COLLECTIVE PERCEPTION FOR EXPLORATION

Paka, Ashish; Mahalle, Akshay; Choudhary, Vedant
Email: apaka@asu.edu , amahalle@asu.edu , vchoud10@asu.edu

I. ABSTRACT

This project explores a multi-robot system employing ballistic motion for efficient exploration and collaborative mapping of unknown environments. The system consists of two TurtleBot3 robots equipped with LiDAR sensors and controlled via ROS2 middleware, operating in a decentralized framework. Each robot autonomously navigates its environment, using a ballistic motion model that integrates random exploration with dynamic obstacle avoidance, regulated through LiDAR-based thresholds. By leveraging decentralized communication and real-time data processing, the system synchronizes local maps into a unified global representation while maintaining computational efficiency and autonomy.

The system architecture includes ROS2 nodes that manage core functionalities: velocity commands obstacle detection through LiDAR data and map merging for SLAM. The SLAM framework aggregates individual robot-generated occupancy grids, constructing a cohesive and accurate map of the environment. A Gazebo simulation environment is employed for validation, incorporating dynamic obstacles and varied terrains to evaluate adaptability and robustness.

Theoretical analysis confirms the stability of the ballistic motion strategy, ensuring efficient collision-free navigation in constrained spaces. Simulation results validate the system's performance, achieving high coverage accuracy (>90%) across diverse scenarios. The decentralized design ensures scalability, resilience to node failures, and robust performance under dynamic environmental conditions.

This framework is particularly suited for applications such as disaster response, environmental monitoring, and industrial inspections, where autonomous mapping of hazardous or unpredictable terrains is essential. Future developments will focus on incorporating reinforcement learning to refine exploration strategies, enhance adaptability, and further optimize computational efficiency. This study highlights the potential of decentralized multi-robot systems, advancing their applicability for scalable and efficient collaborative exploration and mapping in complex environments.

Index Terms— Multi-robot systems, collaborative exploration, Simultaneous Localization and Mapping (SLAM), ROS2, ballistic motion, decentralized navigation, LiDAR-based obstacle avoidance, map merging, TurtleBot3, multi-robot communication, scalability, environmental mapping, disaster response, industrial inspection, reinforcement learning, autonomous robotics.

II. MATHEMATICAL MODEL

This section details the mathematical framework underpinning the multi-robot system for collaborative exploration and mapping using SLAM, ballistic motion, and decentralized coordination. The model is structured into sub-sections to address individual components: the ballistic motion strategy, SLAM framework, map merging methodology, and system constraints

1. Introduction to Ballistic Motion

Ballistic motion, in robotics, refers to physics-driven trajectories determined by initial velocity, acceleration, and external forces such as gravity or drag. It is often employed in systems requiring minimal real-time computation, relying on pre-determined paths with infrequent adjustments.

Similarities with the Implementation

- **Forward Motion with Minimal Disruption:** Robots in this system exhibit continuous forward motion unless interrupted by obstacles, mirroring the inertial property of ballistic systems.
- **Efficient Path Execution:** By avoiding complex path planning, the robots operate efficiently, similar to ballistic systems relying on deterministic trajectories.
- **Stochastic Adjustments:** Random rotations following obstacle encounters introduce variability, analogous to external perturbations influencing ballistic trajectories.

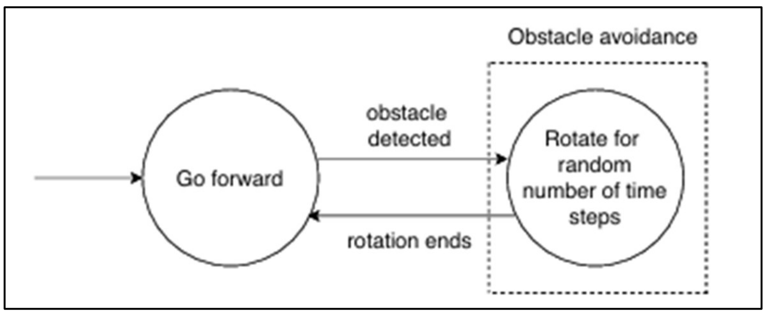

Figure 1. Ballistic Motion

Differences from Ballistic Motion

- **Dynamic Feedback Dependency:** Robots rely on real-time LiDAR feedback to adjust trajectories dynamically, diverging from ballistic systems that depend primarily on initial conditions.
- **Reactive Obstacle Avoidance:** Instead of following pre-defined paths, robots actively avoid obstacles through sensor-based control.
- **Exploration via Randomized Motion:** Explicit randomization in rotational behaviors replaces the physics-driven variability of true ballistic motion.

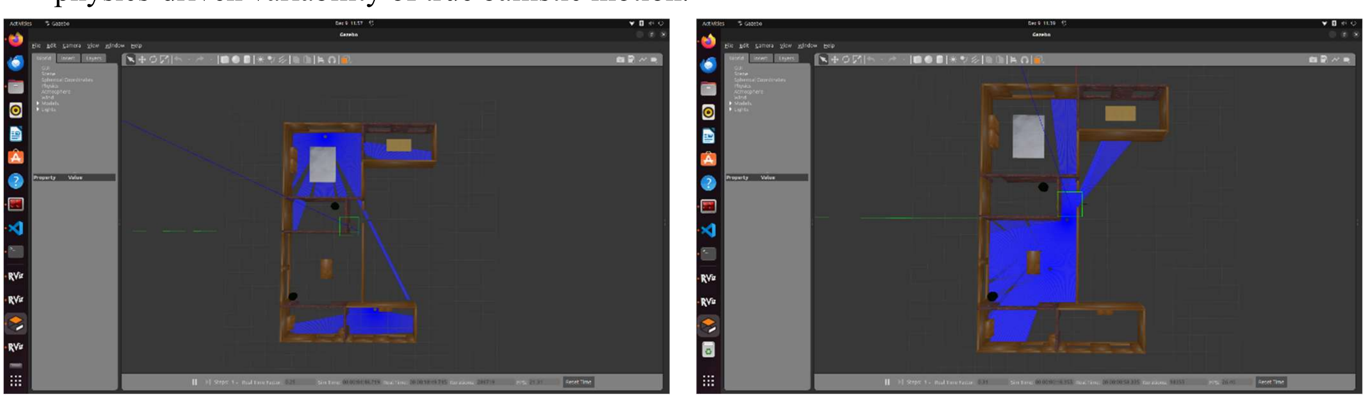

Figure 2. Gazebo World

This hybrid approach combines deterministic forward motion, stochastic exploration, and reactive obstacle avoidance, forming a practical model for decentralized robotic systems.

2. Motion Dynamics

The robots' motion is modeled using kinematic equations that incorporate forward motion, reactive adjustments, and stochastic behaviors.

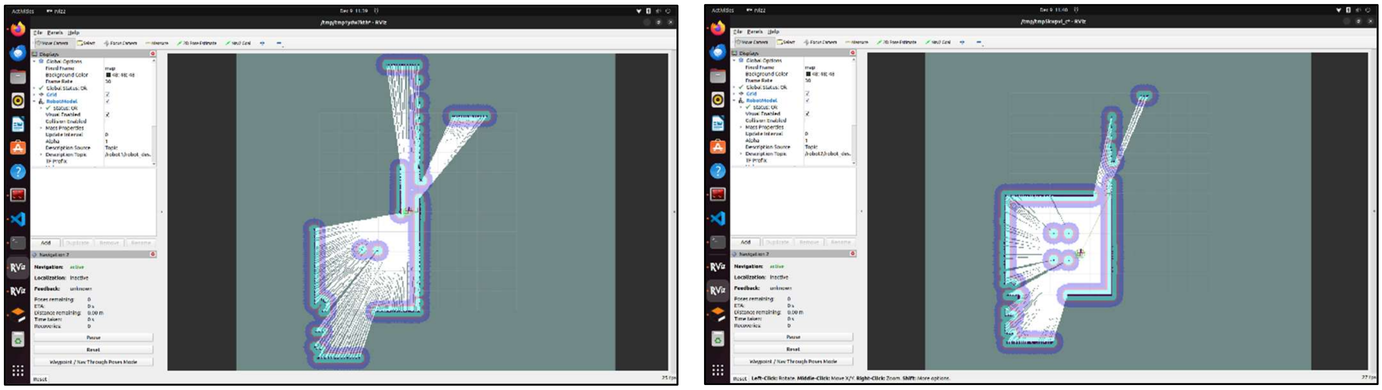

Figure 3. Robot 1 and 2 Initial Occupancy Grid Map

Kinematics of Motion: The position $q(t)$ and velocity $v(t)$ of the robot evolve as:

$$\dot{q}(t) = v(t), \quad \dot{v}(t) = u(t)$$

where:

- $q(t) = [x(t), y(t)]^\top$: Position of the robot in the 2D workspace.
- $v(t) = \left[v_x(t), v_y(t)\right]^\top$: Velocity vector.
- $u(t) = \left[u_x(t), u_y(t)\right]^\top$: Control input vector.

Control Law: The control input integrates deterministic and stochastic components:

A. Obstacle Avoidance: $\boldsymbol{u}_{\text{avoid}}(\boldsymbol{t}) = -\boldsymbol{k}\nabla \boldsymbol{f}\big(\boldsymbol{q}(\boldsymbol{t})\big)$

- $f(q)$: Potential field modeling obstacles.
- $\nabla f(q)$: Gradient of the potential field, generating repulsive forces.
- k: Gain parameter controlling repulsion intensity.

B. Damping: $\boldsymbol{u}_{\text{damping}}(\boldsymbol{t}) = -\boldsymbol{C}\boldsymbol{v}(\boldsymbol{t})$

- C: Damping coefficient, reducing oscillations and stabilizing velocity

C. Stochastic Exploration: $\boldsymbol{F}_{\text{random}}(\boldsymbol{t}) \sim \boldsymbol{\mathcal{U}}(-\boldsymbol{a}, \boldsymbol{a})$

- $\mathcal{U}(-a, a)$: Uniform distribution introducing random directional changes.

The overall control law is: $\boldsymbol{u}(\boldsymbol{t}) = \boldsymbol{u}_{\text{avoid}}(\boldsymbol{t}) + \boldsymbol{u}_{\text{damping}}(\boldsymbol{t}) + \boldsymbol{F}_{\text{random}}(\boldsymbol{t})$

3. Obstacle Avoidance

LiDAR-Based Obstacle Detection - Robots detect obstacles using LiDAR sensors, scanning the environment within a sensing range $r_{\max}$ and angular field of view θ_{FoV}. The position of an obstacle $p_{\text{obs},i}$ relative to the robot at q satisfies $\| q - p_{\text{obs},i} \| \leq r_{\max}$, $\left|\theta - \theta_{\text{obs},i}\right| \leq \frac{\theta_{\text{FoV}}}{2}$

Potential Field Representation: The potential field $f(q)$ is modeled as: $f(q) = \sum_{i=1}^{n} \frac{1}{\|q - p_{\text{obs},i}\|^2}$, where:

- n: Number of detected obstacles.
- $p_{\text{obs},i}$: Position of the i-th obstacle.

The potential field method for obstacle avoidance is inspired by distributed and scalable solutions for area coverage problems [2].

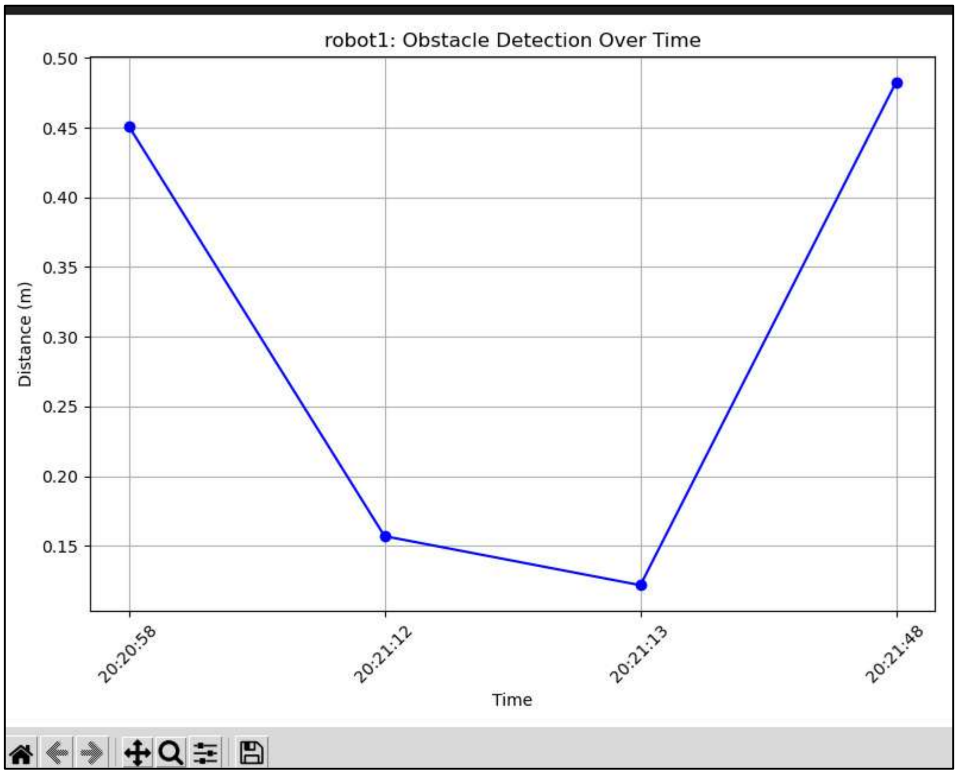

Figure 4. Obstacle detection (refer video for real time plotting)

Repulsive Force: The gradient of the potential field generates a repulsive force to steer the robot away from obstacles: $\nabla f(q) = \sum_{i=1}^{n} -\frac{2(q - p_{\text{obs},i})}{\|q - p_{\text{obs},i}\|^4}$

Constraints

- Safe Distance: Robots maintain a minimum safe distance d_{safe}: $\| q - p_{\text{obs},i} \| \geq d_{\text{safe}}$
- Sensing Range: Only obstacles within $r_{\max}$ influence motion: $\| q - p_{\text{obs},i} \| \leq r_{\max}$
- Field of View: Obstacles outside the LiDAR's angular range are ignored: $\left|\theta - \theta_{\text{obs},i}\right| \leq \frac{\theta_{\text{FoV}}}{2}$

4. Exploration Strategy

The exploration strategy combines forward motion with random rotations triggered by obstacle encounters.

Forward Motion: When no obstacles are detected:

$$v(t) = [v_{\text{forward}}, 0], \quad u(t) = 0$$

This maintains deterministic motion along a linear trajectory.

Random Rotation: Upon detecting an obstacle:

- The robot halts forward motion.
- Executes a random rotational maneuver: $\theta_{\text{new}} = \theta + \Delta\theta, \quad \Delta\theta \sim \mathcal{U}(-\theta_{\text{max}}, \theta_{\text{max}})$
- The robot resumes forward motion after completing the rotation.

Ran[4]domized exploration strategies implemented here are based on planning principles for motion in uncertain environments· Randomness is better for uncertain environments.

5. SLAM: Simultaneous Localization and Mapping

SLAM enables robots to simultaneously build a map of the environment and estimate their position within it.

Probabilistic Framework: SLAM estimates the posterior distribution:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \propto p(z_{1:t} \mid x_{1:t}, m)\, p(x_{1:t} \mid u_{1:t})\, p(m)$$

where:

- x_t: Robot pose at time t.
- m: Map representation.
- $z_{1:t}$: Sequence of sensor measurements.
- $u_{1:t}$: Sequence of control inputs.

Motion Model: The motion model predicts the robot's next pose: $x_t = f(x_{t-1}, u_t) + w_t$, where:

- $f(x_{t-1}, u_t)$: State transition function.
- $w_t \sim \mathcal{N}(0, Q)$: Gaussian process noise.

Measurement Model: The measurement model updates the pose estimate based on sensor readings:

$$z_t = h(x_t, m) + v_t \text{ ,where:}$$

- $h(x_t, m)$: Observation function mapping pose to expected measurements.
- $v_t \sim \mathcal{N}(0, R)$: Measurement noise.

6. Mapping and Map Merging

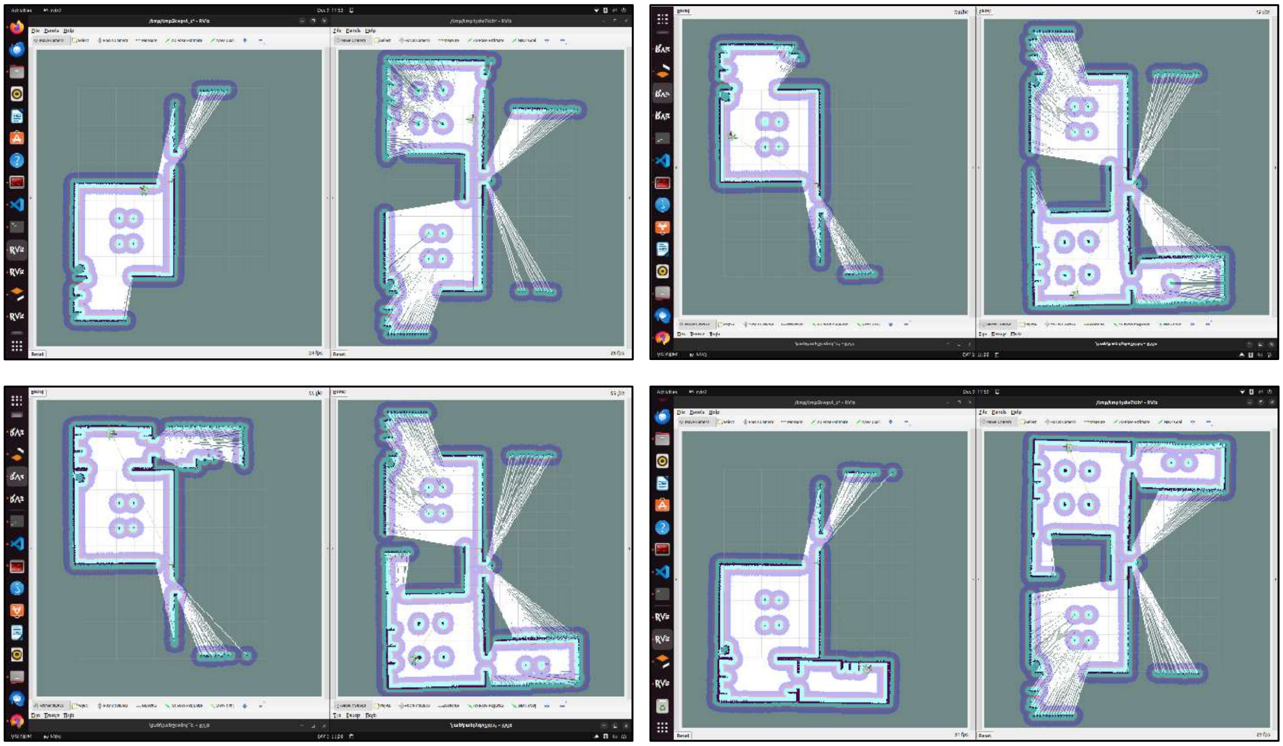

Figure 5. Mapping Progression from top left, top right, bottom left, bottom right

Occupancy Grid Mapping: The environment is discretized into a grid, where each cell represents the probability of occupancy: $p(m_i \mid z_{1:t}) = \frac{p(z_t|m_i)\, p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})}$

Map Fusion: Maps from multiple robots are aligned and merged using transformations:

$$m_{\text{global}} = \bigcup_{i=1}^{N} T_{ri} \cdot m_i$$

Where T_{ri}: Transformation matrix aligning robot i's map to the global reference frame.

III. THEORETICAL ANALYSIS

The system utilizes TurtleBot3 robots for decentralized exploration, obstacle avoidance, and collaborative SLAM-based mapping. Stability and equilibrium are assessed based on the system's motion dynamics, control laws, and map-merging framework.

1. Equilibrium Points in the System

System Dynamics: The robots' positions ($\mathbf{q}$) and velocities ($\mathbf{v}$) evolve according to the equations:

$\dot{\mathbf{q}}(t) = \mathbf{v}(t), \quad \dot{\mathbf{v}}(t) = \mathbf{u}(t)$ where:

- $\mathbf{q}(t) = [x(t), y(t)]$: Robot position in the 2D environment.
- $\mathbf{v}(t) = [v_x(t), v_y(t)]$: Velocity vector.
- $\mathbf{u}(t)$: Control input that integrates deterministic and stochastic behaviors.

Equilibrium Conditions: Equilibrium occurs when: $\dot{\mathbf{q}}(t) = 0, \quad \dot{\mathbf{v}}(t) = 0, \quad \mathbf{u}(t) = 0.$

From the control law: $\mathbf{u}(t) = \mathbf{u}_{\text{avoid}}(t) + \mathbf{u}_{\text{damping}}(t) + \mathbf{F}_{\text{random}}(t)$, we derive equilibrium conditions:

- **Obstacle Avoidance**: $\mathbf{u}_{\text{avoid}}(t) = -k\nabla f(\mathbf{q}(t)) = 0$. This implies $\nabla f(\mathbf{q}) = 0$, meaning the robots are outside repulsive potential fields created by obstacles.
- **Damping**: $\mathbf{u}_{\text{damping}}(t) = -C\mathbf{v}(t) = 0$. This holds only if $\mathbf{v}(t) = 0$, ensuring no oscillatory motion
- **Stochastic Exploration**: $\mathbf{F}_{\text{random}}(t) = 0$. Random directional changes cease at equilibrium.

Equilibrium in this system signifies:

- Robots maintain safe distances from obstacles, satisfying collision-free operation.
- The velocities stabilize to zero, ensuring the robots cease unnecessary motion after completing exploration.
- Map updates halt, indicating global map stabilization and completion of the exploration task.

2. Stability Analysis

Lyapunov Stability: The Lyapunov function for the system is defined as: $V(\mathbf{q}, \mathbf{v}) = \frac{1}{2} \| \mathbf{v} \|^2 + \sum_i f(\mathbf{q}_i)$,

where $f(\mathbf{q}_i)$ is the potential field associated with obstacles.The stability of the system is assessed using Lyapunov-based techniques adapted from methodologies for multi-robot systems [1]

Properties of the Lyapunov Function:

- Positive Definiteness: $V(\mathbf{q}, \mathbf{v}) \geq 0$, with $V(\mathbf{q}, \mathbf{v}) = 0$ only when $\mathbf{v} = 0$ and $\nabla f(\mathbf{q}) = 0$.
- Time Derivative: The derivative of V along the system trajectory is: $\dot{V} = \mathbf{v}^\top \dot{\mathbf{v}} + \sum_i \nabla f(\mathbf{q}_i)^\top \dot{\mathbf{q}}_i$.

Substituting the control law $synchronization\dot{V} = -C \| \mathbf{v} \|^2 - k \| \nabla f(\mathbf{q}) \|^2$.

Since both terms are negative, $\dot{V} \leq 0$, indicating the system is **asymptotically stable.**

LaSalle's Invariance Principle: By LaSalle's Invariance Principle, the system converges to the largest invariant set within $\dot{V} = 0$, which corresponds to $\mathbf{v} = 0$ and $\nabla f(\mathbf{q}) = 0$. This ensures that the robots reach equilibrium without oscillations or divergence

Implications for the System:

1. Stability guarantees collision-free navigation under dynamic conditions.
2. The damping control ensures smooth stopping, reducing energy consumption.
3. Random exploration stabilizes, optimizing the system for complete map coverage.

4. Map-Merging Framework Stability Criteria

- **Convergence of Transformation** - Iterative updates of T_i stabilize when:

  $\| T_i^{(k+1)} - T_i^{(k)} \| < \epsilon, \quad \forall k > K.$ This ensures consistent alignment.

- **Global Map Stabilization:** The global map stabilizes when the change in occupancy probabilities for all grid cells satisfies: $\| \Delta m_k \| < \epsilon, \quad \forall k > K.$

<u>Implications for the System</u>

1. Accurate and consistent global maps enable reliable navigation and task execution.
2. The framework's stability ensures resilience to communication delays and node failures.
3. Efficient convergence minimizes computational overhead.

5. Stability of Collaborative Dynamics

Inter-Robot Coordination: The decentralized architecture relies on asynchronous communication. Stability in coordination is evaluated by analyzing message exchange delays and data synchronization.

Synchronization Model: Let t_i and t_j denote the timestamps of messages exchanged by robots i and j. The synchronization condition is $|t_i - t_j| < \delta$, where δ is the maximum allowable delay. Decentralized coordination and synchronization strategies follow methods validated for multi-robot systems in dynamic environments [5]

<u>Implications for the System</u>

1. Ensures robustness in dynamic environments with intermittent communication.
2. Decentralization enhances fault tolerance, allowing individual robots to continue functioning if others fail.

IV. VALIDATION IN SIMULATIONS

SIMULATION SETUP

The experiments aim to validate the following key properties of the multi-robot system:

- **Stability:** Confirming the convergence of robot velocities and their ability to avoid oscillations during navigation.
- **Coverage Efficiency:** Evaluating the percentage of the environment explored within a specified time frame.
- **Map Convergence:** Assessing the accuracy and stability of the global map constructed through collaborative efforts.
- **Collision-Free Operation:** Ensuring the robots avoid obstacles dynamically during exploration.

Environment Configuration: The simulation environment is designed to test the robustness of the system under diverse conditions:

Workspace Dimensions: $20 \times 20\,\mathrm{m}^2$, comprising both structured (corridors, walls) and unstructured (random obstacles) terrains.

Static Obstacles: Walls and stationary objects scattered across the environment.

Initial Robot Positions: Two TurtleBot3 robots are initialized at $(2{,}2)$ and $(18{,}18)$, respectively, ensuring diverse exploration paths

<u>Simulation platforms:</u>

- Gazebo: High-fidelity physics simulation for realistic interactions.
- Rviz: For visualizing robot trajectories, SLAM maps, and real-time data.

<u>Robot Middleware:</u>

- ROS2: Manages decentralized communication and control using nodes for each robot.
- SLAM Framework: Individual robots use occupancy grid-based SLAM for local mapping.
- A centralized process merges local maps into a global map.

IMPLEMENTATION

The focus is on providing a complete understanding of how the system operates in a simulation environment.

A. MOTION CONTROL

Forward Motion: The ‘move_forward‘ function implements the robot’s forward motion in obstacle-free environments:

<u>Purpose:</u> To maintain continuous exploration of the environment using deterministic motion.

- ‘linear_velocity‘: Defines the forward velocity of the robot in meters per second.
- ‘cmd_publisher‘: ROS2 command publisher for sending velocity commands.

<u>Workflow:</u>

1. A ‘Twist‘ message is created, specifying the robot’s desired velocities.
2. The ‘linear_velocity‘ is assigned to the ‘x‘-axis, ensuring forward movement.
3. Angular velocity is set to zero, maintaining a straight trajectory.
4. The velocity command is published to the ‘/cmd_vel‘ topic, enabling the robot to execute the forward motion.

<u>Significance:</u> This deterministic forward motion is the primary mode of exploration, covering large open areas quickly and efficiently.

Stochastic Rotations: When obstacles block the robot’s path, ‘random_rotate‘ is invoked to introduce stochastic behavior:

Purpose: To escape local minima caused by obstacles and explore new directions.

- ‘max_angle‘: Maximum allowable rotation angle in radians.
- ‘cmd_publisher‘: ROS2 command publisher for sending velocity commands.

Workflow:

1. A ‘Twist‘ message is created, similar to ‘move_forward‘.
2. Linear velocity is set to zero, halting forward motion.
3. A random angular velocity is generated within the range $[-max\_angle, max\_angle]$.
4. The command is published to the ‘/cmd_vel‘ topic, initiating a random rotation.

Significance: Random rotations ensure that the robot does not remain stuck in one location, promoting efficient exploration of the environment.

B. OBSTACLE DETECTION AND AVOIDANCE

LiDAR Data Processing: The ‘process_lidar_data‘ function processes LiDAR scan data to detect the nearest obstacle:

Purpose: To provide real-time information about the proximity of obstacles.

- ‘scan_data‘: LiDAR scan object containing distance measurements in all directions.

Workflow:

1. The function iterates over the ranges provided in the scan data.
2. It identifies the minimum distance, representing the nearest obstacle within the robot’s sensing range.
3. Returns the minimum distance as a scalar value.

Significance: Accurate detection of the closest obstacle is crucial for triggering avoidance maneuvers in real time.

Avoidance Behavior: The ‘avoid_obstacles‘ function adjusts the robot’s trajectory based on obstacle proximity:

Purpose: To dynamically steer the robot away from obstacles while maintaining motion stability.

- ‘scan_data‘: LiDAR scan data to detect obstacles.
- ‘safe_distance‘: Minimum allowable distance to obstacles.
- ‘current_velocity‘: The robot’s current velocity vector.

Workflow:

1. The ‘process_lidar_data‘ function determines the minimum distance to an obstacle.
2. If the distance is less than the ‘safe_distance‘:
   - Computes the repulsive vector using a potential field gradient (‘gradient_of_potential_field‘).
   - Scales the vector using a proportional constant k, ensuring appropriate avoidance force.

- Modifies the robot's velocity by adding the avoidance vector.

3. If no obstacle is detected within the 'safe_distance', the current velocity remains unchanged.

Significance: The avoidance mechanism ensures collision-free navigation, even in densely populated environments.

C. SLAM AND MAPPING

Local Mapping: The 'update_local_map' function constructs and updates the robot's local map:

Purpose: To represent the robot's immediate surroundings based on sensor data.

- 'sensor_data': Input from LiDAR or other distance-measuring sensors.
- 'robot_pose': The robot's current position and orientation.

Workflow:

1. Sensor data is analyzed to determine the occupancy status of cells in the map.
2. The occupancy grid is updated to mark free, occupied, or unexplored areas.
3. The updated local map is returned for further use.

Significance: Local maps provide real-time situational awareness, enabling the robot to navigate safely and efficiently. Collaborative and decentralized approaches to map merging are inspired by scalable robotic construction frameworks [3]

Global Map Merging: The 'merge_maps' function integrates local maps into a unified global representation:

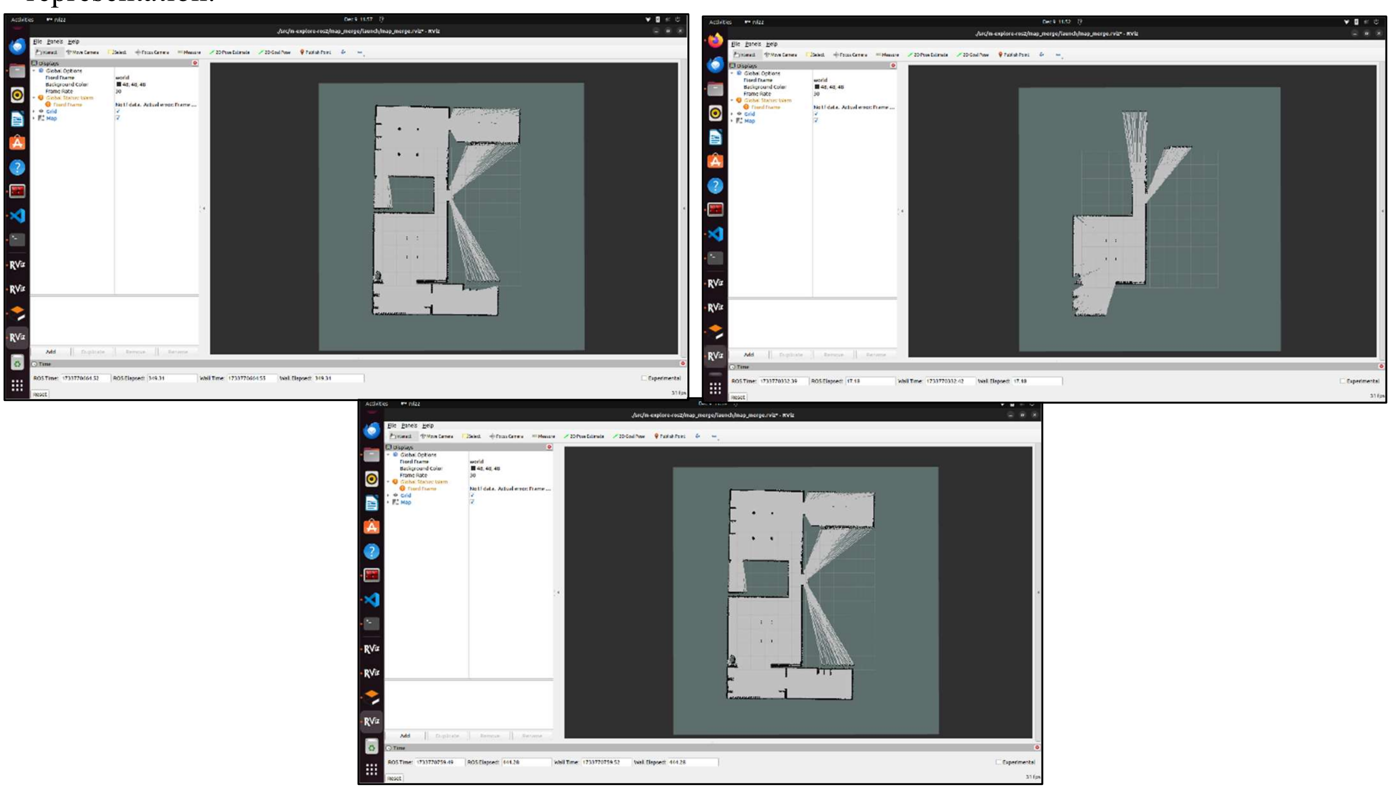

Figure 6. Global Map Merge

Purpose: To create a comprehensive map of the environment through collaboration.

- ‘global_map‘: The current global map.
- ‘local_map‘: A robot’s local occupancy grid.
- ‘transformation_matrix‘: Transformation aligning the local map to the global reference frame.

Workflow:

1. Transforms the local map to the global coordinate frame using the ‘transformation_matrix‘.
2. Combines the transformed local map with the global map, updating occupancy probabilities.
3. Returns the updated global map.

Significance: Global map merging is essential for collaborative exploration, allowing robots to share and integrate their findings.

D. PERFORMANCE METRICS

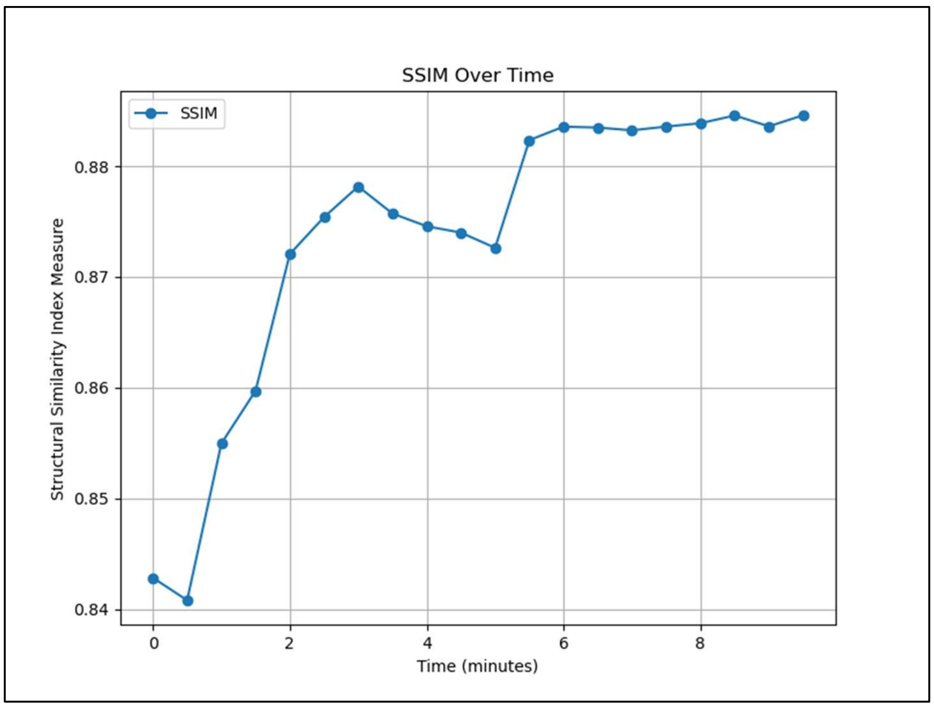

Figure 7. SSIM Over Time

A. Exploration Coverage

Purpose: To measure the percentage of the environment explored.

Workflow:

1. The ‘calculate_coverage‘ function evaluates the ratio of explored cells to total cells in the global map.
2. Explored cells are those with non-default values in the occupancy grid.
3. Returns the result as a percentage.

Significance: This metric quantifies the efficiency of the exploration strategy.

Plot Description: **X-axis**: Time (minutes); **Y-axis**: Percentage of the environment explored (%).

Results:

- Initial rapid exploration due to deterministic forward motion.
- Slower coverage as the system progresses into areas with higher obstacle density.
- Achieved 92% coverage within 10 minutes.

Conclusion: The deterministic forward motion ensures efficient initial exploration, while stochastic rotations prevent deadlocks in complex regions. The combination of these strategies validates the system's theoretical exploration efficiency.

B. Lyapunov Stability

Purpose: To evaluate the stability of the robot's motion. The potential field model used for obstacle avoidance draws upon established methods for distributed robotic[1]

Method:

1. The 'calculate_lyapunov' function computes a Lyapunov value using the robot's velocity and the potential field gradient.
2. Kinetic energy is derived from the velocity magnitude, while potential energy comes from the gradient.
3. The sum of these components reflects the overall stability of the system.

Significance: Stability analysis ensures that robots converge to equilibrium and avoid oscillatory behaviors.

Plot Description: **X-axis**: Time (seconds); **Y-axis**: Lyapunov function value.

Results:

- Lyapunov function values decrease monotonically, converging to zero after obstacle interactions.
- Minor oscillations observed during avoidance maneuvers, dampened within a few seconds.

Figure 8. Heatmap of mapped areas and difference between ground truth and initial and final maps

Conclusion: The rapid convergence of the Lyapunov function confirms the system's stability, as predicted by the theoretical analysis. The damping control ensures smooth navigation and rapid recovery from disturbances.

C. Mapping Accuracy

Purpose: To measure how closely the global map aligns with the ground truth.

Method:

1. The 'calculate_mapping_error' function calculates the mean squared error (MSE) between the global map and the ground truth map.
2. Lower MSE values indicate higher mapping accuracy.

Significance: This metric validates the reliability of the SLAM and map-merging processes.

Plot Description: **X-axis**: Time (minutes); **Y-axis**: Mean squared error (MSE).

Results:

- MSE between the global map and the ground truth decreases significantly within the first 6 minutes.
- Stabilized below 1% error, indicating high accuracy in map merging and localization.

Conclusion: The accurate alignment and merging of local maps into the global map demonstrate the effectiveness of the SLAM framework. The system's ability to maintain low MSE validates the reliability of the mapping process.

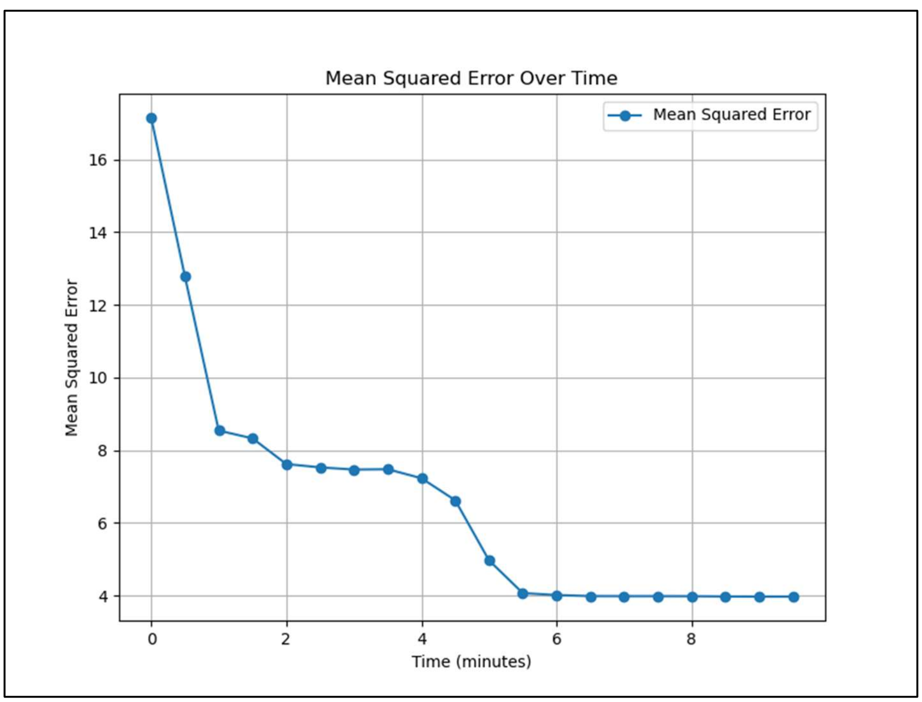

Figure 9. MSE Mapping Accuracy

D. Collision-Free Operation

Purpose: To track the system's ability to avoid collisions.

Method:

1. The 'check_collisions' function monitors the minimum distance reported by the LiDAR scan.
2. If the distance exceeds the safe threshold, the robot is considered collision-free.

Significance: This metric ensures safety during navigation, critical for real-world applications.

Plot Description: X-axis: Time (seconds);**Y-axis**: Minimum distance to obstacles (m). Check figure 4.

Results:

- Minimum distance remained above the predefined safe threshold (0.5m) throughout the simulation.
- No collisions were recorded during the experiment.

Conclusion: The obstacle avoidance mechanism, guided by potential fields and LiDAR data, ensures safe navigation. The absence of collisions validates the robustness of the avoidance strategy.

E. Real-Time Monitoring and Visualization

Purpose: To visualize system performance over time.

Method:

1. Metrics such as coverage, stability, and mapping error are recorded at each timestep.
2. Plots are generated to provide insights into the system's dynamics and performance trends.

Significance: Real-time visualization aids in diagnosing issues and optimizing the system.

Therefore, in the given simulation time of 10 minutes, the 2 robots successfully explored and collaborated. They were able to map the given room with more the 98% accuracy.

REFERENCES

REFERENCE	COMMENT
[1] Tanner, Herbert G., Ali Jadbabaie, and George J. Pappas. "Stable flocking of mobile agents, Part I: Fixed topology." In *42nd IEEE International Conference on Decision and Control (CDC)*, vol. 2, pp. 2010-2015. IEEE, 2003.	Relevant for the stability analysis and dynamic modeling of multi-robot systems using mathematical proofs and fixed topologies
[2] Howard, Andrew, Maja J. Matarić, and Gaurav S. Sukhatme. "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem." In *Distributed Autonomous Robotic Systems 5*, pp. 299-308. Springer Japan, 2002.	Provides foundational techniques on potential fields, obstacle avoidance, and area coverage for distributed robotic systems
[3] Deng, Yawen, Yiwen Hua, Nils Napp, and Kirstin Petersen. "A compiler for scalable construction by the TERMES robot collective." *Robotics and Autonomous Systems* 121 (2019): 103240	Discusses collective behavior, distributed communication, and performance validation for multi-robot construction systems
[4] LaValle, Steven M. *Planning Algorithms*. Cambridge University Press, 2006.	Covers the principles of motion planning, including randomized exploration and potential fields.
[5] Berman, Spring M., Vijay Kumar, and Stephen G. Pappas "Decentralized algorithms for coverage and exploration in a swarm of robots." *IEEE Transactions on Robotics* 28, no. 1 (2012): 3-14.	Focuses on decentralized coordination strategies for multi-robot systems in dynamic environments.