Thirty-first International Symposium on Artificial Life and Robotics 2026 (AROB 31st 2026),
Eleventh International Symposium on BioComplexity 2026 (ISBC 11th 2026),
B-Con Plaza, Beppu, Japan, January 21-23, 2026

Vision-Based Sim2Real Transfer of Map-Free Reinforcement Learning Navigation with Semantic Segmentation and Multi-Sensor Fusion

Daisuke Amano[1†] and Kazuyuki Morioka[2]

[1,2]Graduate School of Advanced Mathematical Sciences, Meiji University, Tokyo, Japan
(E-mail: adaisuke716@outlook.jp)

Abstract: This study proposes a vision-based Sim2Real transfer framework for map-free deep reinforcement learning (DRL) navigation, integrating semantic segmentation and multi-sensor fusion to achieve stable autonomous navigation without prior maps. The main contribution of this work is to demonstrate that a vision-based DRL policy trained entirely in simulation can be successfully transferred to real-world environments through an integrated perception and localization framework. The proposed system enables goal-directed navigation based on multi-sensor fusion of visual, inertial, and wheel-encoder information, providing a lightweight and scalable solution for low-cost mobile robots.

Keywords: Sim2Real, Autonomous Navigation, Reinforcement Learning, Semantic Segmentation, Multi-Sensor Fusion

1. INTRODUCTION

As mobile robots become increasingly integrated into real-world applications, the demand for robust and adaptive navigation capabilities continues to grow. Autonomous mobile robots are increasingly expected to navigate in complex indoor environments without relying on pre-built maps. Conventional navigation pipelines depend on accurate localization against prior maps, leading to high sensor costs and limited practicality in dynamic settings. Reinforcement learning (RL) offers an alternative by learning navigation behaviors directly from sensor observations in simulation [1]. However, transferring such policies to real robots remains difficult due to the Sim2Real gap caused by visual appearance differences, sensor noise, and localization errors. Addressing this gap is essential for reliable deployment of vision-based RL navigation in real-world environments.

This paper proposes a vision-based Sim2Real transfer framework for map-free deep reinforcement learning (DRL) navigation that integrates semantic segmentation and multi-sensor fusion. As illustrated in Fig. 1, the framework consists of two main stages: policy learning in simulation and Sim2Real transfer to real-world environments. A navigation policy is trained entirely in simulation using binarized visual observations and structured vector inputs, to learn stable goal-directed behaviors without relying on prior maps.

To bridge the gap between simulation and reality, a visual perception pipeline is designed to reproduce the same simplified visual representation used during training. Real-world RGB images are converted into binary representations through semantic segmentation, allowing the visual input in real environments to remain consistent with the simulation observations. This alignment reduces the appearance gap between simulation and real-world images, thereby enabling reliable Sim2Real transfer.

In addition, a self-localization module based on multi-sensor fusion is introduced to support real-world navigation. An Extended Kalman Filter (EKF) integrates information from vision, inertial sensing, and wheel encoders to provide robust ego-motion estimation. The resulting pose information is used to generate vector observations compatible with the simulation-trained policy, thereby enabling stable execution of the learned navigation behavior in real environments.

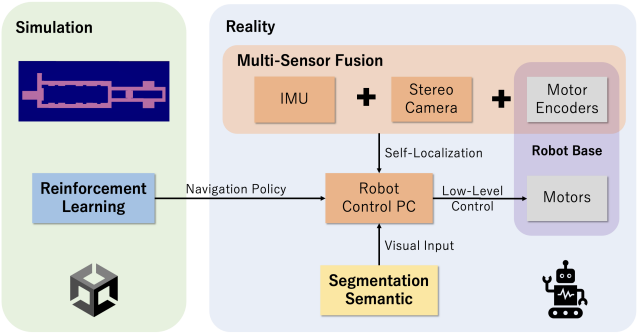

Fig. 1. Proposed vision-based Sim2Real transfer framework

2. RELATED WORKS

Mobile robot navigation based on DRL has been widely studied. Early approaches demonstrated that RL can be applied to navigation and obstacle avoidance using range sensors or visual information [1]. With the development of deep neural networks, more recent studies have focused on learning navigation policies directly from high-dimensional sensor observations [2-4].

Many navigation systems employ range-based sensors such as 2D-LiDAR, since distance measurements provide stable geometric information that is suitable for learning collision avoidance and goal-directed behaviors. LiDAR-based DRL approaches have shown reliable performance in structured indoor environments [2]. However, LiDAR sensors increase hardware cost and system complexity, which may limit their applicability to lightweight platforms.

To reduce dependency on costly sensors, visual-based navigation using DRL has attracted attention [3]. Similar to LiDAR, depth cameras provide explicit geometric distance information that can be directly used for navigation. However, depth cameras are sensitive to strong illumination and may fail under adverse lighting conditions.

† Daisuke Amano is the presenter of this paper.

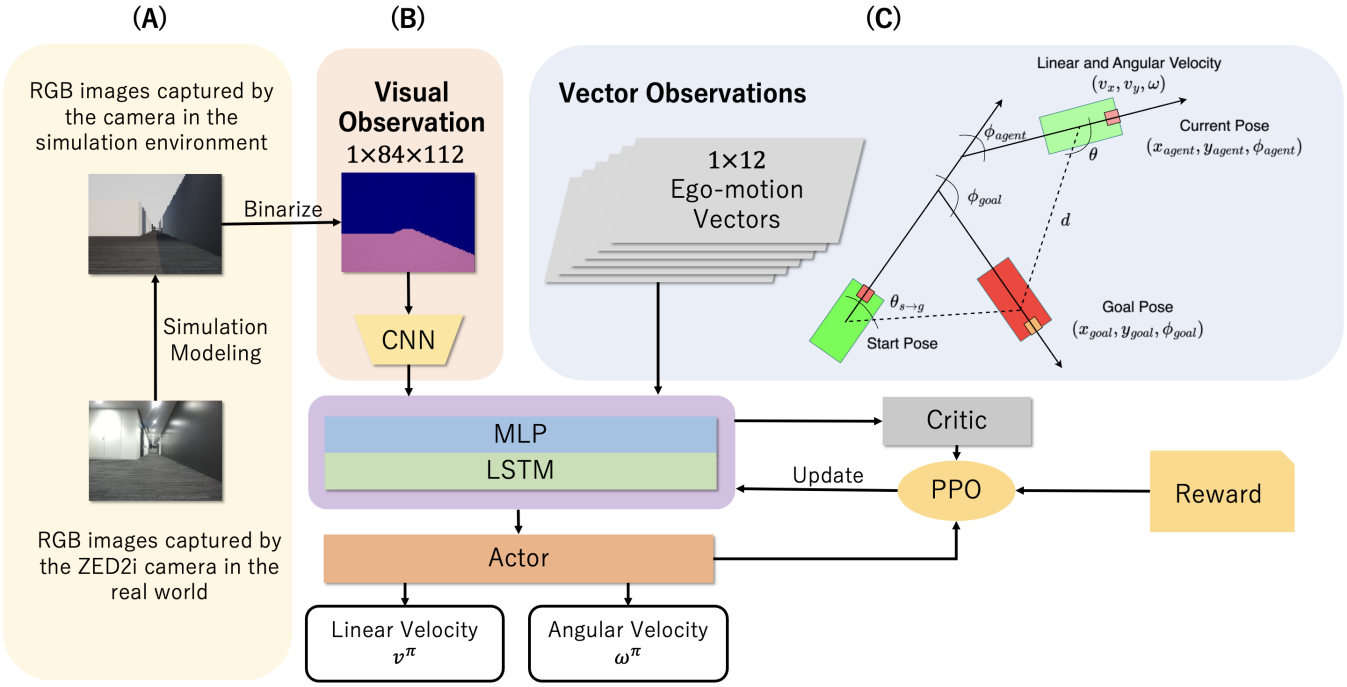

Fig. 2. Overview of the RL system architecture. (A) real-world environment modeling in Unity, with RGB images shown to illustrate the visual consistency between the simulated and real scenes, (B) RGB images are binarized into a binary environment representation to simplify visual observation input, and encoded by a CNN, (C) structured vector observations representing ego-motion and goal-relative geometry.

Alternatively, monocular camera images can be used to infer depth information and construct range-like representations for navigation, achieving effects similar to those of depth-sensing cameras [4]. However, the depth estimation from monocular images is sensitive to visual ambiguity and environmental variations.

To improve robustness of visual perception, semantic segmentation has been incorporated into navigation frameworks. A navigation method using binary semantic segmentation abstracts visual input and stabilizes policy learning [6]. However, the self-localization is still performed using LiDAR, and relative distance and orientation information to the target are derived from LiDAR-based self-localization.

From these studies, it can be observed that range- and depth-based approaches provide stable geometric information at the cost of additional sensors, whereas monocular vision-based navigation enables simpler sensor configurations but suffers from perceptual ambiguity. This motivates research on Map-Free navigation systems that leverage abstracted visual information and auxiliary observations to achieve robust action learning without relying on prior maps.

3. LEARNING A NAVIGATION POLICY IN SIMULATION

To enable map-free navigation without relying on prior environmental knowledge, we train an RL policy entirely in a simulation environment implemented using Unity. This section describes the key components involved in learning such a navigation policy, including the simulation environment that employs a binary-simplified representation of the scene, the policy network architecture, the observation design, and the reward function used during training.

3.1. Simulation Environment

The navigation policy is trained in a simulation environment implemented in Unity. To enable Sim2Real transfer, the corridor scene in Unity is modeled by reproducing the geometry and camera viewpoint of the real indoor corridor, as illustrated in Fig. 2(A). In the real environment, RGB images are captured by the camera mounted on the robot, while in the simulation environment, a virtual camera is placed at the corresponding position and orientation. By aligning the corridor structure and the first-person RGB view between simulation and reality, the simulated images can be used to learn a navigation policy that is consistent with the visual observations obtained in the real world. These simulated RGB images are then passed to the visual observation pipeline described later.

3.2. Policy Network Architecture

This study adopts Proximal Policy Optimization (PPO) [7] and constructs an actorcritic policy network. Visual observations are encoded by a CNN, while vector observations are directly fed into an MLP. The extracted features are concatenated and processed by an LSTM layer to retain temporal information under partial observability. From the LSTM output, the actor module generates continuous control actions $a = (v_\pi, \omega_\pi)$ for the robot. The critic module evaluates the current state based on the same feature representation. PPO updates both the actor and the critic using collected trajectories and rewards, enabling stable improvement of the policy during training.

3.3. Observation Space

The observation space consists of two complementary modalities: a binary-simplified visual observation and a structured vector observation, as illustrated in Fig. 2(B)-(C). The configuration is chosen to facilitate stable learning in simulation and to provide a representation that naturally extends to real-world sensing.

Visual observation: As shown in Fig. 2(B), RGB images in simulation are converted into a binarized representation that separates the floor from the surrounding structures. This representation suppresses variations in color and illumination, enabling stable learning of a navigation policy. The resulting 112×84 grayscale image is encoded by a CNN and passed to the subsequent MLP-LSTM module in the policy network.

Vector observation: In addition to visual input, we incorporate vector observations that encode the agent's geometric relationship to the goal, as depicted in Fig. 2(C). These vector quantities are defined as follows:

1) **Relative position and orientation of the agent** $(p_{\mathrm{agent}}, \phi_{\mathrm{agent}})$: The current position and orientation of the agent are represented as the difference from its initial pose. The orientation angle is constrained to the range $[-\pi, \pi]$. Such a representation has been reported to be effective for map-free spatial understanding [3].
2) **Relative position and orientation of the goal** $(p_{\mathrm{goal}}, \phi_{\mathrm{goal}})$: The goal pose is expressed as the relative position and orientation with respect to the agent's initial pose [3]. The orientation angle is also constrained to the range $[-\pi, \pi]$.
3) **Relative direction and distance to the goal** (θ, d): The bearing and Euclidean distance from the agent to the goal are denoted by θ and d, respectively. The angle θ is defined as the directed angle measured from the agent's forward direction and is normalized to the range $[-1, 1]$ for use as input to the neural network.
4) **Linear and angular velocities** (v_x, v_y, ω): These are dynamic observations that mimic an IMU: the linear velocities v_x and v_y and the angular velocity ω are computed from the differences between the current and previous poses. This information helps the policy to capture the agent's motion response and pose changes, and has been reported to contribute to stabilizing the learned policy [5].
5) **Global bearing from start to goal** $(\theta_{s \to g})$: This is the unique bearing from the start position to the goal at the beginning of the episode. The value is provided as a constant throughout the episode during training, and is normalized to the range $[-1, 1]$.

By integrating these structured geometric cues with the visual input, the policy can infer navigation-relevant spatial relationships that are difficult to estimate reliably from vision alone, thereby enhancing robustness and supporting later Sim2Real transfer.

3.4. Reward Design and Training Setup

To learn a stable and efficient navigation policy, we employ a composite reward function consisting of task-oriented rewards and several penalties designed to suppress undesirable behaviors. The overall reward configuration is summarized in Table 1, and the details of each component are described below.

Task-oriented rewards: A success reward is provided when the agent reaches the goal, directly reinforcing task completion. In addition, to encourage progressive movement toward the target, we introduce a distance reward based on Yokoyama et al.'s [3] weighted-sum index of distance and directional consistency:

$$J_{\mathrm{nav}} = d + w \cdot \theta, \tag{1}$$

where d denotes the Euclidean distance to the goal, θ denotes the relative heading angle, and the weight coefficient is set to $w = 0.1$. The reward at each step is given by the change ΔJ_{nav} between consecutive time steps. The relative angle θ is evaluated only when the distance to the goal is less than $2\,\mathrm{m}$; otherwise, a constant value of 3.14 is substituted. This design suppresses excessive alignment behavior when the agent is far from the target and helps prevent wall-collision-prone motions.

Penalties for unstable behaviors: Multiple penalties are incorporated to stabilize learning, applied as follows:

1) **Collision penalty**: when the agent contacts obstacles.
2) **Slack penalty**: at every step to discourage stagnant motion.
3) **Move-back penalty**: to suppress negative linear velocity.
4) **Oscillatory penalty**: when repeated small-amplitude shaking occurs.
5) **Switch penalty**: when forward-backward switching occurs more than 3 times within 5 steps.

These penalties prevent unstable or unsafe actions and promote smooth forward navigation.

Table 1. Reward settings

Reward Type	Reward Value
Success Reward	+30
Distance Reward	ΔJ_{nav}
Collision Penalty	-20
Slack Penalty (per step)	-0.01
MoveBack Penalty (negative linear velocity)	-0.01
Oscillatory Penalty (movement <0.05 m within 0.2 s)	-0.01
Switch Penalty ($\geq$ 3 switches within 5 steps)	-0.01

Training setup: The CNN encoder, MLP module, LSTM layer, and the actor and critic heads are jointly optimized throughout training, as shown in Fig. 2. The agent is trained for 700,000 steps in the Unity simulation environment, during which the navigation performance gradually converges.

4. SIM2REAL TRANSFER APPROACH

To apply the simulation-trained policy to the real robot, it is necessary to compensate for the Sim2Real gap caused by differences in observations. This section describes two key

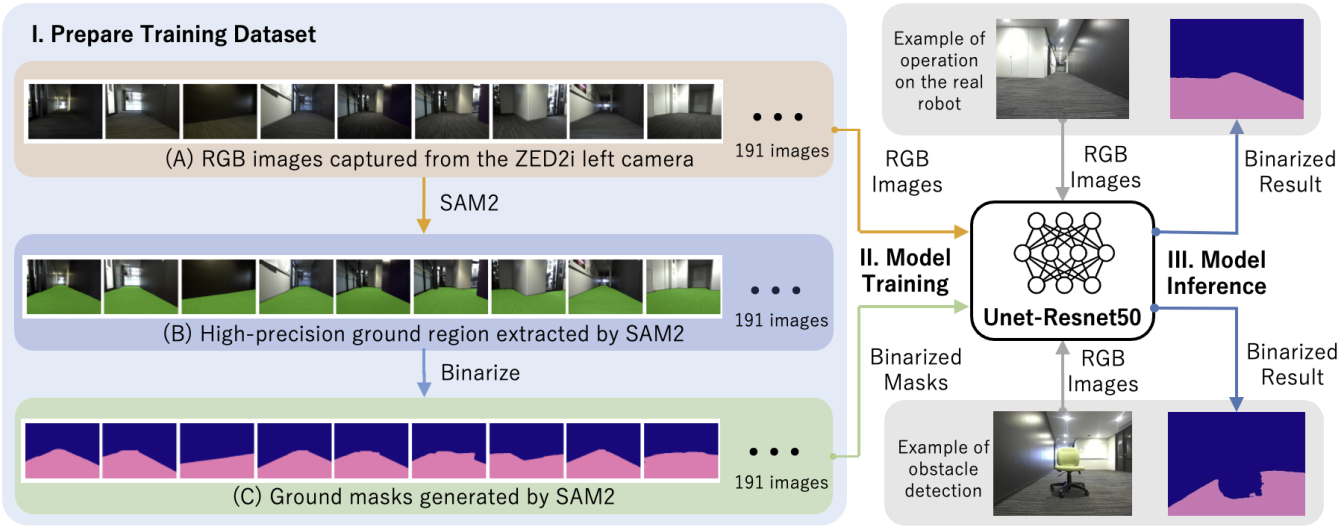

Fig. 3. Overview of the training-data preparation and segmentation pipeline, including I. construction of training dataset using SAM2 to generate ground-truth masks, II. training of a UNet-ResNet50 segmentation model, and III. onboard inference for real-time ground-region extraction.

components for real-world deployment: construction of a binary semantic segmentation model to extract ground regions, and multi-sensor fusion for robust localization.

4.1. Visual Alignment via Semantic Segmentation

In the simulation environment, the observation was simplified by binarizing the scene into ground and obstacles, which contributed to stable policy learning. However, in the real world, raw RGB images cannot reproduce the same representation, resulting in a mismatch with the simulation. To resolve this discrepancy, we construct a binary segmentation model that extracts ground regions from real-world RGB images, enabling observations that closely match the binarized representation used in Unity simulation.

Dataset Construction: To reproduce the binarized visual representation used in simulation, a dataset with high-quality ground labels is required. Conventional plane-detectionbased mask generation [6] suffers from insufficient label accuracy, which can negatively affect training stability. We then applied SAM2 [8] to extract ground regions with high precision, as illustrated in Fig. 3(B). Based on these extracted regions, we generated binarized masks as in Fig. 3(C). Finally, we constructed the training dataset by pairing the RGB images with their corresponding binarized masks.

Training and Evaluation of the Binary Segmentation Model: Although SAM2 provides highly accurate segmentation, its high computational cost makes it unsuitable for real-time onboard deployment. Therefore, a lightweight UNet-ResNet50 model was trained using SAM2-generated masks as supervision to enable efficient inference on the robot.

Fig. 4 shows the training curves, indicating stable convergence in terms of Dice loss and IoU. Representative real-world inference results are shown in Fig. 3, which summarizes the overall pipeline from training-data generation to onboard deployment.

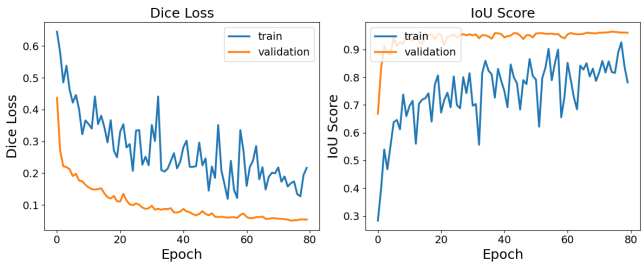

Fig. 4. Learning curves of the segmentation model

4.2. Multi-Sensor Fusion for Robust Localization

To obtain a stable and drift-resistant estimate of the robot's pose, we integrate wheel-encoder odometry, visual odometry (VO), and IMU data using an extended Kalman filter (EKF). Each sensing modality provides complementary information, and their fusion compensates for the individual weaknesses of the sensors.

The EKF receives nine observation variables from three sensing sources, summarized as follows:

1) **ZED2i Visual-Inertial Odometry (VIO):**
ZED2i VIO estimates the relative camera motion by jointly tracking visual features and inertial measurements, providing planar position observations $(x_{\text{vio}}, y_{\text{vio}})$. Although VIO can output a yaw estimate ϕ_{vio}, its heading is sensitive to texture-poor environments, illumination changes, and rapid motion, which may cause sudden jumps. Therefore, in this work, we use VIO only for (x, y) observations, and do not use ϕ_{vio} for EKF correction.
2) **Kobuki Wheel Odometry (WO):**
Wheel odometry provides ego-motion estimates of forward velocity v_x and yaw rate ω from motor encoder readings. Since the robot follows a differential-drive mechanism, lateral motion is physically constrained, and v_x and ω are used for state propagation in the EKF. Here, v_x is used in the prediction step, while ω

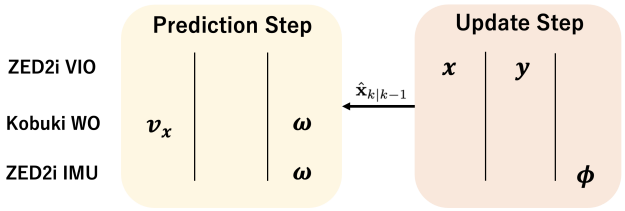

Fig. 5. EKF framework for multi-sensor localization

from WO is used as a redundant rotational-rate input together with IMU. Nevertheless, systematic errors such as wheel slip cause drift over time.

3) **ZED2i IMU:**
The ZED2i provides a high-frequency IMU, which outputs orientation ϕ_{imu} and angular velocity ω_{imu}. Unlike vision-based estimates, IMU measurements are not affected by visual degradation. In this work, ω_{imu} is used as the rotational-rate input in the prediction step, while ϕ_{imu} is incorporated in the update step to suppress abrupt heading jumps in ϕ_{vio}.

The EKF integrates heterogeneous observations through a prediction–update cycle, as illustrated in Fig. 5. The system state is defined as

$$\hat{\mathbf{x}}_k = [x_k,\ y_k,\ \phi_k,\ v_{x,k},\ \omega_k]^\top, \tag{2}$$

where (x_k, y_k, ϕ_k) denote the planar position and heading, and $(v_{x,k}, \omega_k)$ represent the forward velocity and yaw rate.

During the **Prediction Step**, the prior state estimate is propagated using a differential-drive motion model and wheel-encoder-based velocity estimates:

$$\hat{\mathbf{x}}_{k|k-1} = f\big(\hat{\mathbf{x}}_{k-1|k-1},\ \mathbf{u_k}\big) + \mathbf{w_k}, \tag{3}$$

where $\mathbf{u}_k = [v_x,\ \omega]^\top$ represents the estimated linear velocity and yaw rate, and $\mathbf{w_k}$ represents zero-mean process noise. This prediction provides a smooth short-term estimate, but accumulated uncertainty leads to drift over time.

During the **Update Step**, sensor observations are incorporated through the measurement model

$$\mathbf{z}_k = h\big(\hat{\mathbf{x}}_{k|k-1}\big) + \mathbf{v_k}, \tag{4}$$

where $\mathbf{z_k}$ is a composite observation vector consisting of velocity estimates from VIO, planar pose observations (x, y, ϕ) from wheel odometry, and rotational measurements $(\phi_{\text{imu}}, \omega_{\text{imu}})$ from the IMU. The measurement function $h(\cdot)$ maps the predicted state to the observation space by selecting and stacking the corresponding state components associated with each sensor modality. The posterior state estimate is obtained as

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K_k}\left(\mathbf{z_k} - \mathbf{h}\big(\hat{\mathbf{x}}_{\mathbf{k|k-1}}\big)\right), \tag{5}$$

where $\mathbf{K_k}$ denotes the Kalman gain. By combining these complementary observations, the EKF corrects accumulated linear and rotational errors. Specifically, VIO refines short-term linear motion, wheel odometry stabilizes planar pose estimates, and IMU measurements suppress yaw drift through redundant angular constraints [9].

Fig. 6 compares three localization methods: WO, VIO, and EKF fusion. The robot is manually driven along a closed-loop trajectory of approximately 40 m, completing one full lap and returning to the starting position. Wheel odometry exhibits gradual drift over time, while visual odometry is susceptible to local fluctuations, particularly in texture-poor regions. As shown in the trajectory comparison, the EKF-fused estimate does not strictly coincide with either individual source, but achieves the smallest deviation from the starting position after completing a full loop. This indicates that the proposed fusion framework effectively suppresses cumulative drift and local instability by balancing complementary error characteristics of different sensors, leading to a more consistent and stable pose estimation.

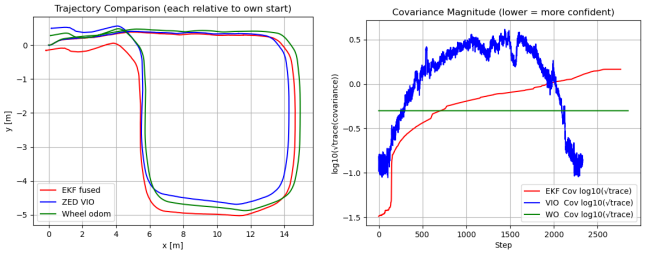

Fig. 6. Comparison of localization accuracy and covariance among WO, VIO, and EKF fusion

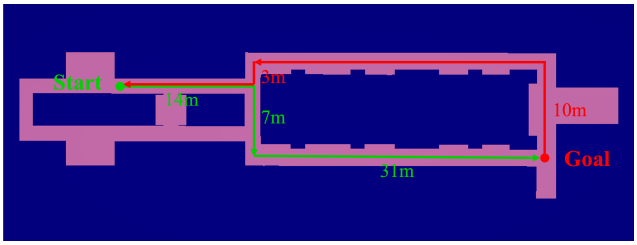

Fig. 7. Experimental environment map

5. EXPERIMENTAL EVALUATION

5.1. Experimental Setup and Procedure

Real-world navigation experiments were conducted to evaluate the proposed Sim2Real transfer framework on a mobile robot platform. The experimental system integrates visual perception, localization, and navigation modules for end-to-end autonomous navigation in real environments. The detailed hardware and software configuration is summarized in Table 2.

Table 2. System configuration of the experimental system

Component	Specification / Purpose
Mobile Robot Base	Kobuki (base platform)
Stereo Camera	ZED2i (RGB input, VIO)
Computing Unit	NVIDIA RTX4060 Laptop
Software Environment	Ubuntu 24.04, ROS2 Jazzy

The experiments were performed in an indoor corridor environment. As shown in Fig. 7, the straight-line distance between the start position and the goal position is approximately 50 m. First, the robot was manually driven from the start position to the goal using a remote controller, and the goal position estimated by self-localization was recorded.

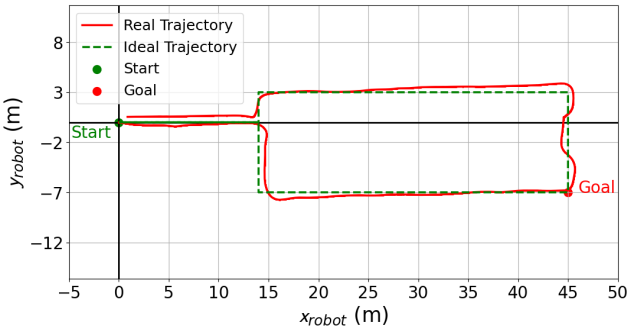

Fig. 8. Trajectory of the robot during the experiment

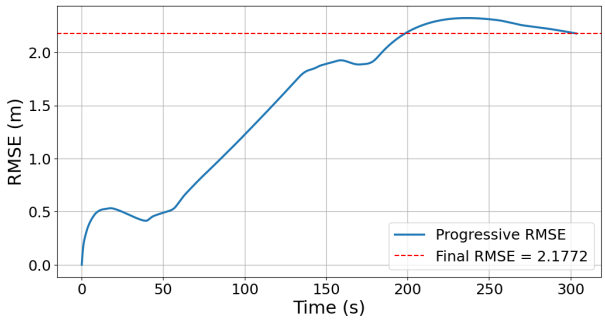

Fig. 9. Progressive RMSE of trajectory vs ground truth

After returning the robot to the start position and reinitializing the system, autonomous navigation was executed continuously from the start to the goal (approximately 52 m) and from the goal back to the start (approximately 58 m), resulting in a total travel distance of about 110 m. This step was designed to evaluate the stability of the learned navigation policy and the robustness of self-localization.

5.2. Experimental Results and Discussion

Experimental results confirmed that the robot was able to navigate stably from the start to the goal and return to the start in both directions. The corresponding round-trip trajectories are shown in Fig. 8. In particular, the robot exhibited highly stable motion in straight corridor segments. Although situations were observed in which the robot approached the wall near corners, it successfully corrected its heading and continued navigation without collisions. These observations indicate that the RL navigation model obtained in this study exhibits a high degree of stability.

On the other hand, during the return navigation from the goal to the start position, the robot reached the vicinity of the initial location in a generally stable manner; Furthermore, by comparing the recorded trajectories with the ideal path inferred from the corridor layout, the cumulative root mean square error (RMSE) over the trajectory is shown in Fig. 9. As shown in the figure, the overall RMSE converges to approximately 2.18 m, reflecting the cumulative effect of localization uncertainty over the long-horizon navigation task.

From these results, it is confirmed that the DRL-based navigation model trained in the simulation environment can operate stably in real-world environments. In particular, the experiments demonstrate that even without using LiDAR and relying on visual information, and without constructing a prior map, the robot is able to achieve stable navigation toward the target destination.

6. CONCLUSION AND FUTURE WORK

This study explored the effectiveness of a map-free RL navigation framework trained in simulation and transferred to real-world environments through a vision-based Sim2Real approach. By integrating semantic segmentation for visual alignment and EKF-based multi-sensor fusion for egomotion estimation, the proposed system demonstrates that stable goal-directed navigation can be achieved without relying on prior maps or LiDAR. These results indicate that a vision-based DRL policy learned entirely in simulation can be effectively deployed on a real mobile robot when appropriate perception and localization support is provided.

Although EKF-based sensor fusion improves the accuracy and stability of self-localization, accumulated error remains inevitable as the distance increases. Such drift originates from odometry-based estimation and cannot be completely eliminated by local sensor fusion alone. To address this limitation, future work will explore more advanced localization approaches, such as SLAM-based methods, to achieve more robust pose estimation during long-distance navigation.

REFERENCES

[1] P. Mirowski, R. Pascanu, F. Viola, et al., “Learning to navigate in complex environments,” *arXiv preprint arXiv:1611.03673*, 2016.

[2] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, “Self-supervised deep reinforcement learning for robotic navigation,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2754–2761, 2018.

[3] N. Yokoyama, “ASC: Adaptive Skill Coordination for Robotic Mobile Manipulation”, *IEEE Robotics and Automation Letters*, 2023.

[4] K. Yokoyama and K. Morioka, “Autonomous Mobile Robot with Simple Navigation System Based on Deep Reinforcement Learning and a Monocular Camera”, *IEEE/SICE International Symposium on System Integration (SII)*, pp. 525–530, 2020.

[5] N. Cohen and I. Klein, “Inertial Navigation Meets Deep Learning: A Survey of Current Trends and Future Directions,” *Results in Engineering*, 2024, Art. no. 103565. doi:10.1016/j.rineng.2024.103565.

[6] R. Tsuruta and K. Morioka, “Autonomous navigation of a mobile robot with a monocular camera using deep reinforcement learning and semantic image segmentation”, *IEEE/SICE International Symposium on System Integration (SII)*, pp. 1107–1112, 2024.

[7] J. Schulman, “Proximal policy optimization algorithms”, *arXiv preprint arXiv:1707.06347*, 2017.

[8] N. Ravi, “SAM 2: Segment Anything in Images and Videos”, *arXiv preprint arXiv:2408.00714*, 2024.

[9] J. Kelly and G. S. Sukhatme, “Visual–Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-Calibration”, *International Journal of Robotics Research*, Vol. 30, No. 1, pp. 56–79, 2011.