SurveillanceBot: Autonomous Navigation and Mapping

Robotics Assignment Report

School of Computer Science & Applied Mathematics
University of the Witwatersrand

Banzile Nhlebela
2571291

Daniel Ngobe
2556833

Palesa Rapolaki
2550752

Mixo Khoza
2429356

June 1, 2025

Introduction

This report presents the implementation of a SurveillanceBot system for autonomous navigation and mapping using a TurtleBot robot in a simulated environment. The project encompasses three main components: environment mapping using gmapping, path planning using Rapidly-exploring Random Trees (RRT), and robot navigation using PID control.

System Architecture

Overview

The SurveillanceBot system consists of three integrated modules:

- **Mapping Module**: SLAM-based environment mapping using gmapping
- **Path Planning Module**: RRT algorithm for collision-free path generation
- **Navigation Module**: PID-controlled robot movement with state-machine behavior

Environment Setup

The robot operates in the provided environment. Gmapping was used to obtain the obstacle locations and then matplotlib was used to visualise the path planning algorithm in action on the map.

Mapping Implementation

SLAM Configuration

We utilized the gmapping package for Simultaneous Localization and Mapping (SLAM) with optimized parameters:

- **minimumScore**: Set to 1000 for improved mapping accuracy in complex environments
- **particles**: 30 particles provide sufficient localization accuracy while maintaining computational efficiency
- **Update rates**: Linear and angular updates of 0.2 ensure frequent map updates during movement
- **Sensor parameters**: maxRange and maxUrange optimized for the TurtleBot's sensor capabilities

Path Planning Algorithm

RRT Implementation

We implemented a Rapidly-exploring Random Tree (RRT) algorithm for path planning. The RRT algorithm incorporates:

- **Sampling Strategy**: 95% random sampling within map boundaries, 5% goal-biased sampling
- **Steering Function**: Limited step size of 5 units to ensure smooth path generation
- **Collision Detection**:
  - Liang-Barsky algorithm for line-rectangle intersection
  - Ray-casting algorithm for point-in-polygon tests
  - Line-polygon intersection for angled obstacles
- **Goal Threshold**: 5-unit radius for goal reaching tolerance

Robot Control System

PID Control Implementation

The navigation system employs separate PID controllers for linear and angular motion where:

- Linear motion: $K_P = 1.0$, $K_I = 0.0$, $K_D = 0.1$
- Angular motion: $K_P = 2.0$, $K_I = 0.0$, $K_D = 0.1$

System Integration

The complete navigation system follows this workflow:

1. Accept target coordinates as command-line arguments
2. Generate collision-free path using RRT algorithm
3. Execute sequential waypoint navigation using PID control
4. Implement turn-then-move behavior for each waypoint
5. Terminate upon reaching final destination within tolerance