CARE: Enhancing Safety of Visual Navigation through Collision Avoidance via Repulsive Estimation

Joonkyung Kim[†1], **Joonyeol Sim**[†1], **Woojun Kim**[2], **Katia Sycara**[2], **Changjoo Nam**[1]
[1]Department of Electronic Engineering, Sogang University
[2]Robotics Institute, Carnegie Mellon University
[†]Equal contribution

Abstract: We propose CARE (Collision Avoidance via Repulsive Estimation) to improve the robustness of learning-based visual navigation methods. Recently, visual navigation models, particularly foundation models, have demonstrated promising performance by generating viable trajectories using only RGB images. However, these policies can generalize poorly to environments containing out-of-distribution (OOD) scenes characterized by unseen objects or different camera setups (e.g., variations in field of view, camera pose, or focal length). Without fine-tuning, such models could produce trajectories that lead to collisions, necessitating substantial efforts in data collection and additional training. To address this limitation, we introduce CARE, an attachable module that enhances the safety of visual navigation without requiring additional range sensors or fine-tuning of pretrained models. CARE can be integrated seamlessly into any RGB-based navigation model that generates local robot trajectories. It dynamically adjusts trajectories produced by a pretrained model using repulsive force vectors computed from depth images estimated directly from RGB inputs. We evaluate CARE by integrating it with state-of-the-art visual navigation models across diverse robot platforms. Real-world experiments show that CARE significantly reduces collisions (up to 100%) without compromising navigation performance in goal-conditioned navigation, and further improves collision-free travel distance (up to 10.7×) in exploration tasks. Project page: https://airlab-sogang.github.io/CARE/

Keywords: Vision-based Navigation, Reactive Planning, Collision Avoidance

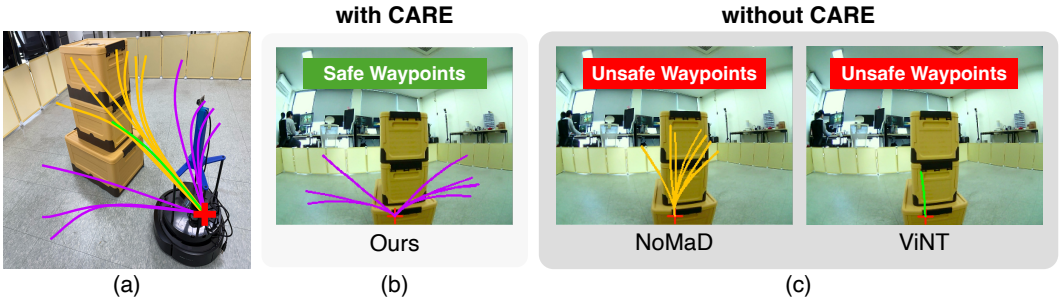


Figure 1: Comparison of trajectory outputs under out-of-distribution (OOD) obstacle settings. **(a)** A test environment with trajectories with and without CARE. **(b)** Trajectories adjusted using CARE to avoid collision. **(c)** Trajectories from the original models without CARE result in collisions.

1 Introduction

Recent advances in image representation learning and large language models (LLMs) have led to the development of vision-language models (VLMs) capable of contextual reasoning and high-level instruction following [1]. These models enable robust robotic navigation across varied tasks with minimal task-specific engineering, achieving strong performance even in few-shot [2] or zero-shot [3]

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

settings. VLMs further allow robots to localize goals from language [4] and perform long-horizon navigation [5] based on human-language instructions.

To extend such high-level generalization to low-level control on physical robots, vision-based navigation models have been developed to handle variations in camera intrinsics, robot dynamics, and hardware configurations [6]. Foundation models such as GNM [7] and ViNT [8] are designed for this purpose, enabling generalization across heterogeneous platforms with minimal fine-tuning. These models support a range of tasks including goal-directed navigation, long-horizon planning with high-level policies [9], and exploration [10], relying solely on monocular RGB input.

Despite their generalization capabilities, vision-based foundation models face safety limitations, particularly in out-of-distribution (OOD) environments (see Figure 1). As they rely on appearance-based reasoning without explicit geometric understanding, they may produce unsafe trajectories that lead to collisions [11]. Adapting them to new robot platforms or camera setups often requires retraining and data collection [12], limiting their real-world applicability.

To address these limitations, we propose CARE (Collision Avoidance via Repulsive Estimation), a plug-and-play safety module that enhances vision-based navigation without additional sensors or fine-tuning. CARE combines monocular depth estimation with reactive local planning to adjust trajectories from pretrained policies. It uses a pretrained monocular depth model (in this work, UniDepthV2 [13]) to infer a top-down obstacle map from RGB input, and applies Artificial Potential Fields (APF) [14] to compute repulsive forces that steer trajectories away from obstacles. CARE can be integrated with any RGB-based navigation policy, improving safety while minimally deviating from the original path. We validate CARE on three distinct robot platforms (LoCoBot, TurtleBot4, and RoboMaster), demonstrating consistent collision reduction in unseen environments and improved collision-free distance, without compromising navigation performance.

Our contributions are summarized as follows. **(i)** We introduce CARE, a plug-and-play module that improves collision avoidance by integrating vision-based navigation model with reactive planning based on monocular RGB input, without requiring additional sensors or retraining. **(ii)** CARE generalizes across diverse vision-based navigation models, robot platforms, and camera configurations, while preserving navigation performance. **(iii)** We extensively validate CARE in real-world experiments across three robot platforms, achieving up to 100% collision reduction in goal-conditioned navigation and up to 10.7× improvement in collision-free travel distance during exploration.

2 Related Work

Recent advances in vision-language and vision-based navigation have enabled robots to perform complex tasks using only RGB inputs. However, ensuring safe operation in dynamic and unfamiliar environments remains a key challenge. We review prior work on vision-based navigation and recent efforts to improve safety in learning-based navigation.

VLM-based navigation approaches such as LM-Nav [5], LFG [15], and NavGPT [3] leverage pretrained VLMs to interpret natural language commands and perform high-level planning without task-specific supervision. VLMaps [4] integrates vision-language features into 3D spatial maps, while Obstructed VLN [16] and Knowledge-Enhanced Scene Understanding [17] aim to improve robustness to domain shifts. However, these methods mainly focus on high-level semantic reasoning and rely on discrete or graph-based action spaces, leaving safety-critical aspects such as collision avoidance largely unaddressed.

To bridge the gap between abstract high-level commands and low-level control, recent efforts have developed vision-based foundation models for navigation that directly map RGB observations to robot actions. ViKiNG [9] combined learned visual traversability with geographic hints for kilometer-scale navigation, while ViNG [6] introduced topological navigation without requiring spatial maps. GNM [7] and ViNT [8] further advanced by enabling generalization across heterogeneous robot platforms through minimal fine-tuning, leveraging embodiment-agnostic policies or Transformer [18] architectures. NoMaD [10] explored diffusion-based multimodal action generation, and Navigation World Models (NWM) [19] proposed leveraging video prediction for policy

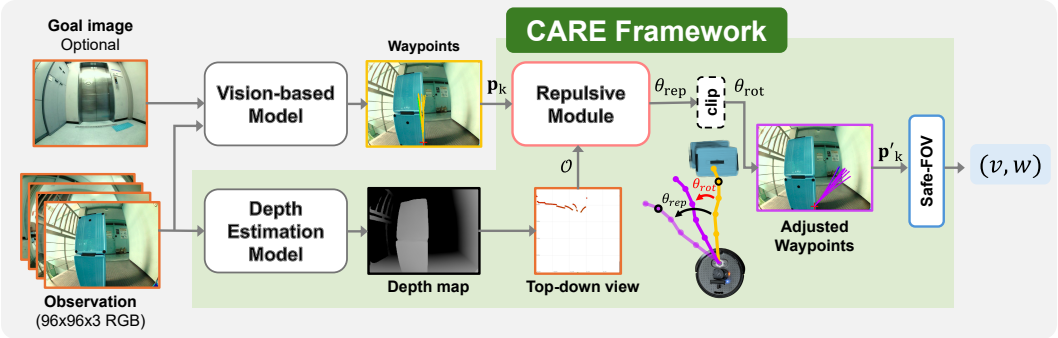


Figure 2: Overview of CARE. The system takes RGB observations and optionally a goal image as input. A vision-based model (e.g., NoMaD [10], ViNT [8]) generates waypoints or trajectories, while a top-down local map is generated from depth map. A repulsive module then adjusts the trajectory to avoid obstacles, with the adjustment angle θ_{rep} clipped to θ_{rot} for stable navigation.

planning without explicit mapping. Despite impressive generalization, these models primarily rely on vision-only reasoning without explicit geometric understanding. This makes it difficult to guarantee safety or reactive obstacle avoidance, especially when encountering unseen obstacles or when precise spatial reasoning is required. Moreover, although they are designed for minimal fine-tuning, adapting them to new robot platforms, camera intrinsics, or dynamic environments often demands nontrivial data collection and retraining [12], limiting their practical applicability.

Ensuring reliable collision avoidance has therefore emerged as a key concern in vision-based navigation. Safe-VLN [20] introduces waypoint filtering and fallback strategies using simulated 2D-LiDAR. Failure Prediction [21] estimates risk from visual input, while PwC [22] calibrates perception models for robustness under domain shift. Adaptive methods such as online constraint updates [23], control barrier functions [24], and velocity-obstacle-based shielding [25] have also been explored. MonoNav [11] reconstructs local 3D metric maps from monocular depth to enable safer planning. However, its conservative obstacle avoidance strategy often sacrifices task success, leading to lower completion rates compared to other vision-based models. Moreover, it assumes static environments, limiting performance in dynamic scenarios. Similarly, NavRL [25] integrates a velocity-obstacle-based safety shield on top of reinforcement learning policies, but still depends on retraining and policy adaptation for different robot platforms.

In contrast, we propose CARE, an attachable module that integrates the outputs of vision-based navigation models and monocular depth estimation with a reactive planning method based on APF, a widely used approach for collision avoidance using local observations with low computational cost [26, 27, 28, 29]. CARE dynamically adjusts planned trajectories using the repulsive force formulation of APF, enabling safer, reactive trajectory adaptation across diverse robotic platforms without additional sensors, explicit 3D reconstruction, or fine-tuning of the navigation model.

3 Collision Avoidance via Repulsive Estimation (CARE)

CARE is a plug-and-play module that augments the output of any vision-based navigation policy producing waypoints or trajectories. It uses the same RGB observation as the navigation model to estimate depth and adjust the predicted path accordingly. CARE operates in three stages (see Figure 2 and Algorithm 1), enabling real-time correction of unsafe paths when encountering previously unseen obstacles.

3.1 Top-Down Range Estimation

CARE first constructs a top-down local map (see Figure 3a) by predicting the scene geometry from a depth image predicted from an RGB observation through the function *ConstructTopDownObstacleMap*$(I, \mathcal{D})$ (see Algorithm 1, Line 1).

Given an RGB observation $I \in \mathbb{R}^{H \times W \times 3}$ captured by a calibrated monocular camera, a pretrained metric depth model predicts a dense depth map. This depth map can be projected into a set of

Algorithm 1 CARE: Collision Avoidance via Repulsive Estimation

Input: RGB observation I, trajectory $\mathcal{T}$, depth estimation model $\mathcal{D}$
Output: Control command (v, ω)
1: $\mathcal{O} \leftarrow$ *ConstructTopDownObstacleMap*$(I, \mathcal{D})$ // Stage 1: Top-down range estimation
2: $\theta_{\text{rep}} \leftarrow$ *EstimateRepulsiveDirection*$(\mathcal{T}, \mathcal{O})$ // Stage 2: Repulsive force estimation
3: $\mathcal{T}' \leftarrow$ *RotateTrajectory*$(\mathcal{T}, \theta_{\text{rep}})$
4: $\theta_{\text{des}} \leftarrow$ *ComputeDesiredHeading*$(\mathcal{T}')$ // Stage 3: Safety-enhancing mechanism
5: **if** $|\theta_{\text{des}}| > \theta_{\text{thres}}$ **then**
6: $(v, \omega) \leftarrow$ *RotateInPlace*(θ_{des})
7: **else**
8: $(v, \omega) \leftarrow$ *MoveForwardAndTurn*(θ_{des})
9: **end if**
10: **return** (v, ω)

three-dimensional points $(X_i, Y_i, Z_i) \in \mathbb{R}^3$ by applying the inverse intrinsic matrix to each pixel location [30, 31, 32]. In our implementation, we employ UniDepthV2 [13], which directly outputs 3D point clouds without requiring explicit back-projection, though CARE can be replaced with any absolute depth estimation model capable of producing similar output formats.

The predicted point cloud $\mathcal{P} = \{(X_i, Y_i, Z_i) \mid i = 1, \ldots, N\}$ is filtered to retain only points satisfying $(Z > 0) \wedge (Z \leq \tau_z) \wedge (Y \geq -\epsilon)$, where $\tau_z \in \mathbb{R}$ denotes the maximum sensing range and $\epsilon \in \mathbb{R}$ defines a vertical margin to exclude ceiling points. The resulting set of valid projected points is defined as $\mathcal{P}_{\text{valid}} = \{(X_i, Z_i) \mid (X_i, Y_i, Z_i) \in \mathcal{P},\ \text{Mask}(X_i, Y_i, Z_i) = \text{True}\}$.

The x-axis is discretized into M uniform bins, and for each bin, the closest point along the Z-axis is selected:

$$(x^\star, z^\star) = \underset{(x,z) \in \mathcal{P}_{\text{valid}}}{\text{argmin}}\ z,$$

forming the obstacle set $\mathcal{O} = \{(x_j^\star, z_j^\star) \mid j = 1, \ldots, M\}$. All coordinates are represented in the local robot frame where the positive x-axis points forward and the positive y-axis points to the left.

3.2 Repulsive Force Estimation and Trajectory Adjustment

To adjust trajectories in response to nearby obstacles, CARE estimates repulsive forces using the APF method. This repulsive vector provides a reactive adjustment direction to steer the robot away from collisions.

Given the obstacle set $\mathcal{O}$ derived from monocular depth estimation, CARE computes repulsive forces applied to a generated trajectory $\mathcal{T} = \{\mathbf{p}_1, \ldots, \mathbf{p}_K\}$ or a single waypoint $\mathbf{p}_k \in \mathbb{R}^2$, depending on the output format of the vision policy. We describe the method assuming a trajectory-based policy such as ViNT or NoMaD, which outputs a sequence of K waypoints $\mathcal{T} = \{\mathbf{p}_1, \ldots, \mathbf{p}_K\}$ at each step, though the procedure naturally extends to single-waypoint outputs.

The repulsive force at each waypoint is computed via *EstimateRepulsiveDirection*$(\mathcal{T}, \mathcal{O})$ (Algorithm 1, Line 2), following the same formulation used in a recent work on APF with limited sensing range [26]:

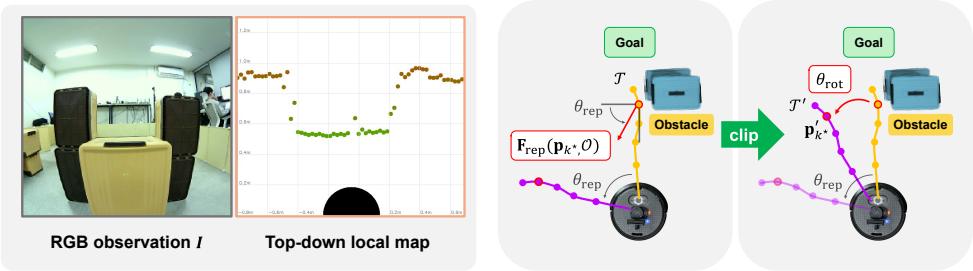
$$\mathbf{F}_{\text{rep}}(\mathbf{p}_k, \mathcal{O}) = \sum_{m=1}^{M} \frac{-1}{\|\mathbf{p}_k - \mathbf{o}_m\|_2^3} \cdot \frac{\mathbf{p}_k - \mathbf{o}_m}{\|\mathbf{p}_k - \mathbf{o}_m\|_2}, \tag{1}$$

where $\mathbf{o}_m \in \mathbb{R}^2$ denotes the m-th obstacle in $\mathcal{O}$ and $\|\cdot\|_2$ is the Euclidean norm.

To determine the adjustment amount and direction, CARE selects the waypoint experiencing the maximum repulsive magnitude:

$$k^\star = \text{argmax}_{k \in \{1, \ldots, K\}} \|\mathbf{F}_{\text{rep}}(\mathbf{p}_k, \mathcal{O})\|_2. \tag{2}$$

For waypoint-only policies, the single predicted point is directly used in place of $\mathbf{p}_{k^\star}$. The repulsive vector at $\mathbf{p}_{k^\star}$ is converted to a heading adjustment angle $\theta_{\text{rep}} = \arctan 2(F_{\text{rep},y}(\mathbf{p}_{k^\star}), F_{\text{rep},x}(\mathbf{p}_{k^\star}))$,



(a) Top-down Depth Projection

(b) Repulsive Rotation Adjustment

Figure 3: (a) Top-down projection of estimated depth. Obstacle points are sampled every 10 pixels from the depth map (Left: input RGB, Right: top-down view), with colored dots indicating sampled points and the black circle denoting the robot. (b) Trajectory adjustment using repulsive force $\mathbf{F}_{\text{rep}}$. The original trajectory $\mathcal{T}$ (yellow) is rotated by θ_{rep}, clipped to θ_{clip}, resulting in the adjusted trajectory $\mathcal{T}'$ (purple).

which determines the direction of deviation from the current trajectory. To avoid excessive deviation from the goal, the adjustment angle is clipped as $\theta_{\text{rot}} = \text{clip}(\theta_{\text{rep}}, -\theta_{\text{clip}}, \theta_{\text{clip}})$, where $\text{clip}(\cdot, a, b)$ limits the value between a and b.

Finally, *RotateTrajectory*$(\mathcal{T}, \theta_{\text{rot}})$ (Algorithm 1, Line 3) applies a 2D rotation matrix $R \in \mathbb{R}^{2\times 2}$ to each waypoint, yielding the adjusted trajectory $\mathcal{T}' = \{\mathbf{p}'_1, \ldots, \mathbf{p}'_K\}$, where each waypoint is updated as $\mathbf{p}'_k = R(\theta_{\text{rot}}) \cdot \mathbf{p}_k$. This rotation steers the trajectory away from nearby obstacles while preserving its overall direction toward the goal (see Figure 3b).

3.3 Safety-Enhancing Mechanism

Since vision-based models are limited to the field of view (FOV) of the camera, they cannot account for obstacles outside it. Sharp turns during forward motion can cause nearby obstacles to enter the view suddenly, giving the robot little time to react. To mitigate this risk, CARE introduces a Safe-FOV mechanism that suppresses forward motion when large heading changes are required.

First, we select $\mathbf{p}'_{k^\star} = (p'_{k^\star,x}, p'_{k^\star,y}) \in \mathcal{T}'$, the adjusted waypoint with the strongest repulsive response (as defined in Section 3.2), as the basis for computing the desired heading. The desired heading θ_{des} is then computed using the function *ComputeDesiredHeading*$(\mathcal{T}')$ (Algorithm 1, Line 4). Since waypoints are generated relative to the local frame of the robot, θ_{des} can be directly calculated using the two-argument $\arctan 2$ function $\theta_{\text{des}} = \arctan 2(p'_{k^\star,y}, p'_{k^\star,x})$. We define a rotation threshold $\theta_{\text{thres}} \in \mathbb{R}$, above which forward motion is suppressed to prioritize in-place safe turns by *RotateInPlace*(θ_{des}). Otherwise, *MoveForwardAndTurn*(θ_{des}) is performed to move forward and turn simultaneously. In other words, the safety-enhancing mechanism (Lines 5–9 of Algorithm 1) follows

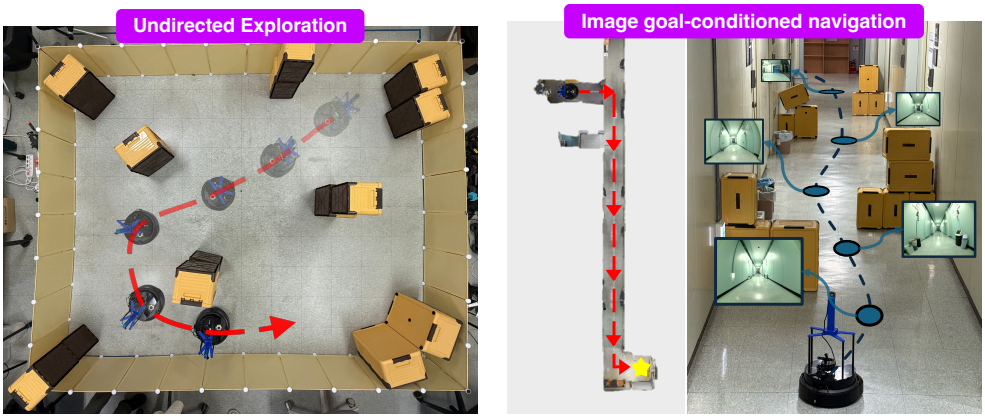
$$(v, \omega) = \begin{cases} (0, \omega_{\text{rot}}), & \text{if } |\theta_{\text{des}}| > \theta_{\text{thres}} \\ (v_{\text{fwd}}, \omega_{\text{rot}}), & \text{otherwise} \end{cases} \tag{3}$$

where v_{fwd} denotes the forward linear velocity, and ω_{rot} is the angular velocity constrained by the maximum linear and angular velocity $(v_{\text{max}}, \omega_{\text{max}})$. If the heading deviation $|\theta_{\text{des}}|$ exceeds the threshold, CARE commands in-place rotation until the deviation falls within $|\theta_{\text{des}}| \leq \theta_{\text{thres}}$, thereby reducing the risk of collisions incurred by the limited FOV of cameras, especially in dense environments.

In summary, CARE selectively adjusts trajectories based on obstacle presence. When nearby obstacles are detected via monocular depth estimation (i.e., within sensing range τ_z), repulsive forces modify the trajectory. If no obstacles are found ($\mathcal{O} = \emptyset$), the original trajectory from the vision-based model is preserved. This design allows CARE to enhance safety while maintaining the generalization of pretrained models without fine-tuning or additional hardware.

4 Experiments

We comprehensively evaluate CARE across unseen real environments and different robot platforms to validate our primary claim: CARE enhances collision avoidance capabilities of pretrained vision-



(a) (b)

Figure 4: Real-world experiments with CARE. (a) The unidirected exploration task in a confined environment ($3.5\,\mathrm{m} \times 2.8\,\mathrm{m}$) tests the ability to explore among unseen obstacles (boxes) as much as possible. (b) The image goal-conditioned navigation task in a corridor ($24\,\mathrm{m}$) tests the ability to reach the goal while avoiding seen obstacles (walls and doors) and unseen obstacles (boxes). The topological graph shows nodes with only image data.

based navigation models without fine-tuning, while preserving original navigation performance. To validate the generalizability of CARE, we deploy our method on three mobile robot platforms with different camera configurations: (i) LoCoBot with a 170° FOV fisheye camera[1], (ii) Clearpath TurtleBot4 equipped with an 89.5° FOV OAK-D Pro camera, and (iii) DJI RoboMaster S1 with a 120° FOV camera. All robots operate under consistent velocity constraints (maximum linear velocity v_{max} of $0.2\,\mathrm{m/s}$ and angular velocity w_{max} of $0.8\,\mathrm{rad/s}$) for comparable evaluation. We optimize CARE parameters experimentally for each platform while maintaining consistent evaluation across all platforms. Detailed robot-specific parameters are provided in the Appendix.

Undirected exploration: This task evaluates the ability of the robot to explore safely in an environment with unseen objects. We place 10 OOD obstacles (plastic boxes) in a $3.5\,\mathrm{m} \times 2.8\,\mathrm{m}$ area as shown in the left of Figure 4a. Since the robot explores without a destination, we measure the distance traveled until the first collision. To avoid indefinitely long trials, we stop if the robot travels above $30\,\mathrm{m}$. We compare NoMaD and CARE-integrated NoMaD.[2] For each combination of robot platform and navigation method, we conduct 20 trials and report the average distance traveled.

Image goal-conditioned navigation: This task tests the image goal-conditioned navigation capability of the robot in environments with random unseen obstacles (a $24\,\mathrm{m}$ corridor). Initially, a topological graph of corridors (see Figure 4b) is constructed, and we place OOD obstacles at random locations along the corridor. We conducted 10 distinct setups, testing all robot and method combinations. The robot must navigate to the goal using the topological graph while avoiding obstacles as much as possible. Evaluation metrics include average path length (shorter is more efficient), completion time (faster is more efficient), and collision count (lower is safer). We also measure the arrival rate (higher is better), which is defined as the proportion of successful goal reaches, even if collisions occur. We compare NoMaD and ViNT and CARE-integrated NoMaD and ViNT.

4.1 Undirected Exploration task in Unseen Environments

Figure 5 and Table 1 present our exploration experiment results. Across all settings, CARE enables robots to travel $2.9\times$ to $10.7\times$ longer before the first collision compared to NoMaD and ViNT without CARE. These performance variations correlated with the camera specifications of each configuration and the resulting depth estimation quality.

[1]The camera setup is consistent with that used in the baseline models NoMaD and ViNT.

[2]We excluded ViNT from exploration experiments because, as a goal-conditioned model, it requires a goal image to generate trajectories and cannot operate without one. Supporting exploration would require a subgoal generator, typically implemented as a diffusion model trained in the target environment, which conflicts with our zero-shot assumption that prohibits additional training.

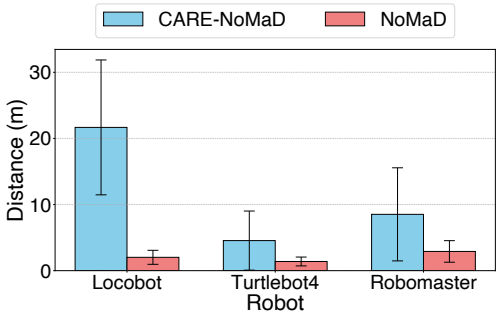


Figure 5: Comparison of distance traveled before first collision between CARE-NoMaD and baseline NoMaD across different robots.

Robot	Method	Distance (m)
Locobot	CARE	**21.67 ± 10.20**
	NoMaD	2.02 ± 1.06
Turtlebot4	CARE	**4.55 ± 4.47**
	NoMaD	1.39 ± 0.67
RoboMaster	CARE	**8.52 ± 7.04**
	NoMaD	2.91 ± 1.63

Table 1: Mean distance traveled with standard deviation before collision for each robot platform and navigation method.

Table 2: The comparison of the navigation performance between baseline vision-based models and their CARE-integrated counterparts across three robot platforms. Metrics include path length (m), completion time (s), collision counts, and goal arrival rate (%).

Robot	**Model**	**Vision-based Model**				**CARE-integrated Vision-based Model**			
		Length (m)	**Time (s)**	**# Collision**	**Arrival Rate**	**Length (m)**	**Time (s)**	**# Collision**	**Arrival Rate**
Locobot	NoMaD	26.39 ± 1.02	132.98 ± 5.02	0.7	50%	26.06 ± 0.30	136.06 ± 3.90	**0.0**	**90%**
	ViNT	25.79 ± 0.19	129.98 ± 0.85	0.4	50%	25.88 ± 0.18	134.87 ± 2.40	**0.0**	**80%**
Turtlebot4	NoMaD	25.27 ± 0.18	127.28 ± 0.55	0.7	20%	26.35 ± 1.10	149.13 ± 9.03	**0.3**	**50%**
	ViNT	26.42 ± 0.75	133.73 ± 3.82	0.2	70%	26.55 ± 0.98	145.91 ± 14.47	**0.1**	**100%**
RoboMaster	NoMaD	25.24 ± 0.61	126.33 ± 3.05	0.8	30%	25.89 ± 0.96	146.76 ± 10.34	**0.1**	**80%**
	ViNT	24.79 ± 0.48	125.60 ± 2.28	1.0	50%	25.75 ± 1.10	155.60 ± 11.35	**0.2**	**80%**

LoCoBot with CARE exhibits both the longest distance traveled and the most significant improvement (10.7×). This performance increase can be attributed to two main factors: the wide-angle fisheye camera enables extensive environmental perception, effectively detecting peripheral obstacles; and NoMaD, which has been predominantly trained on fisheye camera data, performs better with the camera setup of LoCoBot, which CARE further enhances.

TurtleBot4 with CARE exhibits moderate improvement (3.3×) despite having the lowest distance traveled. This enhancement is partly due to its camera mounting position. When navigating corners, the rear-mounted camera maintains visibility of obstacles throughout the turn, allowing CARE to make safer adjustments. However, its narrow FOV still limits overall performance, particularly when approaching walls head-on.

RoboMaster without CARE achieves higher distance traveled due to its wider FOV, but shows the smallest improvement (2.9×) with CARE. This limited gain stems from its front-mounted camera. During turning, the camera moves past obstacles before the body of the robot completes the turn, creating blind spots where side collisions can occur without detection. While the wider FOV achieves relatively higher distance traveled, it cannot fully compensate for these cornering blind spots.

4.2 Image Goal-Conditioned Navigation with Unseen Obstacles

We evaluate NoMaD and ViNT and their CARE-integrated variants in image goal-conditioned navigation scenarios involving unseen obstacles with the three robot platforms. We first establish a topological graph in a corridor environment to enable global planning. During testing, we place several OOD obstacles not represented in the topological map, creating scenarios that would typically require model fine-tuning to handle appropriately. Table 2 presents comprehensive results comparing the baseline navigation models (NoMaD and ViNT) against the same models integrated with CARE. All experiments are conducted across 10 distinct obstacle settings, with randomized positions for each trial.[3]

[3]Detailed information about obstacle placements for each trial is available in Appendix B.1.

Path length and time: The integration of CARE results in slightly longer paths (up to 4.27% increase) and completion times (up to 23.89% increase). This can be attributed to the adjusted trajectories around detected obstacles for safer navigation. Despite these inefficiencies, the overall navigation performance remained reasonable.

Collisions: CARE consistently reduces collisions throughout all platforms. Most notably, LoCoBot achieves complete collision elimination with both NoMaD and ViNT models. For RoboMaster, CARE integration yields notable improvements (80–88% reduction), while TurtleBot4 demonstrates modest but meaningful reductions (50–57%). Despite these improvements, CARE does not completely eliminate all collisions in the cases of TurtleBot4 and RoboMaster. One reason is that monocular depth estimation occasionally produces inaccurate results, particularly on obstacle surfaces without patterns or in challenging lighting conditions. Another reason is the limited FOV or mounting position of the camera, which causes obstacles outside the view to lead to side collisions with the robot body.

Arrival rate: The integration of CARE significantly enhances arrival rates across all platforms. On LoCoBot and RoboMaster, both NoMaD and ViNT models with CARE show substantial improvements. For TurtleBot4, ViNT already performs well (70% baseline) and reaches perfect goal-reaching (100%) with CARE, while NoMaD struggles even with safety enhancements, improving from only 20% to 50%. This shows that while CARE consistently adjusts trajectories to avoid collisions, the quality of trajectories generated by the base navigation model significantly influences results. Nevertheless, even with NoMaD on TurtleBot4, CARE still provides considerable improvement over the baseline model. Overall, CARE improves goal arrival reliability across different robots and camera configurations without requiring additional fine-tuning or sensor hardware.

Navigation with dynamic obstacles. We conducted additional tests in the same environment as the goal-conditioned experiments (see Figure 4b), where up to three people abruptly crossed the robot path during navigation. A trial was considered successful if the robot reached the goal without collision. Unlike prior works, we introduced dynamic obstacles that appeared suddenly from outside the camera field of view. NoMaD often failed to react, while CARE-integrated robots consistently exhibited reactive stopping and safe avoidance, demonstrating the ability to handle unpredictable dynamic obstacles where vision-only models without fine-tuning often fail to generalize. Further details are provided in Appendix A.1.

These results demonstrate the consistent ability of CARE to enhance navigation safety across various robot platforms without additional training. While performance varies based on camera configuration, CARE considerably improves collision avoidance in all tested scenarios, augmenting the capabilities of baseline models to handle complex environments with OOD objects. Supplementary videos of these experiments are available on our project page: https://airlab-sogang.github.io/CARE/

5 Conclusion

We presented CARE, an attachable module that enhances the safety of vision-based navigation models without requiring fine-tuning or additional range sensors. By combining monocular depth estimation with repulsive force-based trajectory adjustment, CARE significantly improves collision avoidance in out-of-distribution environments and across diverse robot platforms with varying camera configurations. CARE enables zero-shot deployment of vision-based models in unseen environments where fine-tuning would otherwise be necessary, overcoming their limited collision avoidance capabilities. Extensive real-world experiments demonstrate that CARE substantially reduces collision rates and improves task success, while preserving the strong generalization performance and efficiency of the underlying navigation models. Our results highlight CARE as a simple and effective solution for safer real-world deployment of visual navigation systems.

6 Limitations

Although CARE does not require fine-tuning of the navigation policy or additional range sensors, it has several limitations arising from its reliance on monocular depth estimation and vision-based navigation models.

First, CARE is sensitive to errors in monocular depth prediction. Reflective or transparent surfaces often lead to inaccurate depth values, which may cause the system to underestimate obstacle proximity or miss obstacles entirely. Additionally, depth estimation models may misclassify ground or low-texture areas, resulting in false avoidance maneuvers that deviate unnecessarily from the desired path.

Second, CARE only reacts to obstacles within a limited sensing range τ_z derived from the estimated depth. Distant obstacles beyond this range are ignored, and the system cannot anticipate them. Furthermore, as with most vision-based systems, perception range of robot is restricted to the FOV of attached camera, making it unable to respond to dynamic obstacles approaching from behind or from occluded regions.

Third, being entirely vision-based, CARE inherits the limitations of RGB-only systems. Its performance may degrade under poor lighting conditions or in environments where camera orientation fails to capture critical obstacles. These constraints reduce the effectiveness of the repulsive adjustment in unpredictable or low-visibility scenarios.

Fourth, CARE adjusts but does not generate trajectories. Its effectiveness depends on the quality of the underlying navigation model. If the base policy predicts unsafe paths (such as directly toward an obstacle) CARE may be unable to fully correct them, even with strong repulsive forces. This limits its performance in densely cluttered or highly dynamic environments.

Finally, CARE introduces computational overhead from the depth estimation step. However, the model we employed (UniDepthV2) is lightweight and uses only 100MB of GPU memory, which does not impact real-time performance.

7 Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00461409), Korea Planning & Evaluation Institute of Industrial Technology (KEIT) grant funded by the Korea government (MOTIE) (RS-2024-00444344), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2022-00143911, AI Excellence Global Innovative Leader Education Program).

References

[1] M. Awais, M. Naseer, S. Khan, R. M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang, and F. S. Khan. Foundation models defining a new era in vision: a survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[2] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2998–3009, 2023.

[3] G. Zhou, Y. Hong, and Q. Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 7641–7649, 2024.

[4] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE, 2023.

[5] D. Shah, B. Osiński, B. Ichter, and S. Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Proc. of the Conference on Robot Learning (CoRL)*, pages 492–504. PMLR, 2023.

[6] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine. Ving: Learning open-world navigation with visual goals. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE, 2021.

[7] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. Gnm: A general navigation model to drive any robot. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.

[8] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. Vint: A foundation model for visual navigation. In *Proc. of the Conference on Robot Learning (CoRL)*, pages 711–733. PMLR, 2023.

[9] D. Shah and S. Levine. ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints. In *Proc. of the Robotics: Science and Systems (RSS)*, 2022.

[10] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[11] N. Simon and A. Majumdar. Mononav: Mav navigation via monocular depth estimation and reconstruction. In *Proc. of the International Symposium on Experimental Robotics (ISER)*, pages 415–426. Springer, 2023.

[12] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, B. Ichter, D. Driess, J. Wu1, C. Lu, and M. Schwager. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research (IJRR)*, page 02783649241281508, 2023.

[13] L. Piccinelli, C. Sakaridis, Y.-H. Yang, M. Segu, S. Li, W. Abbeloos, and L. Van Gool. Unidepthv2: Universal monocular metric depth estimation made simpler. *arXiv preprint arXiv:2502.20110*, 2025.

[14] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research (IJRR)*, 5(1):90–98, 1986.

[15] D. Shah, M. R. Equi, B. Osiński, F. Xia, B. Ichter, and S. Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Proc. of the Conference on Robot Learning (CoRL)*, pages 2683–2699. PMLR, 2023.

[16] H. Lu, J. Wu, X. Wang, T. Darrell, and C. Chen. Navigating beyond instructions: Evaluating and enhancing vln agents under realistic environment changes. In *Proc. of ACM International Conference on Multimedia (ACM MM)*, 2024.

[17] F. Gao, J. Tang, J. Wang, S. Li, and J. Yu. Enhancing scene understanding for vision-and-language navigation by knowledge awareness. *IEEE Robotics and Automation Letters (RA-L)*, 9(12):10874–10878, 2024.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[19] A. Bar, G. Zhou, D. Tran, T. Darrell, and Y. LeCun. Navigation world models. *arXiv preprint arXiv:2412.03572*, 2025.

[20] L. Yue, D. Zhou, L. Xie, F. Zhang, Y. Yan, and E. Yin. Safe-vln: Collision avoidance for vision-and-language navigation of autonomous robots operating in continuous environments. *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[21] A. Farid, D. Snyder, A. Z. Ren, and A. Majumdar. Failure prediction with statistical guarantees for vision-based robot control. In *Proc. of the Robotics: Science and Systems (RSS)*. MIT Press Journals, 2022.

[22] A. Dixit, Z. Mei, M. Booker, M. Storey-Matsutani, A. Z. Ren, and A. Majumdar. Perceive with confidence: Statistical safety assurances for navigation with learning-based perception. In *Proc. of the Conference on Robot Learning (CoRL)*, 2024.

[23] L. Santos, Z. Li, L. Peters, S. Bansal, and A. Bajcsy. Updating robot safety representations online from natural language feedback. *arXiv preprint arXiv:2409.14580*, 2024.

[24] S. Sanyal and K. Roy. Asma: An adaptive safety margin algorithm for vision-language drone navigation via scene-aware control barrier functions. *arXiv preprint arXiv:2409.10283*, 2024.

[25] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada. Navrl: Learning safe flight in dynamic environments. *IEEE Robotics and Automation Letters (RA-L)*, 2025.

[26] J. Kim, S. Park, W. Lee, W. Kim, N. Doh, and C. Nam. Escaping local minima: Hybrid artificial potential field with wall-follower for decentralized multi-robot navigation. *arXiv preprint arXiv:2409.10332*, 2024.

[27] K. Bektaş and H. I. Bozma. Apf-rl: Safe mapless navigation in unknown environments. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7299–7305, 2022.

[28] D. Zhang, X. Zhang, Z. Zhang, B. Zhu, and Q. Zhang. Reinforced potential field for multi-robot motion planning in cluttered environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 699–704, 2023.

[29] Z. Zhang, D. Zhang, Q. Zhang, W. Pan, and T. Hu. Dacoop-a: Decentralized adaptive cooperative pursuit via attention. *IEEE Robotics and Automation Letters (RA-L)*, 9(6), 2024.

[30] X. Weng and K. Kitani. Monocular 3d object detection with pseudo-lidar point cloud. In *Procc of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019.

[31] L. Piccinelli, Y.-H. Yang, C. Sakaridis, M. Segu, S. Li, L. Van Gool, and F. Yu. Unidepth: Universal monocular metric depth estimation. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10106–10116, 2024.

[32] G. Elazab, T. Gräber, M. Unterreiner, and O. Hellwich. MonoPP: Metric-scaled self-supervised monocular depth estimation by planar-parallax geometry in automotive applications. In *Proc. of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2777–2787, 2025.

Appendix

A Additional Experiments **12**
A.1 Dynamic Obstacles in Goal-Conditioned Navigation 12
A.2 Performance in Seen Environments 13

B Experimental Details **13**
B.1 Test Instances for Goal-Conditioned Navigation 13
B.2 Robot Platforms and Specifications 14
B.3 Common Parameters for CARE 15
B.4 Depth Estimation Model: UniDepthV2 16
B.5 Trajectory Selection in NoMaD and ViNT 16

A Additional Experiments

A.1 Dynamic Obstacles in Goal-Conditioned Navigation

To assess the robustness of CARE under dynamic conditions, we evaluate the system in an image goal-conditioned navigation setting with abrupt human intervention. Unlike the evaluations in the paper, this setup contains no static out-of-distribution (OOD) obstacles. All tests are conducted using the LoCoBot platform.

Dynamic Obstacle Scenarios. We introduce up to three humans acting as dynamic, unexpected obstacles, designed to test reactive collision avoidance. Each test trial includes one of the following scenarios[4].

(i) A person appears from a side outside the field of view.

(ii) A person appears from behind and overtakes the robot, stopping in front.

(iii) A person walks toward the robot from the front and stops directly in its path.

Metrics and setup. For each of the three human intervention types, we run 10 trials with four methods: NoMaD, NoMaD+CARE, ViNT, and ViNT+CARE. We record the number trials where collisions occur.

Table 3: Number trials where collisions occur (out of 10 trials) for each dynamic obstacle type

Model	(i) Corner-appear	(ii) Behind-to-front	(iii) Front-approach
NoMaD	8/10	10/10	10/10
NoMaD + CARE	**0/10**	**0/10**	**0/10**
ViNT	7/10	10/10	10/10
ViNT + CARE	**0/10**	**0/10**	**0/10**

Results and Analysis. Table 3 shows the number of collisions in each dynamic obstacle scenario. Without CARE, both NoMaD and ViNT fail to react effectively to abrupt human interventions, resulting in 7 to 10 collisions. In contrast, CARE-integrated policies complete all trials without a single collision, where this improvement is attributed to the repulsive force estimation based on predicted depth, enabling reliable detection of human legs even under sudden appearances.

In scenario **(i)**, where a person enters from the side, both NoMaD and ViNT occasionally succeed in generating avoidance waypoints in the opposite direction, thereby avoiding some collisions. However, in scenarios **(ii)** and **(iii)**, where the human appears directly in front of the robot, both models

[4]Please refer to the supplementary video for better understanding.

fail in all trials. The key reason is the lack of geometric awareness in NoMaD and ViNT: although the models sometimes attempt avoidance by turning slightly, the generated waypoints do not ensure sufficient clearance. As a result, the robots consistently collide after small heading changes, or fail to respond to sudden frontal appearances.

In contrast, CARE-enabled policies successfully avoid all collisions in these scenarios. The estimated depth information allows CARE to compute repulsive directions with adequate lateral displacement, while the Safe-FOV mechanism suppresses forward motion until the robot achieves a sufficiently safe heading. This combination enables reliable and robust collision avoidance even under abrupt dynamic disturbances.

A.2 Performance in Seen Environments

We evaluate NoMaD and ViNT in a fully seen environment without added OOD obstacles as a sanity check to verify that our implementations reproduce the original models. All experiments are conducted using the LoCoBot platform.

Setup. The test environment and overall procedure follow the same setup as in the goal-conditioned navigation experiments in our paper. A topological graph is predefined using a sequence of image goals. For fair comparison, the subgoal image sequences fed to both NoMaD and ViNT are fixed and reused across all trials. This sequence is also identical to those used in the OOD experiments presented in the main paper.

Metric. We measure the success rate, defined as the percentage of trials in which the robot reaches the image goal without any collisions. Each configuration is tested for 10 trials.

Table 4: Success rate in seen environments without OOD obstacles

Model	Success Rate (10 trials)
NoMaD	100%
ViNT	100%

Results and Analysis. Both NoMaD and ViNT achieve a 100% success rate in the seen environment, completing all 10 trials without collision or failure. These results confirm that both the pretrained vision-based models and the constructed topological navigation pipeline function reliably under seen conditions.

The experiments are conducted on the LoCoBot platform (see Sec. B.2), which closely matches the setup used in the original NoMaD and ViNT papers and is also employed to train and tune the released pretrained weights[5]. The consistent performance thus validates the correct integration and implementation of the models. This outcome supports the analysis that the performance degradation observed in other experiments (e.g., with OOD obstacles or dynamic human interventions) is not due to flaws in the experimental setup or execution, but rather reveals the inherent limitations of vision-only navigation models when deployed in unfamiliar environments without fine-tuning.

B Experimental Details

B.1 Test Instances for Goal-Conditioned Navigation

Figure 6 presents the 10 test instances used in the goal-conditioned navigation experiment described in the main paper. Each instance includes 15 OOD obstacles, shown as brown boxes, grouped into four to six clusters of varying shapes and placed randomly throughout the environment. These settings are designed to evaluate the robustness of navigation policies under unseen and cluttered conditions.

[5]https://github.com/robodhruv/visualnav-transformer

Figure 6: Test instances for goal-conditioned navigation

B.2 Robot Platforms and Specifications

We run experiments on three mobile platforms: LoCoBot, TurtleBot4, and RoboMaster S1. Figure 7 shows the actual hardware used in our experiments.

(a) LoCoBot (b) TurtleBot4 (c) RoboMaster S1

Figure 7: Mobile robot platforms used for evaluation. Each robot was equipped with a monocular RGB camera.

A note for LoCoBot. Although the original LoCoBot is built on a TurtleBot2 base, we implement our LoCoBot using a TurtleBot4 as TurtleBot2 has been discontinued. Nevertheless, two models are both differential drive type with very similar wheelbase width, wheel diameter, and etc. A custom 3D-printed camera mount is designed to match the fisheye camera position used in prior NoMaD and ViNT implementations.

Notes for parameters in Table 5.

Table 5: Robot specifications and camera configuration. All resolutions are in pixels, and dimensions are in millimeters (mm).

Specification	**LoCoBot**	**TurtleBot4**	**RoboMaster S1**
Max Depth Range τ_z (m)	1.0	1.2	1.0
Depth Offset (m)	0.05	0.2	-0.1
Published Image Resolution (pixels)	320×240	320×200	640×360
Robot Size (L × W × H, mm)	$341 \times 339 \times 350$	$341 \times 339 \times 351$	$320 \times 240 \times 270$
Camera Height (mm)	340	245	240
Camera X-offset (mm)	10	-60	70

- Max Depth Range τ_z (m): The maximum range (in meters) for depth sensing used during obstacle detection. This value is empirically tuned to prevent false positives caused by distant walls or floor misclassification.
- Depth Offset (m): An offset subtracted from the estimated depth to make obstacles appear closer (or further) than predicted. This compensates for the physical body radius of robots and camera mounting position, and is further tuned empirically to improve depth estimation.
- Published Image Resolution: The resolution of the RGB images published by each camera over ROS 2.
- Robot size (L × W × H): Physical dimensions of each robot, measured in millimeters (mm).
- Camera Height: The vertical distance from the ground to the optical center of the camera.
- Camera X-offset: The horizontal distance (in mm) between the geometric center of robots and the camera. A positive value indicates a forward-facing offset (camera mounted ahead of center), while a negative value indicates a rear-facing offset.

B.3 Common Parameters for CARE

CARE uses a small set of task-agnostic parameters that are experimentally tuned for effective and safe navigation across all platforms. Table 6 summarizes the key scalar parameters used in the CARE module.

Table 6: CARE module parameters used across all experiments.

Parameter	**Value**	**Description**
θ_{clip} (rad)	$\pi/4$	Maximum heading adjustment angle induced by repulsive force
θ_{thres} (rad)	$\pi/6$	Threshold for triggering in-place rotation when the desired heading change exceeds this value
ϵ (m)	-0.05	Vertical offset for filtering out ceiling points during top-down projection. A negative value indicates upward exclusion

The value of θ_{clip} is set to allow sufficient rotation in response to repulsive forces while preventing excessive deviation or backward-pointing waypoints. The value of θ_{thres} is chosen to be smaller than θ_{clip} to trigger in-place rotation only in high-risk situations with large heading changes. Finally, the vertical offset ϵ is used to exclude points located above the robot (e.g., ceilings or high shelves) during depth-based top-down projection, improving the relevance of obstacle detection.

B.4 Depth Estimation Model: UniDepthV2

For monocular metric depth estimation, we employ UniDepthV2 with the ViT-S backbone (the smallest variant)[6], enabling real-time inference. The model is used without any fine-tuning or adaptation. Top-down projections are computed using platform-specific parameters for maximum sensing range τ_z and vertical offset ϵ, as summarized in Table 5 and 6.

B.5 Trajectory Selection in NoMaD and ViNT

For NoMaD, we use the default diffusion-based sampling setup to generate 8 trajectories, each consisting of 8 waypoints in the robot's local frame. In our experiments, we consistently select the second waypoint $\mathbf{p}_2$ from the first generated trajectory for control (as implemented in the open-source codes). ViNT, on the other hand, produces a single predicted trajectory using a Transformer decoder. Similarly, we select the second waypoint from this predicted trajectory as the control target (also the same with the original implementation of the open-source codes).

This fixed selection strategy is to isolate and evaluate the contribution of our proposed collision avoidance mechanism at the level of local planning. Since our focus is not on high-level trajectory selection or adaptive replanning, we maintain the same trajectory and waypoint index to control for variation across trials. All experiments are conducted under this consistent setting, using fixed random seeds and sampling strategies to ensure fair comparison.

[6] https://github.com/lpiccinelli-eth/UniDepth