Comparing Classical and Reactive Local Planners for Dynamic-Obstacle Navigation in CoppeliaSim

Bruno L. Lima
Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte, Brazil
bruno.lima@dcc.ufmg.br

***Abstract*—This paper presents an experimental comparison of three local planners for dynamic-obstacle navigation in simulation: Dynamic Window Approach (DWA), Timed Elastic Band (TEB), and Optimal Reciprocal Collision Avoidance (ORCA). All planners are implemented in Python and evaluated in CoppeliaSim using a Kobuki differential-drive robot equipped with a FastHokuyo LiDAR, with a Pioneer 3-DX acting as a dynamic obstacle. The study examines three scenarios: a static L-shaped corridor, a head-on encounter in a straight hallway, and a cross-shaped intersection with crossing traffic. Safety and efficiency metrics—including success rate, collisions, deadlocks, time to goal, path length, minimum clearance, and near misses—are used to characterize typical behaviors and failure modes of each planner. Results indicate that DWA offers consistent performance across all scenarios, TEB generates high-quality trajectories in static settings, and ORCA performs well in intersection-like contexts where its assumptions are approximately met. The paper also examines the effects of deploying ORCA unilaterally on the controlled robot. At the same time, the Pioneer 3-DX follows a scripted path, thus breaking reciprocity and revealing practical limitations often hidden in idealized setups. The presented framework is designed as a reproducible baseline for future research on enhanced ORCA variants and learning-based local planners.**

***Index Terms*—Mobile robot navigation, dynamic obstacles, local planning, CoppeliaSim**

I. Introduction

Local navigation in dynamic environments remains a core challenge for mobile robots operating in warehouses, service domains, and human-populated areas. Classical local planners, such as the Dynamic Window Approach (DWA) and Timed Elastic Band (TEB), are widely adopted due to their robustness and ease of integration into navigation stacks [1]–[3]. These methods perform well in structured or slowly changing settings, where global plans can be followed with minimal need for reactive maneuvers. In contrast, reactive multi-agent strategies, such as Optimal Reciprocal Collision Avoidance (ORCA), are designed to manage interactions with moving agents by reasoning directly in velocity space [4], [5]. ORCA has been applied in various multi-robot and crowd-navigation contexts.

Recent surveys emphasize that classical planners, velocity-obstacle methods, and learning-based approaches exhibit distinct strengths and failure modes in the presence of dynamic obstacles [3], [6]. Understanding the behavior of DWA, TEB, and ORCA in controlled but challenging environments is a critical step before transitioning to hybrid or reinforcement-learning-based solutions [7], [8].

This work conducts a simulator-based evaluation of DWA, TEB, and ORCA across three representative corridor-like scenarios featuring both static and dynamic obstacles. All experiments are executed in CoppeliaSim [9], where a Kobuki robot serves as the controlled agent and a Pioneer 3-DX acts as a moving adversary. The planners are implemented from scratch in Python and share a unified global planner, map representation, and metric logging system.

The main contributions of this work are:

- A reproducible experimental setup for comparing DWA, TEB, and ORCA across dynamic-obstacle scenarios in CoppeliaSim.
- A quantitative evaluation using safety and efficiency metrics, including success rate, collisions, deadlocks, time to goal, path length, minimum robot/wall distances, and near misses.
- A qualitative analysis of failure modes, particularly the impact of unilateral ORCA use, where only the Kobuki runs ORCA and the Pioneer 3-DX follows a scripted policy.
- A discussion of implementation factors and design trade-offs that account for deviations from idealized planner behavior reported in previous literature.

Although the current study focuses on classical and reactive planners, the framework is designed to support future extensions to learning-based or hybrid strategies [8].

II. Related Work

Velocity-space planners such as DWA [1] and optimization-based methods such as TEB [2] are standard components of mobile robot navigation stacks. Both are widely used in research and practical applications and are frequently included as baselines in comparative surveys on sidewalk and urban navigation [3]. Recent work continues to refine these methods, for example, by enhancing prediction, obstacle handling, and integration with higher-level planners [10], [11].

ORCA extends velocity-obstacle concepts to reciprocal collision avoidance, offering sufficient conditions for safe multi-agent navigation under symmetry assumptions [4]. Variants such as VR-ORCA [5], as well as adaptations for non-holonomic robots and UAVs, relax these assumptions and dy-

namically distribute collision-avoidance responsibility among agents, thereby improving performance in dense or constrained environments.

Hybrid approaches that combine classical planners with reactive modules or path-finding layers have been proposed to address issues such as deadlocks and oscillations in multi-agent contexts [6]. Learning-based local planners, as well as schemes that switch between classical and learned policies, have been investigated for crowd navigation and human–robot interaction [7], [8], [12]–[14].

Although learning-based planners are not explicitly evaluated in this study, the proposed benchmark provides a robust, extensible foundation for future comparisons of such methods.

III. Methodology

A. Simulation Environment and Robots

All experiments are conducted in CoppeliaSim 4.10 [9] using the ZMQ Remote API for Python. The controlled agent is a Kobuki-like differential-drive robot equipped with a FastHokuyo LiDAR (maximum range 5.0 m). A Pioneer 3-DX model acts as a moving obstacle in the dynamic scenarios, following scripted motion patterns. Only the Kobuki executes the evaluated local planners; the Pioneer 3-DX does not run ORCA, intentionally violating the reciprocity assumption of the original method. The consequences of this unilateral use of ORCA are revisited in the discussion.

The Kobuki is modeled with the following key parameters: radius $r = 0.175$ m, maximum linear velocity $v_{\max} = 0.30$ m/s, maximum angular velocity $\omega_{\max} = 1.00$ rad/s, maximum linear acceleration 0.6 m/s^2, and maximum angular acceleration 1.5 m/s^2.

B. Mapping and Global Planning

An orthographic vision sensor mounted above the environment captures an RGB image of the floor plan. The map extraction pipeline converts this image to grayscale, thresholds it to obtain a binary occupancy grid, and inverts the vertical axis so that world and map coordinates are aligned. The map resolution is computed from the orthographic size (12.0 m) and the largest image dimension, and the map origin is set to the sensor position shifted to the lower-left corner.

Static obstacles in the map are dilated by three times the robot radius using morphological operations to account for the robot footprint and pose uncertainty. A global A* planner operates on this inflated occupancy grid, producing a sequence of waypoints that guide the robot from start to goal, following standard practice in combining graph search with local reactive planning [11]. The local planners track these waypoints while performing obstacle avoidance.

C. Scenarios

Three scenarios are implemented, as illustrated in Fig. 1. In all cases, the Kobuki starts near one end of the corridor and must reach a goal at the opposite side.

- **Scene 1 – Static L-shaped corridor:** only static walls are present. This scenario is used to assess baseline navigation performance, distance to walls, and trajectory smoothness without dynamic interactions.
- **Scene 2 – Head-on encounter in straight corridor:** the Pioneer 3-DX starts at the opposite end of the corridor and moves toward its own goal, creating a head-on interaction with the Kobuki.
- **Scene 3 – Cross-shaped corridor:** the robots move along perpendicular branches of a cross-shaped corridor, creating intersection-like interactions and potential near misses.

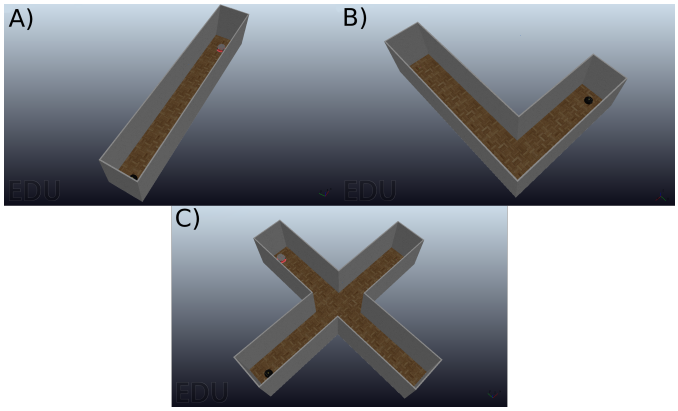

Fig. 1. Navigation scenarios used for evaluation in CoppeliaSim.

D. Local Planner Implementations

All planners share the same low-level interface: given the current pose, the next goal waypoint, LiDAR data, and, when needed, the estimated pose of other robots, they output linear and angular velocity commands.

1) Dynamic Window Approach (DWA): The DWA implementation follows the standard formulation with a prediction horizon of 2.0 s, discretized with a time step of 0.1 s [1]. Candidate velocity pairs are sampled within a dynamic window constrained by the current velocity and acceleration limits. For each candidate, the resulting trajectory is simulated and scored using a weighted sum of:

- heading alignment toward the next waypoint;
- minimum distance to obstacles, with a target safety clearance of 0.5 m;
- linear velocity magnitude.

If no collision-free trajectory exists, the algorithm selects the trajectory with maximum clearance, penalized by misalignment with the goal.

2) Timed Elastic Band (TEB): The TEB implementation maintains a fixed-size sequence of waypoints (12 nodes) between the current pose and the next global waypoint. At each control cycle, a limited number of gradient-descent iterations update the band to minimize a cost function combining:

- distance to the global waypoint;
- distance to static obstacles;
- distance to other robots (when present);
- smoothness via curvature and acceleration penalties.

Velocity commands are obtained by tracking the first segment of the optimized band, with adaptive speed scaling based on heading error and proximity to obstacles. The overall structure follows the widely used TEB local planner and is conceptually related to recent improvements that better account for obstacle motion and curve handling [2], [10].

3) Optimal Reciprocal Collision Avoidance (ORCA): The ORCA implementation maintains a short history of robot poses to estimate the velocities of the Kobuki and the Pioneer 3-DX, as well as clusters of LiDAR points representing moving obstacles. For each neighbor within a 3.0 m radius, a velocity obstacle is constructed and converted into an ORCA half-plane in velocity space. A two-dimensional linear program then searches for a feasible velocity vector as close as possible to the preferred velocity (pointing from the Kobuki to the next waypoint), following the original formulation [4] and its variable-responsibility variant [5].

In the original ORCA formulation, all agents solve similar optimization problems and share responsibility for collision avoidance. In this work, only the Kobuki runs ORCA, while the Pioneer 3-DX follows a scripted controller that ignores ORCA constraints. As a result, the theoretical safety guarantees no longer hold, and ORCA behaves as an advanced unilateral avoidance strategy. This design choice is later analyzed as a source of unexpected behaviors, especially in Scenes 2 and 3.

E. Metrics and Experimental Protocol

For each combination of planner and scenario, 20 runs are executed with identical initial conditions and randomized variations in the robot's start orientation or timing. The following metrics are collected:

- **Success rate:** percentage of runs where the Kobuki reaches the goal within a 0.2 m tolerance without collisions or early termination.
- **Collisions:** number of events where the distance between the Kobuki and any obstacle falls below $r + 0.15$ m.
- **Deadlocks:** events where the robot moves less than 0.1 m within 5 s, indicating local minima or mutual blocking.
- **Time to goal:** elapsed time in successful runs.
- **Path length:** total traveled distance in successful runs.
- **Minimum wall distance:** (Scene 1) minimum distance to static walls.
- **Minimum robot distance:** (Scene 2) minimum distance between the Kobuki and the Pioneer 3-DX.
- **Intersection distance and near misses:** (Scene 3) minimum distance at the intersection and number of events where robots come closer than 0.25 m.

Simulation time is capped at 300 s per run. All pose, velocity, and metric data are logged for offline analysis. Plots are generated using Python and exported to PNG files for inclusion in this paper.

F. System Architecture and Execution Pipeline

The software architecture follows a modular design, separating sensing, mapping, planning, control, and logging into distinct components implemented in Python.

The *DifferentialRobotController* encapsulates the low-level interface with CoppeliaSim, handling pose retrieval, FastHokuyo LiDAR acquisition, and wheel velocity commands for the Kobuki. All planners interact with the simulator exclusively through this controller, which provides a unified API for reading the robot state and applying control inputs.

The *Mapper* module is responsible for extracting and maintaining the occupancy grid used by the global planner. It processes RGB images from the orthographic vision sensor, performs grayscale conversion, thresholding, coordinate frame inversion, and obstacle dilation, and exposes utility functions for world-to-map and map-to-world conversions.

Global planning is handled by the *RoutePlanner*, which runs A* on the dilated occupancy grid to compute a collision-free route from start to goal. The route is discretized into waypoints, and the planner exposes the current target waypoint based on the robot's position and a look-ahead rule. This keeps the global path fixed while allowing the local planners to deviate locally for obstacle avoidance.

Local planners (DWA, TEB, ORCA) are implemented as interchangeable modules that receive the current pose, LiDAR data, the next global waypoint, and (when applicable) estimated states of other robots, and output linear and angular velocity commands. This design makes it straightforward to add new local planners or ablation variants while keeping the rest of the pipeline unchanged.

A dedicated *Metrics* module runs in parallel with the control module, logging robot poses, velocities, distances to obstacles, collision flags, deadlocks, and near misses at each control step. After each run, it computes summary statistics and exports JSON files for offline analysis.

The overall execution flow is:

1) Initialization: connect to CoppeliaSim, load the scene, extract the map, and compute the global path.
2) Control loop: read robot and LiDAR data, detect other robots if present, query the local planner for a velocity command, send the command, and update metrics and optional visualization.
3) Termination: stop when the goal is reached, the time limit is exceeded, or a failure condition is triggered; then aggregate and save results.

G. Planner-Specific Heuristics and Implementation Details

Beyond the core algorithms described earlier, several practical heuristics were required to obtain stable behavior in CoppeliaSim. These choices are often omitted in high-level descriptions but have a direct impact on the observed performance.

For DWA, an explicit escape strategy is used when no collision-free trajectory is found within the dynamic window. Instead of stopping the robot, the planner selects the trajectory with the largest minimum clearance and penalizes misalignment with the goal. This prevents the robot from freezing in narrow passages but can temporarily bring it closer to obstacles, contributing to some of the minor contacts observed

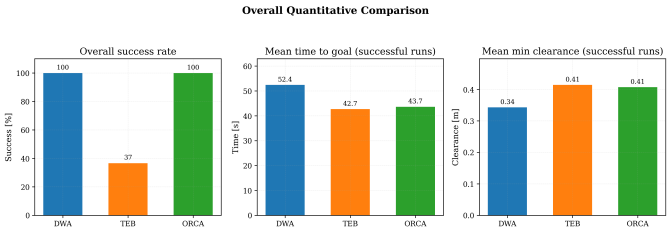

Fig. 2. Overall comparison of DWA, TEB, and ORCA across all scenes.

in dynamic scenes. Additionally, when the computed dynamic window becomes too small due to numerical issues or very low current velocities, the implementation falls back to the full velocity bounds to avoid getting stuck around zero velocity.

For TEB, the implementation uses an iterative optimization scheme with a limited number of gradient descent steps per control cycle. To keep computation bounded in real time, the band size is fixed to 12 nodes, and the learning rate is capped. Velocity commands are derived by tracking the first segment of the optimized band with an adaptive speed factor that depends on the heading error and the minimum distance to obstacles. This ensures smoother motion in static environments but makes the planner sensitive to highly constrained situations, where the band can converge to locally feasible but globally unproductive configurations, contributing to deadlocks in Scenes 2 and 3.

For ORCA, several engineering decisions are needed to bridge the gap between the ideal multi-agent model and LiDAR-based perception. The implementation maintains short histories of positions for the Kobuki and the Pioneer 3-DX, as well as clusters of LiDAR points, from which approximate velocities are estimated. Moving obstacles are associated across time using a distance threshold of 0.5 m, and separate time horizons are used for static and dynamic entities. When solving the two-dimensional linear program, the preferred velocity is clipped to the maximum linear speed and blended with the previous command to reduce abrupt changes in angular velocity. Finally, when the distance to another robot drops below 2.0 m, an additional speed reduction factor is applied on top of the ORCA solution as a safety margin.

These implementation details help explain some of the empirical differences between the tested planners and the idealized behaviors reported in prior work. They also highlight where future improvements—for example, better velocity estimation, more robust band optimization, or refined sampling in DWA—are likely to have the most significant impact within this benchmarking framework.

IV. Results

A. Overall Comparison

Figure 2 summarizes the overall success rates, average time to goal, and minimum clearances across all scenarios (only successful runs are considered for temporal and clearance metrics).

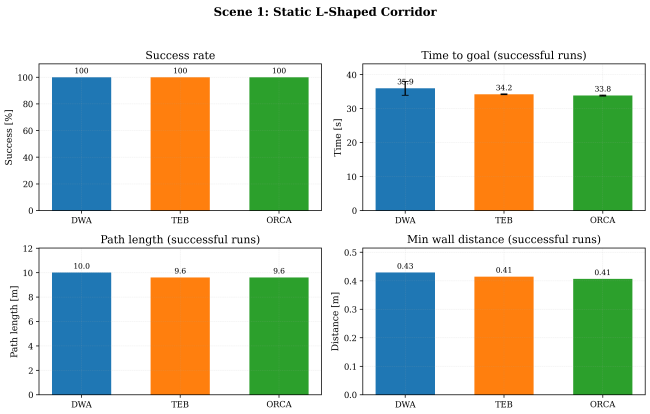

Fig. 3. Results for Scene 1: static L-shaped corridor.

DWA and ORCA achieved a 100% success rate, while TEB succeeded in only 37% of the trials, primarily due to failures in dynamic scenes. In successful runs, TEB and ORCA reached the goal faster than DWA and generally maintained slightly higher clearances. However, these aggregate results conceal significant scenario-specific behaviors and failure modes, discussed in detail below.

B. Scene 1: Static L-Shaped Corridor

Figure 3 reports detailed results for the static L-shaped corridor. All planners achieved 100% success and exhibited similar path lengths around 9.6–10.0 m. ORCA yielded the lowest average time to goal, followed closely by TEB, while DWA was marginally slower but still competitive.

The minimum wall distance plot shows that all planners maintained safe margins, with DWA producing the largest mean clearance (about 0.43 m). TEB delivered qualitatively smoother trajectories and incurred no collisions in this scene, aligning with prior reports on its strengths in static environments [2], [10].

Overall, in purely static corridor navigation with a reliable global plan, all three planners perform well. The observed differences are relatively small and appear to be influenced more by implementation details (such as cost weights and tuning) than by fundamental algorithmic properties.

C. Scene 2: Head-On Encounter in Straight Corridor

In the head-on scenario, the Kobuki and Pioneer 3-DX move toward each other from opposite ends of a narrow corridor. Detailed results are shown in Fig. 4.

DWA and ORCA maintained a 100% success rate, while TEB succeeded in only 2 out of 20 runs. DWA exhibited a moderate number of collisions and deadlocks, indicating that the robot sometimes grazed the opponent or temporarily stalled before resolving the conflict. TEB, despite a low collision count, suffered frequent deadlocks and long detours, rendering it an ineffective practical solution for this scenario.

ORCA presents a more nuanced pattern: it preserves complete success but accumulates a very high number of collision events (182 over 20 runs). Visual inspection shows that many

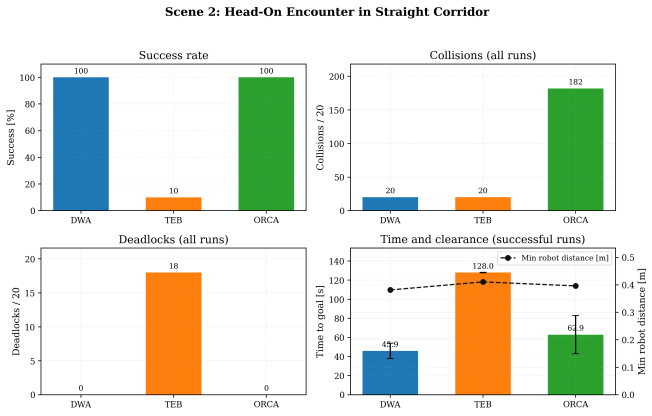

Fig. 4. Results for Scene 2: head-on encounter in a straight corridor.

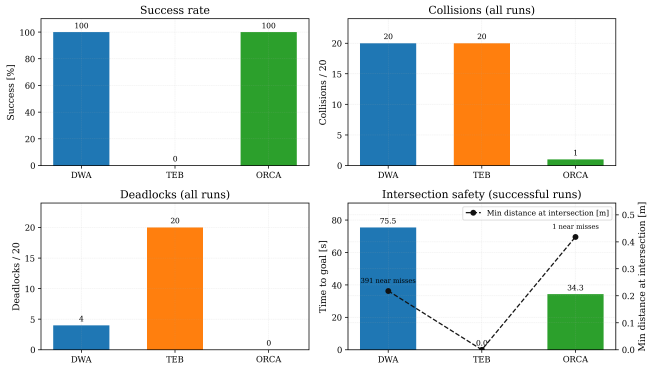

Fig. 5. Results for Scene 3: cross-shaped corridor with intersecting traffic.

of these events correspond to short-lived penetrations of the safety radius when the opponent robot does not contribute to avoidance. Since only the Kobuki runs ORCA, and the Pioneer 3-DX follows a scripted trajectory, the reciprocity assumption is violated. As a result, ORCA often selects aggressive maneuvers that theoretically rely on the opponent also adjusting its velocity; when this does not occur, the Kobuki cuts too close, triggering frequent contact detections.

This exposes a crucial limitation also noted in benchmarks that combine ORCA with additional coordination or path-finding layers [6]. Without full agent reciprocity, ORCA's theoretical guarantees degrade, and empirical outcomes may even be worse than those of simpler methods in terms of collision statistics. From a benchmarking perspective, this is an essential negative result, suggesting that ORCA should either be applied to all interacting agents or be complemented with additional safety mechanisms.

D. Scene 3: Cross-Shaped Corridor

The cross-shaped corridor represents an intersection where robots traverse perpendicular arms and must coordinate at the center. Results are shown in Fig. 5.

In this setting, ORCA clearly outperforms the other planners: it achieves 100% success with almost no collisions and no deadlocks, and it completes the task roughly twice as fast as DWA. The minimum distance at the intersection remains reasonable, with only a single near miss recorded.

DWA also achieves 100% success, but with significantly more collisions and deadlocks. The planner often exhibits oscillatory behavior at the intersection when both robots attempt to pass, resulting in long negotiation phases and multiple minor contacts. This reflects DWA's inherently myopic prediction horizon and its difficulty in reasoning about mutual interactions.

TEB fails to complete the runs in this scenario under the tested configuration, resulting in 0% success. The optimizer frequently converges to locally feasible but globally ineffective configurations near the intersection, where multiple constraints from static walls and the moving robot compete. These results suggest that the current TEB implementation and parameterization, which perform well in static corridors, are insufficient for dense multi-agent interactions without additional modeling of dynamic behaviors [3], [10].

Interestingly, ORCA's collision behavior in Scene 3 differs from that in Scene 2. At the intersection, the Pioneer 3-DX's scripted motion aligns better with ORCA's assumptions (roughly constant velocity along a predictable path), allowing the Kobuki to exploit velocity obstacles more effectively. This reinforces the importance of matching ORCA's modeling assumptions with the actual motion of other agents [4], [5].

V. Discussion

The experiments reveal several implementation- and scenario-dependent effects that complement, and in some cases contradict, idealized expectations from the literature and recent benchmarks [3], [6].

First, DWA emerges as a robust baseline: it always reaches the goal in all three scenarios, despite occasional collisions and deadlocks in dynamic settings. Its simplicity, limited prediction horizon, and hand-tuned cost weights make it relatively easy to deploy, but also expose it to oscillations and suboptimal negotiation behaviors when interacting with other robots.

Second, TEB demonstrates its strengths in static environments but degrades sharply in the presence of dynamic obstacles. In the current implementation, the planner treats the opponent robot primarily as a moving obstacle in the elastic band optimization, without explicitly modeling its future trajectory. This leads to infeasible or highly constrained optimization problems in Scenes 2 and 3, where the optimizer settles into configurations that satisfy constraints locally but do not progress toward the goal. This failure mode is consistent with reports that TEB and similar trajectory optimizers require careful tuning and additional heuristics to handle rapidly changing obstacle patterns [2], [10].

Third, ORCA demonstrates both its promise and its sensitivity to modeling assumptions. When the Pioneer 3-DX follows a predictable path that roughly matches ORCA expectations (Scene 3), the planner excels, offering fast, collision-free navigation with no deadlocks. However, in the head-on corridor (Scene 2), the unilateral ORCA setup produces many minor

collisions because the opponent does not share responsibility for avoidance. This illustrates that applying ORCA only on the ego robot, without instrumenting other agents or adding fallback safety checks, can lead to deceptively poor collision statistics even when macro-level behavior (success rate) looks good, a phenomenon also reported in multi-agent navigation studies that combine ORCA with explicit deadlock-resolution mechanisms [6].

From an implementation perspective, these findings suggest several directions for improvement:

- Extending ORCA to the Pioneer 3-DX or implementing a reciprocal behavior model to restore its theoretical guarantees.
- Refining deadlock and collision detectors to distinguish between brief grazing contacts and persistent unsafe behaviors, which are currently counted equally.
- Enhancing TEB with better models of dynamic obstacles or integrating it with a higher-level collision-avoidance layer to reduce failures in multi-agent scenes.
- Exploring improved DWA variants, such as IA-DWA [11], within the same benchmark to assess how far classical sampling-based planners can be pushed before resorting to learning-based approaches.

VI. Conclusions and Future Work

This paper presented a controlled comparison of DWA, TEB, and ORCA for local navigation with dynamic obstacles in CoppeliaSim. Using a Kobuki robot with a FastHokuyo LiDAR and a Pioneer 3-DX as a moving adversary, three corridor-like scenarios were implemented to capture static navigation, head-on encounters, and cross-shaped intersections.

The main conclusions are:

- DWA provides a strong baseline with 100% success in all scenarios but suffers from collisions and deadlocks in dense interactions.
- TEB excels in static environments but, under the tested configuration, fails to handle multi-agent dynamics, leading to low success rates and frequent deadlocks.
- ORCA can outperform both methods in intersection-like scenarios when its assumptions are approximately satisfied. Still, unilateral application (only on the Kobuki) leads to many collisions in head-on encounters due to broken reciprocity.

The experimental framework, including map extraction, global planning, local planners, and metric collection, is fully implemented in Python and can be extended to support reproducibility and new planners. Future work will focus on:

- Implementing fully reciprocal ORCA between both robots and adding additional reactive baselines.
- Incorporating learning-based local planners and hybrid schemes that switch between classical and learned policies.
- Expanding the set of scenarios to include wider corridors, bottlenecks, and human-like trajectories.
- Refining metrics and detectors to better distinguish between transient contacts, safe negotiation behaviors, and truly unsafe situations.

Additional related work, particularly on learning-based navigation and hybrid planning, will be integrated in an extended version of this paper, together with an expanded bibliography.

References

[1] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1997, pp. 378–385.

[2] J. Wu, X. Ma, T. Peng, and H. Wang, "An improved timed elastic band (teb) algorithm of autonomous ground vehicle (agv) in complex environment," *Sensors*, vol. 21, no. 24, p. 8312, 2021.

[3] D. Gómez-Ayalde and V. A. Romero-Cano, "Local planning methods for autonomous navigation on sidewalks: A comparative survey," in *2022 IEEE Colombian Conference on Applications of Computational Intelligence (ColCACI)*, 2022, pp. 1–6.

[4] J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha, "Optimal reciprocal collision avoidance for multi-agent navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 262–267.

[5] K. Guo, D. Wang, T. Fan, and J. Pan, "Vr-orca: Variable responsibility optimal reciprocal collision avoidance," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4815–4822, 2021.

[6] S. Nair, R. Chandra, B. Ichter, A. R., K. Bharath, P. Abbeel, S. Levine, N. Naik *et al.*, "DynaBARN: Benchmarking metric ground navigation in dynamic environments," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2022, pp. 63–70.

[7] K. Nakhleh, M. Raza, M. Tang, M. Andrews, R. Boney, I. Hadzic, J. Lee, A. Mohajeri, and K. Palyutina, "Sacplanner: Real-world collision avoidance with a soft actor critic local planner and polar state representations," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023. [Online]. Available: https://arxiv.org/abs/2303.11801

[8] V. D. Sharma, J. Lee, M. Andrews, and I. Hadžić, "Hybrid classical/rl local planner for ground robot navigation," *CoRR*, vol. abs/2410.03066, 2024. [Online]. Available: https://arxiv.org/abs/2410.03066

[9] G. Montenegro, R. Chacón, E. Fabregas, G. Garcia, K. Schröder, A. Marroquín, S. Dormido-Canto, and G. Farias, "Modeling and control of a spherical robot in the CoppeliaSim simulator," *Sensors*, vol. 22, no. 16, p. 6020, 2022.

[10] J. Ou, Y. Liu *et al.*, "Local path planner for mobile robot considering future positions of obstacles," *Processes*, vol. 12, no. 5, p. 984, 2024.

[11] H. He, Y. Zhang *et al.*, "Autonomous navigation of mobile robots based on an improved IA-DWA algorithm," *Scientific Reports*, vol. 15, p. 2099, 2025.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," in *arXiv preprint arXiv:1707.06347*, 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[13] 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. [Online]. Available: https://arxiv.org/abs/1801.01290

[14] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018, pp. 1587–1596. [Online]. Available: https://arxiv.org/abs/1802.09477