SC 627 - Motion Planning & Coordination of Autonomous Vehicles
Professor Arpita Sinha
TAs- Anik Mitra
Date: - 24-January-2026

Week 1: The "Digital Twin" (Environment & Mapping)

Goal: Establish the simulation environment and generate the map required for planning.

1. Problems to be Solved

- **The Blind Robot Problem:** The robot exists in the "Gauntlet" world but has no map to plan on.
- **The Configuration Space Problem:** The robot is not a point; it has a physical radius (r). How much must obstacles be "grown" so we can treat the robot as a point?
- **The Tooling Gap:** Setting up the ROS 2 workspace correctly with custom World files.

2. Methods to be Followed

- **SLAM (Simultaneous Localization and Mapping):** Launch the provided gauntlet.world. Use slam_toolbox or cartographer while teleoperating the robot to build an Occupancy Grid.
- **C-Space Calculation:** Manually measure the robot's radius. Apply the Minkowski Sum concept to "inflate" the obstacles in the generated map by radius r + safety margin delta.
- **Map Server:** Save the map and configure the nav2_map_server to publish it on the /map topic.

3. Deliverables

- Map Files: gauntlet.pgm and gauntlet.yaml (clean, loop-closed map).
- C-Space Diagram: A visual overlay showing original obstacles vs. inflated obstacles.
- Workspace Check: A screenshot of rviz2 showing the map and the robot localized correctly.

Week 2: The Algorithm (Dual Planner Implementation)

Goal: Implement the intelligence. Write the pathfinding logic from scratch.

1. Problems to be Solved

- **The Optimality Problem:** Finding the strictly shortest path (A*).
- **The Safety/Exploration Problem:** Implementing the safe planner (GVD) to contrast with A*.
- **The Grid-to-Graph Conversion:** Converting the 2D occupancy grid array into a graph structure suitable for search.

2. Methods to be Followed

- **A* Implementation:** Use a Priority Queue (Python heapq) to manage the Open List. Implement Euclidean Heuristic for the cost-to-go. (Constraint: Manual solver logic only).
- **Alternative Planner:** GVD (Skeletonization/Brushfire) or RRT (Random sampling).
- **Visualization:** Publish the resulting trajectory to the nav_msgs/Path topic.

3. Deliverables

- Simulation Video: Sequential video showing A* vs. Alternative path execution.
- Code Submission (v1): The Python node planner_server.py.

Week 3: The "Reality Check" (Hardware Transfer)

Goal: Deploy code to the physical TurtleBot and mitigate real-world noise.

1. Problems to be Solved

- **The Drift Problem:** Real motors slip. Odometry is not perfect like in Gazebo.
- **The Inflation Problem:** Real walls are imperfect. If inflation is too small, the robot clips corners.
- **The Network Problem:** Configuring ROS_DOMAIN_ID and connectivity.

2. Methods to be Followed

- **AMCL Integration:** Launch amcl to correct odometry drift using the LIDAR scan.
- **Parameter Tuning:** Adjust inflation_radius in costmap. Shrink until the robot fits through Zone B.
- **Controller Tuning:** Adjust Lookahead Distance in Pure Pursuit or PID to stop oscillation.

3. Deliverables

- Failure Log: Documenting at least 3 specific hardware failures.
- Hardware Checkpoint: Live lab demo of Zone C (Open Field) navigation.

Week 4: Final Validation & Reporting

Goal: Comparative analysis, defense, and final grading.

1. Problems to be Solved

- **The "Which is Better?" Problem:** Quantifying planner differences.
- **The Robustness Problem:** Ensuring the demo works live.
- **The Viva Defense:** Preparing to modify code on the fly.

2. Methods to be Followed

- **Data Collection:** Run both planners 5 times each. Measure Time to Goal, Path Length, and Minimum Distance to Obstacle.
- **Final Report:** Synthesize Sim-to-Real gap analysis.

3. Deliverables

- Final Report (PDF): C-Space theory, algorithm explanation, and results table.
- Live Demo: Full Gauntlet map navigation.
- Source Code: Final Git repository link.