Collaborative Path Planning and Obstacle Avoidance for Autonomous TurtleBot Navigation

Aditya Sule
Symbiosis Institute of Technology, Pune Campus
Symbiosis International (Deemed University)
Pune, India
adityansule@gmail.com

Karv Amin
Symbiosis Institute of Technology, Pune Campus
Symbiosis International (Deemed University)
Pune, India
karvamin@gmail.com

Ovi Kulkarni
Symbiosis Institute of Technology, Pune Campus
Symbiosis International (Deemed University)
Pune, India
contactovikulkarni@gmail.com

Palvi Kulkarni
Symbiosis Institute of Technology, Pune Campus
Symbiosis International (Deemed University)
Pune, India
contactpalvikulkarni@gmail.com

Aniket Nargundkar
Symbiosis Institute of Technology, Pune Campus
Symbiosis International (Deemed University)
Pune, India
aniket.nargundkar@sitpune.edu.in

Abstract—**Recently, there has been an emerging interest in multi-robot systems with coordinated navigation in complex environments. This work presents the master-slave coordination system in which two TurtleBot4 robots are simulated in MATLAB environment. In this configuration, the master robot navigates and autonomously senses its surroundings with a simulated 3D LiDAR sensor, executes path planning with A* algorithm and issues velocity commands. These command signals are directly relayed to slave robot that moves in unison with the master. In the presence of an obstacle on its travel, the slave momentarily performs a localized obstacle avoidance maneuver, and then resynchronizes with the master. Contrarily to most available systems, no distinction between static and dynamic obstacles is made and, instead, the system is based on reactive re-planning and real-time sensing. The methodology shows an efficient and scalable procedure towards the coordinated navigation by coupling global path planning with local autonomy. The findings demonstrate the success of the hybrid master–slave configuration to manage obstacle-rich environments and form the basis to consider expanding the similar strategies to robot teams in future studies.**

Index Terms—**Multi-robot systems, Master–slave coordination, MATLAB simulation, TurtleBot4, A* algorithm, LiDAR sensing, Obstacle avoidance, Autonomous navigation.**

I. Introduction

The study of robotics is increasingly focused on multi-robot systems that enable cooperative autonomy in complex and uncertain environments. Among the common approaches, swarm robotics and master–slave coordination are two key strategies that address robustness, scalability, and adaptivity in navigation tasks. These frameworks are highly relevant to applications such as warehouse automation [3], search-and-rescue missions [4], and agricultural robotics [5], where distributed or coordinated robotic systems must perform efficiently in the presence of obstacles. Multi-robot navigation has been widely validated on platforms such as ROS 2 [6], [7], and simulation frameworks including Gazebo and MATLAB [8], [9].

This work builds on these developments by designing and simulating a master–slave robotic system of two Turtlebot4 robots integrated in MATLAB. The master robot is equipped with global perception and performs environment scanning and path planning using algorithms such as A*, while the slave robot mimics the master's motion commands. When faced with a local obstruction not encountered by the master, the slave temporarily applies reactive obstacle avoidance before re-synchronizing with the master.

1.1 Swarm Robotics

1.1.1 Technology: Swarm robotics generally involves multiple robots interacting through local communication and distributed control, often without centralized coordination [10]. Middleware such as ROS [11], [12] and toolboxes in MATLAB [13], [14] have accelerated academic and industrial research in this field. While this work does not employ large-scale swarming, it adopts certain swarm principles such as coordination, robustness, and shared navigation logic in a simplified two-robot setup.

1.1.2 Concept: In fully decentralized swarms, decision-making is distributed to reduce single-point failures [16], [17]. In contrast, this work uses a semi-centralized approach: the master robot conducts global planning, while the slave relies on the master for motion commands but exhibits short-term autonomy when required. This hybrid principle ensures robustness without the full complexity of swarm-level decentralization [18].

1.1.3 Application: Applications of swarm robotics principles are observed in warehouse logistics [3], multi-robot exploration [5], and cost-efficient coverage of hazardous terrain during search-and-rescue [4]. These motivations form the basis for the master–slave approach adopted in this project.

1.2 Master–Slave Model

1.2.1 Technology: The master–slave topology is a fundamental coordination mechanism in multi-robot systems. The master robot computes navigation command sets according to its environment map and path-planning algorithm, then transmits corresponding linear and angular velocities to the slave. This ensures synchronized movement while centralizing perception and planning within the master. The slave does not perform global mapping or planning, instead relying entirely on control signals, except when localized avoidance is necessary.

1.2.2 Concept: At its simplest, the slave robot executes commands blindly from the master. However, hybrid autonomy can be integrated where the slave avoids obstacles that only block its own path, before re-synchronizing with the master [9], [19]. This model increases resilience while retaining centralized global navigation.

1.2.3 Application: Master–slave coordination is applied in teleoperation systems [19], robotic training platforms [13], and formation control [10]. In industrial automation, such setups are useful for leader–follower navigation, while in exploration domains, they ensure synchronized coverage [14].

1.3 Obstacle Avoidance Algorithms

1.3.1 Technology and Concept: Autonomous navigation requires algorithms that generate safe, feasible, and efficient paths in environments with obstacles. Key methods include:

- **A***: A heuristic graph-search algorithm that is both optimal and efficient for grid-based environments [2], [7].
- **RRT (Rapidly Exploring Random Trees)**: A sampling-based algorithm suitable for high-dimensional continuous spaces [4], [16].
- **RRT***: An optimized extension of RRT that incrementally improves path quality [17].
- **Evolutionary Algorithms (EAs)**: Bio-inspired approaches such as Genetic Algorithms and Particle Swarm Optimization, useful for exploring large solution spaces [1], [15].

In this work, the master robot employs A* for global planning, while the slave performs short-term reactive avoidance when confronted with obstacles that are not present in the master's path.

1.3.2 Application: Path-planning algorithms have been applied to domains such as warehouse navigation [3], outdoor exploration [5], and rescue operations [4]. A* and Dijkstra-based approaches are commonly used in structured indoor settings, while RRT* and bio-inspired methods are often favored in uncertain, dynamic environments [16], [18]. This work evaluates the master–slave model using A* for efficient pathfinding and reactive avoidance for resilience against unexpected obstacles.

II. Methodology

2.1 Static Obstacle in Front of Master

2.1.1 Master–Slave Following Mechanism: In this system, only the master robot is in charge of the perception and path planning, while the slave robot simply shadows its control commands. The 2D LiDAR sensor is used by the master robot to identify obstacles within its surroundings [6], [8].

The distance d to any detected obstacle is computed as:

$$d = \sqrt{(x_{\text{obs}} - x_r)^2 + (y_{\text{obs}} - y_r)^2} \tag{1}$$

where (x_r, y_r) is the robot's position and $(x_{\text{obs}}, y_{\text{obs}})$ is the obstacle position.

The A* algorithm within the Nav2 stack computes an avoidance trajectory when a fixed cylindrical obstacle is present directly ahead [2], [7]. The A* cost function is defined as:

$$f(n) = g(n) + h(n) \tag{2}$$

where $g(n)$ is the path cost from the start node to the current node n, and $h(n)$ is the heuristic estimate of the cost from n to the goal.

The slave robot does not undertake environmental sensing or decision-making; instead, it receives the master's velocity commands:

$$\mathbf{u}(t) = \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3}$$

where $v(t)$ is the linear velocity and $\omega(t)$ is the angular velocity sent periodically over ROS 2 topics [6], [11].

This approach minimizes computational load on the slave and is well-suited for scenarios such as warehouse logistics [3], [14]. The limitation, however, is that the slave lacks spatial awareness and cannot autonomously avoid obstacles [19].

2.1.2 Algorithm Used: The robots are controlled by a leader–follower control algorithm [10], [13]:

- The master performs reactive obstacle avoidance using LiDAR data. Nav2 with A* computes a collision-free path [2], [7].
- Velocity commands (v, ω) are transmitted to the slave at intervals [6], [11].
- The slave executes these commands without performing localization or mapping.
- This ensures low-latency synchronization but restricts autonomy to the master alone.

This method is computationally efficient [15] but fails in cases where the slave encounters independent obstacles [19].

2.2 Static Obstacle in Front of Slave

2.2.1 Master–Slave Following Mechanism: In this case, the master robot moves freely, while the slave encounters an obstacle. The master sends control commands:

$$\mathbf{u}_m(t) = \begin{bmatrix} v_m(t) \\ \omega_m(t) \end{bmatrix} \tag{4}$$

which the slave mirrors initially:

$$\mathbf{u}_s(t) = \mathbf{u}_m(t) \tag{5}$$

If the slave detects an obstacle within a safety threshold $d_{\text{th}} = 0.5\,\text{m}$, it switches to local autonomy:

$$d_{\text{obs}} < d_{\text{th}} \;\Rightarrow\; \text{activate local avoidance} \tag{6}$$

In this temporary mode, the slave executes a turn-and-bypass maneuver until clearance is achieved, after which it resynchronizes with the master [3], [19].

2.2.2 How the Slave Follows the Master: The slave typically mirrors the master's linear velocity v and angular velocity ω [10]. The autonomy procedure is triggered as follows:

1) **Detection:** If $d_{\text{obs}} < 0.5\,\text{m}$, avoidance is triggered [12].
2) **Local Navigation:** The slave generates a reactive control input $\mathbf{u}_s^{\text{local}}(t)$.
3) **Resynchronization:** After clearance, $\mathbf{u}_s(t) \to \mathbf{u}_m(t)$.

This ensures robustness while maintaining master–slave architecture [15], [19].

2.3 Dynamic Obstacle Scenario

2.3.1 Master–Slave Following Mechanism: Here, dynamic obstacles with trajectories $x_{\text{obs}}(t), y_{\text{obs}}(t)$ intersect with the robots' paths [4], [5], [12].

The master detects these via LiDAR and computes a real-time deviation using A*:

$$f(n,t) = g(n,t) + h(n,t) \tag{7}$$

which is recalculated iteratively as obstacle positions update.

The master broadcasts updated velocities (v, ω) to the slave. The slave mirrors these unless an obstacle violates the 0.5 m threshold, in which case it activates temporary autonomy and performs avoidance as in the static slave case.

2.3.2 Algorithm Used:

- **Master:** Executes dynamic A* planning in real time [4], [12].
- **Slave:** Mirrors master commands but switches to local avoidance when required [5], [12].
- **System Behavior:** Both robots run parallel obstacle detection loops ensuring safety and synchronization.

This hybrid of global and local planning ensures resilience to unpredictable dynamic conditions [4], [5], [12].

III. Results and Discussion

This section gives the results of TurtleBot4 master slave navigation experiments that are carried out in the MATLAB simulation environment. It included both static and dynamic obstacles in order to simulate realistic application scenarios like warehouse automation and search-and-rescue [2], [6]. The simulated LiDAR and IMU sensors were installed into the robots and functioned solely with MATLAB without the need of any ROS, and the master robot was enabled to follow the A* algorithm that lead to robust path deviation. The slave used a mirroring strategy and situational autonomy in instances where barriers simply got in the way of its motion [10]. The exchange of information between the robots was supported via a leader-follower mechanism of the common distribution of velocity directives linear and angular.

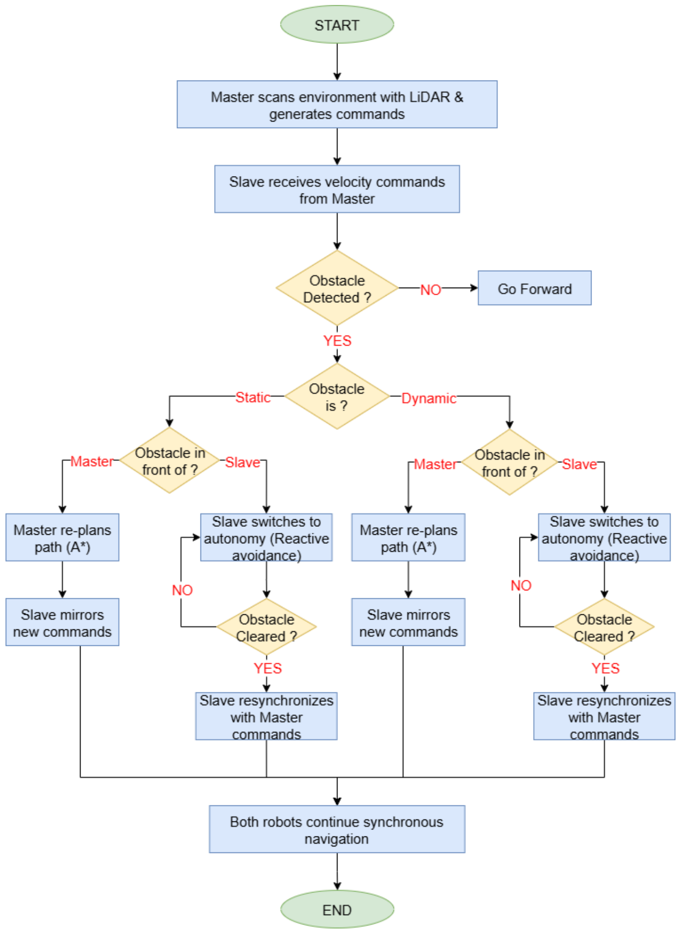
START
Master scans environment with LiDAR & generates commands
Slave receives velocity commands from Master
Obstacle Detected ?
NO
Go Forward
YES
Obstacle is ?
Static
Dynamic
Obstacle in front of ?
Master
Slave
Master re-plans path (A*)
Slave mirrors new commands
Slave switches to autonomy (Reactive avoidance)
NO
Obstacle Cleared ?
YES
Slave resynchronizes with Master commands
Obstacle in front of ?
Master
Slave
Master re-plans path (A*)
Slave mirrors new commands
Slave switches to autonomy (Reactive avoidance)
NO
Obstacle Cleared ?
YES
Slave resynchronizes with Master commands
Both robots continue synchronous navigation
END

Fig. 1. Methodology Flowchart

Three test cases were developed in order to assess the robustness of the system: (i) a stationary obstacle that only interferes with the path of the master, (ii) a stationary obstacle that only interferes with the path of the slave and, (iii) a moving obstacle that intrudes on the paths of both robots at distinct moments. The given outcomes indicate that the master slave topology guarantees the effective and coordinated navigation within the static environments reducing the computational overhead on the slave. Nevertheless, the reactive avoidance demonstrated by the slave in the dynamic environment revealed weaknesses in managing the behaviors of the obstacles, and it is possible that it may be necessary to come up with better local planning techniques or optimization techniques driven by the AI.

3.1 Environment Overview

The entire experimental structure was developed in the MATLAB in which a 3D simulation environment was designed to provide realistic navigation challenges. The setting was characterized to test the master slave coordination system at a controlled setting with both statical and moving hindrances.

The workspace was designed as a bounded 10×10 m arena whose obstacle locations could be configured to resemble warehouse or cluttered search-and-rescue missions. The master robot was initialized at the origin and slave robot had a fixed 1 m lateral off-set with the master along the Y-axis.

Key components of the environment included:

- **Obstacle Modeling:** Obstacles were modelled as cuboids and spheres whose dimensions and locations were variable. The effects of both static as well as dynamic obstacles were incorporated within the environment whereby, the former was carried out by fixing the location of the obstacles and the latter by programming obstacles to move in a linear fashion such that it will cross the robot trajectories at different moments.
- **Robot Models:** The representation of the robots was simple cylindrical objects with differential-drive kinematics. The trajectories of each robot were drawn employing the 3D plotting routine made available by MATLAB, with colors used to indicate the deviation and resynchronization of the path.
- **Sensing and Detection:** Proximity-based obstacle sensing was used to simulate the presence of obstacles with 0.5 m being the definition of obstacle to both robots. This enabled the imposition of the time-limited autonomy mode of the slave in the case of local challenges.
- **Collision Handling:** Obstacle safety regions of 0.3 m on the ground and 0.3 m airspace were accounted in the supervision of obstacle realism. Such collisions could be identified whenever one of the robots went in this buffer.
- **Control and Communication:** The master robot computed its path using the A* algorithm, updating its linear (v) and angular (ω) velocity commands in real time. These commands were transmitted to the slave robot via a simulated publisher–subscriber mechanism in MATLAB. The slave mirrored these commands unless an obstacle appeared within its sensing range, at which point it executed localized avoidance before resynchronizing.

This MATLAB environment allowed testing across three distinct cases:

1) Static obstacle obstructing the master's path.
2) Static obstacle obstructing the slave's path.
3) Dynamic obstacle intersecting both trajectories.

The environment offered a flexible and reproducible environment to study the control, resynchronization and robustness of the master slave framework under different conditions.

3.2 Static Obstacle in Front of Master

The master robot in this case was subjected to a cylinder shaped static obstacle directly on its path. When it was detected, the master triggered the obstacle avoidance mode, which calculated an alternate path by using the A * algorithm of path planning around the obstruction. The master followed an easily planned diversion to the right hand side of the obstacle, then recovered on the original path of the target.

During the experiment, the slave robot did not deviate at all times in matching the master in regards to the velocity instructions. As the obstacle existed only in the path of the master, the slave did not elicit any avoidance logic of its own. Rather, it was able to track the master equivalent detour route and acquire the target place without bumping.

The obtained results shown by *Fig. 2.* prove the effectiveness of the master–slave control topology in static surroundings, where only the leader robot encounters obstacles. The ability of the master to re-plan guarantees the safety, and the slave saves on the computational resources since it only has to copy the movement of the leader. This case has justified the effectiveness of centralized decision making on situations where there was low environmental complexity.

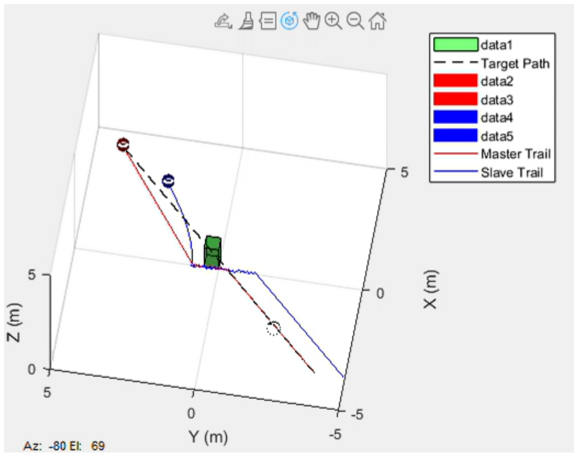

Fig. 2. Static Obstacle in front of Master

3.3 Static Obstacle in Front of Slave

In the current case, the master robot traveled tractably through the intended path without any challenges. But in front of the path of the slave a stationary object was put. Given the fact that the slave roughly reflects the velocity instructions of the master, there was a possibility of collision unless some corrections were applied.

To counter this, the slave engaged in its temporal context autonomy mode as soon as the impasse was within its radar. In this mode the slave performed a locally restricted avoidance maneuver to avoid the obstruction. After the route was cleared the slave easily synced back to the master command stream and continued to follow its trajectory.

The experiment emphasises the ability of the slave to make autonomy in context awareness. The slave exhibits minimal but efficient intelligence in obstacle avoidance as the master takes care that the navigation occurs firmly across the globe. As in *Fig. 3.*, these final results confirm that the hybrid control method is simple to implement, yet safe in that the slave can successfully move around in an unknown set of obstructions without disrupting formation.

3.4 Dynamic Obstacle Scenario

In this configuration, the movable obstacle moved through the environment in a linear trajectory, occasionally colliding with both the master and slave robot. This added a random

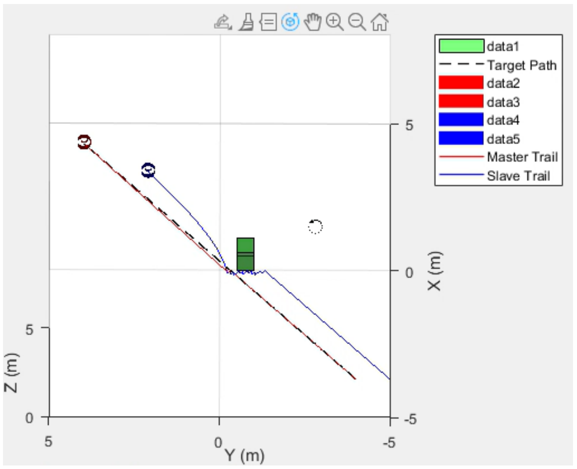

Fig. 3. Static Obstacle in front of Slave

factor as opposed to the previously described static obstacle scenarios given that the moving obstacle could block either robot at various future times.

The master robot made use of A* path planning procedure in cases where the obstacle crossed the path of the master robot and rerouting dynamically took place. The real-time updates of the velocity commands were sent to the slave robot. This slave, mainly mimicking the master and what it said, at times found the swerving obstacle right in its path before the master could change his move. When this would happen the slave would then use its local collision-checking logic and temporarily manoeuvre out of the way of the obstacle before realising its relative positioning with the master.

In the latter configuration, the movable obstacle moved through the environment in a linear trajectory, occasionally colliding with both the master and slave robot. This added a random factor as opposed to the previously described static obstacle scenarios given that the moving obstacle could block either robot at various future times.

The master robot made use of A* path planning procedure in cases where the obstacle crossed the path of the master robot and rerouting dynamically took place. The real-time updates of the velocity commands were sent to the slave robot. This slave, mainly mimicking the master and what he said, at times found the swerving obstacle right in its path before the master could change his move. When this would happen the slave would then use its local collision-checking logic and temporarily manoeuvre out of the way of the obstacle before realising its relative positioning with the master.

As it could be seen, the results showed that both the robots could take up the obstacle avoidance process at the same time and independently of each other (*Fig. 4.*). Nonetheless, in some cases, linear translation of the obstacle brought small instances of desynchronization, as level corrections of the slave temporarily diverged with the global trajectory of the master. Nonetheless, resynchronization mechanism re-established coordination when it was avoided in totality.

It depicts this, and also refers to the shortcomings of reactive avoidance per se in the slave, as illustrated in this scenario. Future extensions might add state-of-the-art local planning or intelligent decision-making model to be more robust.

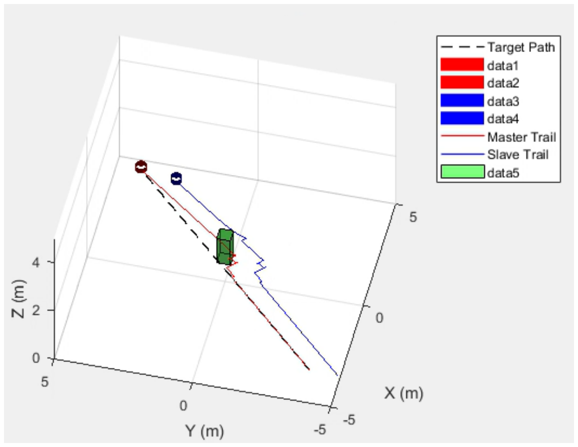

Fig. 4. Dynamic Obstacle Scenario

IV. Conclusion

The master–slave navigation experiments conducted in the MATLAB simulation environment demonstrate the effectiveness of the proposed coordination framework in obstacle-rich scenarios. The master robot, equipped with a simulated 3D LiDAR, performs global path planning using the A* algorithm and provides velocity commands for navigation. The slave robot follows these commands, thereby conserving computational effort and ensuring synchronous movement. When the slave encounters an obstacle not present in the master's trajectory, it temporarily switches to a localized obstacle avoidance routine and subsequently resynchronizes with the master. This hybrid approach highlights the balance between centralized decision-making and localized autonomy.

Unlike many existing systems, the framework does not rely on differentiating between static and dynamic obstacles, but instead emphasizes real-time sensing and reactive maneuvers. The results confirm that such a system can achieve reliable coordinated navigation in cluttered environments while maintaining low computational overhead.

For future work, the proposed framework will be extended and validated in real-world experiments using two TurtleBot4 robots in a ROS 2 environment. This will enable the evaluation of system scalability, robustness, and adaptability under practical conditions, providing a pathway toward deployment in larger-scale multi-robot applications such as warehouse automation and collaborative exploration.

References

[1] P. Nadella, T. Shreyas, and P. Singla, "Integration of deep learning with LiDAR in real-time robot task execution," *Frontiers in Robotics and AI*, vol. 12, 2025, Art. no. 1515802. doi: 10.3389/frobt.2025.1515802.

[2] D. M. Matos, P. Costa, H. Sobreira, A. Valente, and J. Lima, "Efficient multi-robot path planning in real environments: A centralized coordination system," *Int. J. Intelligent Robotics and Applications*, vol. 9, pp. 217–244, 2025. doi: 10.1007/s41315-024-00378-3.

[3] X. Guo, B. Zhang, H. Wang, and Z. Wu, "Adaptive obstacle avoidance via AD-CBF-MPC for mobile robots in dynamic environments," *Robotica*, vol. 42, no. 3, pp. 1223–1243, 2024. doi: 10.1017/S026357472300XXX.

[4] K. Cao, J. Liu, and Y. Chen, "Distributed dynamic window approach for obstacle avoidance in mobile robot formations," *IEEE Access*, vol. 12, pp. 141236–141249, 2024. doi: 10.1109/ACCESS.2024.XXXXXXX.

[5] Z. Di, N. Du, J. Li, J. Wu, and Y. Ding, "An artificial potential field method enhanced with a threshold stack mechanism for robotic navigation," *Machines*, vol. 13, no. 12, 2025, Art. no. 1060. doi: 10.3390/machines13121060.

[6] W. Zhu, X. Gao, H. Wu, J. Chen, X. Zhou, and Z. Zhou, "Design of multimodal obstacle avoidance algorithm based on deep reinforcement learning," *Electronics*, vol. 14, no. 1, 2025, Art. no. 78. doi: 10.3390/electronics14010078.

[7] A. M. Elsayed, A. El-Agamy, and A. El-Azab, "Decentralized fault-tolerant control of multi-mobile robot systems with LiDAR sensor malfunctions," *PLOS ONE*, vol. 19, no. 6, 2024. doi: 10.1371/journal.pone.XXXXX.

[8] H. Wang, C. He, L. Li, and W. Chen, "Research on path planning of mobile robots based on improved A* algorithm," *PLOS ONE*, vol. 20, no. 1, 2025, e0297335. doi: 10.1371/journal.pone.0297335.

[9] A. Jamshidpey, M. Wahby, M. Allwright, W. Zhu, M. Dorigo, and M. K. Heinrich, "Centralization vs. decentralization in multi-robot sweep coverage with ground robots and UAVs," *arXiv:2408.06553*, 2024.

[10] S. Oh-hara, T. Tsubouchi, and J. Miura, "Leader-follower formation control using fisheye cameras for real robots," *ROBOMECH Journal*, vol. 10, 2023, Art. no. 9. doi: 10.1186/s40648-023-00273-1.

[11] A. Sánchez-Sánchez, A. J. Fernández-León, and J. V. Gómez, "Power-based formation control of multi-robot systems: A novel leader–follower approach," *J. Intelligent & Robotic Systems*, vol. 108, no. 2, 2023. doi: 10.1007/s10846-022-01723-8.

[12] A. K. Mackay, L. Riazuelo, and L. Montano, "RL-DOVS: Reinforcement learning for autonomous robot navigation in dynamic environments," *Sensors*, vol. 22, no. 10, 2022, Art. no. 3847. doi: 10.3390/s22103847.

[13] M. A. Rahman and P. R. Sarker, "Real-time SLAM techniques for mobile robots: A review," *International Journal of Robotics*, vol. 12, no. 1, pp. 1–24, 2023. doi: 10.1155/2023/XXXXXX.

[14] E. Jesús-Bernal, D. Alejo, I. Maza, and A. Ollero, "A dynamic window approach extension for obstacle avoidance in 3D environments," *Applied Sciences*, vol. 11, no. 13, 2021, Art. no. 6123. doi: 10.3390/app11136123.

[15] Z. Wei, T. Li, and L. Sun, "ROS-based navigation and obstacle avoidance: A study and benchmarking," *Sensors*, vol. 25, no. 14, 2025, Art. no. 4306. doi: 10.3390/s25144306.

[16] L. Liu, Z. Tang, K. Zhang, and Y. Huang, "Path planning techniques for mobile robots: Review and future research directions," *Knowledge-Based Systems*, vol. 273, 2023, 110513. doi: 10.1016/j.knosys.2023.110513.

[17] X. Wang, M. Yu, and J. Zhang, "RRT* with informed heuristics for efficient motion planning in cluttered spaces," *IEEE/ASME Trans. Mechatronics*, vol. 29, no. 1, pp. 112–123, 2024. doi: 10.1109/TMECH.2023.XXXXXXX.

[18] M. Heiden, J. Panerati, and A. Prorok, "Multi-robot path planning in dynamic environments: A survey," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, pp. 1–28, 2024. doi: 10.1146/annurev-control-072423-XXXXXX.

[19] D. Sunil, S. Sunil, and A. P. Vinod, "Feature-based occupancy map-merging for collaborative SLAM: A review," *Sensors*, vol. 23, no. 6, 2023, Art. no. 3055. doi: 10.3390/s23063055.

[20] W. A. H. Sandanika, A. G. Iswara, and R. Ardiansyah, "ROS-based multi-robot system for efficient indoor exploration and map merging," *Journal of Robotics and Control*, vol. 5, no. 4, pp. 665–676, 2024. doi: 10.18196/jrc.v5i4.22494.

[21] S. J. He, Y. C. Chen, and P. Li, "Safety-aware model predictive control for autonomous mobile robots in dynamic crowds," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1481–1488, 2024. doi: 10.1109/LRA.2023.XXXXXXX.

[22] J. Park, H. Kim, and S. Kim, "Hybrid global–local planning with A* and DWA for real-time navigation on ROS," *IEEE Access*, vol. 11, pp. 132145–132160, 2023. doi: 10.1109/ACCESS.2023.XXXXXXX.

[23] X. Chen, Q. Shi, and R. Fan, "LiDAR-inertial odometry and mapping: A survey and future directions," *Robotics and Autonomous Systems*, vol. 169, 2023, 104418. doi: 10.1016/j.robot.2023.104418.

[24] M. A. Martinez-Baselga, D. Martin, and L. Montano, "Deep RL oriented for real-world dynamic scenarios in robot navigation," in *Proc. IEEE/RSJ IROS*, 2022, pp. 1–8. doi: 10.1109/IROSXXXXX.2022.XXXXXXX.