People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdès

Institute of Electrical and Electronic Engineering
Department of Control and Power Engineering

Project Report Presented in Partial Fulfillment of

the Requirements of the Degree of

'MASTER'

In Control Engineering

"An indoor lidar based autonomous mobile robot using SLAM and RRT* algorithms"

Presented By:

ABBAD Aicha Manar

Supervisor:

Dr. Ouarda Hachour

Registration Number:……../2022

An indoor lidar based autonomous mobile robot using SLAM and RRT* algorithms

Aicha Manar ABBAD

June 2022

Abstract

In the past few years, we have seen a rise in autonomous robots that are able to navigate and operate in a variety of different environments. Autonomous navigation robots are now capable of moving around in a given space without human intervention. They can navigate independently, avoid obstacles and find their way back to a starting point. Research in the development of autonomous navigation robots is intended to enhance their autonomy in executing different tasks of varying complexity levels. One of these tasks is localization, mapping, and path planning. Robot localization is the process of defining the location of the mobile robot as it navigates through its environment. Mapping addresses the problem of collecting spatial models of the environment by the robot as it navigates. Path planning is the process of planning an obstacle-free trajectory for the mobile robot to navigate from a starting position toward a given goal destination. This project introduces a simulation of a mobile robot navigating in a closed indoor environment. The robot uses a lidar sensor to collect data about its environment in order to build a model map of it that will be used later on for path planning purposes. Both the SLAM (Simultaneous localization and mapping) and RRT* algorithms are used for map building and trajectory planning respectively.

Dedication

I would like to dedicate this work to my parents and my little brother, for their help, support, and assistance during my whole academic curriculum. May Allah help me to give them back even a small portion of what they gave me during all my life.

I would like to dedicate this work to my friends who have been there for me during all these years in my journey at IGEE.

Acknowledgement

I would like to express my out-most gratitude to Almighty Allah for providing me with knowledge, power, and for keeping me faithful to continue working on this project even when I was at my lowest.

I would like to appreciate Miss Hachour for her efforts, supervision, and for providing me with guidance and correction during the execution of this project.

I also would like to appreciate everyone who helped me on the fulfilment of this project, from teachers, friends, and colleges.

Nomenclature

$C - space$	The configuration Space
C_{free}	The free space
E	The tree Edge
q_i	Configuration Space
$q_{nearest}$	The nearest sample point
Q_{near}	The set of the neighboring nearest nodes
q_{new}	The new sample point
q_{rand}	The random sample point
T	The tree Graph
V	The tree vertex

Acronyms

AAV Autonomous Aerial Vehicle.

AGV Autonomous Ground Vehicle.

AUV Autonomous Underwater Vehicle.

CCD Charge-Coupled Device.

CMOS Complementary Metal-Oxide Semiconductor.

DOF Degrees of Freedom.

LIDAR Light Detection and Ranging.

RRT Rapidly Exploring Random Trees.

RRT* Improved Rapidly Exploring Random Trees.

SLAM Simultaneous Localization and Mapping.

UAV Unmanned Aerial Vehicle.

UUV Unmanned Underwater Vehicle.

UV Ultraviolet.

Contents

List of Figures

General Introduction

In the past few decades, robotics has witnessed a significant advancement. Autonomous machines are now able to navigate and operate independently in a variety of different environments without any human intervention or guidance, which makes them invaluable for emergency response situations as well as indoor environments.

There are numerous types of autonomous robots, which differ in terms of the tasks they can perform and the environment in which they can operate. For this project, we will focus on autonomous navigation mobile robots. Mobile robots are autonomous navigation robots that move around and operate in human-inhabited areas. They're often used in warehouses and distribution centers to move inventory around, or perform other types of logistics tasks.

Mobile robot navigation is considered an active research field. For a robot to navigate it needs to perform a set of tasks which are classified as follows: Localization, mapping, trajectory planning and path following, and motion planning.

This report discusses two main problems in mobile robotics which are SLAM (Simultaneous localization and mapping) and path planning. SLAM deals with the problem of simultaneously defining the location of the robot as it navigates and build a model map of its environment. Path planning deals with the problem of finding an obstacle-free trajectory for the mobile to navigate from its starting location to its goal destination within a short amount of time.

The suggested work aims to implement a simulation of an autonomous mobile robot navigating in a closed indoor environment. The robot uses a lidar sensor to gather environment data, the SLAM algorithm for map construction, and the RRT* algorithm for path planning.

This report is divided into three main chapters. The first chapter gives a comprehensive overview of autonomy and highlights basic information and definitions of autonomous navigation mobile robots. The second chapter dives deep into the theory and mathematics behind two major issues in autonomous navigation systems: SLAM (Simultaneous Localization and Mapping) and path planning. The third chapter illustrates the overall system design and demonstrates the simulation and obtained results of each section of it. The last part of this report concludes the project and proposes future work.

Chapter 1

Basic definitions

1.1 Introduction

The globe is currently approaching a tipping point in the mobile robots revolution. A wide range of robotics systems are being developed in order to solve daily life problems. In order to ensure the success of these challenges and achieve optimal execution of tasks, we need to add some intelligence to our autonomous system. However, inserting intelligence may cause a rise of many problems within our robot, from which: localization, mapping, and path planning.

This chapter provides a general overview and some basic knowledge about autonomous systems, as well as discuss briefly the problem of localization, mapping, and path planning in autonomous navigation robots.

1.2 Autonomous Robots

Autonomy in robotics systems is the ability to act without resources to human control. An autonomous robot can perceive its environment, make decisions, and then actuate a movement or a manipulation, using different sensors and actuators [1].

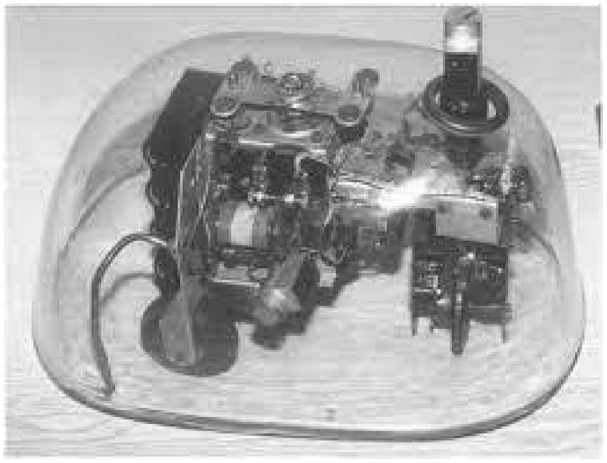


Figure 1.1: W.Grey Walter personal tortoise [2]

The first robots that were programmed to think the way biological brains do were known as Elmer and Elsie, and they were constructed in the 1940s by W.Grey Walter (see figure 1.1). They are often labeled as tortoises because of how they were shaped and the manner in which they moved [3].

1.3 Types of autonomous robots

According to their locomotion we have different types of autonomous robots:

1. **Stationary robots (Arm/Manipulators)**: These are industrial robots and manipulators. They are mostly used to provide solutions for tasks that are difficult for humans to perform, an excellent example of that is robotic arms.

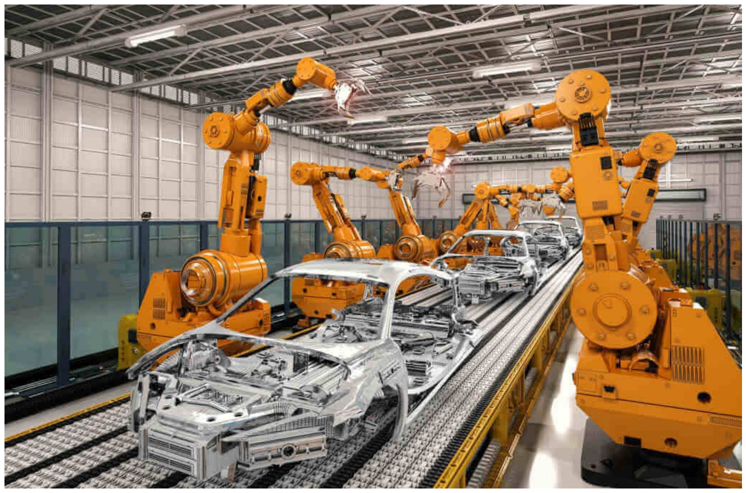


Figure 1.2: Industrial Robotic Arms [4]

2. **Autonomous Ground Vehicle (AGV)**: These are vehicles that operate while in contact with the ground. A good example is autonomous intelligent cars, which use aspects like pattern recognition and image processing to be able to understand the environment around and navigate easily.

Figure 1.3: Autonomous mobile robot [5]

3. **Autonomous Aerial Vehicle (AAV)**: These are aircraft that can make intelligent decisions without the assistance of a human pilot. They constitute a part of a large group of aerial systems called Unmanned Aerial Vehicle (UAV). Autonomous drones are a good example of AAVs.

Figure 1.4: Autonomous drone [5]

4. **Autonomous Underwater Vehicle (AUV)**: These are robots that travel underwater. They constitute a part of a large group of underwater systems named Unmanned Underwater Vehicle (UUV) [6]. A good example of AUVs is an underwater drone.

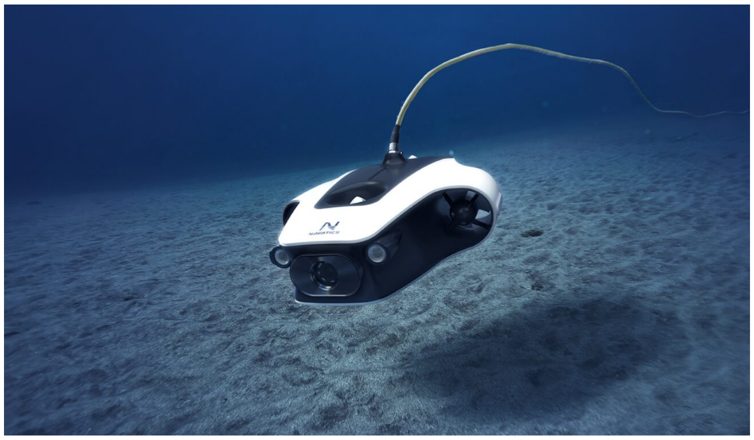


Figure 1.5: Navatics MITO underwater drone [7]

1.4 Autonomous Navigation System

An autonomous navigation system is a system that is able to understand and determine its location within an environment and plan its path and movements without any human intervention [8]. The vehicle determines its location using different sensors such as lidar, video cameras, and radar. As presented in figure 1.6, there are different levels of autonomy ranging from a vehicle that is controlled by a human from a distance with some simple algorithms onboard, to a fully autonomous vehicle with no human control.

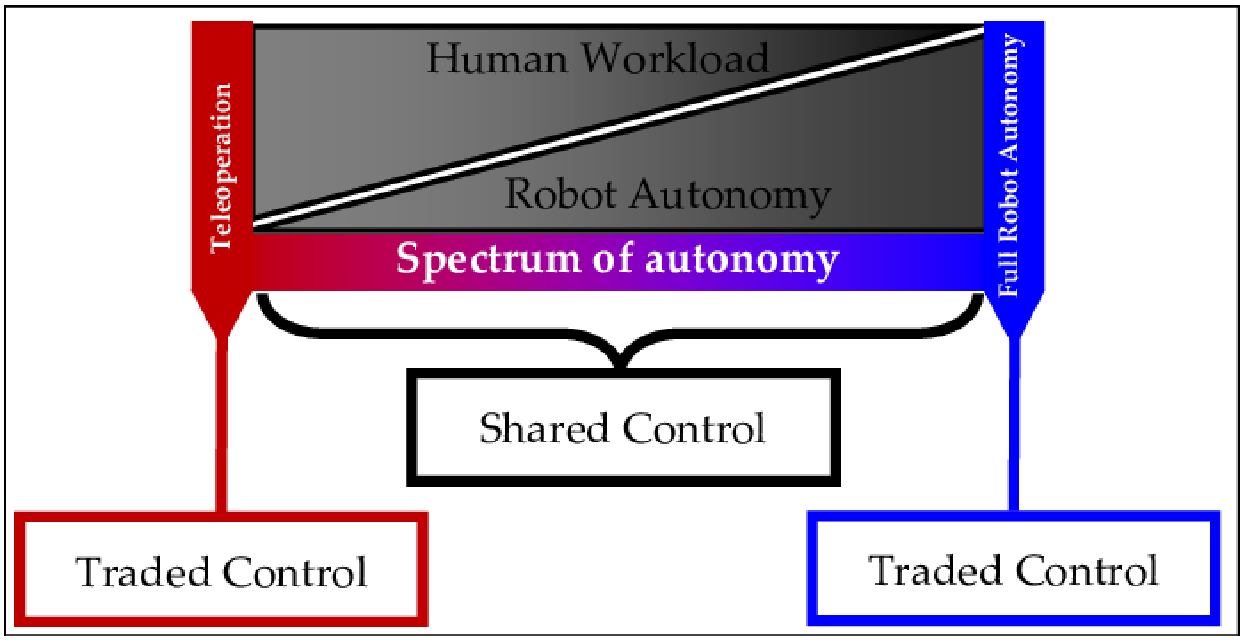


Figure 1.6: The spectrum of autonomy [9]

For the case of fully autonomous navigation systems, we can say that it is divided into two different approaches: Heuristic and optimal approaches.

A "heuristic approach", is a method where it is not necessary to have full data of the environment and the robot may not follow an optimal path. An "Optimal approach", is a method in which the vehicle requires more information about the environment and follows the best optimal path. The systems that employ this strategy construct/update a model of the environment in which they operate, and then determine the shortest path to their destination [10].

1.5 Capabilities of Autonomous Systems

For the autonomous systems to navigate, they need to perform a set of steps. The robot has to collect data about its environment using sensors, try to understand and define obstacles, build a model map, and determine the state, localization, and orientation of the vehicle itself within that map (see figure 1.7). This information will be used later on to plan a path to its goal [10].

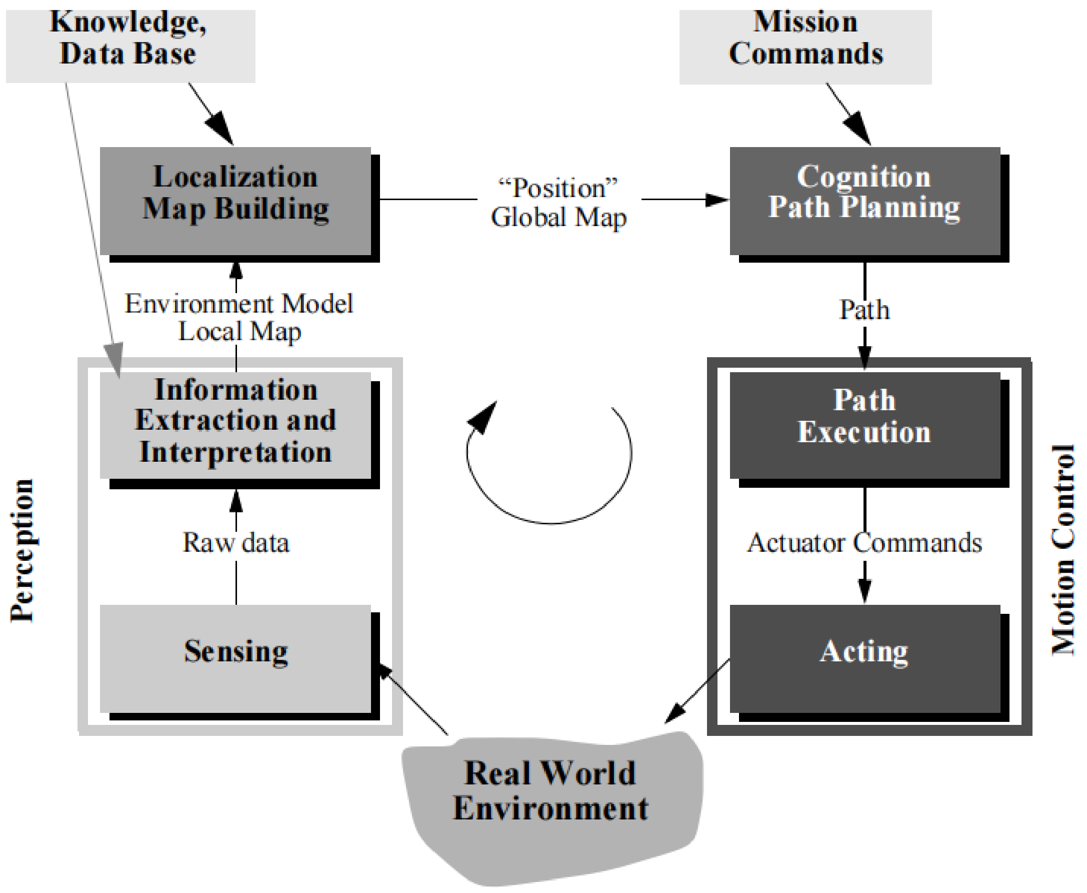


Figure 1.7: Reference control scheme for mobile robot [11]

1.5.1 Sense

The robot collects data of the environment around using different sensors. There is a wide variety of sensors, and they are differentiated between Proprioceptive/Exteroceptive and Passive/Active [11].

1. **Proprioceptive Sensors** Used for measurements related to robot's internal state, like position, voltages and currents, temperature, battery . . . etc.

2. **Exteroceptive Sensors** Used to gather information about the external environment, for example, range measurement, robot orientation, light intensity . . . etc.

3. **Passive Sensors** Used to detect and respond to some form of input energy from the physical environment. Temperature probes, microphones, and CCD or CMOS cameras are examples of passive sensors [11].

4. **Active Sensors** They send energy into the environment and then measure the response [11]. An ultrasonic sensor is an example of an active sensor.

For this project, we will use a lidar sensor for environment data collection.

Environment data extraction using LIDAR sensor

Lidar stands for "Light detection and ranging" or "Laser imaging, detection, and ranging", is a remote sensing technique that uses light in form of a pulsed laser to measure ranges, and create 2D and 3D models and maps of objects and environments.

Lidar uses Ultraviolet (UV), visible, or near-infrared light to image objects [12]. It consists of a laser transmitter and a light receiver. A light beam is emitted by the transmitter nearby objects and will be reflected later on to the sensor when hitting an object. The lidar sensor records each beam and measures the time needed to reach each obstacle, and the angle relative to the sensor frame.

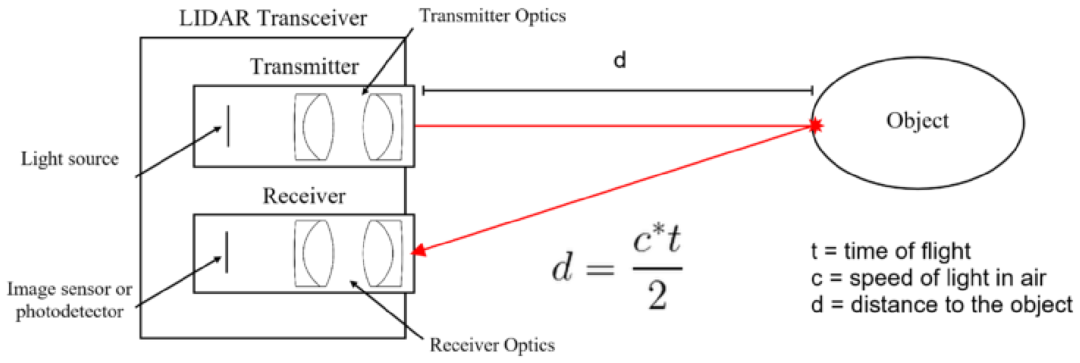


Figure 1.8: lidar working principle [13]

Lidar determines the distance to an object using the following formula:

$$d = \frac{c * t}{2} \tag{1.1}$$

Where c is the speed of light and t is the time spent for the laser light to travel to the object or surface being detected, then travel back to the detector [12].

Lidar Wavelengths range from around 10 micrometers (infrared) to roughly 250 nanometers (UV) depending on the target.

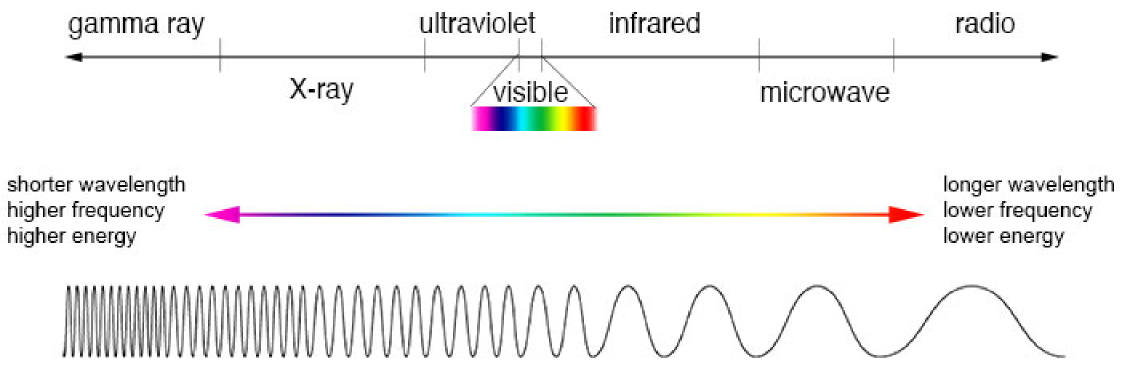


Figure 1.9: Comparison of wavelength, frequency and energy for the electromagnetic spectrum. (Credit: NASA's Imagine the Universe) [14]

1.5.2 Localization

Robot localization is the process of determining where a mobile robot is in relation to its environment. In a typical case, the robot is equipped with a map of the environment and sensors in order to observe its surroundings as well as monitor its motion. Using the information received from these sensors, the localization challenge becomes one of estimating the robot's location and orientation inside the map. A localization system must be able to cope with noisy observations and produce an estimate of the robot's location, and a measure of the location estimate's uncertainty [15].

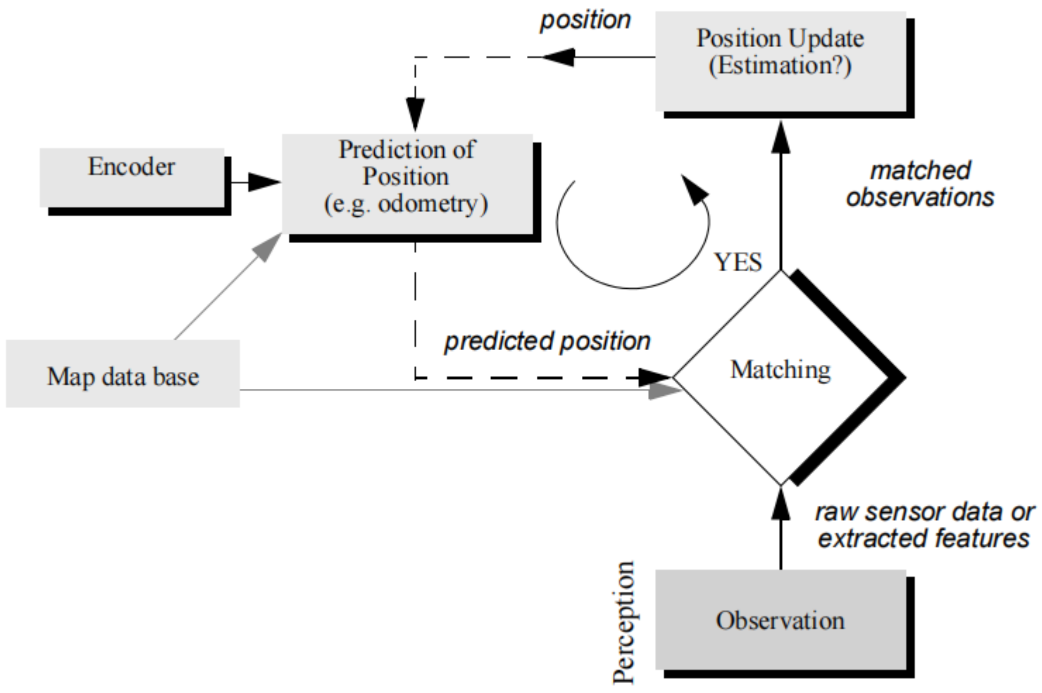


Figure 1.10: General schematic for mobile robot localization [11]

There are two types of localization: local and global.

Local localization:

Local localization is the process of determining the current position of the robot as a function of past position and displacement. This technique requires that the initial location of the robot to be known, as it is not able to recover if it loses track of its position [16]. Its main disadvantage is the accumulation of errors from measurements.

Global localization:

Global localization techniques always require a previous representation of the environment around, hence it can localize a robot without any previous knowledge about its position. Global localization techniques are more powerful than local localization approaches, as they are able to handle situations in which the robot is likely to experience serious positioning errors [16].

More details about the different localization methods used will be discussed in the second chapter.

1.5.3 Map representation

Robotic mapping addresses the problem of acquiring spatial models of physical environments through mobile robots [17].The study of robotic mapping has a long history. In the 1980s and early 1990s, this field has been widely divided between metric and topological approaches. Topological maps describe the connection of multiple places, whereas metric maps record the geometric aspects of the environment [17].

Topological Map:

A topological map is a type of graph that has been reduced such that only essential information remains and extraneous detail has been removed (see example in figure 1.11). These maps lack scale, and distance and direction are subject to change and variation, but the relationship between points is maintained [18].

There are several methods to extract such a graph. More modern techniques add nodes to the graph every time the robot goes a specific distance or captures a unique input sensory pattern. Nodes are connected when it is possible to find a path free of obstacles between them. Nodes inserted consecutively in the graph are implicitly connected by an arc. However, in order to link recently inserted nodes to previous ones, it is necessary to analyze the graph. This process is known as disambiguation [19].

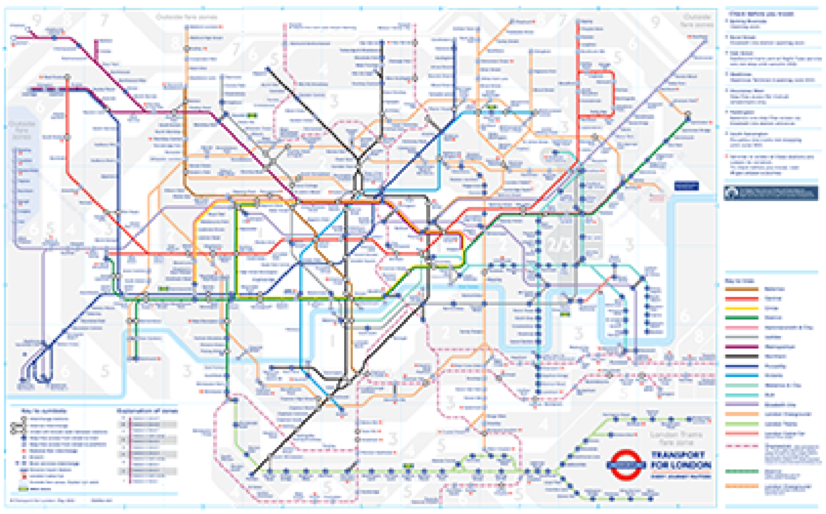


Figure 1.11: London Underground map of May 2022 [20]

Metric Map:

A metric maps are made by collecting accurate geometric descriptions of the environment.These descriptions are gathered from different sensors. There are many metric schemes: Configuration space, Voronoi diagrams, grid models, etc. The major benefit of these representations is that their geometry is identical to the geometry of the represented environment.

Probabilistic Occupancy Grid:

A probabilistic occupancy map is a location-based metric map, where each cell is associated with a given probabilistic value ranging from 0 to 1. Given a map m, $z_{1:t}$ the set of measurements from time 1 to t, and the set of robot poses $x_{1:t}$, the estimated posterior probability is defined as follow $P(m|z_{1:t}, x_{1:t})$ [21].

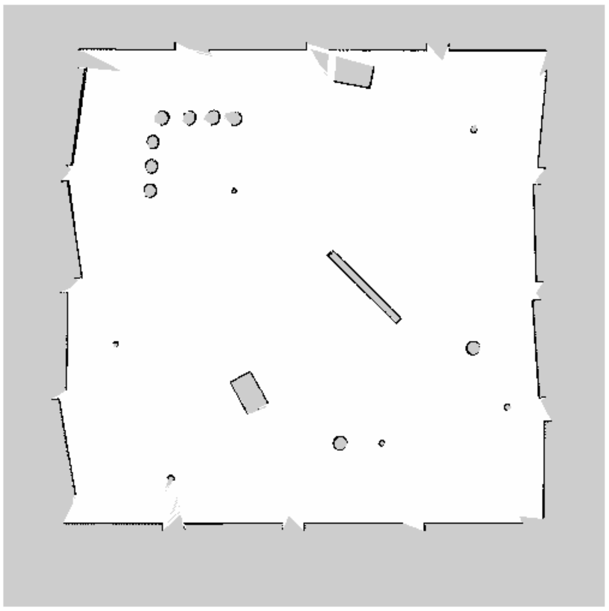


Figure 1.12: Example of a Probabilistic Occupancy Grid [21]

Binary occupancy map:

The binary occupancy grid is a map where each cell is associated with a given binary value: 0 if the cell is empty and 1 if the cell is occupied.

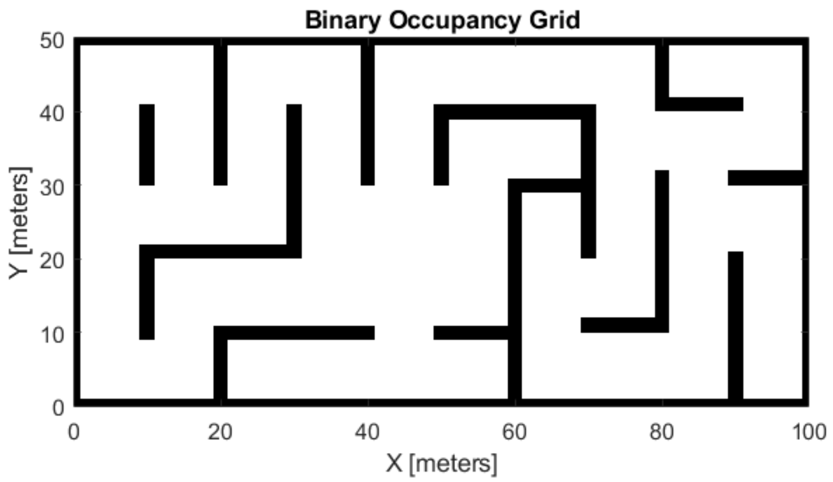


Figure 1.13: Binary Occupancy Grid [22]

1.5.4 Path planning

Path planning is a computational problem that involves determining a set of optimal configurations for moving a robot from one location to another. In the ideal case, path planning is done in a static environment with stationary obstacles and targets, however, the reality imposes the opposite of that. Hence, different ways to plan the robots' path in an unknown environment have been developed. The task of deducing a path increases in complexity as the configuration space (C-space) expands. The C-space represents the space that the robot may attain with respect to external constraints [23].

Problem formulation:

We can say that the path planning problem is defined as follows [24]:

1. $A \subset W$: The robot, is a single moving rigid object in world W represented in the Euclidean space as R2 or R3.
2. $O \subset W$: The obstacles are stationary rigid objects in W.
3. The geometry, the position, and the orientation of A and O are known a priori.
4. The localization of the O in W is accurately known.

Given a start and goal positions of $A \subset W$, plan a path $P \subset W$ denoting the set of positions so that $A(p) \cap O = \emptyset$ for any position $p \in P$ along the path from start to goal, and terminate and report P or $\emptyset$ if a path has been found or no such path exists [24].

Path planning categories:

Path planning methods for a mobile robotic system have some properties that vary according to the environment. These properties are whether it is static or dynamic, local or global, and exact or heuristic [25].

Static path planning refers to an environment with no moving objects or barriers, whereas dynamic path planning refers to an environment with dynamic moving and changing obstacles [25].

Global path planning is for when we have information about the environment based on a given map, grid, or cells. Local path planning is when the mobile robot has no information about the environment. The robot has to sense its surroundings before deciding how to move toward its target.

For completeness, the exact algorithms for mobile robot path planning find an optimal solution toward its goal if it exists, or simply prove that no optimal path is present. Whereas heuristic algorithms search for a good optimal solution in a shorter amount of time.

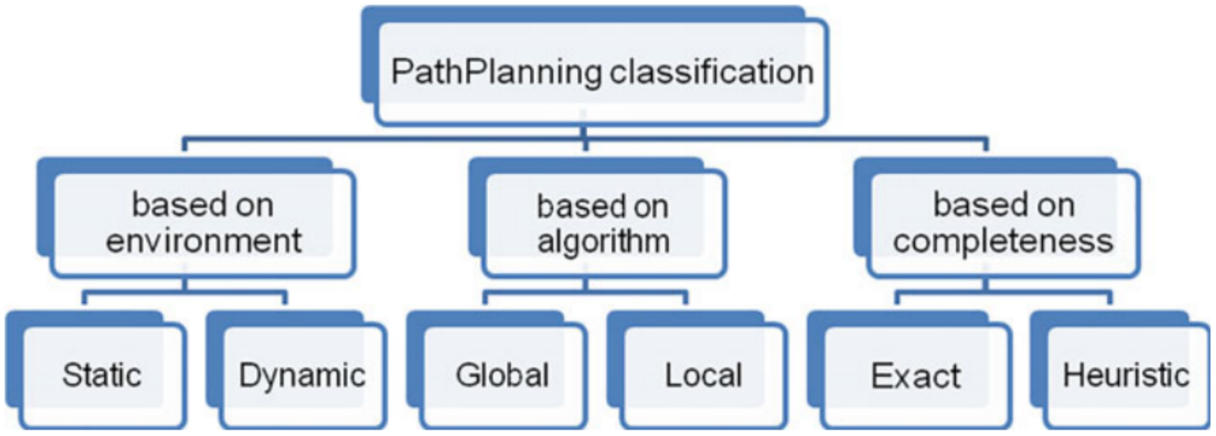


Figure 1.14: Images/Path Planning Categories [24]

1.6 Conclusion

This chapter presented a general knowledge and basic definitions of autonomous navigation systems and the different problems within autonomy. The next chapter will discuss two main problems in autonomous navigation which are Simultaneous Localization and Mapping (SLAM) and path planning.

Chapter 2

Theoretical background

2.1 Introduction

In robotics, SLAM (Simultaneous Localization and Mapping) is the process of constructing a map of an unknown environment in which the robot is navigating. The robot gathers information from its surroundings via sensors and generates a map over time.

Path planning is the process of finding the optimal path from a start to a goal state, in order for the robot to navigate freely. The generated path can be in a form of a set of states or way-points. To plan a path a map of the environment along with a start and a goal state input is required.

This chapter provides the theoretical background and the mathematical description behind these two problems, as well as discusses the methods that will be used in this project.

2.2 Simultaneous Localization and Mapping

2.2.1 Formulation of SLAM problem

Simultaneous Localization and mapping abbreviated as SLAM is one of the most fundamental problems in robotics. It was first introduced in 1986 at the IEEE Robotics and Automation Conference in San Francisco by the researchers Peter Cheeseman, Jim Crowley, and Hugh Durrant-Whyte, where they were trying to apply some theoretical estimation methods to mapping and localization [26].

SLAM asks about the possibility of placing some mobile robot in an unknown environment and simultaneously building a model map and determining the location of the robot within that map using different sensors. One major problem with SLAM is that the measurements read from the sensors will invariably contain noise, and the motion performed by the robot too will produce uncertainties in its positioning [27].

2.2.2 Mathematical description of SLAM

Given $X_T = \{x_0, x_1, x_2....x_T\}$ the robot path, where x_0 is known. The robot relative motions are given as follow $U_T = \{u_0, u_1, u_2....u_T\}$, where u_t is the robot motion between $t-1$ and t. $M_T = \{m_0, m_1, m_2....m_{n-1}\}$ where M is the true map of the environment and m_i are the set of vectors that represents the positions of landmarks. $Z_T = \{z_0, z_1, z_2....z_T\}$ are the robot measurements at each time.

SLAM is defined as the problem recovering a map M, robot path X_T from odometry U_T and observations Z_T [11].

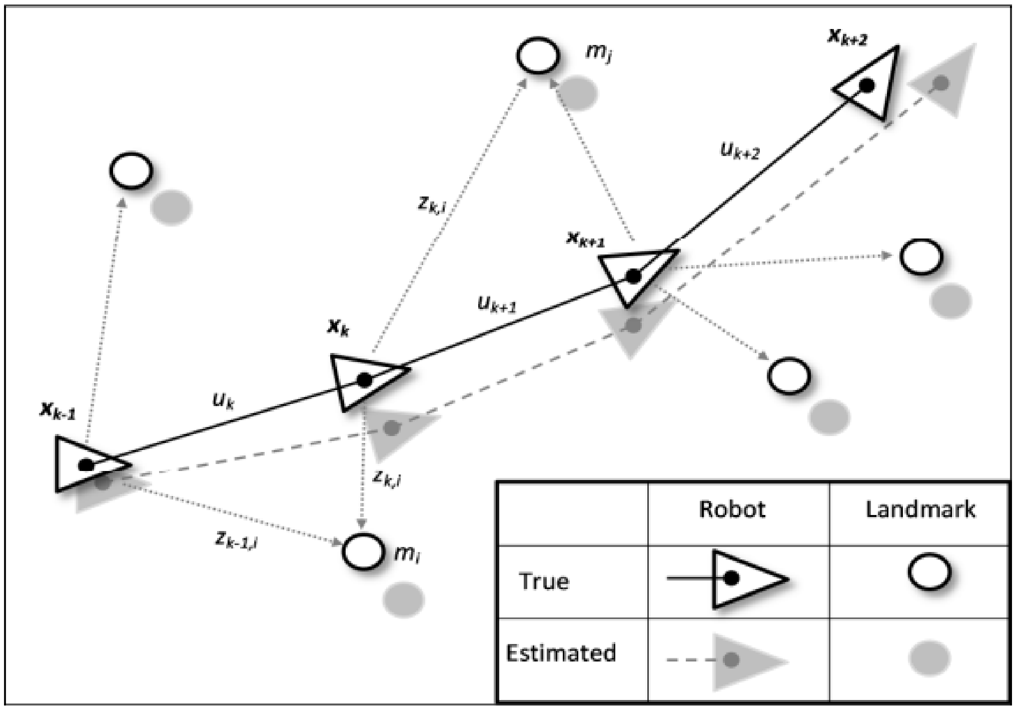


Figure 2.1: SLAM problem example [26]

Full SLAM problem:

It estimates the posterior probability over full data X_T and M and it is given by:

$$p(x_T, M|Z_T, U_T)$$

Online SLAM problem:

It estimates the posterior probability over current pose x_t and M and it is given by:

$$p(x_t, M|Z_T, U_T)$$

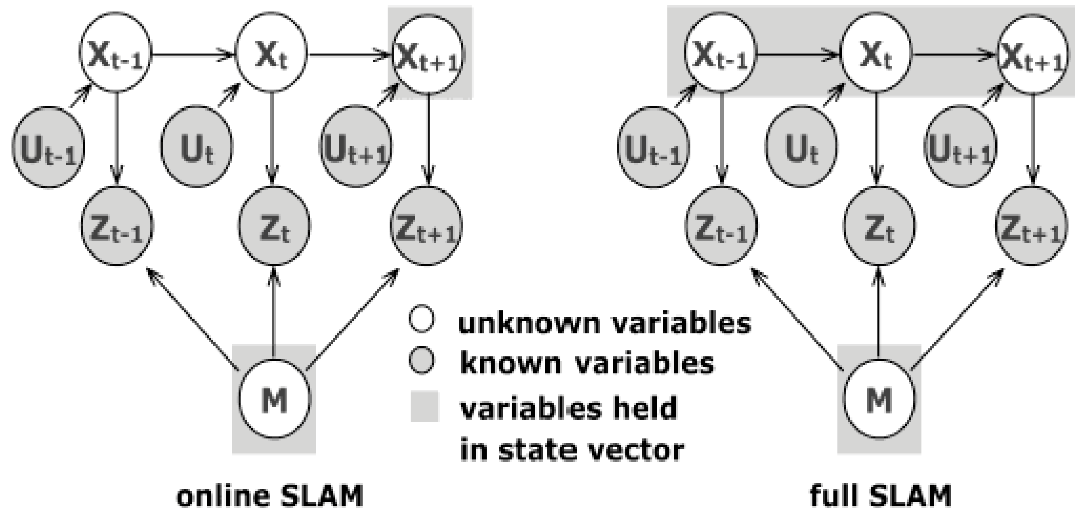


Figure 2.2: Forms of SLAM problem [28]

2.2.3 Solutions to the SLAM problem

A solution to the SLAM problem requires finding a representation for both the observation and the motion model which allows efficient computation of the prior and posterior distribution. Several algorithms have been developed to solve the SLAM problem. They are classified into two groups depending on whether we are dealing with full or online SLAM.

2.2.4 Graph-Based SLAM using pose-graphs

It is an approach that aims to solve the full SLAM problem. It attempts to model a map using a sparse graph of nodes and constraints [29].The nodes represent the robot poses $x_0, x_1, x_2....x_T$ and the features of the map $m_0, m_1, m_2....m_{n-1}$, whereas the relative position between two poses x_{t-1}, x_t are represented by constraints [11].

According to graph-based SLAM formalization a map is represented in a form of a graph $G = \langle X, C \rangle$, where each node is associated to a robot pose and each edge $e = \langle x_i, x_j \rangle \in C$ expresses the spatial relationship between poses [29]. Nodes are also referred to pose graphs.

Let $X = (x_1, x_2, ..., x_n)^T$ a vector of poses, we define the mean z_{ij} and Ω_{ij} the information matrix of an edge between two nodes i and j. z_{ij} can be seen as an equilibrium point of a spring with stiffness Ω_{ij} connecting the masses at X_i and X_j.

We let $\hat{z_{ij}}(z_i, z_j)$ the relative transformation between two nodes. The error function $e(x_i, x_j, z_{ij})$ that computes the distance from the two poses to the equilibrium point is given by:

$$e(x_i, x_j) = Z_{ij} - \hat{z_{ij}}(x_i, x_j) \tag{2.1}$$

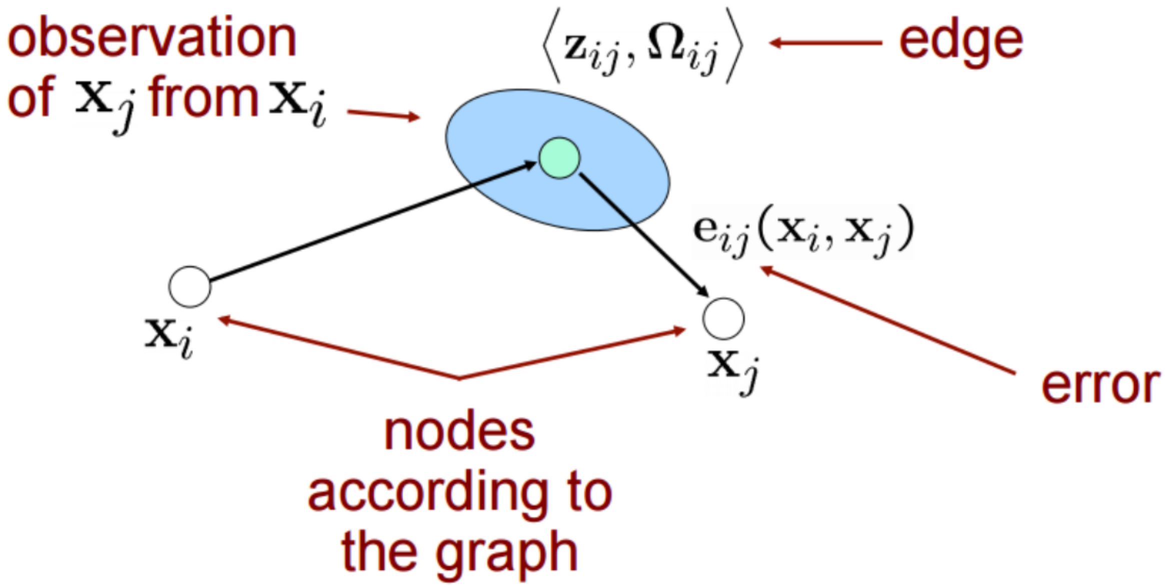

observation of $\mathbf{x}_j$ from $\mathbf{x}_i$
$\langle \mathbf{z}_{ij}, \mathbf{\Omega}_{ij} \rangle$
edge
$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j)$
$\mathbf{x}_i$
$\mathbf{x}_j$
error
nodes according to the graph

Figure 2.3: A pose-graph representation of a SLAM process [30]

Pose-graphs optimization:

To minimize the error in the graph, we need to find the configuration of the nodes x^* that minimize the energy of all edges:

$$F(x) = \sum_{\langle i,j \rangle \in C} e_{ij}^T \Omega_{ij} e_{ij} \tag{2.2}$$

hence we seek to solve the following equation:

$$x^* = \arg\min_x F(x) \tag{2.3}$$

To solve this problem several algorithms has been developed. Some useful techniques has been discussed in [31].

2.3 Path planning

A path is a sequence of pose states or way-points that smoothly connect the start and the goal, while avoiding obstacles. Determining this sequence is referred to as path planning [32]. Path planning requires a map of the environment along with start and goal positions. The map can be represented in different ways such as grid-maps, state spaces, and topological road-maps [33].

Consider the system $\dot{x}(t) = f(x(t), u(t))$ where $x \in X$ the state space and $u \in U$ the input space. The path planning problem consists of finding the input $u : [0, T] \rightarrow U$ that yields to a feasible path $x \in C_{free}$ (The obstacle-free space) for $t \in [0, T]$ from the initial state to the goal state [34].

The optimal path planning problem imposes the additional requirement that the resulting feasible path minimize a given cost function $c(x)$ [34]. To solve the optimal path planning problem, several algorithms have been developed, some of which will be discussed later in this chapter.

2.3.1 Configuration space

The configuration space also referred to as C-space, is defined as the n-dimensional space that gathers the set of all possible configurations of the robot (see figure 2.4). For every configuration, there is a unique point in the C-space, and for every point in the C-space, there is a unique configuration of the robot [35].

Definition 01: The configuration of a robot is a complete specification of the position of every point of the robot [35].

Definition 02: The dimension of the configuration space is defined by the Degrees of Freedom (DOF), which is basically the number of variables specified.

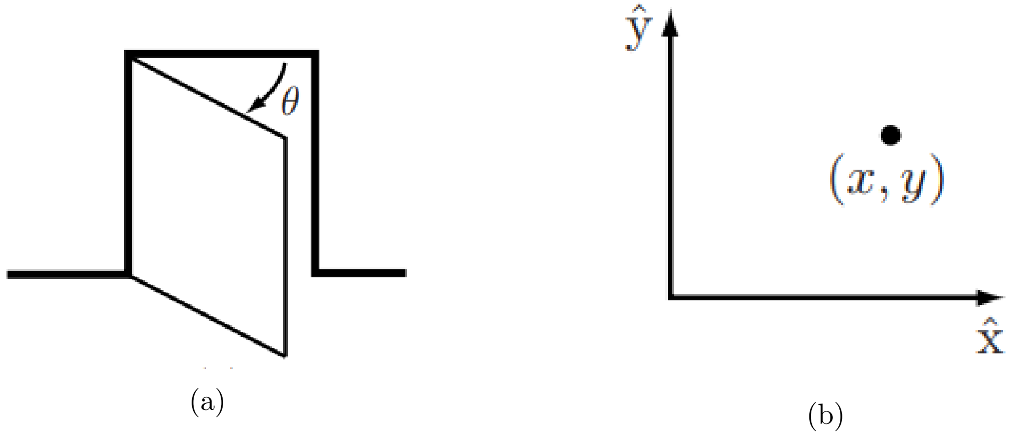


Figure 2.4: (a) The configuration of a door is described by the angle θ. (b) The configuration of a point in a plane is described by coordinates (x, y) [35]

For this project, the robot configuration is described by the coordinates (x, y, θ). x and y are the coordinates of along the x and y axes, and θ is the orientation of the robot at the specific position.

Free space:

The free space C_{free} is the set of configurations that avoids collision with obstacles [36].

Obstacle space space:

The obstacle space is a space that the robot can not move to [36].

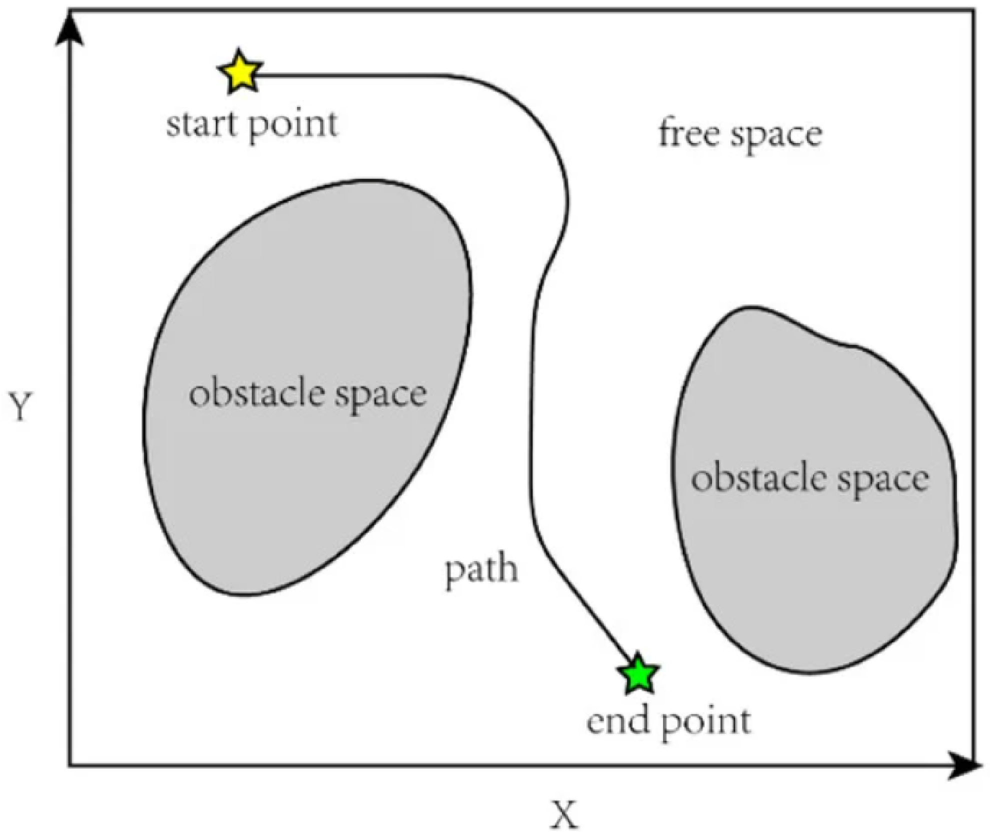


Figure 2.5: Schematic of a two-dimensional C-space [37]

2.3.2 Path planning methods

There are two major types of algorithms used for path planning: Grid-based and Sampling-based search algorithms.

Grid based methods

It overlays on a grid configuration space and assumes that each configuration is defined with a grid point. At each grid point, the robot can move to the adjacent grid point as long as the line between them is within the free space C_{free} [36].

The main disadvantage of this method is that it fails to find a path through narrow portions of C_{free}. Moreover, it is not useful for high-dimensional spaces as it requires more memory for computation. Several grid-based algorithms have been developed from which A* and Hybrid A*.

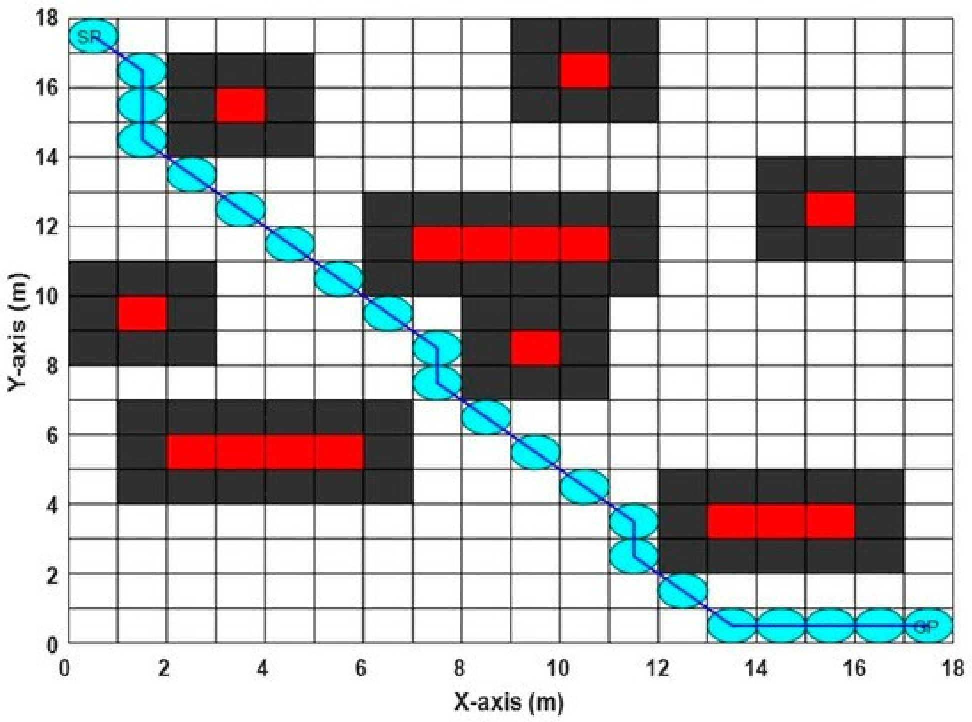


Figure 2.6: Grid-based mobile robot path planning [38]

Sampling based methods

It represents the configuration space with a road-map of sampled configurations [36]. It creates a searchable tree by randomly sampling new nodes in the state space. They are useful for both low and high dimensional search spaces [33]. Several sampling-based algorithms have been developed, the most used ones are RRT and RRT* (rapidly exploring random trees).

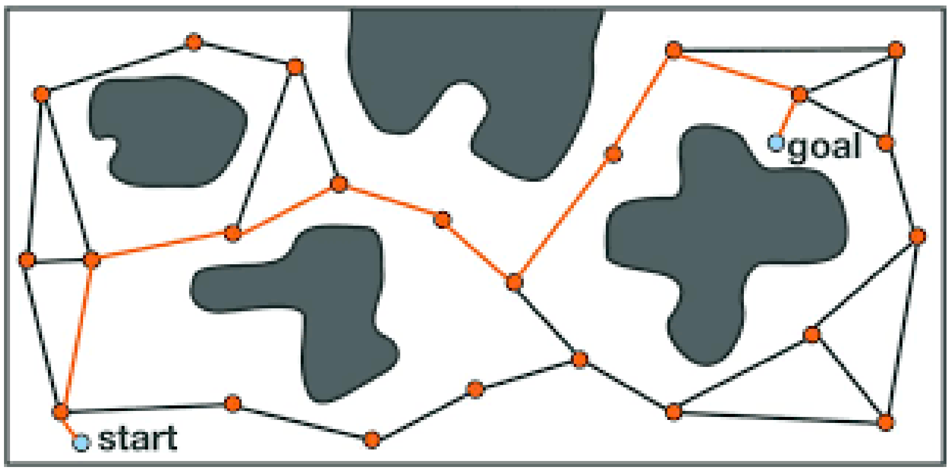


Figure 2.7: Sampling-based Motion Planners [39]

2.4 Rapidly Exploring Random Trees

2.4.1 RRT Algorithm

It attempts to generate random nodes within the state space and connect them to form a path to the goal position. These nodes need to be validated or excluded based on the map constraints so that the robot does not collide with obstacles. The path generated by the RRT algorithm may be fast but it does not validate an optimal path.

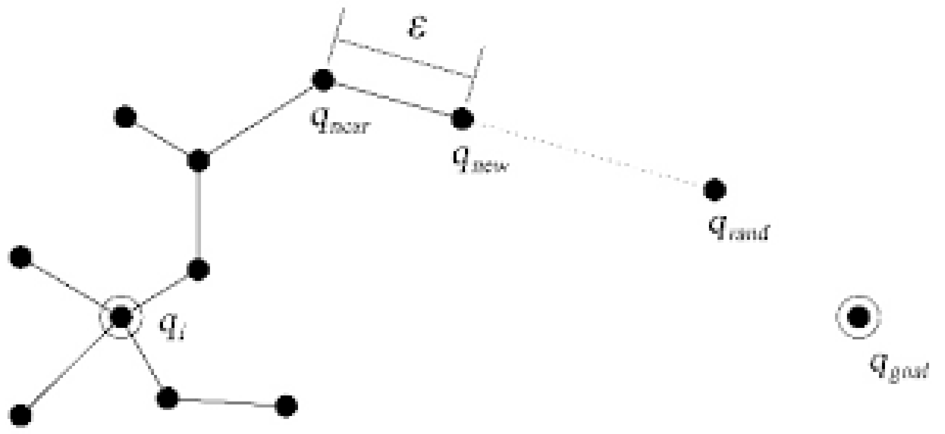


Figure 2.8: RRT Algorithm [40]

1. Set a graph that includes the initial pose q_i and its single vertex.
2. For every incrementation we select a random point q_{rand} in the free space C_{free}.
3. A connection q_{near} is made to relate the nearest vertex $v \in V$ to the new sample q_{new} within a distance ϵ.
4. if there is an obstacle in the space of the new connection, q_{rand} is added to the vertex set, and (v, q_{rand}) is added to the edge set.

Algorithm 1 RRT Algorithm

```
$V \leftarrow q_i$;
$E \leftarrow \emptyset$;
**for** i=1, ...., k **do**
    $q_{rand} \leftarrow SampleRandomNode()$;
    $q_{near} \leftarrow Nearest(T = (V, E), q_{rand})$;
    $q_{new} \leftarrow Steer(q_{near}, q_{rand})$;
    **if** $NoObstacle(q_{near}, q_{new})$ **then**
        $V \leftarrow V \cup q_{new}$;
        $E \leftarrow E \cup (q_{near}, q_{new})$;
    **end if**
**end for**
**return** $T = (V, E)$;
```

Sampling random node function: The $SampleRandomNode()$ function samples a random point in the free space.

Nearest neighbor: The $Nearest(T = (V, E), q_{rand})$ function returns the nearest random point in terms of the distance function.

Steering function: The $Steer(q_{near}, q_{rand})$ function deduces the direction between q_{near} and q_{rand} along the path $x : [0, T] \to X$.

Collisions avoidance: The $NoObstacle((q_{near}, q_{new})$ function checks for collisions between the two states (q_{near} and q_{new}).

2.4.2 RRT* Algorithm

It is an optimized version of the RRT algorithm. It aims to find the shortest path to the goal position within a small amount of time and with less memory consumption. That is achieved by constantly choosing the right parent node at each step.

Algorithm 2 $T = (V, E) \leftarrow RRT^*\ (q_i)$

```
$T \leftarrow InitializeTree()$;
$T \leftarrow InsertNode(\emptyset, q_i, T)$;
for i=1, ...., N do
  $q_{rand} \leftarrow RandomNode(i)$;
  $q_{nearest} \leftarrow Nearest(T, q_{rand})$;
  $q_{new} \leftarrow Steer(q_{nearest}, q_{rand})$;
  if $NoObstacle(q_{new})$ then
    $Q_{near} \leftarrow Near(T, q_{new})$;
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_{new}, q_{nearest})$;
    $T \leftarrow Insertnode(q_{min}, q_{new}, T)$;
    $T \leftarrow Rewire(q_{min}, q_{new}, Q_{near}, T)$;
  end if
end for
return $T$;
```

RRT* incrementally builds a tree by sampling a random node q_{rand} in the free space, and solves for the trajectory that connects the new node q_{new} to the nearest node $q_{nearest}$ [34].

The basic RRT algorithm inserts q_{new} into the tree with $q_{nearest}$ as its parent and continues with the next iteration. It is here that the operation of the RRT* differs. The algorithm checks all the nodes in the neighborhood of q_{new} and evaluates the cost of choosing each parent. The node that yields the smallest cost is selected as a parent node [34]. This process is done through the $ChooseParent()$ function.

Near function: The $Near(T, q_{new})$ function returns the vertices in V that are near the node q_{new}.

Algorithm 3 $q_{min} \leftarrow ChooseParent(Q_{near}, q_{nearest}, q_{new})$

```
$q_{min} \leftarrow q_{nearest}$;
$c_{min} \leftarrow Cost(q_{nearest}) + C(q_{new})$;
for $q_{near} \in Q_{near}$ do
  $q_{new} \leftarrow Steer(q_{near}, q_{new})$;
  if $NoObstacle(qpath)$ then
    $c_{new} = Cost(q_{near}) + C(qpath)$;
    if $c_{new} < c_{min}$ then
      $q_{min} \leftarrow q_{near}$;
      $c_{min} \leftarrow c_{new}$;
    end if
  end if
end for
return $q_{min}$;
```

The *Rewire* function checks each node q_{near} in the neighborhood of q_{new} to see if reaching q_{near} through q_{new} would achieve a lower cost than doing so through its current parent.

Algorithm 4 $T \leftarrow Rewire(q_{min}, q_{new}, Q_{near}, T)$;

```
for $q_{near} \in Q_{near}$ do
  $qpath \leftarrow Steer(q_{near}, q_{new})$;
  if $NoObstacle(qpath)$ and $Cost(q_{new}) + C(qpath) < Cost(q_{near})$ then
    $T \leftarrow Reconnect(q_{new}, q_{near}, T)$;
  end if
end for
return $T$;
```

2.5 Conclusion

This chapter discussed the mathematical description behind both the SLAM (Simultaneous Localization and Mapping) and the path planning problems. It also covered the most used methods in order to resolve these problems, which are the pose-graph estimation and the rapidly exploring random trees. The next chapter will cover the simulation and the results of the project.

Chapter 3

Simulation and Results

3.1 Introduction

This project aims to implement a simulation of an autonomous mobile robot navigating in a closed indoor environment. The proposed model uses a lidar sensor for environment data extraction, the SLAM algorithm for map construction, and the RRT* algorithm for path planning.

All the functions and algorithms used for this project were implemented and tested using MATLAB2021a.

This chapter will describe the overall system design of the project, then dive deeply into each part of it.

3.2 System design

In order for our autonomous mobile robot to navigate, it has to perform a set of tasks. For this project, we developed a simulation of an autonomous mobile robot in a closed indoor environment that is able to perform the following tasks:

1. Environment data extraction using a lidar sensor.
2. Map construction from lidar scans using the SLAM Map Builder app from MATLAB.
3. Path planning using RRT* algorithm.

The next sections of this chapter will discuss the simulation and results obtained from each of these steps.

Figure 3.1 demonstrates the flowchart of the overall system.

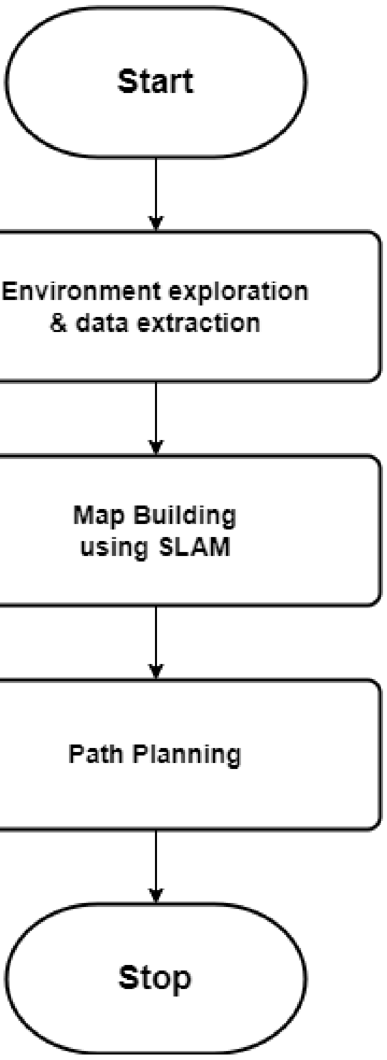


Figure 3.1: The full system design

3.2.1 Environment data extraction

For the robot to navigate, it needs to have a map that represents its environment. To construct a map, the robot needs to explore its environment first and capture data.

For this project, we assume that our robot has no information about its surroundings. To make the robot explore its environment, we give it a path with a set of poses it needs to follow while capturing data within a reference map we already have previous information about.

The robot uses a lidar sensor to capture data. The detection range of the lidar is 0 to 50 meters, with a 360-degree detection angle.

Figure 3.2 describes the environment data extraction process.

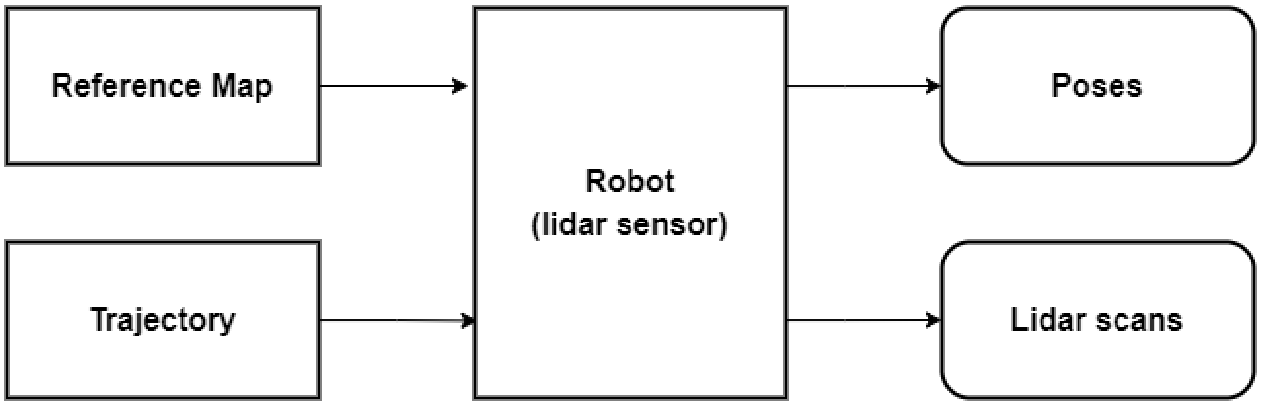
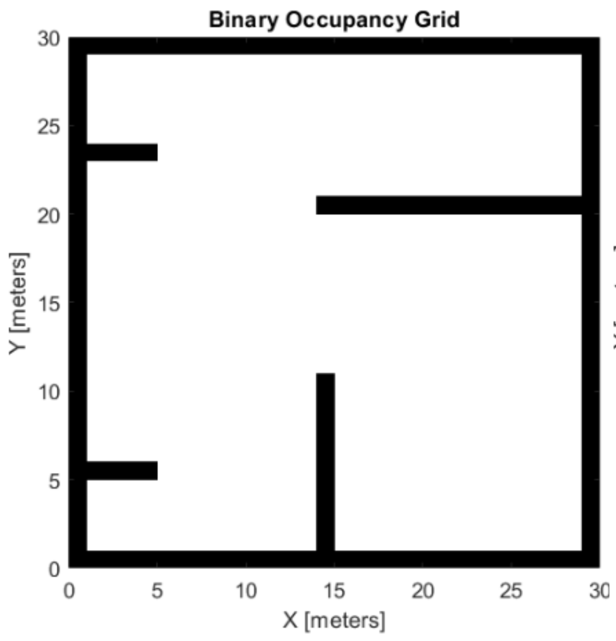
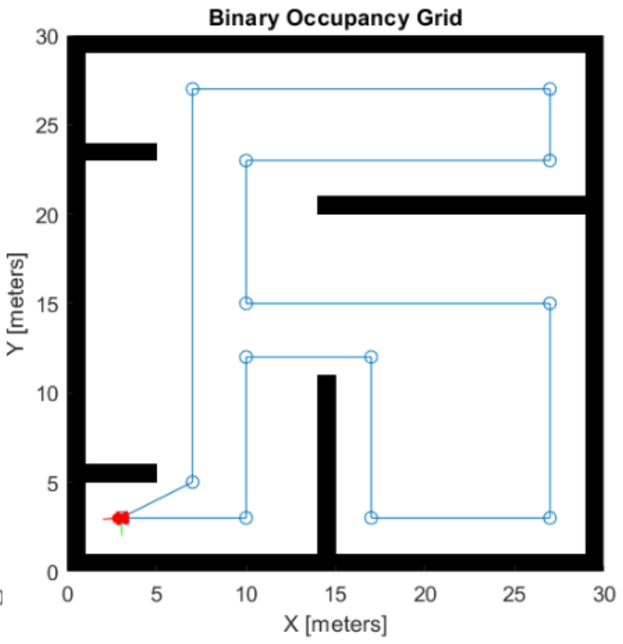


Figure 3.2: Environment data extraction process

Figures 3.3a and 3.3b demonstrates the reference map and the given input trajectory for environment exploration.



(a)



(b)

Figure 3.3: (a) The reference map, (b) Mobile robot following the input trajectory

After this process is done we get two output matrices: An odometry sensor matrix ($n \times 3$) and lidar-scans matrix ($n \times 1$), where for every given pose in the odometry sensor matrix there is a unique corresponding lidar scan in the lidar-scans matrix.

The reference map is of size 30×30 and the robot poses are given by (x, y, θ). x and y are the coordinates of the robot along the x and y axes, and θ is the orientation of the robot at the specific position.

The obtained outputs will be used on the upcoming section to build a model map of the environment.

3.2.2 Map building

To construct an occupancy map of the environment, we need the lidar scans and their corresponding odometry sensor data (poses) obtained from the previous section. The poses and scans are used to generate and update a pose graph, this is done by matching the scan corresponding to each pose to previous scans each time. To optimize our pose graph we need to simultaneously compensate for the robot odometry drift as the robot moves. This process is referred to as loop closures detection and it is accomplished by detecting the previously visited positions each time. After exploring all the data, we obtain an occupancy map that is close to the given input reference map.

Figure 3.4 summaries the map building process.

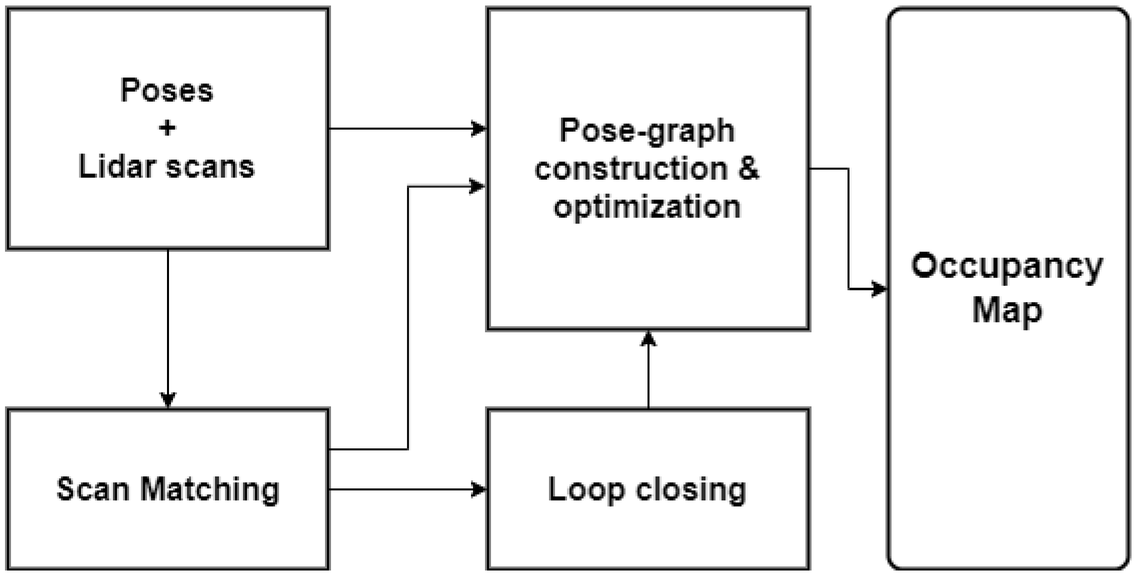


Figure 3.4: Map Building Process

For this project, the map building process is done using the SLAM Map Building Map from MATLAB.

The SLAM Map Builder app:

The SLAM Map Builder app loads recorded lidar scans and odometry sensor data to build a 2-D occupancy grid using simultaneous localization and mapping (SLAM) algorithms. Incremental scan matching aligns and overlays scans to build the map. Loop closure detection adjusts for drift of the vehicle odometry by detecting previously visited locations and adjusting the overall map [41].

Simulation and results:

As it is illustrated in the figures 3.5, we start first by uploading our data (Lidar scans and their corresponding poses) into the SLAM MAP Builder.

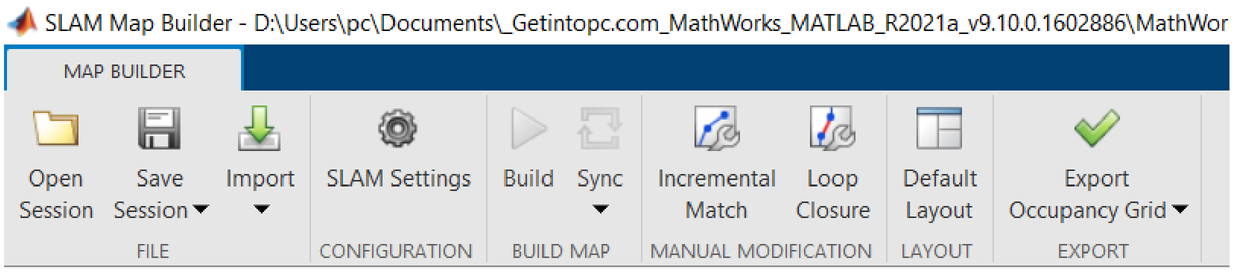
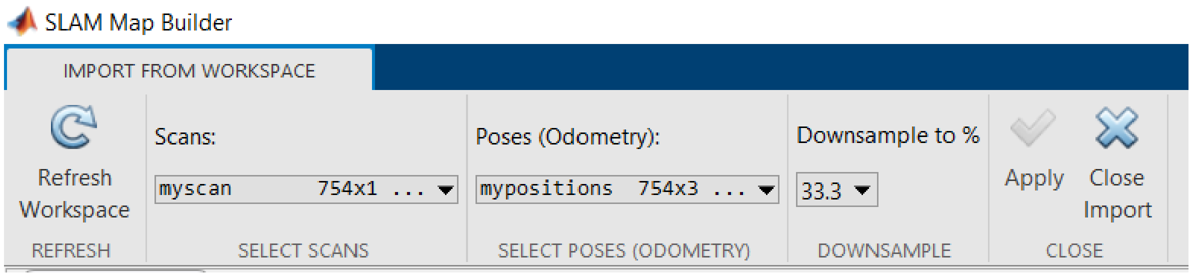


Figure 3.5: Importing data

For a faster computational time, we down-sample our data by 33.3% (see figure 3.6).



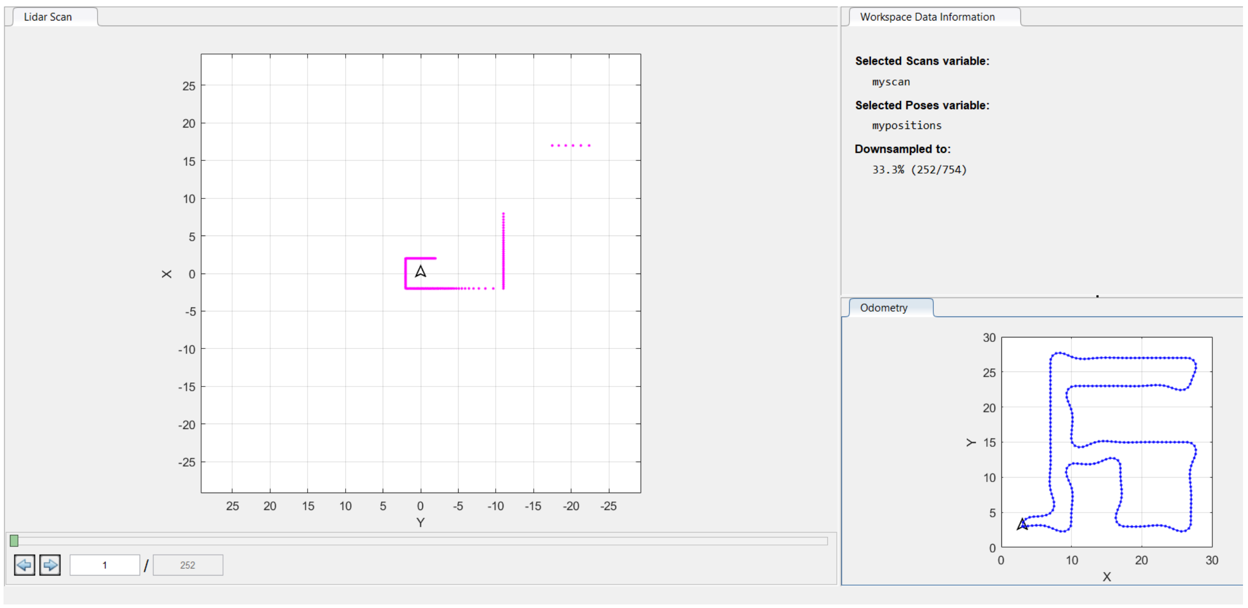
Scans: myscan 754x1 ...

Poses (Odometry): mypositions 754x3 ...

Downsample to %: 33.3

Figure 3.6: Selecting scans and poses

The imported data is shown in figure 3.7.



Selected Scans variable:
myscan
Selected Poses variable:
mypositions
Downsampled to:
33.3% (252/754)

Figure 3.7: Imported data down-sampled by 33.3%

Figure 3.8 demonstrates the SLAM algorithm in process.

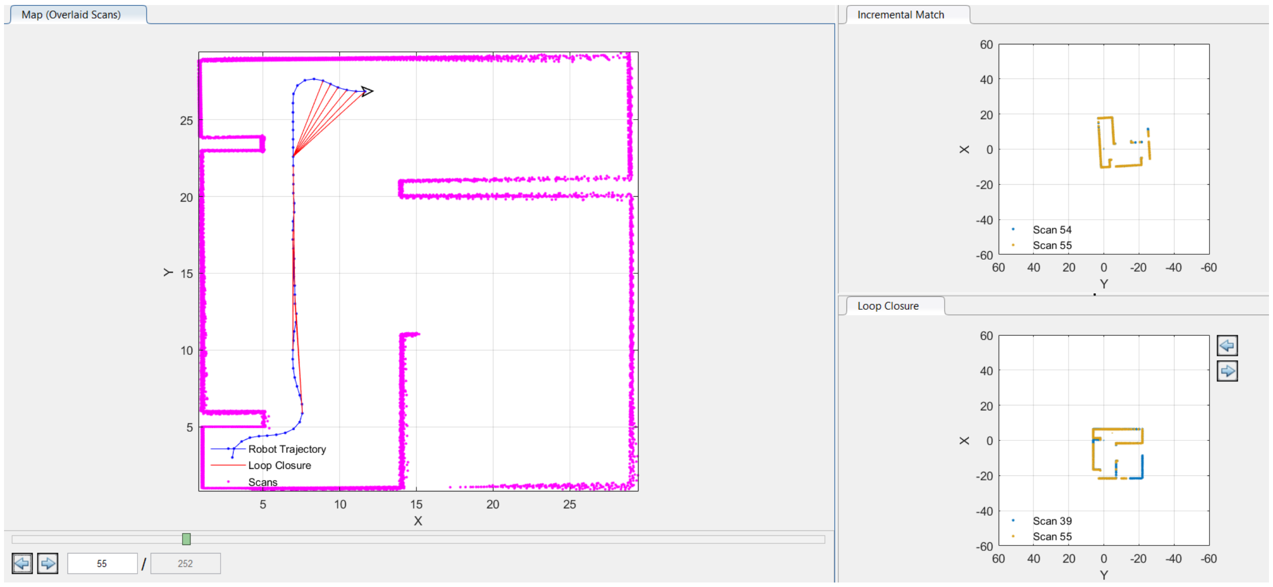


Figure 3.8: SLAM algorithm in process

For every single pose, the software overlay its corresponding scan with the scan of its precedent in order to build a map (see figure 3.9).

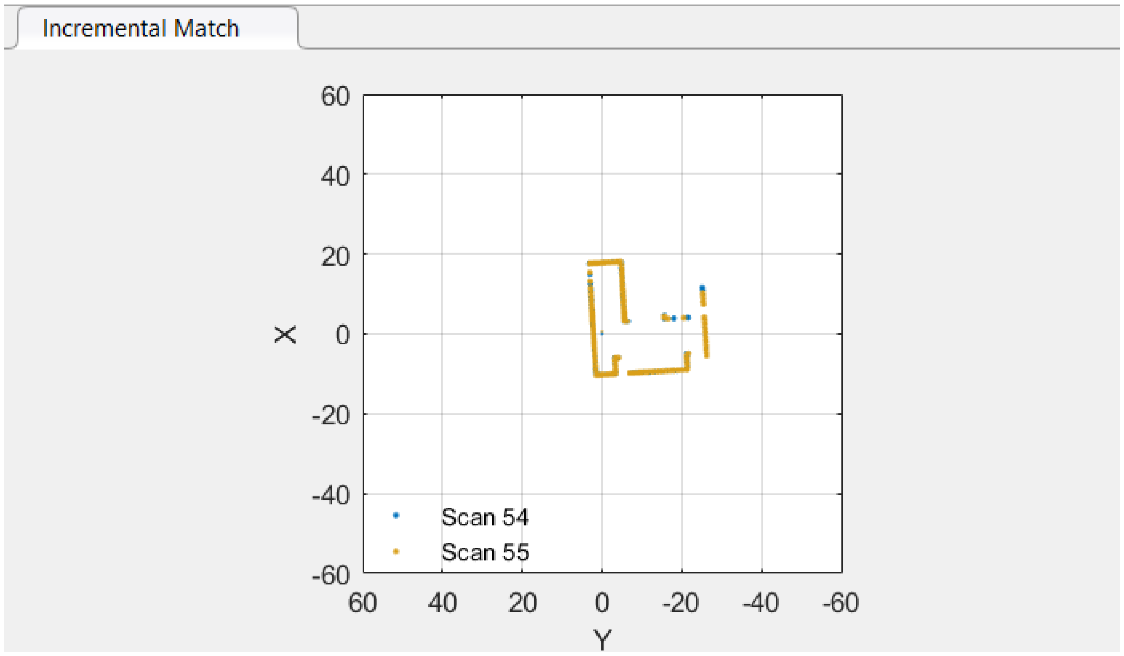


Figure 3.9: Scan matching the observation of the current pose with its precedent

The software tries to compare the scans corresponding to each odomotry data with old scans. If a match exists we say a loop closure has been detected. As it is demonstrated in figure 3.10, the current scan (scan 55) matches the scan 39. The software connects the two poses with a red line to indicate the detection of a loop closure (see figure 3.8).

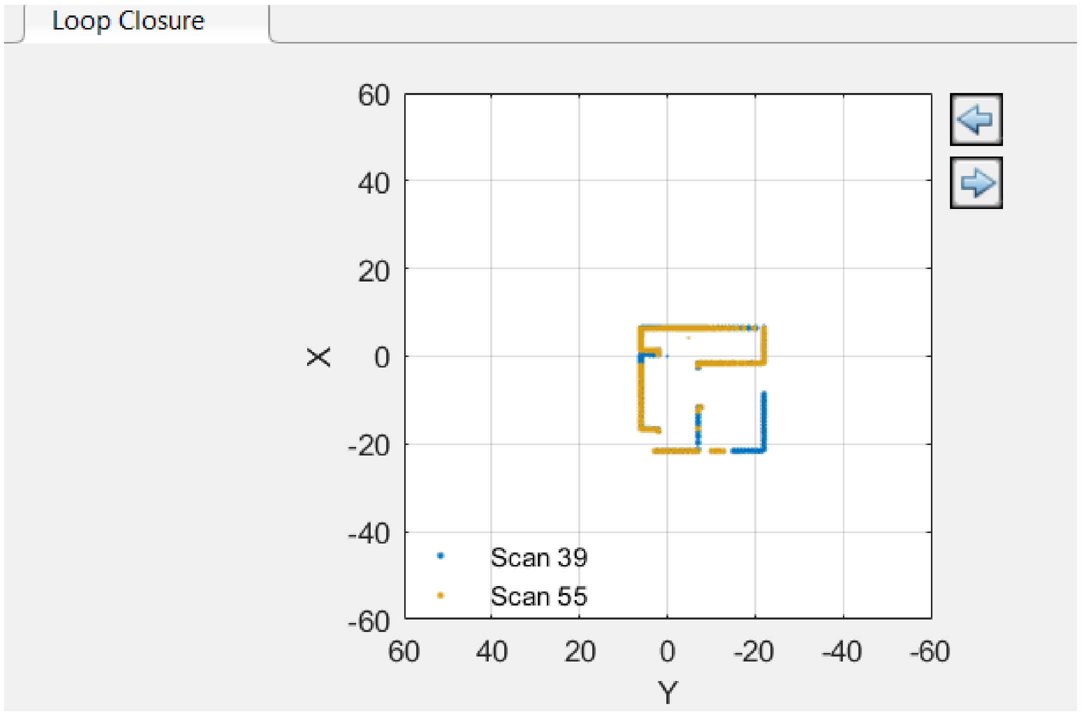


Figure 3.10: Detection of a loop closure

The software continues matching scans and detecting loop closures for all the given poses. This process helps optimize the map each time, which leads to better results. After the end of the simulation, we get the following map (see figure 3.11).

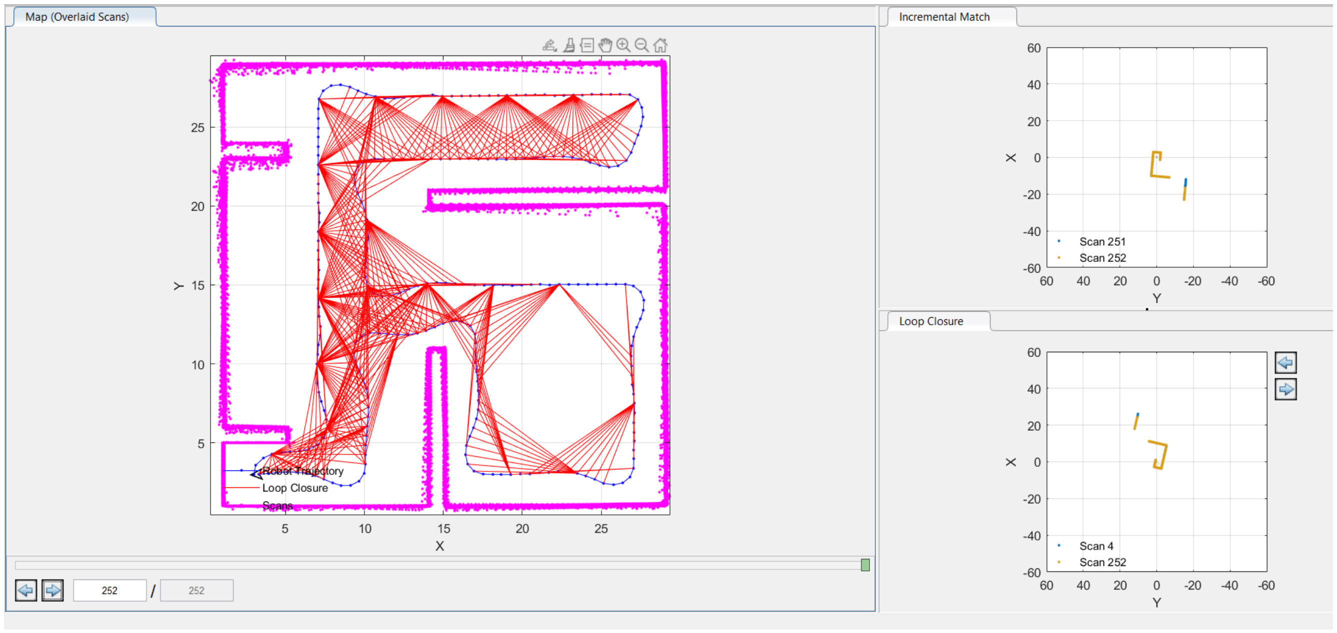


Figure 3.11: SLAM algorithm after the end of the simulation

The corresponding occupancy map is demonstrated in figure 3.12.

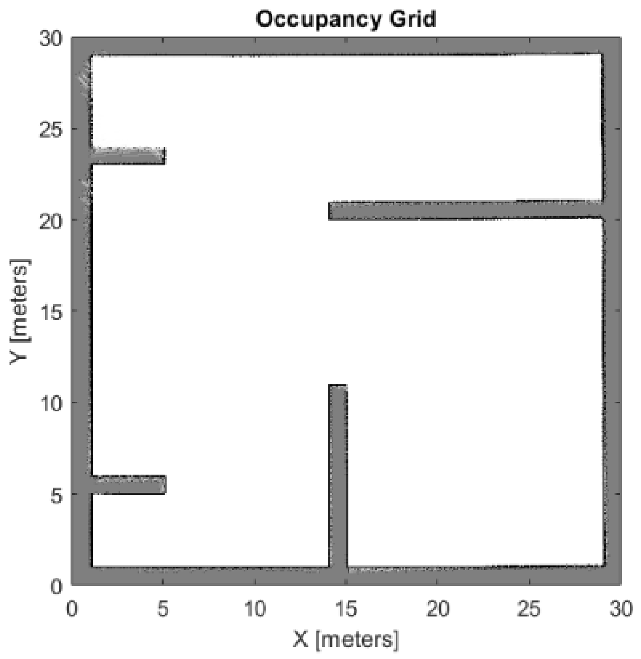


Figure 3.12: Occupancy map

We save the map obtained in the MATLAB workspace, so that we can use it later for path planning.

3.2.3 Path planning

In order for the robot to plan its path, it needs to have a map of the static environment it is moving in as well as a starting and a goal position within that map. On this part we will be using the occupancy map we obtained on the previous section (The map represented on figure 3.12). Figure 3.13 illustrates the path planning process.

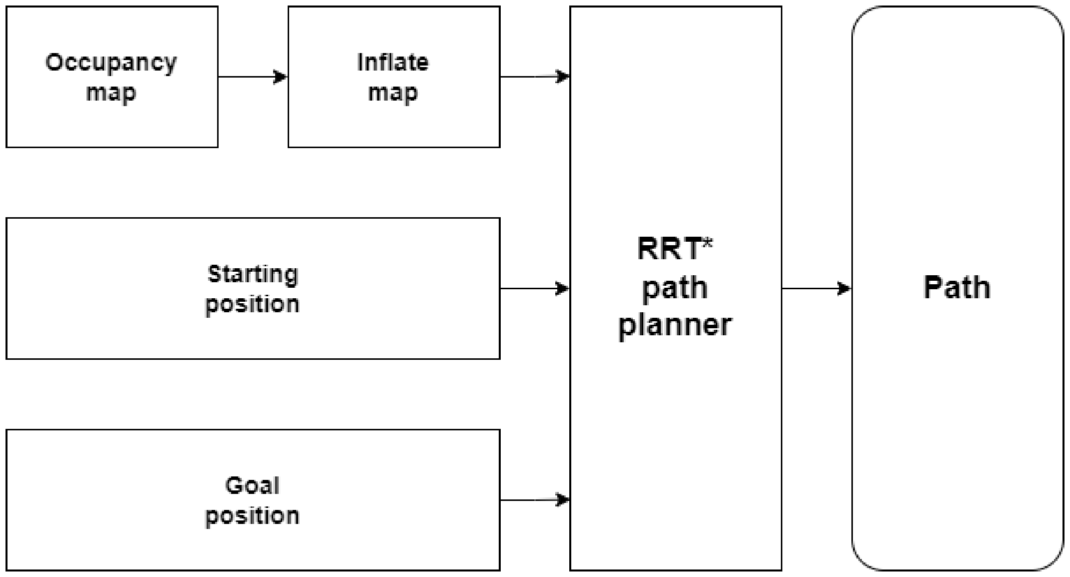
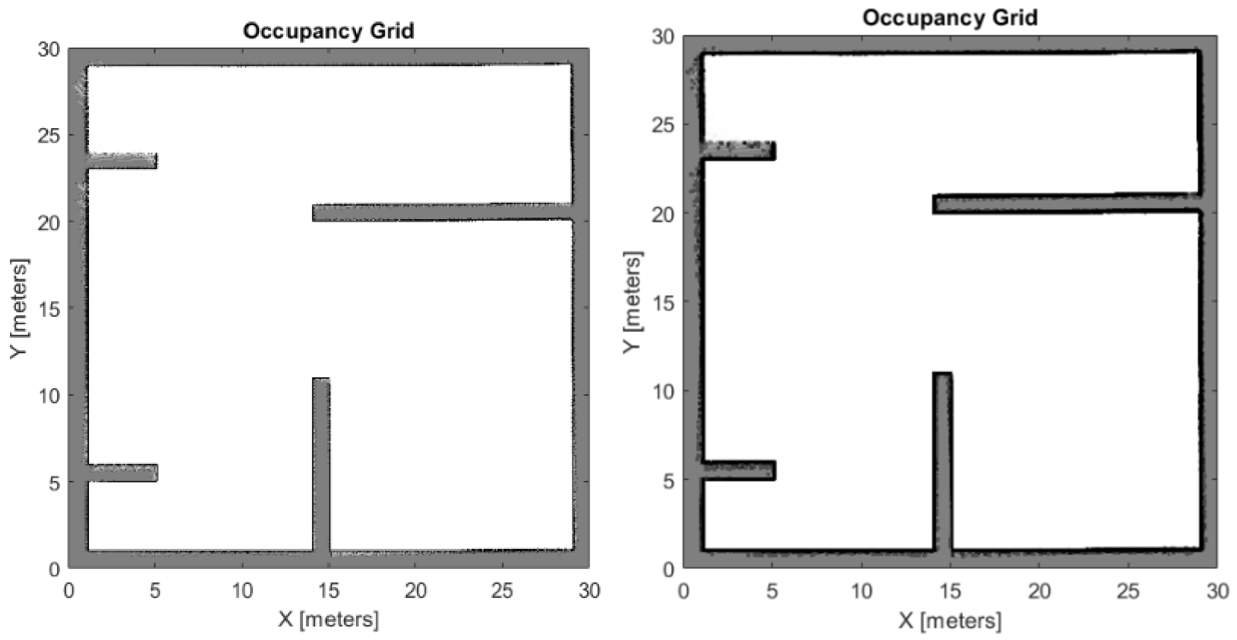


Figure 3.13: Path planning process

To use our occupancy map we need to inflate the occupied position, this aids in avoiding collision with walls. Figure 3.14 demonstrates the occupancy map before and after inflation.



(a) (b)

Figure 3.14: (a) Map before inflation, (b) Map after inflating the occupied positions

Simulation and results:

For the first testing scenario, we set the starting position to $(5, 3, 0)$ and the goal position to $(15, 26, 0)$. Figure 3.15 illustrates the first path planning scenario.

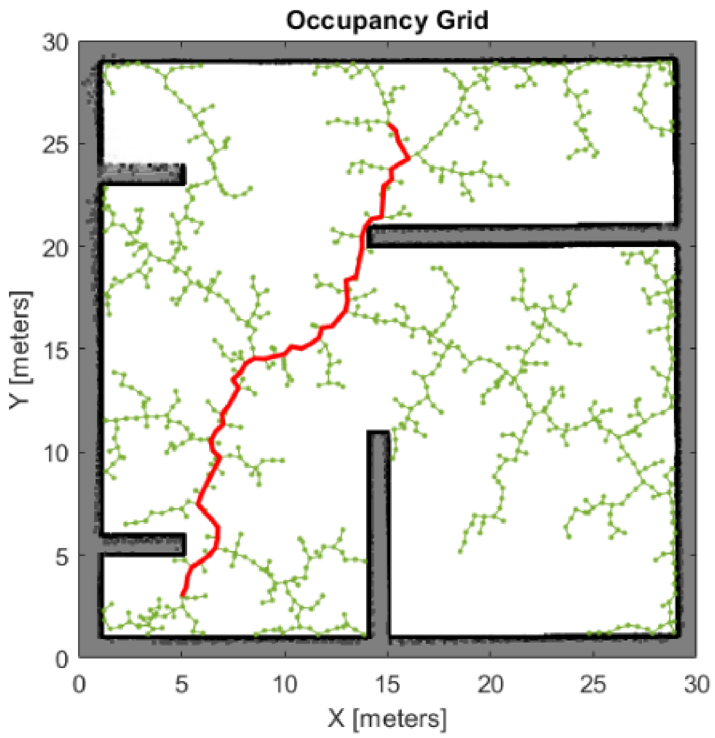


Figure 3.15: The first path planning scenario

For the second testing scenario, we set the starting position to $(5, 3, 0)$ and the goal position to $(25, 7, 0)$. Figure 3.16 illustrates the second path planning scenario.

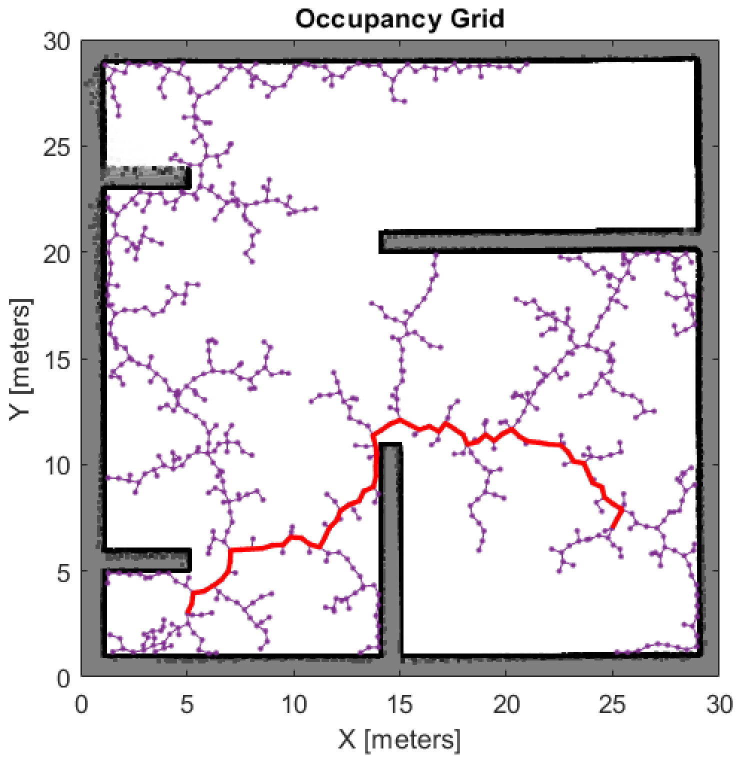


Figure 3.16: The second path planning scenario

For the third testing scenario, we set the starting position to $(5, 3, 0)$ and the goal position to $(5, 27, 0)$. Figure 3.20 illustrates the third path planning scenario.

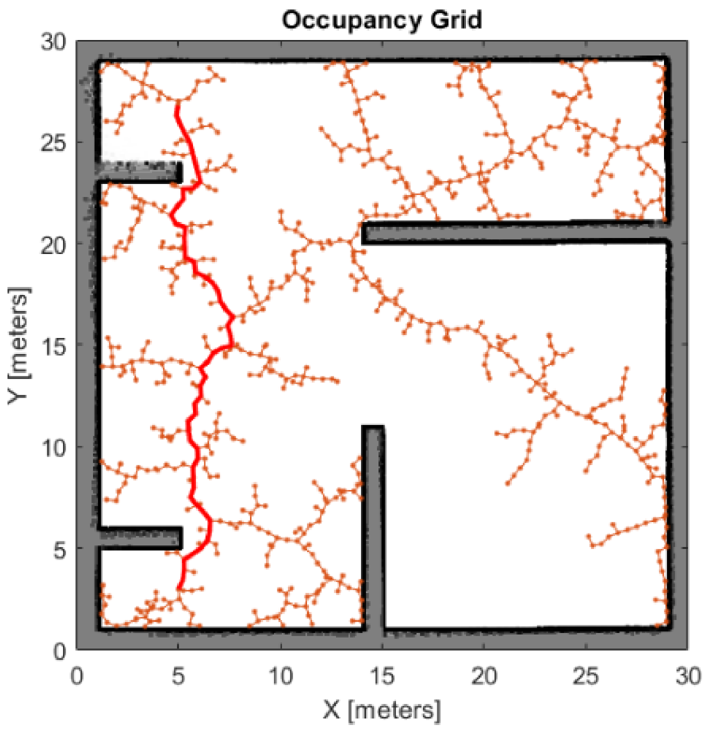


Figure 3.17: The third path planning scenario

3.2.4 Obstacle avoidance

Obstacle avoidance is the task of accomplishing some control objective while adhering to non-intersection or non-collision position constraints.

We used the same path planning scenarios, but this time by adding different obstacles each time and applying RRT* algorithm. We get the following results:

First scenario, with starting goal $(5, 3, 0)$ and goal $(15, 26, 0)$.

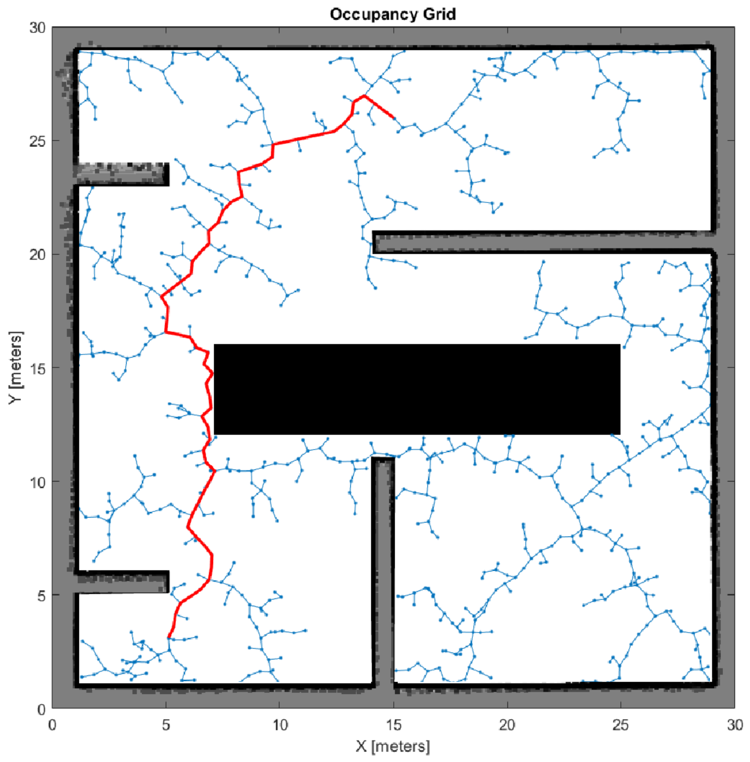


Figure 3.18: The third path planning scenario

Second scenario, with starting goal $(5, 3, 0)$ and goal $(25, 7, 0)$.

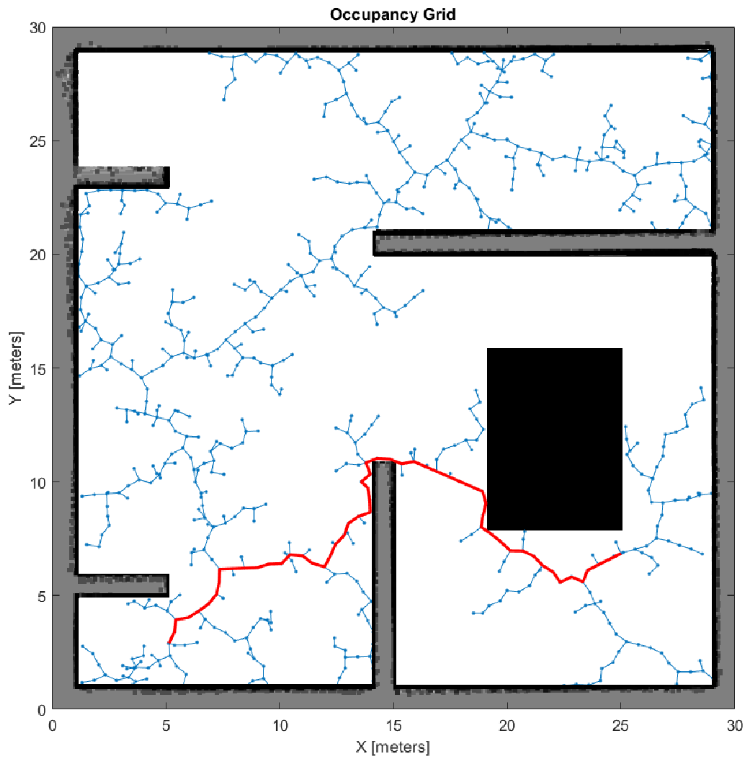


Figure 3.19: The third path planning scenario

Third scenario, with starting goal $(5, 3, 0)$ and goal $(5, 7, 0)$.

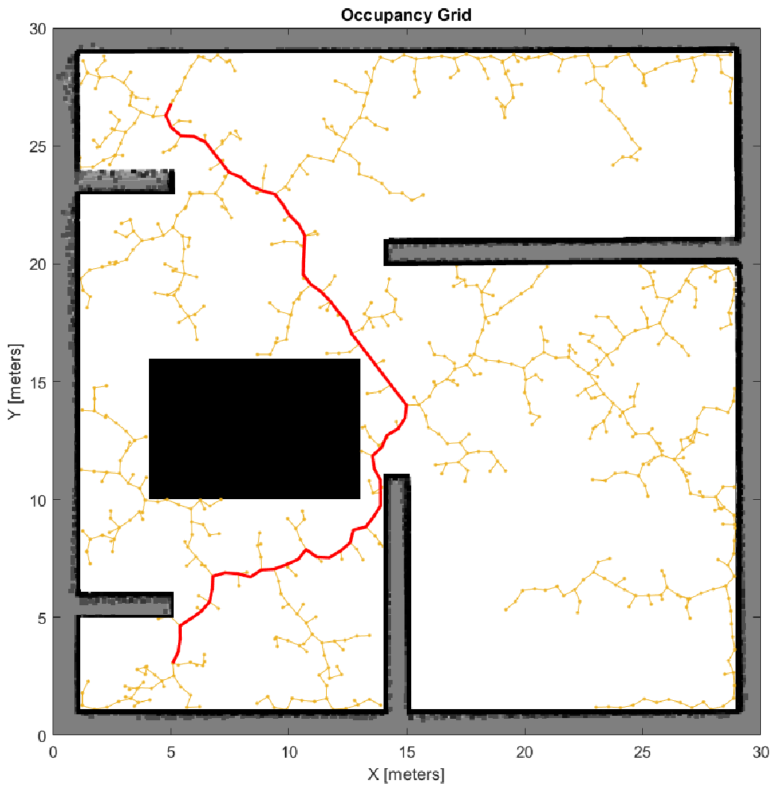


Figure 3.20: The third path planning scenario

3.3 Discussion

Based on the simulation results, many interesting insights can be made:

- The detection angle of the lidar sensor plays an important role in the environment data extraction and the map-building process. Decreasing the lidar range leads to less efficient results since the robot will not be able to detect all the obstacles in its environment.
- Down-sampling the input lidar scan in the SLAM Map Builder app from MATLAB, gave us a good output occupancy map. However, using the full input sensor data will lead to a better and more efficient result, but due to hardware limitations, we down-sampled the lidar scans to 33.3% to be able to perform the simulation.
- The generated paths obtained in the last section using the RRT* algorithm were shown to be good. The algorithm also managed to avoid the obstacles added successfully.

3.4 Conclusion

This chapter illustrated our mobile robot's overall system design and discussed the simulation and the obtained results of each part: environment data extraction using lidar, map building using the SLAM Map Builder app from MATLAB, and path planning using the RRT* algorithm. The following section draws a broad conclusion regarding the entire project as well as discusses proposed future work.

General conclusion and future Work

In this project, we handled two fundamental problems in robotics which are the SLAM problem (Simultaneous localization and mapping) and the path planning problem.

The proposed work aimed to implement a simulation of an autonomous mobile robot in a closed indoor environment using a lidar sensor for environment data extraction, the SLAM algorithm for map building, and the RRT* algorithm for path planning.

In order for the robot to extract data from an unknown environment, we gave it a trajectory to follow on a map that we already have previous information about. While navigating the robot captured data from its surroundings using a lidar sensor. At the end of the process, we got a set of lidar scans with their corresponding odometry poses which we used later on to build a model map of the environment.

To build the model of the environment, we used the SLAM Map Builder app from MATLAB, which utilizes SLAM algorithms to construct a map. The app took the lidar scans and their corresponding odometry data as an input and returned the model map of the environment in a form of a probabilistic occupancy map. The provided map was then used for path planning.

To plan a path within the given environment, we used the RRT* algorithm. The RRT* algorithm assists the robot to navigate from a particular starting location to a goal position in a short period of time by attempting to determine the shortest path toward its destination.

All the functions, algorithms, and tests of this project have been implemented in MATLAB2021a using a laptop with an Intel Core i7 processor and 8Go of RAM.

Future work:

For future work, I aim to develop my project in the following areas:

- Consider dealing with dynamic obstacles.
- Optimize the path obtained by the RRT* algorithm using curve smoothing techniques.
- Improve the environment exploration process by using local path planning algorithms such as the Bug algorithm.
- Extend the work to the outdoor environment.

Bibliography

[1] What are autonomous robots? 8 applications for today's amrs. `https://waypointrobotics.com/blog/what-autonomous-robots/`. Accessed: 2022-05-25.

[2] Owen Holland. Exploration and high adventure: The legacy of grey walter. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 361:2085–121, 11 2003.

[3] Autonomous robot. `https://en.wikipedia.org/wiki/Autonomous_robot_Outdoor_navigation`. Last edited on 25 May 2022, Accessed: 2022-05-25.

[4] Training a robotic arm to do human-like tasks using rl. `https://medium.datadriveninvestor.com/training-a-robotic-arm-to-do-human-like-tasks-using-rl-8d3106c87aaf`. Last edited: February 28, 2019, accessed: 2022-05-26.

[5] Autonomous mobile robots could change metal fabrication. `https://www.thefabricator.com/thefabricator/article/shopmanagement/autonomous-mobile-robots-could-change-metal-fabrication`. Last edited: April, 16,2022 , accessed: 2022-05-26.

[6] Autonomous underwater vehicle. `https://en.wikipedia.org/wiki/Autonomous_underwater_vehicle`. Last edited on 24 February 2022, Accessed: 2022-05-25.

[7] Navatics mito underwater drone. `https://www.cined.com/navatics-mito-underwater-drone/`. Last edited: January 24, 2019, accessed: 2022-05-26.

[8] Autonomous navigation. `https://www.accessscience.com/content/autonomous-navigation/YB000130`. Article By: Michelson, Robert C. School of Aerospace Engineering, Georgia Institute of Technology, Smyrna, Georgia, last reviewed:2000. Accessed: 2022-05-26.

[9] Choon Wong, G. Seet, and Siang Sim. Multiple-robot systems for usar: Key design attributes and deployment issues. *International Journal of Advanced Robotic Systems*, 8, 03 2011.

[10] Autonomous navigation, part 1: What is autonomous navigation? `https://www.mathworks.com/videos/autonomous-navigation-part-1-what-is-autonomous-navigation-1592993748308.html`. Accessed: 2022-05-24.

[11] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[12] Lidar. `https://en.wikipedia.org/wiki/Lidar`. Last edited on 28 May 2022, Accessed: 2022-06-5.

[13] Lidar: The eyes of an autonomous vehicle. `https://medium.com/swlh/lidar-the-eyes-of-an-autonomous-vehicle-82c6252d1101`. Last edited on 12 September 2019, Accessed: 2022-06-4.

[14] The electromagnetic spectrum. `https://imagine.gsfc.nasa.gov/science/toolbox/emspectrum1.html`. Last edited on March 2013, Accessed: 2022-06-21.

[15] Shoudong Huang and Gamini Dissanayake. Robot localization: An introduction. *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 1–10, 1999.

[16] Wolfram Burgard, Frank Dellaert, Dieter Fox, and Sebastian Thrun. Monte carlo localization for mobile robots. 1999.

[17] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1, 2002.

[18] Topological map. `https://en.wikipedia.org/wiki/Topological_map`. Last edited on 29 May 2022, Accessed: 2022-06-5.

[19] Alberto Poncela, Eduardo J Perez, Antonio Bandera, Cristina Urdiales, and Francisco Sandoval. Efficient integration of metric and topological maps for directed exploration of unknown environments. *Robotics and Autonomous Systems*, 41(1):21–39, 2002.

[20] Tube map. `https://en.wikipedia.org/wiki/Tube_map`. Last edited on 5 June 2022, Accessed: 2022-06-6.

[21] Occupancy grid mapping. `https://en.wikipedia.org/wiki/Occupancy_grid_mapping`. Last edited on 21 February 2022, Accessed: 2022-06-6.

[22] Mobile robot modelling and navigation in an unstructured environment. `https://matlabexamples.wordpress.com/2021/07/11/mobile-robot-modelling-and-navigation-in-an-unstructured-environment/`. Accessed: 2022-06-19.

[23] Sean Campbell, Niall O'Mahony, Anderson Carvalho, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Path planning techniques for mobile robots a review. In *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, pages 12–16. IEEE, 2020.

[24] Anis Koubâa, Hachemi Bennaceur, Imen Chaari, Sahar Trigui, Adel Ammar, Mohamed-Foued Sriti, Maram Alajlan, Omar Cheikhrouhou, and Yasir Javed. *Robot path planning and cooperation*, volume 772. Springer, 2018.

[25] Norlida Buniyamin, W Wan Ngah, Nohaidda Sariff, Zainuddin Mohamad, et al. A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*, 5(2):151–159, 2011.

[26] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[27] Abu Bakar Sayuti HM Saman and Ahmed Hesham Lotfy. An implementation of slam with extended kalman filter. In *2016 6th International Conference on Intelligent and Advanced Systems (ICIAS)*, pages 1–4. IEEE, 2016.

[28] Luciano Buonocore, Cairo L. Nascimento, and Areolino de Almeida Neto. Solving the indoor slam problem for a low-cost robot using sensor data fusion and autonomous feature-based exploration. In *ICINCO*, 2012.

[29] Taigo Maria Bonanni, Giorgio Grisetti, and Luca Iocchi. Merging partially consistent maps. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 352–363. Springer, 2014.

[30] [slam] graph-based slam (pose graph slam). `http://jinyongjeong.github.io/2017/02/26/lec13_Least_square_SLAM/`. Last edited on 26 February 2017, Accessed: 2022-06-9.

[31] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE, 2011.

[32] Autonomous navigation, part 4: Path planning with a* and rrt. `https://www.mathworks.com/videos/autonomous-navigation-part-4-path-planning-with-a-and-rrt-1594987710455.html`. Accessed: 2022-06-12.

[33] Design, simulate, and deploy path planning algorithms. `https://www.mathworks.com/discovery/path-planning.html`. Accessed: 2022-06-12.

[34] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483. IEEE, 2011.

[35] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[36] Motion planning. `https://en.wikipedia.org/wiki/Motion_planning`. Accessed: 2022-06-12.

[37] Jiping An, Xinhong Li, Zhibin Zhang, Guohui Zhang, Wanxin Man, Gangxuan Hu, and Junwei He. Path planning for self-collision avoidance of space modular self-reconfigurable satellites. *Aerospace*, 9(3):141, 2022.

[38] Fatin Hassan Ajeil, Ibraheem Kasim Ibraheem, Ahmad Taher Azar, and Amjad J Humaidi. Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. *Sensors*, 20(7):1880, 2020.

[39] Sampling-based motion planners. `http://www.kumarakshay.me/sampling-mp/`. Accessed: 2022-06-15.

[40] Burak Boyacioglu and Seniz Ertugrul. Time-optimal smoothing of rrt-given path for manipulators. In *ICINCO (2)*, pages 406–411, 2016.

[41] Slam map builder. `https://www.mathworks.com/help/nav/ref/slammapbuilder-app.html`. Accessed: 2022-06-17.

[42] Ankit Deo, Ayush Gupta, Himanshu Khemani, and Rashmi Ranjan. Path tracking mobile robot using steppers. *E3S Web of Conferences*, 87:01028, 01 2019.