2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Confidence-Based Robot Navigation Under Sensor Occlusion with Deep Reinforcement Learning

Hyeongyeol Ryu[†], Minsung Yoon[†], Daehyung Park[†,‡], Sung-eui Yoon[†,‡]

***Abstract*— This paper considers the problem of prolonged occlusions on navigation sensors due to dust, smudges, soils, etc. Such uncontrollable occlusions often cause lower visibility as well as higher uncertainty that require considerably sophisticated behavior. To secure visibility (i.e., confidence about the world), we propose a *confidence*-based navigation method that encourages the robot to explore the uncertain region around the robot maximizing its local confidence. To effectively extract features from the variable size of sensor occlusions, we adopt a point-cloud based representation network. Our method returns a resilient navigation policy via deep reinforcement learning, autonomously avoiding collisions under sensor occlusions while reaching a goal. We evaluate our method in simulated and real-world environments with either static or dynamic obstacles under various sensor-occlusion scenarios. The experimental result shows that our method outperforms baseline methods under the highly occurring sensor occlusion, and achieves maximum 90% and 80% success rates in the tested static and dynamic environments, respectively.**

I. INTRODUCTION

We consider the problem of mobile robot navigation in an unknown environment using a light detection and ranging (LiDAR) sensor. The LiDAR-based robot navigation often suffers from unexpected adverse occlusions on the sensor surface due to dust, water, or smudge [1], [2], [3], [4] (see Fig. 1). Such occlusions are often uncontrollable and prolong until human intervention while lowering the visibility as well as increasing uncertainty [1], [3]. Our goal is to build a robot navigation policy robust to the occlusion in the real world.

One way of handling occlusions is the traditional uncertainty-aware navigation that lets the robot know where it has to care more about, particularly for collision avoidance [5], [6]. The approach makes the robot obtain better visual features to reduce the epistemic uncertainty of observation given noisy sensor measurements [7], [8], [9]. This often requires additional behaviors, such as back-and-forth movements, to gather more valid measurements. However, the information-seeking behavior is a non-trivial decision-making problem.

Another way is deep reinforcement learning (DRL), which is one of the most successful decision-making methods that enable the robots to deal with future uncertainties by learning from observation features [10]. However, DRL-based approaches suffer from obtaining features given variable size or sudden occurrence of sensor occlusions [11], [12]. To resolve it, researchers utilize a local map as an auxiliary input to get more features of the surroundings via the action selection [13], [14]. Unfortunately, those methods require precise state information of other agents, such as position and velocity, which enforces the excessive use of multiple sensors and techniques to build the whole framework.

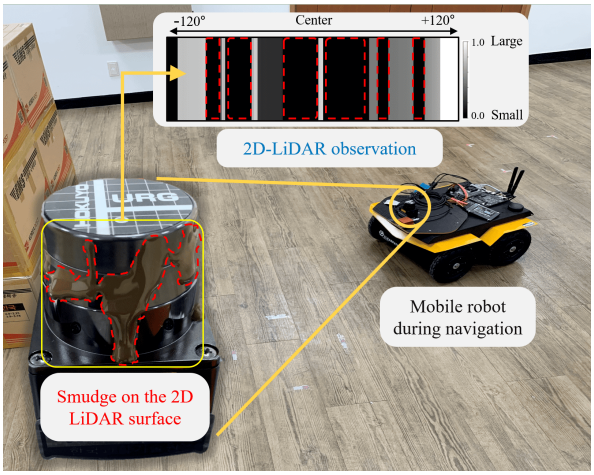

Fig. 1: An example of the sensor occlusion during robot navigation. The unexpected smudge occludes the 2D LiDAR sensor, and corresponding observation data gives empty (i.e., zero) values.

To address the challenges, we propose a *confidence*-based navigation method trained through DRL (CBN-DRL) that selects the best action maximizing the local *confidence*-based rewards to produce both safe exploration and efficient reaching behaviors. To do that, our method combines two modules of networks: *confidence*-prediction network (CP-Net) and point-cloud network. We first build a local *confidence* map from 2-dimensional (2D) LiDAR observations around the robot. To keep track of the uncertainty on the map, CP-Net encodes the dynamics of the *confidence* map and predicts its change on the sensed region given prolonged or eliminated occlusions. We then adopt a point-cloud based network [11] to extract sensor features robust to varying sizes of occlusions. Combining the two outputs of the modules, our CBN-DRL provides a robust and effective navigation performance under sensor occlusions.

We evaluate the proposed method, CBN-DRL, via simulated and real-world environments with static and dynamic obstacles. We show how CBN-DRL robustly works with the complexity of the environments with various sensor occlusion scenarios. By comparing it with three baseline methods, we show the proposed approach gives superior performance in terms of navigation success, collision, and

[†]H. Ryu, M. Yoon, D. Park and S. Yoon are with the School of Computing, Korea Advanced Institute of Science and Technology, South Korea; {hy.ryu, minsung.yoon, daehyung}@kaist.ac.kr, sungeui@kaist.edu [‡]D. Park and S. Yoon are co-corresponding authors.

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9812090

timeout rates. We also demonstrate the robustness of the proposed method with both a simulated mobile robot and a real mobile navigation platform, Jackal from Clearpath Robotics, under highly occurring sensor occlusions.

Our main contributions are as follows.

- We propose a *confidence*-based navigation method that is robust to local uncertainty due to sensor occlusions during 2D-LiDAR based mobile navigation.
- We introduce a sensor-feature extractor that returns occlusion-size agnostic features from the sensor observation adopting a point-cloud based network.
- We demonstrate our method outperforms baseline methods in terms of effectiveness and robustness via simulation with various sensor-occlusion scenarios. We also conduct proof-of-concept experiments with a real robot.

II. Related Work

Sensor occlusion is known as a type of sensor data uncertainty related to partial visibility or observation. In the navigation domain, it is often addressed as the problem of sight restriction due to external objects. For example, Coenen et al. solve the problem of unpredictable moving obstacles behind the doorway as a sight occlusion problem by modeling the probability of the collision [5]. Isele et al. investigate occluded intersection problems in autonomous driving systems [15]. The authors use deep Q networks [16] to learn the optimal behaviors in the tasks. Although these works tackle the occlusion problem, such region lies on the outside of the robot (*altruistic*) while this paper targets to solve the occlusion that occurs on the sensor surface (*egocentric*).

Another way of handling partial visibility is the use of local auxiliary maps or sensor readings. Choi et al. build a local-map based RL agent that can efficiently navigate with a limited field of view (FOV) sensor [13]. Liu et al. also utilize an occupancy grid map and angular map to compensate for the partial observation and demonstrate successful navigation in a crowded environment [14]. Although these works address the limited visibility problem, the agent could sense the front of itself. This narrow FOV is a part of sensor occlusions that we consider, but we aim to handle broader types of occlusion cases where the occlusion occurs on unpredictable regions of the sensor.

In recent years, DRL has been widely studied for the ability to solve complex navigation tasks while handling uncertainty in navigation [17]. The method often takes raw sensory measurements as input and predicts a desired action velocity through a convolutional neural network (CNN) or multi-layer perceptrons (MLP). This end-to-end learning scheme enables direct adoption of the learned policy to the real robots and allows successful navigation in unknown environments [18]. Further, Long et al. propose a DRL-based multi-agent navigation method using the 2D LiDAR observation for each decentralized agent [19]. Jin et al. also prove the RL-agent using 2D laser scanners can perform well in a social navigation task [20]. While these approaches demonstrate the effectiveness for the predefined input observation, the network structure cannot take a variable size of the input,

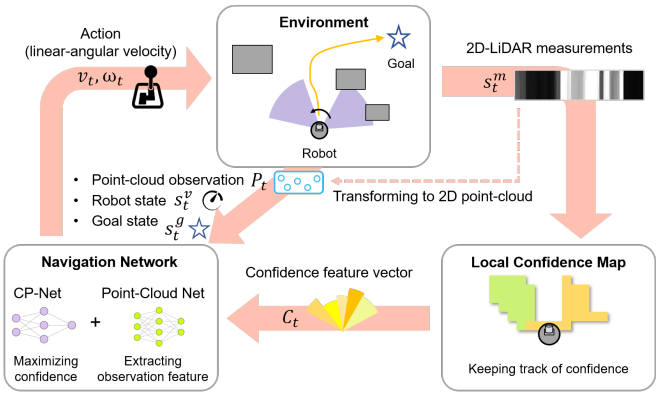

Fig. 2: Overview of CBN-DRL. Under the occlusions, our method tracks the local *confidence* values with the LiDAR observation and calculates the *confidence* feature vector C_t. Our proposed navigation network, which takes input states from the environment and C_t through the *confidence* prediction network (CP-Net) and point-cloud based network, outputs the robot's action, i.e., linear-angular velocity.

such as varying FOV or input dimension. To overcome the restriction, recent works utilize point-cloud based architecture (i.e., PointNet [21]) to process the raw sensor readings as 2D point cloud data [11], [22]. The network processes each point data through MLPs and sends out a fixed size of the features via max-pooling. In this work, we utilize this representation to handle a variable size of the input observation and represent the geometric features more effectively.

III. Approach

We propose a *confidence*-based navigation method that learns to select the best exploration and exploitation action via deep reinforcement learning given sensor occlusions hindering part of the sensor receiving the measurements.

A. Formulation of Navigation Problem under Occlusions

We first formulate the navigation task as Markov decision process (MDP) defined by a tuple (S, A, P, R, γ). Here, S, A, P, and R denote the state space, the action space, a transition function, and a reward function, respectively. $\gamma \leq 1$ represents a discounted factor. Our MDP aims to find an optimal policy $\pi^* : S \rightarrow A$ that maximizes the discounted cumulative reward $G = \sum_{t=0}^{T} \gamma^t \mathbb{E}[R(s_t, a_t)]$, where $s_t \in S$ and $a_t \in A$ at each time step $t \leq T$ [23].

The raised navigation problem requires additional exploratory behavior to complement the partial visibility due to the occluded sensor. To do that, we re-define the objective function to encourage balancing between exploration and exploitation behaviors, motivated by soft actor-critic (SAC) framework [24]:

$$G = \sum_{t=0}^{T} \gamma^t \mathbb{E}\left[R(s_t, a_t) + \alpha H(\pi(\cdot|s_t)) + \beta \xi_{conf}(s_t)\right], \quad (1)$$

where $H(\pi(\cdot|s_t))$ and $\xi_{conf}(s_t)$ denote the entropy of the policy π for randomness and total *confidence* at s_t for uncertainty-guided exploration, respectively. α and β are coefficients to regulate their explorativeness in the policy. To

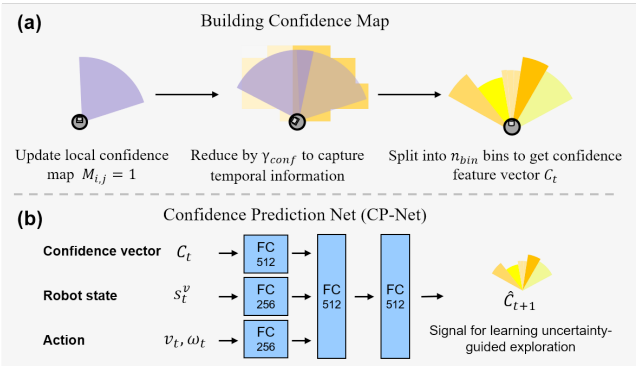

Fig. 3: The overview of building the *confidence* map (a) and *confidence*-prediction network (CP-Net) (b).

obtain the total *confidence* robust to varying size of occlusions, we introduce two networks: *confidence*-prediction network in Sec. III-B and point-cloud network in Sec. III-C. By combining these two via DRL-based navigation framework, we show a *confidence*-based navigation method in Sec. III-D

B. Confidence-Prediction Network (CP-Net)

We propose a *confidence*-prediction network (CP-Net) to predict the future *confidence* values on the perceptible region around the robot given the current observation of the robot. To do that, CP-Net requires a local grid map, we call *confidence* map, tracking the *confidence* values around the robot (see Fig. 3-(a)). The value of each cell $M_{i,j} \in [0,1]$ on the map is 1 when the sensor ray reaches the region. We decrease the value by γ_{conf} to capture the temporal information. Then, we split the forward field of vision (i.e., 2D LiDAR) evenly into the n_{bin} number of cone-shape bins where each bin contains the sum of the belonged *confidence* values. We represent a set of bins as a *confidence* vector $C_t = [c_t^1, c_t^2, ..., c_t^{n_{bin}}]$.

We model the CP-Net that estimates the next confidence vector $\hat{C}_{t+1}$ given a set of state features during navigation as:

$$\hat{C}_{t+1} = \text{CP-Net}(s_t^v, v_t, \omega_t, C_t), \tag{2}$$

where s_t^v is the current velocity, $[v_t, \omega_t]$ is the desired linear-angular velocity pairs, and C_t is the current confidence. To train CP-Net, we minimize a mean-squared-error loss function: $L(\theta) = \frac{||C_{t+1} - \hat{C}_{t+1}||^2}{2}$ where C_{t+1} is a ground-truth *confidence* vector. We design CP-Net to have three independent fully-connected (FC) layers (see Fig. 3-(b)). We calculate the total *confidence* in Eq. (1) using $\xi_{conf}(s_t) = \sum_{i=1}^{n_{bin}} w_{bin}^i \cdot c_t^i$, where w_{bin}^i is the weight of the i-th bin. In this work, we split 120° of the forward field into 60 bins and set all weights as 1.

C. Point-Cloud Network

We introduce a feature extraction network that is robust to varying sizes of sensor occlusions. Conventional navigation methods often lack the flexibility of input in terms of the size or the invalid pattern of data. The methods often filter out the occluded part of data using an occlusion mask that contains the sensor indices of having sensor values lower than the minimum distance threshold defined in [25]. However, masking the occluded indices increases the difficulty for the network to fully represent the valid features from the entangled data. Alternatively, this paper adopts a point-cloud based representation network [11], [22] inspired by PointNet [21] that processes the input point cloud via MLP and then conducts max-pooling over the MLP output to yield a fixed size of outcome features. The point-cloud representation enables the network to extract the valid sensor data with the help of handling a variable input size. Additionally, capturing the spatial information allows a better representation of the geometric information around the robot.

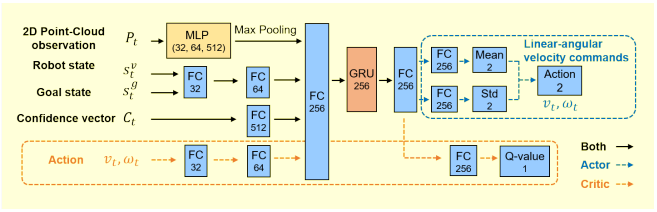

Fig. 4: The proposed navigation network to learn the resilient actions under the occlusions.

We transform the raw sensor measurements into the 2D point-cloud set $P_t \subset \mathbb{R}^{k\times 2}$, where k is the number of unoccluded sensor measurements. In detail, $P_t = \{(s_t^i \cdot cos(\theta_i), s_t^i \cdot sin(\theta_i))\}_{i=1}^k$ where s_t^i is the i-th sensor measurement and θ_i is the corresponding angle of it.

D. Training and Testing

We feed the two network outputs as input for the objective function of SAC. For each time step t, our method builds a *confidence* map and feeds its *confidence* vector C_t as an auxiliary feature of the network input. The network then concatenates the latent features from CP-Net as well as the output from the point-cloud network to feed it into a sequence of FC layers with Gated Recurrent Units (GRU), which is a recurrent neural network [26] to make the network implicitly learn the temporal property of the given states. The network then outputs 2D action distribution parameters for the actor network and 1D Q-value for the critic network.

In this work, we model a sensor occlusion mask $M_t^{occ} = \{m_t^i\}_{i=1}^{n_s}$ where n_s is the dimension of the entire sensor measurements. we set the minimum range distance of the sensor specification as $d_{threshold}$ and specify the sensor index m^i of having a lower value than $d_{threshold}$ as the occluded index. We use this mask to filter out the occluded part of data for training and evaluation.

Below represents our MDP setup with details:

1) State space: A state s_t consists of the raw sensor measurements $s_t^m \in \mathbb{R}^{n_s}$, which we later transform into the point-cloud set P_t, current robot velocity $s_t^v \in \mathbb{R}^2$, goal state $s_t^g \in \mathbb{R}^2$ represented in the polar coordinate of relative distance and angle from the robot, and current confidence vector C_t.

2) Action space: The action of a mobile robot with nonholonomic constraints consists of linear velocity $v_t \in [0, v^{max}]$ and angular velocity $\omega_t \in [\omega^{min}, \omega^{max}]$, where v^{max}, ω^{min}, and ω^{max} are the maximum linear, the minimum angular, and maximum angular velocities, respectively. We normalize the linear velocity into $[0,1]$ and angular velocity into $[-1,1]$ when using in the network.

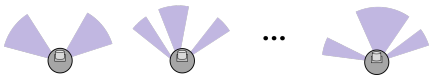

Fig. 5: Illustrations of the various types of occlusion cases.

3) Reward function: We set the reward function as follows:

$$R(s_t, a_t) = R_{nav} + R_{conf}, \tag{3}$$

$$R_{nav} = \begin{cases} r_{goal} & \text{if arrival,} \\ r_{coll} & \text{else if collision,} \\ r_{dist} \cdot (d_{t-1} - d_t) & \text{otherwise,} \end{cases} \tag{4}$$

where d_t is the relative distance between the robot and the goal at time step t and encourages the robot to get closer to the goal. Note that we omit the augments in the reward function of R_{nav} and R_{conf} for clarity. In this work, we set $r_{goal} = 5$, $r_{coll} = -5$, and $r_{dist} = 0.03$. We additionally define *confidence*-related rewards as follows.

$$R_{conf} = r_{secure} + r_{safety}. \tag{5}$$

$r_{secure} = g_{weight} \cdot max(\xi_{conf}(s_t) - \xi_{conf}(s_{t-1}), 0)$ gives small incentive to increase the *confidence* and $g_{weight} \in [0,1]$ is the value of the relative distance weight to the goal expressed as $g_{weight} = \frac{d_t}{d_{init}}$ where d_{init} is the initial distance between the robot and goal at the beginning of the episode. This produces higher exploratory action of increasing *confidence* when the robot locates far from the goal. We then reduce it gradually as the robot reaches the goal, referring to the concept of the exploration strategy of SAC.

Additionally, we define *confidence* safety reward $r_{safety} = -w^{safety} \cdot (1 - \xi_{conf}(s_t^{safety}))$ that the robot gets a negative penalty if entering the region with lower *confidence*. Inspired by the social-safety zone in [20] that gives penalties when the robot enters pedestrians' safety zone, we represent the *confidence* safety zone as a rectangular shape with $width = r_{robot}$, $length = \Delta d \cdot r_{robot}$, where r_{robot} is the radius of the robot model and Δd is the relative distance that the robot can move within a short time step. We set $\Delta d = 3.5$ and the safety weight $w^{safety} = 0.05$. This safety zone is projected into the robot frame.

IV. Experimental Setup

We performed quantitative and qualitative studies in simulated and real-world navigation setups.

A. Quantitative Evaluation

Our quantitative evaluation uses a Python-based simulated navigation setup. We set a mobile robot with a 240° FOV of 2D LiDAR sensor, which is similar to the sensor used in the real-world experiment (see Fig. 9). The simulated environment consists of static obstacles (i.e., wall) as well as dynamic obstacles (i.e., movable objects), where we controlled the motion of the dynamic obstacles using the optimal reciprocal collision avoidance (ORCA) method [27]. Given a randomly selected goal, our robot with a random initial position is designed to reach a goal. We stopped the simulation when reaching a goal (i.e., success) or running over 500 steps (i.e., failure). The robot is a car-like robot of a rectangular shape with $width = 0.48\,\text{m}$ and $length = 0.6\,\text{m}$ represented in a 2D space. The maximum linear and angular velocities of the robot are $1.2\,\text{m/s}$ and $1.0472\,\text{rad/s}$, respectively. The dynamic obstacle is a circular shape with a radius of $0.3\,\text{m}$, and its maximum velocity is $0.2\,\text{m/s}$.

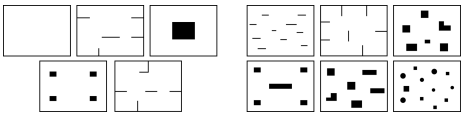

Fig. 6: Simulation environments used for training (left) and evaluation (right).

We evaluate our method and three baseline methods, explained in Sec. IV-B, by randomly generating start and goal positions as well as an occlusion mask M^{occ} per episode in a given map, where the mask is activated around 10-to-20th steps during navigation. Fig. 5 shows the illustration of the arbitrary sensor occlusions. After training a method until the success rate converged in a group of randomly selected training maps such as Fig. 6 left, we evaluated the method in another group of maps (see Fig. 6 right) with and without randomly generated 5 moving obstacles. Finally, we collected 500 times of trials from each static or dynamic test environment per map. Overall, we run total $3,000$ per type of environment.

We then compared the following evaluation metrics:

- **Success Rate:** The ratio of the number of episodes that the robot reaches the goal successfully over the total number of trials.
- **Collision Rate:** The ratio of the number of episodes that the robot collides with any obstacle over the total number of trials.
- **Timeout Rate:** The ratio of the number of episodes that the robot fails to reach the goal within the maximum steps except for collision cases over the total number of trials.

In addition, to train the DRL-based method including baselines, we used SAC with an Adam optimizer [28] with a $3 \cdot 10^{-4}$ learning rate and γ=0.99. We automatically tuned the entropy coefficient α, referring to the improved version of SAC [29]. For our method, we trained the model for around four days and used rectified linear unit (Relu) for CP-Net, and LeakyRelu activation unit [30] for the rest of the layers. We used a $5 \cdot 10^5$ replay-buffer size, a 0.005 target smoothing coefficient, and a 20 sequence length of GRU as well as $\beta = 0.1$ and $\gamma_{conf} = 0.925$.

B. Baseline Methods for Evaluation

We implemented a classic collision avoidance method, dynamic window approach (DWA) [31], and two DRL methods trained with either raw sensor [19] or a point-cloud based architecture [11]:

- **Dynamic window approach (DWA) [31]:** DWA, widely used collision-avoidance algorithms for mobile robots, computes the safe area and selects an optimal action command. We used only the unoccluded part of sensor data to search the safe area.
- **Raw sensor-based architecture *(RawSensor)* [19]:** *RawSensor* takes six consecutive sensor observations

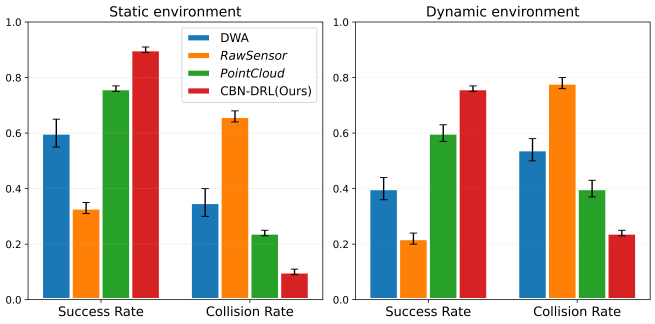

Fig. 7: Evaluation results in static and dynamic environments where the occlusions block 50% of the sensor measurements. The range of success and collision rate is 0 to 1.0.

through the two CNNs and three FC layers and then outputs the command actions. We further concatenated the occlusion mask M^{occ} with the corresponding sensor measurements s_t^m to let the network identify the occluded features as it requires fixed input.

- **Point-cloud based architecture *(PointCloud)* [11]:** *PointCloud* uses SAC instead of DDPG in the original work [32] to raise its exploration performance [29]. We also modified a part of rewards to fairly compare with other methods, and adopted GRU [33] to reduce the training time [26]. We further designed *PointCloud-Upper bound*, trained without sensor occlusion, to measure the best performance which the plain point-cloud method can achieve.

Note that we filtered the sensor occlusion out using the occlusion masks to make each baseline perform at best.

C. Demonstration Setup

We demonstrated the proposed method on a real-world reaching scenario with a real mobile robot, Clearpath Jackal (see Fig. 9). The robot is equipped with NVIDIA Jetson TX2 as a mainboard and Hokuyo URG-04LX-UG01 as the 2D LiDAR sensor. We run our navigation method via Robot Operating System (ROS) [34].

V. Results

We first analyze the overall performance of our method and the baselines in the static and dynamic environments where the sensor occlusion blocks 50% of FOV. Fig. 7 shows the point-cloud based architecture, *PointCloud* (green) and CBL-DRL (red), outperforms DWA (orange) and *RawSensor* (blue) in terms of both success rate and collision rate.

The point-cloud representation helps to filter out occluded data and set remaining measurements into valid points. Accordingly, the network can represent the feature more effectively. Further, CBN-DRL shows the best performance since it utilizes an auxiliary local information from C_t while considering the *confidence* around the robot. Additionally, our method explores the regions with uncertainty to prevent potential collisions. This led to avoid the collision from either static or dynamic obstacles.

In contrast, the plain *PointCloud* method could only prevent collisions when the LiDAR sensor unintentionally perceives the obstacles and could not avoid collisions from the region

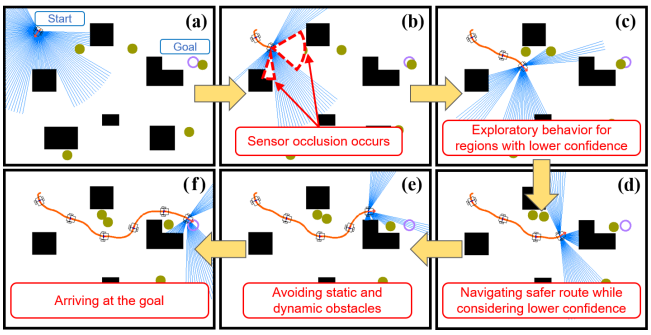

Fig. 8: Trajectories of the proposed method in the dynamic environment of having sensor occlusion (black and olive color indicate the static and dynamic obstacles, respectively). In the middle of the navigation, the occlusion occurs (b) and blocks the sensory perception, but the agent avoids collision with the obstacles and successfully arrives at the goal.

of the sensor that is occluded or dynamically approaching obstacles. As a result, the gap in the success rate between CBN-DRL and *PointCloud* is bigger in dynamic environments than in static environments. The whole trajectories of the CBN-DRL are shown in Fig. 8. Another interesting point is that DWA shows a higher timeout rate in static (0.06 ± 0.02) and dynamic (0.03 ± 0.01) environments than the others; its myopic behaviors were often stuck due to the highly constrained visibility from the occlusion. This result demonstrates that solely using the classical method lacks dealing with the occlusion cases as it has no ability to actively compensate for the partial visibility. Consequently, the success rate drastically dropped in the dynamic environment.

To evaluate the robustness, we conducted experiments with various sensor-occlusion ratios from 0% (no sensor occlusion) to 75%. Fig. 10 shows *PointCloud-upper bound* performs better or comparable to CBN-DRL under no sensor occlusion cases. As *PointCloud-Upper bound* is trained without the sensor occlusion, the network concentrates on extracting features from the unoccluded data. However, as the sensor occlusion ratio grows, our method performs better than the other baseline methods. Compared to the cases under the occlusion ratio of 25%, the performance gap of CBN-DRL and other methods significantly increases under the ratio of 50%. *PointCloud-Upper bound* dealt with a small portion of occlusion, i.e., 25%, but it lacks handling unexpected occluded data. On the other hand, our method, CBN-DRL, achieves higher performance with the help of handling partial visibility. Another point is that *PointCloud* does not perform as much as CBN-DRL even it is trained with the occluded data. This result shows that using only occluded data cannot contribute to dealing with the sensor occlusion, but considering the local *confidence* can be a viable solution to the occlusion cases. Although performances of other methods decrease under the severe occlusion (i.e., 75%), ours still achieved higher success rates. These results demonstrate that our method can behave more resiliently under various sensor occlusions.

Further, we conducted an ablation study of our method in static and dynamic environments with a 50% sensor

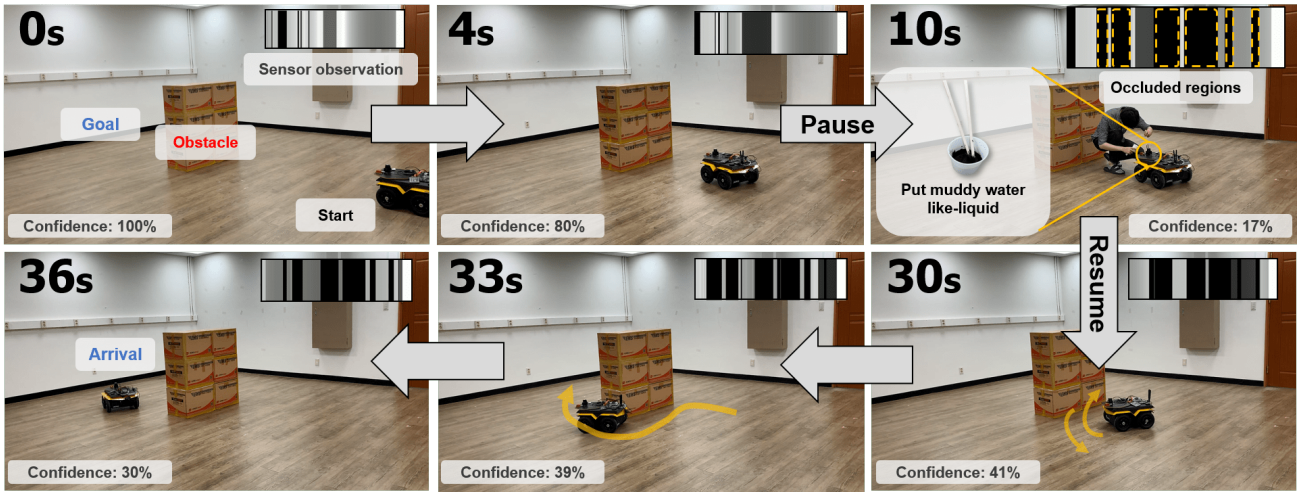

Fig. 9: Demonstration of our method under the sensor occlusions. To simulate the occlusions, we temporarily paused all functionalities of the robot and put the muddy water-like liquid on the sensor surface. The robot then explored the region with lower *confidence* and reached the goal without collisions.

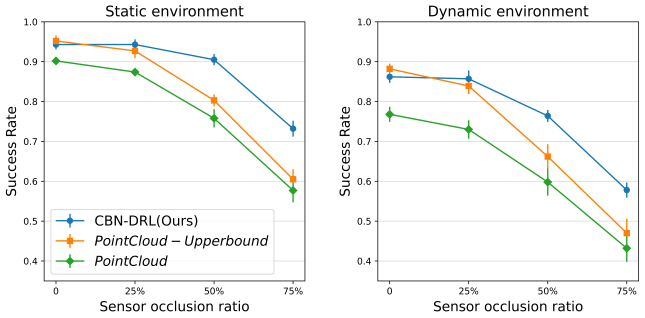

Fig. 10: Evaluation under the various ratio of sensor occlusions from 0% (no occlusion) to 75%. *PointCloud-Upper bound* indicates the method trained with occlusion-free dataset.

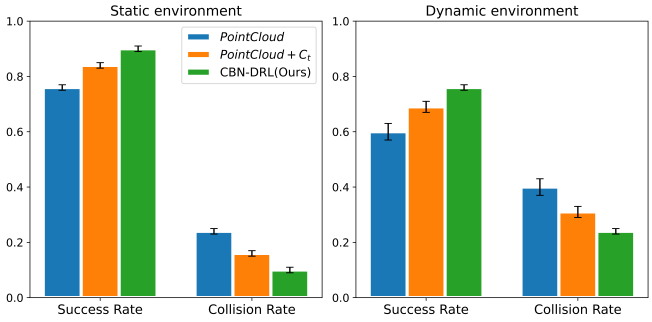

Fig. 11: Ablation study of the proposed method. *PointCloud*+C_t indicates the method using additional *confidence* vector C_t to the network's feature (i.e., a navigation framework using local map information).

occlusion ratio. We compare our method with *PointCloud* and *PointCloud*+C_t, which uses the *confidence* vector C_t as an additional feature to the navigation framework to verify the effectiveness of the local information in the partial visibility cases. Fig. 11 shows the comparison results of using each proposed method. Simply using the local *confidence* information, i.e., *PointCloud*+C_t (orange), contributes a higher success rate compared to the *PointCloud* (blue). As an auxiliary input feature, the local *confidence* inherently helps to represent geometric features of surroundings. Additionally, our method, which uses local map features and encourages to seek higher *confidence*, shows the highest success rate. This result demonstrates that keeping and pursuing higher sensing *confidence* of the surroundings compensate for the insufficient *confidence* region to check.

Finally, we demonstrated the proposed navigation method under occlusion via real-world experiments. We transferred a learned policy to the real robot and conducted experiments in a 6 m by 7 m size room. To simulate the sensor occlusions, we temporarily paused all of the robot's functional nodes during the navigation and put the muddy water-like liquid on the sensor to block its perception as Fig. 9. Once we resumed the operation, the robot showed the behavior of seeking higher local *confidence* and successfully navigated to the goal. For more information, please refer to the attached video.

VI. Conclusion

In this paper, we proposed a *confidence*-based method with DRL (CBN-DRL) robust to local uncertainty due to sensor occlusions during 2D-LiDAR based robot navigation. Our method combines a *confidence*-prediction network (CP-Net) and point-cloud network to encourage the robot to explore the uncertain local region and effectively represent the variable size of the occluded sensor data. The results showed that our method outperformed baseline methods under the highly occurring sensor occlusion both in static and dynamic environments. We also demonstrated that the learned policy robustly works in real-world occlusion scenarios.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) (IITP-2015-0-00199) and the National Research Foundation of Korea (NRF) (No. NRF-2021R1A4A3032834) funded by the Korea government (MSIT).

REFERENCES

[1] J. R. V. Rivero, I. Tahiraj, O. Schubert, C. Glassl, B. Buschardt, M. Berk, and J. Chen, "Characterization and simulation of the effect of road dirt on the performance of a laser scanner," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[2] S. Zang, M. Ding, D. Smith, P. Tyler, T. Rakotoarivelo, and M. A. Kaafar, "The impact of adverse weather conditions on autonomous vehicles: how rain, snow, fog, and hail affect the performance of a self-driving car," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 103–111, 2019.

[3] M. Trierweiler, P. Caldelas, G. Gröninger, T. Peterseim, and C. Neumann, "Influence of sensor blockage on automotive lidar systems," in *Proceedings of IEEE Sensors*, 2019, pp. 1–4.

[4] N. Akai, Y. Kakigi, S. Yoneyama, and K. Ozaki, "Development of autonomous mobile robot that can navigate in rainy situations," *Journal of Robotics and Mechatronics*, vol. 28, no. 4, pp. 441–450, 2016.

[5] S. Coenen, J. Lunenburg, M. van de Molengraft, and M. Steinbuch, "A representation method based on the probability of collision for safe robot navigation in domestic environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 4177–4183.

[6] S. Bouraine, T. Fraichard, and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 174–179.

[7] H. Yang, J. Lim, and S.-e. Yoon, "Anytime rrbt for handling uncertainty and dynamic objects," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4786–4793.

[8] B. Lütjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8662–8668.

[9] T. Fan, P. Long, W. Liu, J. Pan, R. Yang, and D. Manocha, "Learning resilient behaviors for navigation under uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 5299–5305.

[10] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[11] F. Leiva and J. Ruiz-del Solar, "Robust rl-based map-less local planning: Using 2d point clouds as observations," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5787–5794, 2020.

[12] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Safe reinforcement learning with scene decomposition for navigating complex urban environments," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1469–1476.

[13] J. Choi, K. Park, M. Kim, and S. Seok, "Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5993–6000.

[14] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5671–5677.

[15] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2034–2039.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning*, 2013.

[17] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[18] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 31–36.

[19] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6252–6259.

[20] J. Jin, N. M. Nguyen, N. Sakib, D. Graves, H. Yao, and M. Jagersand, "Mapless navigation among dynamics with social-safety-awareness: a reinforcement learning approach from 2d laser scans," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6979–6985.

[21] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[22] W. Zhang, N. Liu, and Y. Zhang, "Learn to navigate maplessly with varied lidar configurations: A support point-based approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1918–1925, 2021.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018, pp. 1861–1870.

[25] T. Raj, F. H. Hashim, A. B. Huddin, M. F. Ibrahim, and A. Hussain, "A survey on lidar scanning mechanisms," *Electronics*, vol. 9, no. 5, p. 741, 2020.

[26] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[27] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 1928–1935.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[29] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," in *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[30] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013, p. 3.

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

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Software*, 2009.