# Technical Overview Document

*TurtleBot3 Autonomous Navigation System – SLAM, Path Planning, Control, and Sensor Integration*

1. Problem Definition

Indoor autonomous navigation requires a robot to move through complex, GPS-denied spaces such as hallways, labs, and office environments. The robot must perceive the environment, build a map, localize, plan a global path, track that path, and avoid obstacles that appear in real time. This requires a tightly coupled stack:

- Sensing: LiDAR, IMU, camera
- Mapping: SLAM to convert raw sensor data into a 2D occupancy grid
  Localization: continuous pose estimation
  Global Planning: A* on the occupancy grid to produce a collision-free path
- Local Control: Pure Pursuit to track the path
- **Reactive Avoidance:** sensor-based safety overrides

The end goal of our project is to enable a TurtleBot3 to autonomously navigate between arbitrary waypoints, continuously update its map, and avoid both static and dynamic obstacles.

2. SLAM Research Summary

SLAM (simultaneous localization and mapping) is a method used for autonomous vehicles that lets you build a map and localize your vehicle in that map at the same time. SLAM algorithms allow the vehicle to map out unknown environments. Engineers use the map information to carry out tasks such as path planning and obstacle avoidance.

Two types of technology components used to achieve SLAM. The first type is sensor signal processing, including the front-end processing, which is largely dependent on the sensors used. The second type is pose-graph optimization, including the back-end processing, which is sensor-agnostic.

2.1. Occupancy Grids

SLAM converts LiDAR + odometry into a 2D occupancy grid, a discrete grid where each cell holds the probability of being occupied. A few key characteristics of SLAM are that the grid resolution is around 0.05-0.10 m, used directly by A* planner and the following are the respective values: 0 (free), 100 (occupied), −1 (unknown).

The occupancy grid provides a structured, planner-friendly representation of the environment.

### 2.2. Loop Closure

Loop closure detects when the robot revisits a previously scanned location. This matters, because it can correct accumulated drift, perform global graph optimization and prevent map distortion.

### 2.3. Localization vs Mapping

SLAM involves two subproblems. Mapping builds the layout of the environment. Localization estimates the robot's pose. Real SLAM algorithms solve these simultaneously.
For Level 2, localization is essential because A* requires an accurate starting pose for path computation.

## 3. Path Planning (A) Research Summary

The A* algorithm is a powerful and widely used graph traversal and path finding algorithm. It finds the shortest path between a starting node and a goal node in a weighted graph. It is an informed search algorithm, meaning it leverages a heuristic function to guide its search towards the goal. This heuristic function estimates the cost of reaching the goal from a given node, allowing the algorithm to prioritize promising paths and avoid exploring unnecessary ones.

The A* algorithm combines the best aspects of Djikstra's and Greedy Best-First Search.

### 3.1 Grid Representation

The A* planner uses the SLAM occupancy grid:

- Free cells = valid nodes
- Occupied cells = forbidden
- Unknown cells = optionally treated as high-cost or blocked
- Neighbors: typically 8-connected (diagonals allowed)

Each grid cell is treated as a vertex in a graph.

### 3.2. Heuristics

A* uses a heuristic to guide search.

Common heuristics:

- Manhattan Distance $h = |x_1 - x_2| + |y_1 - y_2|$
- Euclidian: $h = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
- Diagonal / Octile distance for 8-direction grids

We use Euclidean because TurtleBot3 moves in continuous space.

### 3.3. Path Smoothing

Raw A* paths contain sharp 90° turns due to grid discretization. Smoothing improves control stability, path continuity, execution speed and safety through reducing jerking movements. Possible methods include Line-of-sight simplification, corner cutting, moving-average smoothing.

### 3.4. Line-of-Sight Optimization

For any three sequential waypoints *A*, B, C. If the straight line from A → C does not intersect an occupied cell, then B is removed. Benefits include shorter path, fewer turns, better Pure Pursuit tracking, and lower computational load.

## 4. Research Resources


- TurtleBot documentation: https://emanual.robotis.com/docs/en/platform/turtlebot3/slam/
- Slam https://www.mathworks.com/discovery/slam.html
- TurtleBot3 Github https://github.com/ROBOTIS-GIT/turtlebot3
- A* https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
- A* https://www.datacamp.com/tutorial/a-star-algorithm