Development of a Memory-Based Anti-Oscillation Strategy to Improve Potential Field Path Planning for Robot Swarms

Septedy Indrajannah[1] and Anugerah Wibisana[1]

[1]Jurusan Teknik Elektro, Prodi Teknik Rekayasa Elektronika, Politeknik Negeri Batam, Batam, Indonesia

[*]Email: septedy44@gmail.com

Received on dd-mm-yyyy | Revised on dd-mm-yyyy | Accepted on dd-mm-yyyy

***Abstract*— Potential-field navigation in leader–follower robot swarms often suffers from oscillation and local trapping near obstacles, reducing goal-reaching reliability. This paper proposes PF+GBM, a Potential Field (PF) planner augmented with a goal-direction bias and a short-term memory of recently visited cells to discourage back-and-forth motion during gradient-descent stepping. Unlike baseline PF, the proposed method adds this memory-based anti-oscillation rule while keeping computation lightweight. Experiments in a Webots simulation integrated with Robot Operating System 2 (ROS2) were conducted in corridor, single-obstacle, and random-obstacle scenarios (five runs each) with a fixed start and varied targets. PF+GBM reached the goal in 13/15 runs versus 6/15 for baseline PF, and reduced mean path length in the corridor and random-obstacle cases (about 6.6% and 16.9%), while maintaining comparable travel time in successful runs.**

Keywords: anti-oscillation, leader-follower, path planning, potential field, robot swarms.

I. INTRODUCTION

Robot navigation in obstacle-filled environments is a fundamental capability in mobile robotics, particularly in leader–follower schemes where the leader must reach a target while maintaining safe and stable motion. Variations in obstacle configurations and narrow free spaces may trigger unstable behaviors, such as oscillatory back-and-forth motion or premature stopping before the goal is reached, which are commonly reported in potential-field-based navigation [1], [2].

A widely used method due to its simplicity and low computational cost is the Potential Field (PF) approach, which combines attractive influence toward the goal and repulsive influence away from obstacles, guiding the robot by descending the resulting potential landscape [2], [3]. In 2D planning, PF is often realized by constructing a potential map and performing local minimum-neighbor selection (gradient-descent-like stepping) to generate a path [4]. Despite its efficiency, PF remains prone to local minima and oscillations that may trap the robot, terminate planning prematurely, and yield only a partial path [1], [5]. Prior studies have proposed improvements using historical or memory information, as well as direction-aware weighting, to suppress oscillation and enhance trajectory stability [6]–[8].

However, a lightweight modification that explicitly suppresses back-and-forth motion during local step selection while preserving the simplicity of grid-based PF planning is still desirable for practical leader–follower navigation. Therefore, this study proposes PF+GBM, which augments PF with a goal-direction bias and a visited-cell memory mechanism during gradient-descent step selection. The proposed method is evaluated in a Webots–ROS2 simulation across three scenarios (corridor, single obstacle, and random obstacle) with five runs per scenario [9]–[11]. The initial position is fixed while the target is varied across runs. The leader robot is evaluated using the number of successful runs (n), travel time (T), and path length (L) computed over successful runs; experimental data are recorded and analyzed using ROS 2 tools and trajectory visualization [12], [13].

II. METHOD

The methodology is organized from system-level overview to component-level details. A system block diagram is presented first to summarize the data flow from simulation and sensing to planning and control; subsequent subsections specify the grid-map construction, the potential-field formulations, and the controller implementation employed in the experiments.

A. System Diagram

This study evaluates Potential Field (PF)–based path planning in a leader–follower robot system within a Webots–Robot Operating System 2 (ROS 2) simulation. The system workflow is shown in Figure 1. The Webots arena provides robot poses and obstacle information. The leader pose and target are used as inputs to the planner, while obstacles are converted into a 2D grid map. Based on the pose+goal and the obstacle grid, the Potential Field Path Planner produces a navigation path, which is then tracked by the Motion Controller to generate velocity commands so that the robot moves within the arena. The components in Figure 1 are described in the following

subsections.

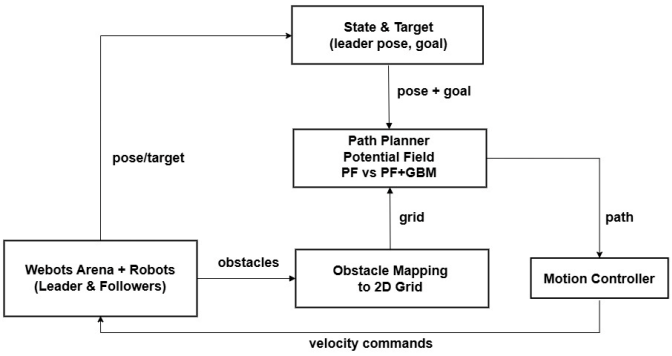

Figure 1. System diagram of the proposed navigation pipeline (PF vs PF+GBM).

B. Simulation Environment and Obstacle Mapping to a 2D Grid

Experiments were conducted in a Webots arena containing a leader robot and several follower robots, with obstacle configurations defined according to the test scenarios. For grid-based planning, an arena of 2.0 m × 1.5 m was discretized into a 40×30 2D grid. Obstacle information from the arena was mapped into an occupancy grid, where 0 denotes free cells and 1 denotes obstacle cells. World coordinates (x, y) were mapped linearly to grid indices gxgy, and clamping was applied to ensure indices remained within grid bounds. Planning used 8-connected neighborhood connectivity; diagonal moves that could potentially "cut corners" around obstacles were rejected using a corner-cutting rule.

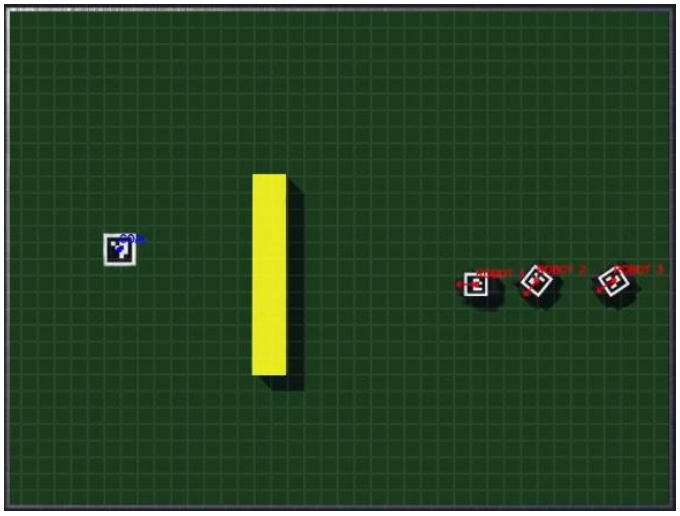

Figure 2. Example of the Webots simulation arena used in the experiments.

C. 2.3 Potential Field Path Planner (PF and PF+GBM)

The artificial potential field (PF/APF) method is widely used for navigation because it is conceptually simple and supports real-time computation, where the robot is guided by the combination of an attractive influence toward the goal and a repulsive influence away from obstacles [3]. In this study, PF is implemented in a grid-based manner by building a global potential map and generating a path through minimum-neighbor selection (gradient descent), following commonly used practical implementations [4]. The total potential at a grid cell (x, y) is defined as the sum of the attractive and repulsive potentials,

$$U(x,y) = U_{att}(x,y) + U_{rep}(x,y). \quad (1)$$

In (1), $U(x,y)$is the total potential, $U_{att}(x,y)$is the attractive potential, and $U_{rep}(x,y)$is the repulsive potential. The attractive potential pulling the robot toward the goal $g = (g_x, g_y)$is written as (2),

$$U_{att}(x,y) = \frac{1}{2} k_p \, d_g(x,y), \quad (2)$$

where $k_p$is the attractive gain and $d_g(x,y)$is the Euclidean distance to the goal, defined as in (3),

$$d_g(x,y) = \sqrt{(x-g_x)^2 + (y-g_y)^2}. \quad (3)$$

The repulsive potential is computed with respect to the nearest obstacle and is activated only when the obstacle distance is within the influence radius r_r(robot radius plus a safety margin),

$$U_{rep}(x,y) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\max(d_o(x,y),\epsilon)} - \frac{1}{r_r}\right)^2, \\ 0, \\ d_o(x,y) \le r_r \\ d_o(x,y) > r_r \end{cases} \quad (4)$$

In (4), $\eta$is the repulsive gain, $d_o(x,y)$is the Euclidean distance to the nearest obstacle, and $\epsilon$is a small constant to avoid division by zero. A path is obtained by computing the potential map over the grid and iteratively selecting the free neighbor with the lowest Uat each step [4]. Although efficient, PF can be trapped in local minima and can produce oscillatory or zig-zag trajectories near obstacles or narrow passages [1], [5]. To improve PF robustness while keeping the computation lightweight, this study proposes PF+GBM, which modifies the step-selection rule during potential-map descent. Hereafter, PF+GBM denotes *Potential Field with Goal-direction Bias and Memory*, i.e., a PF variant that augments the baseline potential-map descent by (i) a goal-direction bias that penalizes neighbor moves deviating from the goal direction and (ii) a short-term visited-cell memory that rejects candidates revisiting recently visited cells to suppress oscillatory back-and-forth motion. For reproducibility, the planner parameters used in the experiments were: attractive gain $k_p = 3.0$, *repulsive gain* $\eta = 30.0$, obstacle influence radius $r_r = 0.14$ *m (robot radius* 0.12 *m plus safety margin* $0.02\,m$), bias gain $k_{bias} = 0.25$, *maximum iterations* $I_{max} = 900$, and goal tolerance $\tau = 1$ grid cells; the memory length was set to $L = 3$. At iteration t, let $c_t$be the current grid cell, $g$the goal cell, and $\mathcal{N}(c_t)$the set of free 8-connected neighbor cells (with corner-cutting prevention). For each candidate neighbor $n \in \mathcal{N}(c_t)$, PF+GBM computes a goal-direction alignment using cosine similarity,

$$align(n) = \frac{(g - c_t) \cdot (n - c_t)}{(\| g - c_t \| + \epsilon_a)(\| n - c_t \| + \epsilon_a)}, \quad (5)$$

where $(\cdot)$ denotes the dot product, $\|\cdot\|$ denotes the Euclidean norm, and $\epsilon_a$is a small constant to avoid division by zero. The candidate is then ranked using a biased score,

$$S(n) = U(n) + k_{bias}(1 - align(n)), \quad (6)$$

Intuitively, the term $(1 - \text{align}(n))$penalizes moves that deviate from the goal direction, encouraging more progressive motion toward the target. In PF+GBM, candidate neighbors $n \in \mathcal{N}(c_t)$are ranked by the biased score in (6), which combines the potential value $U(n)$with a goal-direction preference through $\text{align}(n)$in (5). To suppress short cyclic motions, PF+GBM maintains a short-term visited-cell queue $M_t$of length $L(L = 3$ in this work) and selects the best-ranked candidate that does not revisit any cell within this window; if no such candidate exists, planning terminates and returns a partial path. This step-selection procedure (including corner-cutting prevention and the memory check) is summarized in Algorithm 1.

Algorithm 1. PF+GBM Path Planning (Goal-Direction Bias + Short-Term Memory)

Require: Occupancy grid G (0 free, 1 obstacle); start cell s=(sy,sx); goal cell g=(gy,gx); obstacle points ObsXY (world coordinates); GridToWorld(·); gains kp, η; influence radius rr; bias gain kbias; maximum iterations Imax; goal tolerance τ (cells); memory length L.
Ensure: Path P (sequence of grid cells), or empty/partial path.

```
if s is not free OR g is out of bounds OR g is not free then
    return empty
end if
Build potential map U over the grid using (1)–(4); set U=∞ for obstacle cells
c ← s
P ← [c]
M ← empty queue (capacity L)
for iter = 1 to Imax do
    if dist(c,g) ≤ τ then
        if c ≠ g then append g to P end if
        return P
    end if
    N ← valid free neighbors of c (8-connected, with corner-cutting prevention)
    if N is empty then
        return empty
    end if
    vg ← g – c
    for each n ∈ N do
        vs ← n – c
        align(n) ← (vg · vs) / (||vg|| ||vs||)
        S(n) ← U(n) + kbias (1 − align(n))
    end for
    Sort N by increasing S(n) to form Cands
    chosen ← NONE
    for each n ∈ Cands do
        Mtmp ← copy(M); push n into Mtmp
        if |Mtmp| > L then pop the oldest element end if
        if Mtmp contains no duplicate cell then
            chosen ← n
            M ← Mtmp
            break
        end if
    end for
    if chosen = NONE then
        return P
    end if
    c ← chosen
    append c to P
end for
return empty
```

Algorithm 1 implements PF+GBM as an iterative grid-based descent on a precomputed potential map U. At each iteration from the current cell c, the planner enumerates valid 8-connected neighbors (free cells; diagonal moves are disallowed if they would cut obstacle corners) and assigns a step score $S(n) = U(n) + k_{\text{bias}}(1 - \text{align}(n))$. Here, $\text{align}(n)$denotes the cosine alignment between the step direction $(n-c)$and the goal direction $(g-c)$, yielding values in $[-1,1]$; therefore, steps pointing more directly toward the goal (higher alignment) receive a smaller penalty. Candidate neighbors are then evaluated in ascending $S(n)$while enforcing a short-term memory $M$of the last Lvisited cells: a move is rejected if it revisits any cell still stored in the memory window, which suppresses back-and-forth oscillations. After a move is accepted, the selected cell is pushed into $M$and the oldest entry is dropped when $| M |> L$. The planner terminates when the goal tolerance is satisfied; if all candidates are rejected by the memory constraint, it returns the partial path computed so far, whereas invalid inputs or the absence of feasible neighbors results in an empty output (and the iteration limit $I_{\max}$prevents unbounded looping).

The motion controller tracks the planned path produced by the planner and generates velocity commands for the leader robot. The navigation stack is implemented in Robot Operating System 2 (ROS 2), which provides a modular publish/subscribe middleware for data exchange between software components. In this work, the Motion Controller runs as a ROS 2 node that publishes linear and angular velocity commands (v, ω)as geometry_msgs/Twist messages, while subscribing to the current robot pose $(x, y, \theta)$and the waypoint sequence generated by the planner. The path is represented as an ordered set of waypoints in world coordinates obtained by converting grid cells using the GridToWorld(·) mapping described in

Section 2.2. At each control cycle, the controller selects the active waypoint and advances the waypoint index when the Euclidean distance to the waypoint is below the waypoint tolerance $\tau_w = 0.05$ m. The controller is executed at a fixed period $\Delta t = 0.05$ s. The commanded velocities are saturated to $0 \le v \le v_{max}$and $-\omega_{max} \le \omega \le \omega_{max}$, with $v_{max} = 0.20$ m/sand $\omega_{max} = 1.50$ rad/s, to ensure stable motion in narrow passages. A heading-based proportional controller drives the robot toward the active waypoint: the desired heading is computed from the robot position to the waypoint, the heading error is wrapped to $(-\pi, \pi]$, and the angular velocity is set proportional to this error and limited to $\pm\omega_{max}$. The linear velocity is set proportional to the waypoint distance and limited to v_{max}, and it is reduced when the heading error is large to avoid overshoot and oscillatory turning. The resulting (v, ω)pair is published as a Twist command (cmd_vel). The controller outputs zero velocity after the final waypoint (goal) is reached. All controller parameters (τ_w, v_{max}, ω_{max}, and control gains) are kept constant across all runs and both planners to ensure a fair comparison.

D. Brief Testing Procedure and Evaluation Metrics

Testing was performed in three scenarios (corridor, single obstacle, and random obstacle), with five runs per scenario. The robot's initial position was fixed, while the target position was varied across runs. Pose data and velocity commands were recorded for metric computation on the leader robot (leader/robot1). Travel time and path length are evaluated using the leader robot trajectory. Travel time $T$is computed from the difference between the arrival time and the start time, Travel time $T$is computed from the difference between the arrival time and the start time, as in (7),

$$T = t_{end} - t_{start}, \tag{7}$$

where t_{start} is the first timestamp when the leader begins moving (i.e., $| v | > 0.02$ m/s), and t_{end} is the timestamp when the leader reaches the goal and becomes stable (i.e., distance to the goal ≤ 0.08 m and $| v | \le 0.02$ m/s). Path length L is computed as the sum of Euclidean distances between consecutive leader positions, Path length $L$is computed as the sum of Euclidean distances between consecutive leader positions, as in (8),

$$L = \sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \tag{8}$$

where (x_i, y_i) is the leader position at sample $i$and N is the number of recorded samples along the trajectory.

III. RESULT AND DISCUSSION

The proposed PF+GBM method was evaluated against the baseline Potential Field (PF) planner in a leader–follower Webots simulation integrated with ROS 2. Experiments were conducted in three scenarios: corridor, single obstacle, and random obstacles, with five runs per scenario. The initial pose was fixed, while the goal position was varied across runs to reduce bias toward a single target configuration. Both methods used the same simulation settings, grid-map resolution, and controller parameters to ensure a fair comparison. Representative trajectory patterns are visualized in Figures 3–5, and a quantitative summary is provided in Table 1.

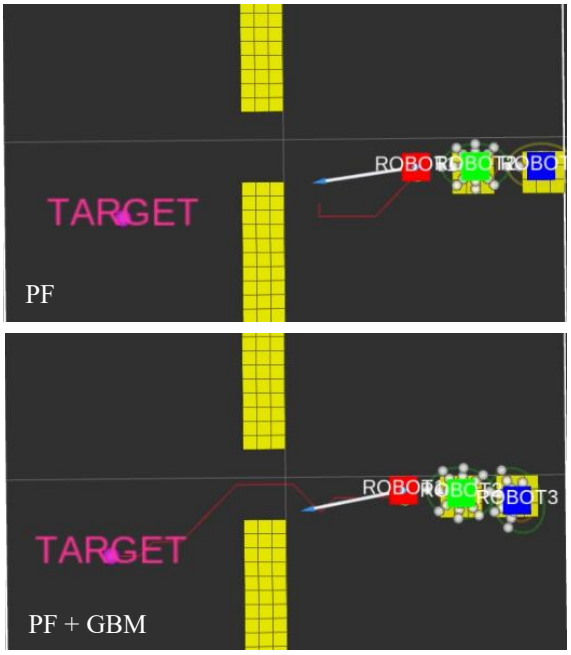

Figure 3. RViz2 trajectory in the corridor scenario, where PF achieves 1/5 success and PF+GBM achieves 3/5 success.

In the corridor scenario (Figure 3), free space is constrained, which tends to amplify oscillatory behavior and premature termination during potential-map descent. The visualization indicates that PF+GBM more often maintains forward progress toward the goal, whereas the baseline PF may stop earlier when oscillation or trapping occurs. This qualitative behavior is consistent with the higher completion robustness of PF+GBM observed in this scenario.

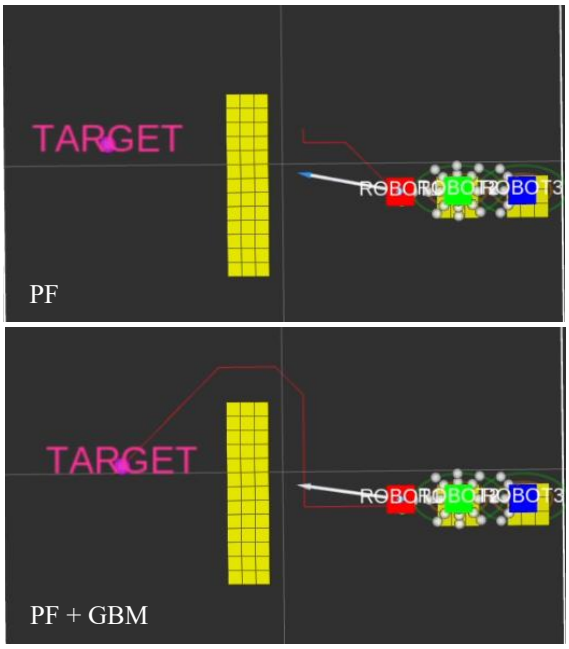

Figure 4. RViz2 trajectory in a single obstacle scenario, where PF achieves 2/5 success and PF+GBM achieves 5/5 success.

In the single-obstacle scenario (Figure 4), PF+GBM trajectories remain stable while navigating around the obstacle and tend to continue progressing toward the goal. In contrast, the baseline PF may terminate earlier in some runs, which is consistent with oscillation or local trapping near obstacle boundaries. Overall, the trajectories support the intended role of the visited-cell memory in suppressing back-and-forth motion during step selection.

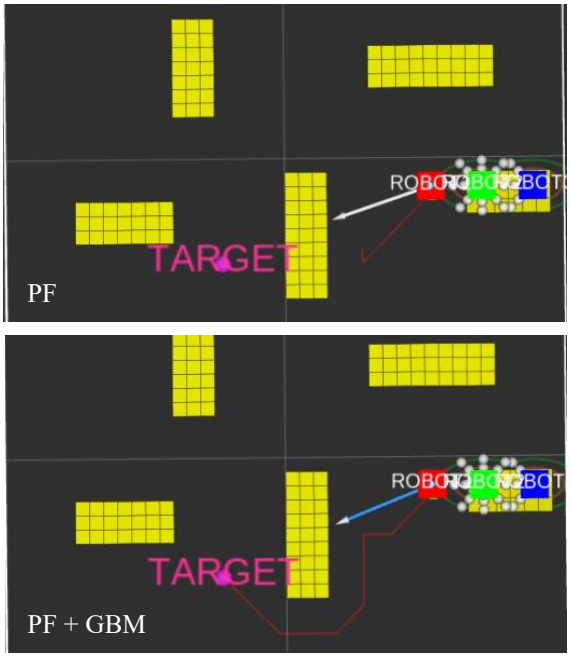

Figure 5. RViz2 trajectory in random obstacle scenarios, where PF achieves 3/5 success and PF+GBM achieves 5/5 success.

In the random-obstacle scenario (Figure 5), PF+GBM trajectories appear more progressive across varied obstacle configurations, whereas PF shows more frequent stagnation or early termination in unsuccessful trials. These qualitative patterns suggest that combining a goal-direction bias with visited-cell memory helps sustain motion toward the goal in environments that can induce oscillation and local minima effects.

Table 1. Summary of travel time T, path length L, and successful runs nacross scenarios; mean Tand mean Lare computed over successful runs only.

Scenario	**Method**	**n (success/5)**	**Mean T (s)**	**Mean L (m)**
Corridor	PF	1	44.40	1.130
Corridor	PF+GBM	3	53.62	1.055
Single obstacle	PF	2	56.52	1.228
Single obstacle	PF+GBM	5	59.87	1.404
Random obstacle	PF	3	68.08	1.435
Random obstacle	PF+GBM	5	67.81	1.193

Table 1 summarizes the quantitative results. PF+GBM consistently yields a larger number of successful runs (higher n) than PF, indicating improved robustness in reaching the target. In the corridor scenario, PF achieves n = 1 while PF+GBM achieves n = 3; therefore, PF's mean travel time in the corridor should be interpreted with caution because it is based on a single successful run. For successful runs, the mean travel time of PF+GBM is comparable to PF, particularly in the single-obstacle (59.87 s vs 56.52 s) and random-obstacle scenarios (67.81 s vs 68.08 s). This indicates that the primary improvement of PF+GBM is completion reliability rather than consistently faster traversal in successful trials. Regarding path length, PF+GBM produces shorter mean L in the corridor (1.055 m vs 1.130 m) and random-obstacle scenarios (1.193 m vs 1.435 m), while it yields a longer mean path in the single-obstacle scenario (1.404 m vs 1.228 m). The longer path in the single-obstacle case may occur when the memory-based anti-oscillation mechanism selects an alternative route to avoid repeated back-and-forth steps. Because goal positions were varied across runs, both T and L are influenced by target geometry; however, within each scenario, PF and PF+GBM were evaluated using the same set of goal indices across runs, so the observed differences primarily reflect planner behavior. For the random-obstacle scenario, a stricter head-to-head comparison additionally requires using identical obstacle layouts (fixed seeds) across both methods for each run.

IV. CONCLUSION

This study demonstrated that augmenting a Potential Field (PF) planner with goal-direction bias and short-term visited-cell memory (PF+GBM) improves navigation robustness in a leader–follower Webots–ROS 2 simulation. Across corridor, single-obstacle, and random-obstacle scenarios (five runs each), PF+GBM achieved 13/15 successful goal-reaching runs compared with 6/15 for baseline PF, indicating higher resistance to oscillation and local trapping during step selection. For successful trials, travel time remained comparable to PF, while path length was generally shorter in the corridor and random-obstacle cases; the single-obstacle case showed a longer path, suggesting a trade-off where anti-oscillation behavior may favor safer detours. This work is limited by the use of simulation-only evaluation, a small number of runs per scenario, and the lack of strictly matched random-obstacle layouts across methods in all trials. Future work will include testing on real robots, increasing the number of trials with fixed obstacle seeds for head-to-head comparison, and extending the

evaluation to multi-robot interactions and adaptive parameter tuning to balance success rate, travel time, and path efficiency.

REFERENCES

[1] Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," in Proc. IEEE International Conference on Robotics and Automation (ICRA), Sacramento, CA, USA, Apr. 1991, pp. 1398–1404.

[2] S. S. Ge and Y. J. Cui, "Dynamic Motion Planning for Mobile Robots Using Potential Field Method," Autonomous Robots, vol. 13, no. 3, pp. 207–222, 2002, doi: 10.1023/A:1020564024509.

[3] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," The International Journal of Robotics Research, vol. 5, no. 1, pp. 90–98, 1986.

[4] A. Sakai, "PythonRobotics," GitHub repository. [Online]. Available: https://github.com/AtsushiSakai/PythonRobotics.

[5] G. B. Bell and M. Livesey, "The Existence of Local Minima in Local-Minimum-Free Potential Surfaces," in Proc. Towards Autonomous Robotic Systems (TAROS 2005), London, UK, 2005.

[6] X. Sun, D. Shen, F. Yu, and X. Niu, "Enhanced Artificial Potential Field for Planning the Safe Return of Wartime Aircraft," Aerospace, vol. 12, no. 12, art. no. 1043, 2025, doi: 10.3390/aerospace12121043.

[7] Q. Han, X. Ma, H. Liu, Y. Xu, Y. Xie, Q. Yang, and F. Meng, "Improved Artificial Potential Field Method for UAV Path Planning," IEEE Access, vol. 13, pp. 177537–177547, 2025, doi: 10.1109/ACCESS.2025.3620220.

[8] J. Liu, Y. Yan, Y. Yang, and J. Li, "An Improved Artificial Potential Field UAV Path Planning Algorithm Guided by RRT Under Environment-Aware Modeling: Theory and Simulation," IEEE Access, vol. 12, pp. 12080–12097, 2024, doi: 10.1109/ACCESS.2024.3351974.

[9] Open Robotics, "ROS 2 Documentation." Available: https://docs.ros.org.

[10] Cyberbotics Ltd., "Webots: Open Source Robot Simulator.". Available: https://cyberbotics.com.

[11] O. Michel, "Webots: Professional Mobile Robot Simulation," International Journal of Advanced Robotic Systems, vol. 1, no. 1, p. 5, 2004.

[12] Open Robotics, "rosbag2_py: Python API for rosbag2," ROS 2 Documentation (Humble). [Online]. Available: https://docs.ros.org/en/humble/p/rosbag2_py/

[13] Open Robotics, "RViz2," GitHub repository. Available: https://github.com/ros2/rviz.