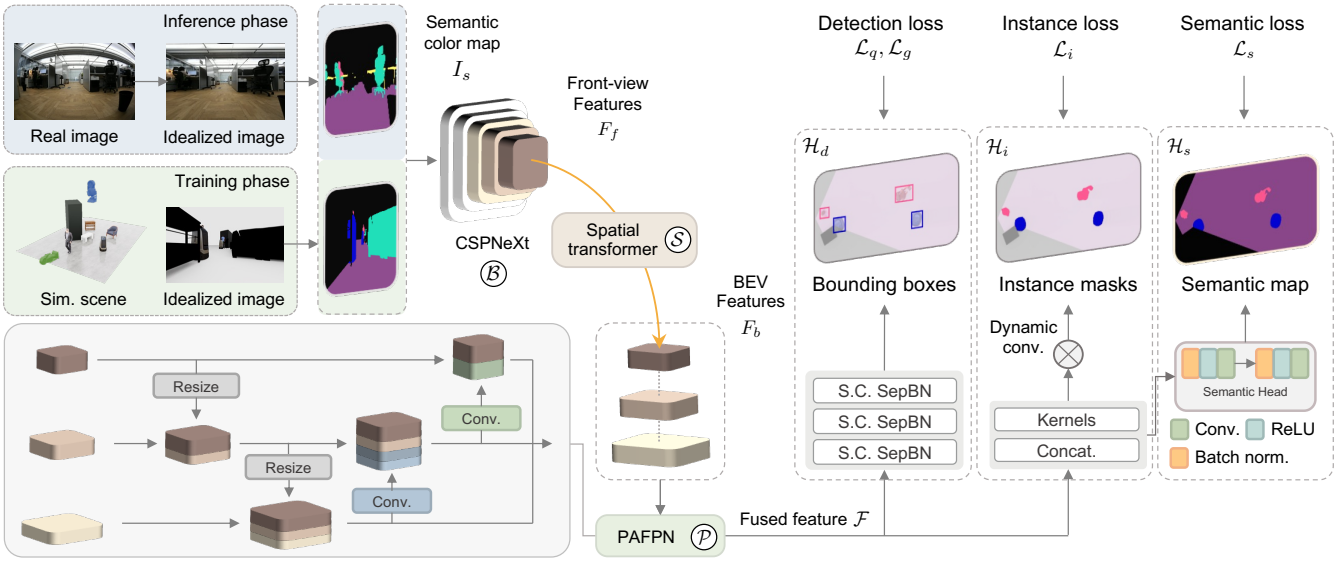

Fig. 1. Overall architecture of PeneBEV. During training, we use the semantic color map from a simulated ideal pinhole camera, and during inference, we convert a real image into an image with the same geometric characteristics as the simulation. PeneBEV extracts features from this map, converting them into BEV features for efficient localization and segmentation.

robot driving environments poses challenging issues and even direct capturing is limited by ceiling constraints. SkyEye [22] employs a self-supervised approach to overcome the challenges of generating accurate ground truth data. This approach is highly valuable as it enables learning of BEV estimation based solely on semantic ground truth from frontal views. This technique mandates the recording of the camera's pose information and and utilize state-of-the-art monodepth to generate pseudolabels. Nevertheless, owing to scale ambiguity and inconsistency, the outcome of pseudolabel generation could fluctuate contingently on post-processing methodologies.

B. BEV segmentation based on semantic map

Some methods [23], [24] do not utilize the camera image as an input, but instead employs a segmentation mask as the input source. Firstly, this approach enables the use of a small network as the semantic information does not need to be processed by the network. Secondly, It simplifies the BEV segmentation process as the network only needs to perform the perspective conversion using the semantic information already known from the input mask. Finally, the greatest advantage is that there is no sim-to-real gap. BEV can be performed without visual aid and thus can be simulated. Our research was inspired by this approach.

III. METHODOLOGY

Since BEV data cannot be collected in real indoor environments, the only way to train a BEV segmentation network is to use data generated by simulation. However, creating a virtual environment for all indoor settings with diverse objects and layouts is impossible. In addition, there will always be a gap between simulation and reality due to various factors such as data characteristics and camera image distortion. We propose methods to minimize this gap by processing the real and simulated images to have the same geometric characteristics (Sec. III-B) and fully automatically generate data that can cover a wide range of indoor environments (Sec. III-C). In addition, we introduce a network architecture that enables real-time BEV panoptic segmentation at edge devices (Sec. III-D).

A. Overall architecture

Fig. 1 illustrates the overall architecture of our proposed method. During the training phase, we directly use the semantic color map obtained from ideal pinhole camera in the simulation. Meanwhile, in the inference phase, we convert distorted images captured by a real camera into images that have exactly same geometric attributes as the simulation.

PeneBEV extracts features from a semantic color map and converts them into BEV features through spatial transformer. The fused features are used by the object detection, instance segmentation and semantic segmentation heads to derive bounding boxes, instance masks and semantic maps in the BEV, respectively.

B. Real-to-Sim camera setup

The frontal images of NLST dataset in Section III-C are captured by a simulated camera without distortion. As a result, there exists a discrepancy when using disorted images from real world as input. To accomplish this issue, we synchronize cameras of simulation and real world. Firstly, we do intrinsic calibration of real world camera with patterned board. In our setting, we use basalt [25] with Double sphere model [26] because our camera has large field of view over 150 degrees. After intrinsic calibration, all images are undistorted and cropped into dense image. Specifically we precompute the valid angle of view preventing empty image region for cropping. Secondly we precisely solve 6DOF camera pose based on AprilTag [27] and PnP. We regards the result as camera pose compared to the ground assuming

AprilTag plane is identical to the ground. Finally we set simulated camera as same as precisely calibrated real camera. Synchronized virtual camera insures that pixel rays from a real image can one-to-one match with those from a simulated image. This synchronization eliminates any discrepancies between images from the simulation and real world. This process is represented in Fig. 2.

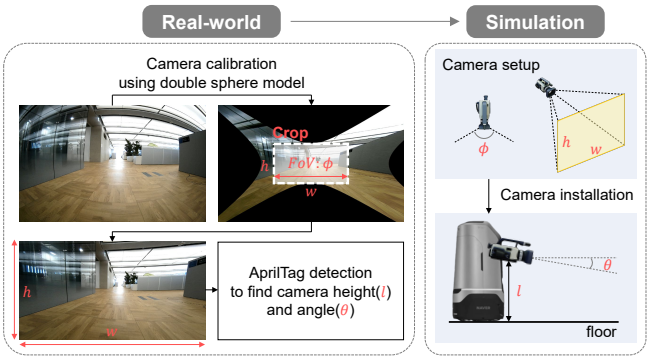

Fig. 2. The process of generating perfectly identical images between the real-world and the simulated camera. Since the camera used in the real-world uses a wide-angle lens, it was calibrated using double sphere camera model. To find the valid angle of view and resolution, crop from the undistorted image and use it in the simulation environment for NLST dataset.

C. BEV data auto-generation

Outdoor driving environments are relatively less dynamic than indoor ones. This is because they are subject to the traffic system and there are a limited number of things that can appear on the road, such as cars, traffic lights and pedestrians. Furthermore, outdoor driving simulation environments such as NVIDIA Drive Sim [28], CARLA [29] and LGSVL [30] are actively used. This enables the acquisition of a significant amount of high-quality simulation data. On the other hand, indoor driving simulation is quite irregular and the shape of objects is inconsistent. This makes it challenging to create a realistic scene.

To solve this problem, we propose NLST (NAVER LABS Sliding Things), a highly effective method for indoor scene generation. NLST works in a similar way to Falling Things [31], using the simulator's physics engine to collect data across various scenarios. Fig. 3 describes how NLST generates a scene. It generates multiple data from a single scene as follows. First, a random combination of people, furniture, a robot and obstacles are generated at different distances from the front camera and allowed to slide towards the front camera. At each simulation step, training data is collected by simultaneously collecting images and semantic maps from the front camera and the BEV camera. During the sliding process, the distance to the objects varies and the objects bounce off each other due to collisions between them. Because of these phenomena, a large variety of data is generated from a single scene. We also imposed a constraint on the physics engine: the objects are fixed for all rotations and translations in all directions except the sliding direction. The reason for this is to prevent objects from being thrown into the air by collisions.

In a real indoor environment, objects are not always close together. When the scene is generated using the method described above, the objects are initially separated, but are brought closer together with each simulation step. To simulate the space between the objects, we create space cubes. These cubes are randomly sized and have no visual model but only a collision model. This is intended to act as a support between objects, creating gaps even though it is not visible in the camera's image.

Sliding Things is based on Isaac Sim [32], which offers a reasonable physics engine and a high level of development comfort. There are other physics-based simulators such as Unreal Engine [33] and Unity [34], but Isaac Sim provides a wide range of tools for AI research, making it easy to obtain the desired labels. For data generation, we used 10 human models, 1 custom robot model, 87 furniture models extracted from Amazon-Berkeley-objects [35]. We then trained a BEV segmentation network using about 80,000 pieces of synthetic data. The occluded area is post-processed after generating the BEV mask and is added as a class for training. This occluded class is different from the obstacle class and is categorized as an area that cannot be traversed during navigation.

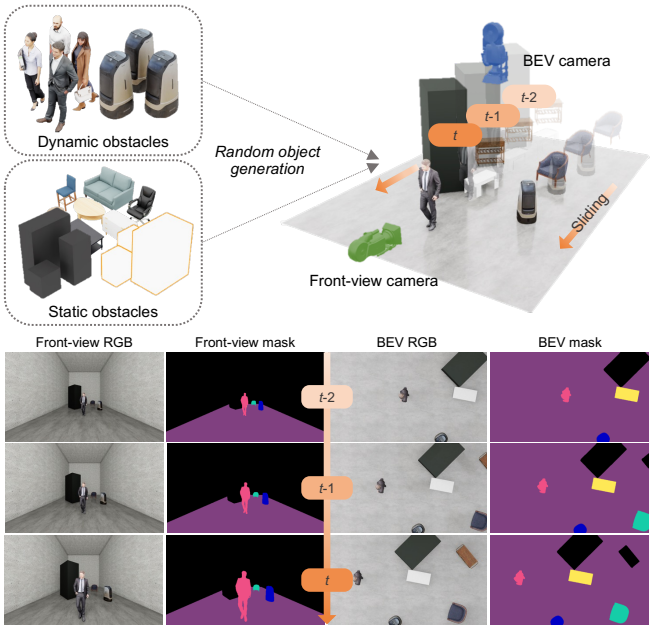

Fig. 3. Conceptual diagram of NLST. It randomly selects various types of obstacles and forces them to slide towards the front-view camera. For each simulation step, RGB image and semantic map for Front-view / BEV are collected.

D. PeneBEV

In this section, we describe the architecture of PeneBEV. It is based on RTMDet-ins [36], a real-time instance segmentation network. We added a Spatial transformer [23] and a segmentation head to this structure. Spatial transformer transforms the front-view features extracted by the backbone into BEV features, and the head performs semantic segmentation with very small features.

PeneBEV takes a monocular semantic color map $I_s \in \mathbb{R}^{3\times H\times W}$ as input. For feature extraction, it uses CSPNeXt

$\mathcal{B}$ [36] as a backbone and derives the front-view features ${F_f}^i \in \mathbb{R}^{C_i \times H_i \times W_i}$ at level i. For efficient inference, we only use features of the last three levels. Given that there are N levels of CSPNeXt, this can be expressed as follows.

$$F_f = \{{F_f}^i\}_{i=N-2}^{N} = \mathcal{B}(I_s) \tag{1}$$

The extracted front-view features F_f are passed through Spatial transformer $\mathcal{S}$ and converted to BEV features ${F_b}^i \in \mathbb{R}^{C_i \times H_i \times W_i}$ at level i. Spatial transformer takes as input the features F_f, intrinsic $K_f \in \mathbb{R}^{3\times3}$, rotation extrinsic $R_f \in \mathbb{R}^{3\times3}$ and translation extrinsic $t_f \in \mathbb{R}^3$ for the front view and the intrinsic $K_b \in \mathbb{R}^{3\times3}$, rotation extrinsic $R_b \in \mathbb{R}^{3\times3}$ and translation extrinsic $t_b \in \mathbb{R}^3$ for the BEV. this can be expressed as follows.

$${F_b}^i = \mathcal{S}({F_f}^i, K_f, [R_f \mid t_f], K_b, [R_b \mid t_b]) \tag{2}$$

Like front-view features F_f, BEV features F_b are composed of three levels of features, and a fused features $\mathcal{F}$ are obtained by PAFPN $\mathcal{P}$ [37]. In this process, multi-level BEV features are fused to produce features that are very small in size but contains rich information. This can be represented as follows.

$$\mathcal{F} = \mathcal{P}(\{{F_b}^i\}_{i=N-2}^{N}) \tag{3}$$

The fused features $\mathcal{F}$ provide effective clues to perform various tasks. Finally, the object detection head $\mathcal{H}_d$, instance segmentation head $\mathcal{H}_i$, and semantic segmentation head $\mathcal{H}_s$ are used to perform each task.

We used multi-task loss to ensure that the backbone and all task heads were well trained. For the object detection loss, we used a combination of QFL [38] and GIoU loss [39]. The expressions below represent each loss.

$$\mathcal{L}_q = -|y_q - \sigma|^{\beta}((1-y_q)\log(1-\sigma) + y_q\log(\sigma)) \tag{4}$$

$$\mathcal{L}_g = 1 - (\frac{|A \cap B|}{|A \cup B|} - \frac{|C \setminus (A \cup B)|}{|C|}) \tag{5}$$

In Eq. 4, y_q is the quality score, which is a joint representation of the classification score and the IoU score, σ is $\sigma(\mathcal{H}_d(\mathcal{F})$ taking the sigmoid of the prediction result, and β is a parameter for the scale factor. In Eq. 5, A, B and C are the ground truth box, the predicted box and the smallest box containing them, respectively. For semantic segmentation loss $\mathcal{L}_s$, we used cross-entropy loss, and for instance segmentation loss $\mathcal{L}_i$, we used Dice loss [40]. Finally, the loss used for training is as follows.

$$\mathcal{L} = \lambda_q\mathcal{L}_q + \lambda_g\mathcal{L}_g + \lambda_s\mathcal{L}_s + \lambda_i\mathcal{L}_i \tag{6}$$

where λ_q, λ_g, λ_s, and λ_i are weights to adjust how much each task loss counts towards the total loss. We used all values as 1.

IV. EXPERIMENT

In this section, we quantitatively and qualitatively compare the performance of our method and other algorithms. In addition, we present the experimental results using a robot in a real environment to demonstrate that indoor robot driving based on a monocular camera is possible using our proposed method. For the quantitative evaluation, since there is no way to obtain BEV ground truth for the indoor environment, there is no open dataset. So we used data collected in a simulated environment. Meanwhile, for the qualitative evaluation, we used data from both real and simulated environments.

A. Quantitative Evaluation

To quantitatively evaluate the generalization performance of the algorithms, we collected 1,000 data from an office scene released as open source by NVIDIA. All the furniture in this environment has never been used for training, and all networks were trained only on automatically generated data. Therefore, we can perform quantitative evaluation without worrying about cheating. The distances were derived by comparing the 2D LiDAR obtained by ray-casting the ground truth BEV mask and the predicted BEV mask. We also measured the inference speed on NVIDIA Orin AGX 32GB to verify that real-time inference is possible.

Table I shows the evaluation results of each algorithm. We did not create a large model because real-time computation is required at the edge. In semantic segmentation, PeneBEV has the highest mIoU, which is about 162% higher than IPM (Inverse Perspective Mapping) [41] and about 14% higher than Cam2BEV for a medium-sized model. In Table II, which shows the IoU for each class, Cam2BEV showed low performance for Person and Robot belonging to instance-level classes, but our method has much higher performance. This means that Cam2BEV is not well trained about instance-level classes because it only performs semantic segmentation. In Table I, the semantic segmentation performance of RTMDet decreases as the size of the model increases. It means that the generalization performance is not good due to the overfitting problem occurs as the capacity of the model increases. In particular, this trend is more pronounced in object detection and instance segmentation. The performance of tiny and medium models dropped from 89.42 to 84.57 and 82.84 to 74.44, respectively. The performance degradation is relatively small with the semantic head, but still shows the same trend. On the other hand, PeneBEV outperformed all other algorithms with a tiny model, and its performance tended to improve as the model size increased. This suggests that our proposed network architecture has good generalization performance. Finally, in distance estimation, Cam2BEV showed an error of about 6.93 cm, while PeneBEV-m performed better with an error of about 6.65 cm.

In addition, We evaluated the inference time of the algorithms on AGX Orin. Cam2BEV exhibited an inference time of 19.66 ms, while the PeneBEV-s model achieved 16.32 ms. Our model is faster than the baseline by 3.21 ms, and the mIoU is higher by about 9.0%p. This means that PeneBEV has an efficient architecture that is superior to existing algorithms in both performance and speed.

B. Qualitative Evaluation

Fig. 5 shows the results of qualitative comparison between Cam2BEV and PeneBEV. The results located at the top are

TABLE I

EXPERIMENTAL RESULTS. PENEBEV-TINY IS THE SMALLEST MODEL, PENEBEV-S IS THE SMALL MODEL, AND PENEBEV-M IS THE MEDIUM MODEL.

Method	Semantic Head	Input Shape	Latency (ms)	Object Detection		Instance Segmentation		Semantic Segmentation	Distance Estimation	
				AP	AP50	AP	AP50	mIoU (%)	MAE (cm)	RMSE
IPM	-	768×384	0.56	-	-	-	-	29.49	9.093	0.2061
Cam2BEV	-	768×384	19.66	-	-	-	-	67.88	6.928	0.1692
RTMDet-tiny		768×384	14.59	44.23	89.42	36.52	82.84	-	-	-
	✓	768×384	15.00	44.86	90.43	35.96	82.82	74.26	6.942	0.1738
PeneBEV-tiny (Ours)	✓	768×384	16.32	47.73	91.90	37.42	82.99	76.07	6.838	0.1676
RTMDet-s		768×384	15.45	44.83	88.09	36.74	82.47	-	-	-
	✓	768×384	15.84	44.04	87.06	36.26	81.34	73.90	6.951	0.1773
PeneBEV-s (Ours)	✓	768×384	16.45	47.60	92.27	37.07	84.91	76.91	6.676	**0.1638**
RTMDet-m		768×384	20.07	40.39	84.57	31.62	74.44	-	-	-
	✓	768×384	20.45	44.79	87.24	36.14	79.88	73.55	7.067	0.1781
PeneBEV-m (Ours)	✓	768×384	21.23	**48.79**	**91.92**	**39.07**	**86.82**	**77.25**	**6.646**	0.1646

TABLE II

IOU (%) FOR EACH CLASS ON THE EVALUATION DATASET

Method	Drivable	Obstacle	Person	Robot	Occluded
IPM	63.37	50.74	7.22	15.56	10.57
Cam2BEV	**90.92**	84.93	32.76	69.22	61.57
RTMDet-tiny	87.27	81.55	50.60	76.64	75.22
PeneBEV-tiny (Ours)	88.16	82.60	52.73	**79.54**	77.31
RTMDet-s	87.18	81.27	48.19	78.13	74.75
PeneBEV-s (Ours)	88.24	82.40	**55.60**	78.34	**79.98**
RTMDet-m	87.37	81.55	46.56	78.70	73.55
PeneBEV-m (Ours)	90.88	**85.01**	54.44	77.72	78.19

from simulation where ground truth for the BEV semantic map exists. The figures on the far right of each row are maps showing the performance differences. In these maps, the white areas are where both algorithms are correct, the green areas are where only PeneBEV is correct, the blue areas are where only Cam2BEV is correct, and the red areas are where both are incorrect. Overall, PeneBEV is more robust than Cam2BEV to objects with parts that are not attached to the floor, such as desks and chairs. It also predicts the area more accurately for instance-level classes than Cam2BEV due to the positive impact of the structure performing instance segmentation together.

The results located at the bottom of Fig. 5 compare the results on real data. Front-view segmentation was performed using PIDNet-s [42] model trained on a small set of annotations. Since it is not possible to collect BEV images in the real world, the predicted BEV results were warped to fit the front-view and overlaid. The results show that when one foot of a walking person is off the ground, Cam2BEV only predicts the area for the other foot, while PeneBEV predicts the area correctly. Also, as mentioned above, Cam2BEV incorrectly predicts the area under the desk as a drivable area, but PeneBEV correctly predicts the area occupied by the desk. In addition, PeneBEV is more robust to noise in front-view segmentation compared to Cam2BEV, and predicts areas more accurately in scenes with many desks and chairs.

C. Real-world Autonomous Driving

We conducted a real-world experiment to verify that BEV segmentation alone is sufficient for safe driving. We applied ray-casting to the BEV segmentation to obtain a virtual 2D LiDAR and used the DWA planner [43] to perform the driving. The results, as shown in the figure, demonstrate that it is possible to accurately locate obstacles in various situations. It also shows that PeneBEV trained with data generated by NLST is robust in real environments.

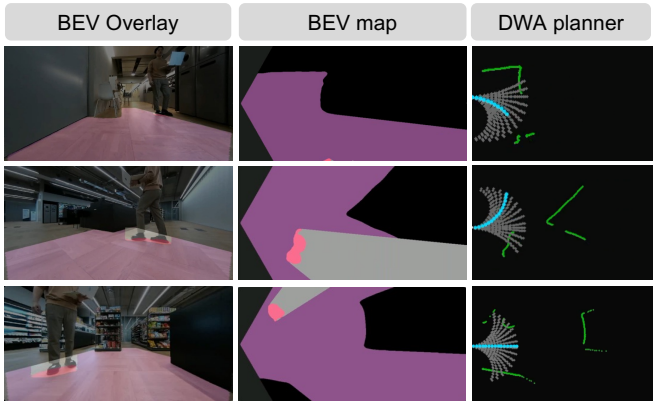

Fig. 4. Experimental results of robot driving in real environment. The input of DWA is the result calculated using only BEV segmentation without any other sensor. The robot used in the experiment is a two-wheeled robot.

V. CONCLUSIONS

In this paper, we propose a method for BEV segmentation in indoor environments, which was previously only possible for outdoor environments. NLST solves the data challenges of BEV segmentation by automatically collecting training data. We also propose PeneBEV, a network capable of real-time panoptic segmentation on edge devices. We demonstrated its high performance compared to other algorithms through qualitative and quantitative experiments. Finally, through experiments with a mobile robot in a real-world environment, we show that safe indoor driving is possible using only a monocular camera based on PeneBEV.

VI. ACKNOWLEDGEMENTS

This work was supported by Korea Evaluation Institute Of Industrial Technology (KEIT) grant funded by the Korea government(MOTIE) (No.20023455, Development of Cooperate Mapping, Environment Recognition and Autonomous Driving Technology for Multi Mobile Robots Operating in Large-scale Indoor Workspace)

Fig. 5. Qualitative evaluation results in simulation (top) and real (bottom) environments. We compare the performance of PeneBEV with baseline. For the simulation, the results of each algorithm were overaid on the ground truth, and for the real environment, they were warped and overaid on the front-view RGB image. The yellow circles show where baseline incorrectly predicts. In both simulated and real-world environments, PeneBEV performs more robustly.

REFERENCES

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[2] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[3] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 784–11 793.

[4] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proc. IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.

[5] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proc. of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[6] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[7] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[8] T. Wang, X. Zhu, J. Pang, and D. Lin, "FCOS3D: Fully convolutional one-stage monocular 3d object detection," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 913–922.

[9] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries," in *Proc. Conference on Robot Learning*. PMLR, 2022, pp. 180–191.

[10] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *Proc. European conference on computer vision*. Springer, 2022, pp. 1–18.

[11] Y. Wei, L. Zhao, W. Zheng, Z. Zhu, J. Zhou, and J. Lu, "SurroundOcc: Multi-Camera 3D Occupancy Prediction for Autonomous Driving," *arXiv preprint arXiv:2303.09551*, 2023.

[12] X. Tian, T. Jiang, L. Yun, Y. Wang, Y. Wang, and H. Zhao, "Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving," *arXiv preprint arXiv:2304.14365*, 2023.

[13] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[15] A. Saha, O. Mendez, C. Russell, and R. Bowden, "Translating images into maps," in *Proc. IEEE international conference on robotics and automation*, 2022, pp. 9200–9206.

[16] B. Zhou and P. Krähenbühl, "Cross-view transformers for real-time map-view semantic segmentation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 760–13 769.

[17] N. Gosala and A. Valada, "Bird's-eye-view panoptic segmentation using monocular frontal view images," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1968–1975, 2022.

[18] R. Xu, Z. Tu, H. Xiang, W. Shao, B. Zhou, and J. Ma, "CoBEVT: Cooperative bird's eye view semantic segmentation with sparse transformers," *arXiv preprint arXiv:2207.02202*, 2022.

[19] C. Pan, Y. He, J. Peng, Q. Zhang, W. Sui, and Z. Zhang, "BAEFormer: Bi-Directional and Early Interaction Transformers for Bird's Eye View Semantic Segmentation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9590–9599.

[20] S. Sirko-Galouchenko, A. Boulch, S. Gidaris, A. Bursuc, A. Vobecky, P. Pérez, and R. Marlet, "OccFeat: Self-supervised Occupancy Feature Prediction for Pretraining BEV Segmentation Networks," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4493–4503.

[21] J. Ross, O. Mendez, A. Saha, M. Johnson, and R. Bowden, "BEV-SLAM: Building a Globally-Consistent World Map Using Monocular Vision," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 3830–3836.

[22] N. Gosala, K. Petek, P. L. Drews-Jr, W. Burgard, and A. Valada, "SkyEye: Self-Supervised Bird's-Eye-View Semantic Mapping Using Monocular Frontal View Images," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 901–14 910.

[23] L. Reiher, B. Lampe, and L. Eckstein, "A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird's eye view," in *Proc. IEEE International Conference on Intelligent Transportation Systems*, 2020, pp. 1–7.

[24] S. Gong, X. Ye, X. Tan, J. Wang, E. Ding, Y. Zhou, and X. Bai, "Gitnet: Geometric prior-based transformation for birds-eye-view segmentation," in *Proc. European conference on computer vision*. Springer, 2022, pp. 396–411.

[25] V. Usenko, N. Demmel, D. Schubert, J. Stueckler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 422–429, 2020.

[26] V. Usenko, N. Demmel, and D. Cremers, "The double sphere camera model," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 552–560.

[27] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 3400–3407.

[28] "NVIDIA DRIVE Sim," https://developer.nvidia.com/drive/drive-constellation, 2018.

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conference on robot learning*. PMLR, 2017, pp. 1–16.

[30] "LGSVL Simulator — An Autonomous Vehicle Simulator," https://www.lgsvlsimulator.com, 2019.

[31] J. Tremblay, T. To, and S. Birchfield, "Falling Things: A synthetic dataset for 3d object detection and pose estimation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2038–2041.

[32] "NVIDIA Isaac Sim," https://developer.nvidia.com/isaac-sim, 2021.

[33] "Unreal Engine," https://www.unrealengine.com, 2019.

[34] "Unity," https://unity.com/, 2005.

[35] J. Collins, S. Goel, K. Deng, A. Luthra, L. Xu, E. Gundogdu, X. Zhang, T. F. Y. Vicente, T. Dideriksen, H. Arora, *et al.*, "ABO: Dataset and benchmarks for real-world 3d object understanding," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 21 126–21 136.

[36] C. Lyu, W. Zhang, H. Huang, Y. Zhou, Y. Wang, Y. Liu, S. Zhang, and K. Chen, "RTMDet: An empirical study of designing real-time object detectors," *arXiv preprint arXiv:2212.07784*, 2022.

[37] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[38] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 002–21 012, 2020.

[39] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 658–666.

[40] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, Springer*, 2017, pp. 240–248.

[41] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological cybernetics*, vol. 64, no. 3, pp. 177–185, 1991.

[42] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A Real-Time Semantic Segmentation Network Inspired by PID Controllers," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 529–19 539.

[43] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.