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

University M’Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Power and Control

Final Year Project Report Presented in Partial Fulfilment of the Requirements of the Degree of

MASTER

In **Electronics**

Option: **Control**

Title:

Path Planning of a Mobile Robot in an Indoor Environment using Dynamic Replanning and RRT Star Algorithm

Presented By:

- **DJEBBAR Amira**

Supervisor:

- **Ms. HACHOUR Ouarda**

Registration Number:……../2021

Acknowledgements

I would love to appreciate Miss Hachour Ouarda's patience and efforts to provide guidance and correction during the execution of this project.

I also would like to show my greatest appreciation to all the teachers and colleagues who in one way or another encouraged me during the entire process.

Dedication

I would love to express my total appreciation to the Almighty Allah for providing me with knowledge, power, and most importantly, for keeping me faithful to continue doing what I am doing even when I was at my lowest situations.

I would love to express my deepest appreciations and thanks to my parents and sister Sabrina, I owe them a big dept for standing beside me and showing me support, and still, during all my academic curriculum and my entire life. May Allah help me to give them back even a small portion of what they gave me during all my life.

I also would love to thank my very best friends, Manel and Roumaissa, for being the best friends I have ever had, and still having, and for being always next to me although we are far apart in different places

Table of Content

Table of Content

List of Figures

List of Tables

List of Abbreviations

C-space	Configuration Space
DOF	Degrees of Freedom
LIDARS	Light Detection and Ranging Sensors
GPS	Global Positioning System
AUV	Arial Underwater Vehicle
RRT	Rapidly exploring Random Trees
RRT*	Improved Rapidly exploring Random Trees
qinit	The initial sample point
qrand	The random point
qnearest	The nearest sample point
qnew	The new sample point
Qnear	The set of the neighboring nearest nodes
qrobot	The current robot position
cmin	The minimum cost
Xfree	The free space
G	The tree Graph
V	The tree vertex
E	The tree Edge

Abstract

Mobile robots are currently being introduced in many fields such as hospitality, healthcare, manufacturing, transportation, delivery. And research in the development of mobile robots is heading towards increasing their level of autonomy in performing different tasks of different levels of complexity. One of these tasks is path planning which is taking a significant amount of importance during the last decade, this is because mobile robots nowadays are requiring more intelligence in order to plan feasible paths in various types of environments. The sampling-based method RRT* is one of the most efficient algorithms to perform path planning because of their many advantages such as the adaptability with complex and large environments. This project introduces a path planning method for a ground wheeled mobile robot in an indoor environment in the presence of random obstacles using RRT* and Dynamic Replanning. the robot is uninformed about the obstacle’s positions, but informed about its static environment. and it combines both global and local path planning in order to achieve the goal. The results demonstrate successful execution of the algorithm using the same number of tree nodes.

General Introduction

The field of robotics has encountered a significant amount of interest in the research and the business industries alike. Robots became and will become a necessity in our daily lives, because they take part of every aspect of it, health, transportation, construction, delivery to mention few. Specifically, mobile robots perception, mapping, navigation and control, is taking a significant amount of research nowadays, especially that these types of robots are being considered as the next revolution in many influential domains such as the transportation industry, the delivery industry, where mobile vehicles of all types are being developed in order to achieve as much autonomy, safety and energy efficiency as possible. Mobile robots' navigation is one of the most active research areas in robotics, it includes four main blocks which are; localization, path planning, trajectory planning and path following[10]. Localization is concerned with driving the robot to answer the question where am I? Path planning is more concerned with answering the question how am I going to achieve the goal in a complete, optimal path?

On the other hand, trajectory planning is concerned with optimizing the found path to achieve minimum energy or minimum time constraints, or sometimes both constraints. Path following is concerned with the control part of the robot and how to align the planned path with the controller signals and the kinematics of the robot. While taking a look at the previously mentioned navigation blocks , it becomes clear that ensuring successful path planning is the main building block of a successful navigation process. Which driven motivation to set focus of this project on path planning in an unknown environment of a ground wheeled mobile robot.

The path planning problem is defined as the concern of finding a feasible obstacle free path that will drive the robot from a start position to a goal position and respects the costs and time metrics. Depending on the environment and application at hand , various path planning methods where proposed to achieve a solution for this problem, A*[1] which is graph based method that was used to achieve a feasible path based on a graph representation of the environment and calculation of the costs of each cell. On the other hand, potential field methods [2] [3] [4] [12] were introduced as a faster method to achieve minimum cost and time path from start to goal , through assuming the obstacles and goal configurations as potentially charged objects in the map. However, these methods suffer from local minima and they may not be always applicable when environments are unknown or when obstacles are moving. Similarly , many researches considered sampling based algorithms , among which we mention Probalistic Road Maps (PRMs) [5] [6] and Rapidly Exploring Random Trees (RRTs) [6]. Sampling based algorithms overcome the previously mentioned methods in terms of performance, because they consider the environment as a discretized workplace, where each cell in is a node . the linkage of these nodes produces a tree that is further optimized in order to achieve optimality.

One of the most efficient sampling based algorithms is RRT*[8] [6] (pronounced RRT Star) , this algorithm is an extension of RRT algorithm which samples the free space in the configuration space into a tree of nodes that are rewired and further optimized in order to achieve path that is converging to an optimal one.

General Introduction

RRT * was widely used in static known environments, where the robot is aware of its environment, and is aware of the placement of the obstacles. However, operational real world environments are rarely static and obstacles are usually random events that the robot can not expect. Which drives necessity into introducing dynamism into the algorithm. For this reason, path planning and replanning in an unknown indoor map environment for a wheeled mobile robot is addressed in this project.

Previous efforts addressing this problem were done with the purpose of achieving a method to store the path waypoints as was done in ERRT [6]. In addition, DRRT[7] was proposed to place the root of the tree at the goal location, so that only a small number of branches may be lost or invalidated when replanning .

The contribution of this project is to combine global and local path planning using RRT* and dynamic replanning in order to plan a sub-optimal path of a ground mobile robot in an indoor environment. The robot does not have any prior knowledge about the location of obstacles, it is only informed about its start and goal locations and the static environment in which it is moving.

This report is organized into three chapters, the first one highlights basic knowledge about mobile robots and path planning. While the second chapter dive deeper into the various and widely used path planning methods under each environment representation. The third chapter provides the path finder strategy and the simulation results, in addition, it discusses the accumulated results. The last part of this report concludes the project with a general conclusion and a proposed future work.

Chapter I :
An Overview of Mobile Robots and Path Planning

1.1 Introduction

Nowadays, the world is going towards a peak in the revolution of mobile robotics. A variety of robotics systems are being developed and are showing their effectiveness in performing simple and complex tasks in different environments, few examples include Amazon delivery robots, manufacturing laboratories robots, exploration robots.

Ensuring successful application of tasks requires applying some kind of intelligence into the mobile robot, in order to achieve near-optimal execution of the missions in terms of time, energy and safety. Embedding this intelligence rises many research problems in mobile robotics, one of which is the path planning problem, which is considered as a sub-problem of the navigation problem.

This chapter provides a general overview about mobile robots and path planning, it presents the basic knowledge points about the topic.

1.2 Basic terms and definitions

1.2.1 Mobility

The word " Mobile" takes its roots from the Latin word " mobilis" . which means the ability to achieve Locomotion, the capacity to move and navigate an environment actively, without explicit need of the human operator. This implies that the mobile system is able to move actively around its environment using a certain level of autonomy with the help of some apparatus, such as sensors, in order to collect data for proper navigation of the environment and decision making.

1.2.2 Autonomy

Robots' autonomy implies that the robot is capable of performing its tasks, such as detecting objects or arriving at certain points in time, by means of sensors or cameras and of converting this information into movement without the need of an external human intervention.

1.3 Configuration Space

The Configuration of a robot is defined by all the position points that the robot takes.

The Configuration Space, or as called the C-Space, Defines the n-dimensional space that gathers the set of all possible configurations of a robot . This means that for every configuration of the robot, there is a corresponding unique point in the C-space, and for every point in the C-Space, there is a governing unique configuration of the robot [9] . In addition, configuration spaces have three characteristics which are:

1.3.1 The degrees of freedom: or DOFs, which are the number of real valued points needed to represent the configuration of the robot. as a global rule, a rigid body in three-dimensional space has six degrees of freedom while in two-dimensional space has three degrees of freedom.

1.3.2 The topology: which specify the shape of the configuration space, where two configuration spaces are said to be topologically equivalent if one can be deformed into the other without cutting or gluing.

1.3.3 The representation: points in the configuration space can be represented implicitly, through choosing an n-coordinate frame to represent the configuration in terms of latitude, longitude and orientation for example. Or we can represent the configuration of a robot explicitly, through viewing the n-dimensional space as embedded in a Euclidean space of more than n-dimensions, just as a two-dimensional unit sphere can be viewed as a surface embedded in a three-dimensional Euclidean space.

1.4 The Workspace and Task space

The space where the robot can do its specified task is known as the task space, while defining this space, we take into consideration only the task that the robot has to perform.

On the other hand, the workspace is a specification of the configurations' reachability of the end effector of the robot, which means that we express it in terms of the robot's structure only[9]

The above two definitions imply the following remarks:

- Both the workspace and the task space relate to the configuration of only the end effector of the robot.
- Choosing the workspace and the task space of a robot is a matter of choice, and it is independent from the C-Space of the robot.
- Not all points in the task space are reachable by the C-space. However, all points in the workspace are achievable by at least one configuration of the robot [9]
- Mechanisms that have either the same or different workspaces, can have different C-spaces.

1.5 Holonomic and non-holonomic mobile robots

These two terms define the imposed constrains on the movement of the mobile robot
Holonomic robots have the ability to move in any direction of the three dimensional space (x, y, θ) , where " x" and " y" refer to the position coordinates and " θ " refer to the orientation coordinate.
On the other hand, non-holonomic mobile robots are restricted in motion and can not move in all directions, For example, a car that drives straight cannot turn right at once, it has to turn for a while until it is headed in the new direction and then start driving straight all over again.

1.6 Kinematic and Dynamic Models

1.6.1 kinematic Models

they describe the geometric relationships that are present in the system [9], meaning the relationship between the control input and the behavior of the systems using state-space representation.

Looking for kinematic models , we are not interested by knowing the effect of the forces, torques. Instead, we are interested by the effect of velocities that are represented in terms of first order differential equation.

1.6.2 Dynamic Models

On the other hand , looking for this type of models means describing a system motion when forces are applied to the system [9]

While looking for this model, we are interested in the causes of robot’s motion such as forces, torques, inertia, system mass .

Knowing the kinematic models of a robot is sufficient for designing Locomotion strategies for wheeled mobile robots. While other types of robots such as walking, air, space robots, having the dynamic models is needed.

1.7 Types of mobile robots

According to the type of mobility, robots can be classified into:

- **Stationary (arm/manipulator)**
  These are industrial robots that provide solutions for performing tasks that are considered often cumbersome for humans[10], among these tasks we can mention, welding, assembling, machining, grasping. Few examples of these kinds of robots are Abb IRB 6640 robot series , which are designed for heavy lifting payloads , and mostly suitable for welding applications , and KR 4 Agilus which is a manipulator robot designed to perform tasks related to handling and assembly of electronic parts. Both robots are shown in Figure 1.1.

(a) **(b)**

Figure1.1 : (a). Abb IRB 6640 robot series[18] **(b).** KR 4 Agilus is a manipulator robot [19]

- **Land-based**

  These types of robots are a very important and widely included in many research areas nowadays, such as in autonomous intelligent vehicles, which rely on aspects like pattern recognition and image processing.

  Land based mobile robots can be walking robots or wheeled robots [10], examples of these types of robots are S5.2 outdoor security and inspection wheeled mobile robot and Atlas humanoid robot , both are shown in Figure1.2

(a) **(b)**

Figure1.2: (a). S5.2 outdoor security and inspection robot [21], **(b).** Atlas humanoid robot [20]

- **Air-based mobile robots or AI** :
  inspired from airplanes operations, unmanned aerial robots or known as drones, are flying robots that perform pre-programmed tasks with or without human intervention, one example of these kinds of robots is the ARSI – aerial robot for sewer inspection that is shown in Figure 1.3.

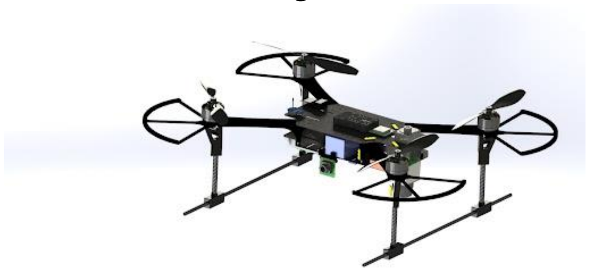
Figure1.3: ARSI, aerial robot for sewer inspection[22]

In addition to the above-mentioned types of mobile robots, there are many more robot types such as Submarine robots, which are one of the most rich research topics nowadays, and that is due to their high contribution to explore underwater areas that are inaccessible to humans. Hybrid robots are another extent to these kinds of robots, they combine two or more robot types in order to achieve flexibility in tasks performance.

1.8 Path Planning

To ensure successful navigation of the robots' environment, three main questions must be answered by the mobile robot, which are where am I ? where am I going? How I am going to get there [9] . These questions are answered through three main functions:

a. Localization: which relates the robot to the environments for which it exists, through providing the necessary knowledge about its current location at any instance of time with the help of sensors, various sensing methods are used for achieving successful navigation including GPS in outdoor environments, Ultrasonic sensors, LIDARS to mention few. And the corresponding position is expressed to the robot as a topological reference, for example (Room 78), or as an absolute coordinate such as latitude, longitude, altitude.

b. Mapping: this function helps the robot acquire knowledge about the environments it is moving in, its boundaries and corners, its shape as an example. The map can be an offline one where it is embedded in the robots' memory, or can be an online map where the robot has to build it gradually while moving in the specified environment.

c. Path Planning: this is the function that answers the question how I am going to get there? In order to achieve it, the robot must know its goal position and avoid the static or moving obstacles being present in between the start and the goal positions. Research in path planning is being driven to making the robot generates the most efficient, accurate and safe paths.

The path planning problem was defined in general in [10] given the following variables:

- $A \subset W$: The robot, it is a single moving rigid object in the world W represented in the Euclidean space as R^2or R^3.
- $O \subset W$: The obstacles space, which are considered as stationary rigid objects in W.
- 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 position 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. The quality of the path (optimal or not) is measured using a set of optimization criteria such as shortest length, runtime, or energy. Where:

This definition of the path planning problem is a general one, since the robot is not always aware of the position of the obstacles.

Given the mission to be achieved, the path planning problem is classified into different types based on the reference being looked at.

- Based on the types of obstacles, it is divided into static, where the obstacles positions are well known to the robot and do not change position or shape. And dynamic path planning where the obstacles are either random or random and moving ones.
- Based on the algorithm, we can have global path planning or local path planning. Where in global path planning the algorithm defines the environment of the robot and how to get to the goal from start to end positions prior to the movement of the robot, in other words, the robot navigates in a known environment.

  While in local path planning the robot plans its path while moving in an unknown environment. It has knowledge about its current and goal position, but no details about the environment or the obstacles positioning.

- Based on completeness, the planned path can be exact where the algorithm finds an exact optimal solution for the robot if one exists or proves that no optimal solution is present, while the heuristic algorithms search for a good quality, sub-optimal solution in the shortest amount of time.

1.8.1 Environment representation for path planning purposes

based on the global and local classification of path planning problems, various methods are used in order to plan a feasible path from a start to a goal position in an efficient way. Among which fall under three main categories:

a. Graph representation

These methods divide the robots' environment into a grid of cells, and apply real-time search algorithms on each grid cell in order to find the path from the start to end position. One drawback of graph-based path planning is that while the environment is being explored the graph is being reconstructed, further processing is needed to extract the path from the graph, which is considered as consuming a significant amount of time and processing resources.

b. Sampling based representation

Environments described by a large number of obstacles may result in an excessive computational burden. The idea to develop sampling-based algorithms came to avoid the computational burden associated representing the environment as a graph. One famous algorithm underlying this category is the Rapidly Exploring Random Trees (RRT) algorithm family.

c. Potential based representation

These methods consider the environment as a summation of attractive and repulsive potentials, where the attractive potentials are that of the goal position, and the repulsive potential are those of the obstacles, the way these methods work is by considering the robot as a particle with an alike potential to that of the obstacles and an opposite to that of the goal, where when the robot is moving will be attracted to the goal and repulsed from the obstacles. A well-known drawback of these methods is the local minima, which affects negatively the responsiveness of the robot to avoiding the obstacles.

1.9 Conclusion

This chapter presented a general knowledge about mobile robots, and crossed through the path planning problem and its different classifications.

The coming chapter will focus on the different widely used path planning algorithms, with a emphasize on RRT* algorithm, being considered as the main algorithm to solve the path planning problem presented in this project.

Chapter II:
Path Planning Methods

2.1 Introduction

Planning a robot’s path means finding a continuous obstacle free and optimal path that connects the start to the goal position.

Ideally, path planning is done considering static environments where obstacles and targets are stationary. However, the reality imposes that in most cases obstacles are random objects, and sometimes even targets are moving. Hence, the development of several approaches to plan the robots’ path in an unknown environment is necessary .

Regardless of the robots’ type, and regarding the problem of path planning, the used approaches are divided into Graph based methods, Sampling Based methods and potential based methods, Intelligence based methods.

The combination of the algorithms underlying the previously mentioned approaches, resulted in the development of various planning methods depending on the robot type and the planning environment, to mention few , [3] considered path planning of underwater AUV robot in dynamic environments through the combination of bidirectional-RRT method and Artificial Potential fields method along with dynamic window approach. On the other hand, .[5] considered using probabilistic roadmaps to plan a path in high dimensional spaces. And [7] considered real time path planning in a dynamic environment. [12] considered path planning of a rover mobile robot using a combination of A* and artificial potential fields method. While others proposed including artificial intelligence methods such as neural networks and deep learning algorithms[14].

This chapter provides a background of the general and mostly used methods for path planning.

2.2 Path Planning methods

2.2.1 Graph-based methods

These methods use a graph representation of the robots’ workspace in order to find the best path to be followed. Generally, the accomplishment of the search begins by the start cell then it extends the searching the neighboring cells. Along this process the explored cells are divided into closed list cells and open list cells, where the closed list cells are the already explored ones, and the open list cells are the ones not yet explored.

The variation between the different graph-based algorithms occur in the way cells are checked, and the calculations of the cost functions.

In Addition, graph based algorithms can be informed[1] or as called heuristic search, where they contain information about the cells, or non-informed where they contain information about the problem description only, and do not distinguish promising cells from non-promising cells.

A*, pronounced A star, is one of few examples of graph-based methods, it is an informed algorithm, since it contains information about the cells or heuristics, where the heuristic is the cost estimate from the current cell to the goal cell for the part that has not been explored yet. This algorithm is also complete and guarantees an optimal path, however, it is memory and time expensive. Figure 2.1 demonstrates the working process of A* .

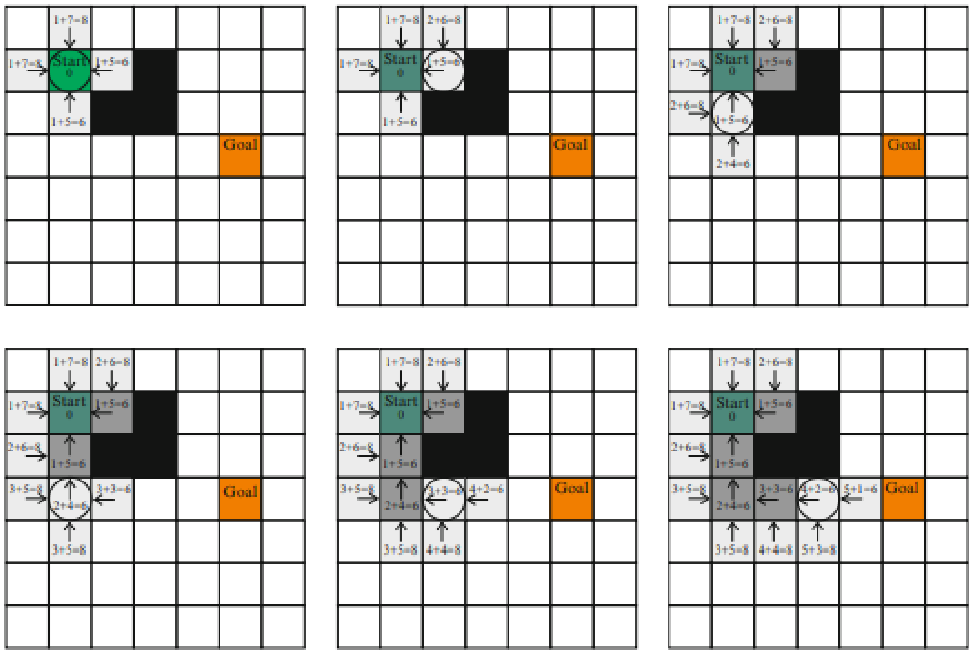

Figure2.1: A* Algorithm [9]

2.2.3 Potential-based methods

Represented in [4], this method considers the robot as an electrically-charged particle. Thus, the obstacles should contain the same electrical charge as the robot, in order to repel it , and the target should contain a different electrical charge than that of the robot in order to attract it to the goal.

Graphically, the obstacles are shown as high peaks as shown in Figure 2.1, while the goal is given a low value as shown in Figure 2.2.

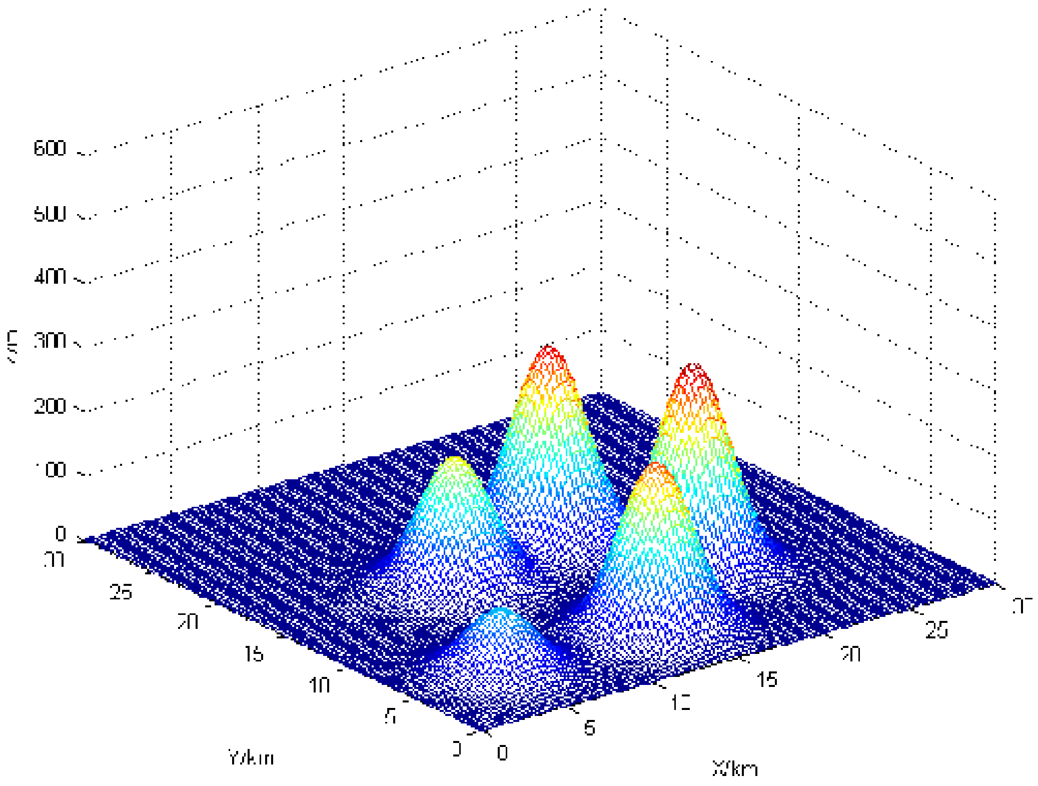

Figure2.2: potential field due to obstacles[15]

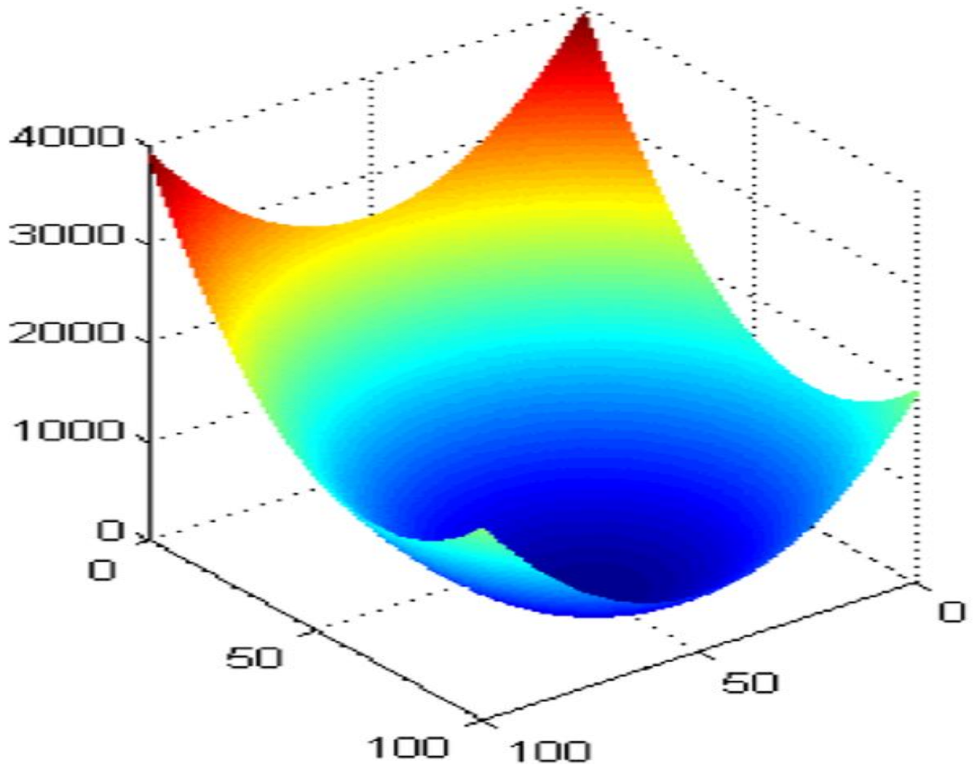

Figure2.3: potential field due to the goal[14]

2.2.2 Sampling-based and Optimal sampling-based algorithms

Using these methods require the choice of random points to be sampled, as well as the application of collision detection methods for the purpose of verifying if these random points fall in the free space. Sampling-based methods have the advantage of presenting the configuration space independently from the environments' geometry, where a path planning solution can be obtained. Figure2.4 demonstrates how sampling based methods sample the nodes and how they incrementally connect them to find a feasible path.

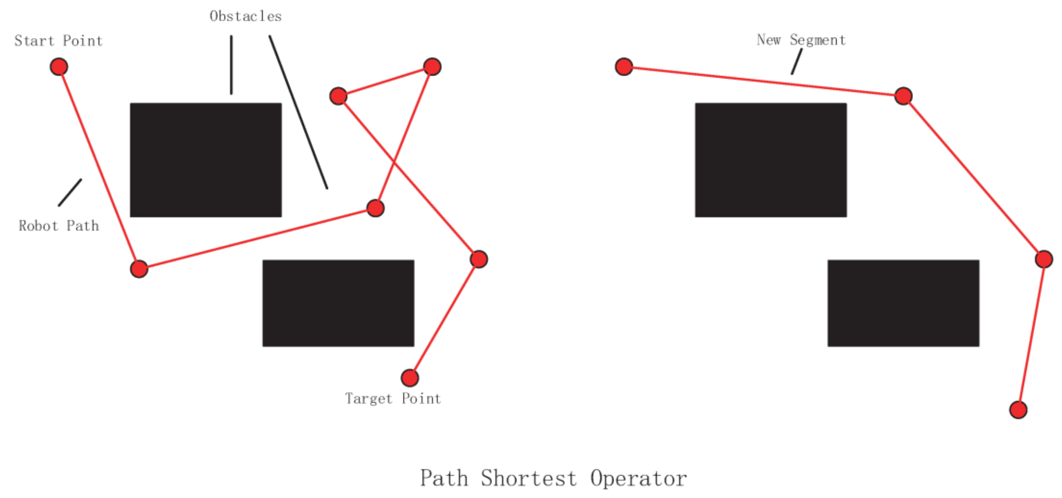

Figure 2.4: The working principle of sampling-based algorithms[16]

- ### Rapidly exploring random trees (RRT)

Rapidly exploring random Tree, was initially built for executing tasks for single query applications. Basically, The algorithm incrementally produces a tree of feasible trajectories through the steps listed below and demonstrated through Figure

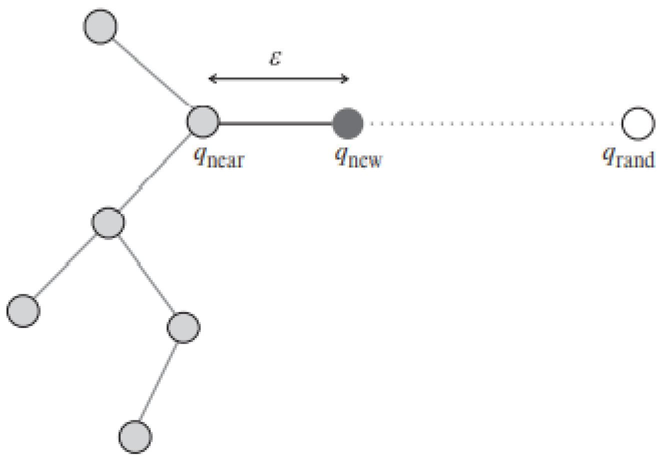

Figure 2.5: The working process of RRT algorithm[9]

- The start of the algorithm is a graph that includes the initial state $qinit$ as its single vertex.
- At each incrementation, a point $qrand \in Xfree$ is sampled.
- Afterwards, a connection intention is made to relate the nearest vertex $v \in V$ in the tree to the new sample $qnew$ within the validation distance ε .
- In case no obstacle exists in the space of the new connection, then $qrand$ is added to the vertex set, and $(v, qrand)$ is added to the edge set.
- Once the algorithm termination criterion is satisfied, a check is made on the goal point to determine whether it can be connected to the tree.

A formulation of RRT algorithm is shown in Algorithm 1 and a demonstration illustrating the resulting tree expansion is shown in Figure 2.6 .

Algorithm1: *RRT*

```
1  V ← {qinit}; E ← ∅;
2  for $i = 1, \ldots, n$ do
3      $qrand \leftarrow SampleFree(i)$ ;
4      $qnearest \leftarrow Nearest(G = (V, E), qrand)$;
5      $xnew \leftarrow Steer(qnearest, qrand)$ ;
6      if $ObtacleFree(qnearest, qnew)$ then
7          $V \leftarrow V \cup \{qnew\}; E \leftarrow E \cup \{(qnearest, qnew)\}$ ;
8  return $G = (V, E)$;
```

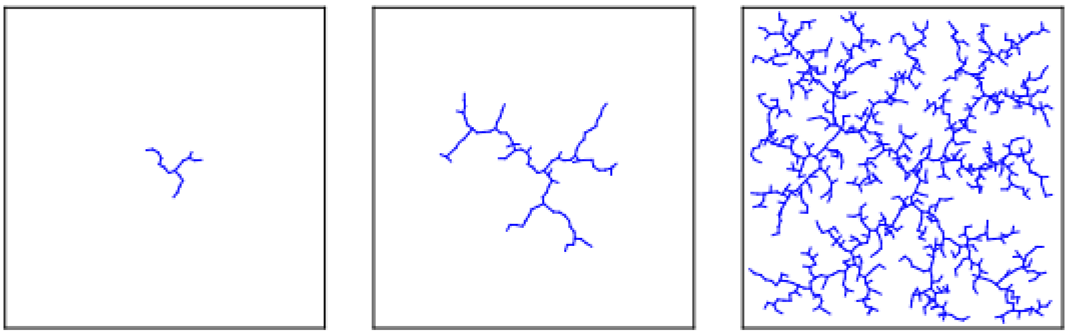

Figure2.6: a sample simulation of RRT method tree expansion [9]

We derive from the above algorithm the following primitive functions that RRT algorithm rely on:

- **The Sampling function:** shown in Algorithm1 as $SampleFree()$, this procedure samples random points from the obstacle-free space.
- **Nearest-Neighbor:** Represented as $Nearest()$ in Algorithm1.this function returns the nearest sample to the random point based on the cost that is calculated by the Euclidean distance,
- **The Steering Function:** represented as *Steer()* , this function drives the direction between the random and the nearest point, given the new sample point between *qrand* and *qnear*.
- **Checking for Collisions:** represented by $ObtacleFree()$ in Algorithm1, this function checks for collisions between two states using the Euclidean distance in order to ensure that the path is not included in an obstacle region.

• RRT Star

It was proven in [5] that RRT planner is not asymptotically optimal nor it is computationally efficient, which gave rise to the development of RRT Star (RRT*). This algorithm achieves asymptotic optimality within minimum time and memory through properly choose the right parent node at each step as well as constantly deforming the tree through the rewiring step . Algorithm 2 shows the steps that this algorithm takes in order to achieve the goal.

Algorithm2: ***T = (V, E) ←RRT*(qinit)***

```
1  $T \leftarrow InitializeTree()$
2  $T \leftarrow InsertNode(\emptyset, qinit, T)$
3  for $k \leftarrow 1$ to $N$ do
4      $qrand \leftarrow RandomSample(k)$
5      $qnearest \leftarrow NearestNeighbor(qrand, Qnear, T)$
6      $qmin \leftarrow ChooseParent(qrand, Qnear, qnearest, \Delta q)$
```

```
7       $T \leftarrow InsertNode(qmin, qrand, T)$
8       $T \leftarrow Rewire(T, Qnear, qmin, qrand)$
9   end
```

The primitive procedures of RRT* include:

- **Insert Node :** represented in Algorithm2 as *InsertNode*(), this function adds a new node the tree , and connects it to an already existing node that is considered as a parent node. Along with making this connection, a cost is associated to the new node which is considered as the summation of the costs of the parent node and the Euclidean distance cost between this new node and the parent node.
- **Calculating the cost:** the cost between two states is estimated using the Euclidean distance in order to decide whether or not the random point should be added to the tree.
- **The choice of the parent:** the $ChooseParent$ *()* function selects the best parent along the nodes in the neighborhood set $Qnear$, where $qmin$ is considered the neighbor having the minimum cost, and $cmin$ is the current best cost of reaching the new random sample $qrand$ through making $qnearest$ a parent.

  The cost $cmin$ is considered as the summation of the costs of the nearest point *qnearest* and the cost of the random point *qrand*, and is calculated through the Euclidean distance.

  using the *Steer()* function shown in step 4 of Algorithm 3, a path from $qnear$ to $qrand$ is returned after performing a search in the neighborhood of $qrand$. If this path is obstacle free and its associated cost is less than $cmin$, then the new neighbor is considered the best neighbor and its cost is the best cost. The procedure from step 3 to 12 of algorithm 2 is repeated on all the nodes in the set *Qnear*. So that in the end , the best neighbor to be considered as the parent is returned.

- **Rewiring :** the *Rewire()* function demonstrated in Algorithm 4, deforms the tree structure depending on the newly chosen node *qrand*, using the set of neighborhood nodes *Qnear*.

  Again, using the *Steer()* function (steps 2 to 5 in Algorithm 4) , if the path between the new node *qrand* and the current chosen parent *qnear* less than the total cost of this path, then the new node *qrand* is a better parent. At this point the tree is rewired using the *ReConnect ()* function, which removes the edge to the current parent *qnear* and adds one to the new node *qrand* that makes it a parent of *qnear*.

Algorithm 3: $\boldsymbol{qmin} \leftarrow \boldsymbol{ChooseParent}(\boldsymbol{qrand}, \boldsymbol{Qnear}, \boldsymbol{qnearest}, \boldsymbol{\Delta q})$

```
1   $qmin \leftarrow qnearest$
2   $cmin \leftarrow \mathrm{Cost}(qnearest) + c(qrand)$
3   for $qnear \in Qnear$ do
4       $qpath \leftarrow \mathrm{Steer}(qnear, qrand, \Delta q)$
5       if $ObstacleFree(qpath)$ then
6           $cnew \leftarrow Cost(qnear) + c(qpath)$
7           if $cnew < cmin$ then
8               $cmin \leftarrow cnew$
9               $qmin \leftarrow qnear$
10          end
11      end
12  end
13  return $qmin$
```

Algorithm 4: $\boldsymbol{T} \leftarrow \boldsymbol{Rewire}(\boldsymbol{T}, \boldsymbol{Qnear}, \boldsymbol{qmin}, \boldsymbol{qrand})$

```
1   for $qnear \in Qnear$ do
2       $qpath \leftarrow Steer(qnear, qrand, \Delta q)$
3       if $ObstacleFree(qpath)$ and $Cost(qrand) + c(qpath)$
                    $< Cost(qnear)$ then
4           $T \leftarrow ReConnect(qrand, qnear, T)$
```

```
5        end
6    end
7    return $T$
```

2.3 Conclusion:

This chapter covered the mostly used path planning methods depending on the environment representation, in addition, focus was set on the sampling-based algorithm RRT*, being considered as the main building block of the proposed path finder strategy, which is further explored in the following chapter .

Chapter III
Simulation Results and Discussion

Chapter III : Simulation Results and Discussion

3.1 Introduction

The proposed path finder strategy focused on using RRT* algorithm in order to plan and dynamically replan a new free-obstacle path once a random obstacle is found during the way In other words, both global and local path planning are combined in order to achieve the goal.

Another main contribution is that the new path calculation is done using the same existing search tree, without the need of regenerating a new tree each time the robot encounters an obstacle crossing its way .

Various scenarios are simulated using MATLAB, and discussed using in the upcoming sections.

3.2 The Path finder Strategy

The robot has knowledge about its starting position and its goal position as well as the static environment it is moving in. However, the details of the environment are kept blinded to the robot.

3.2.1 Executing the path

Using RRT* algorithms 2,3 and 4 stated above, the robot executes the optimal path found by the search tree during the initial planning process

In order to move through the optimal path nodes, the robot chooses the next node and the needed velocity vector to reach it, and this process repeats till it reaches the goal node. This is shown lines 1 and 2 of algorithm 5.

Obstacles are randomly placed in the robot's environment, where they take a region of the configuration space that is considered a collision if the robot was to move towards it.

3.2.2 Obstacles detection

Since the robot does not have any prior knowledge about the obstacles' location, it will need to acquire this data from sensors, usually sensors such as LIDARs are used for this purpose, in this project no specific sensor is used, instead, a detection range that represents the sensors on the robot is used.

Based on sensor readings, if the robot encounters obstacles in its way then it is going to calculate a new path through the *doReplanning()* function (line 6 of algorithm 5), and plans another path based on the existing tree.

Algorithm 5: $Pathexecution()$

```
1  $SetingRobotDestination()$
2  $SetRobotVelocity()$
3    while $robotLocation\ != \ GOAL$ do
4      $UpdateRobotLocation()$
5      if $IsPathOccupied()$ then
6          $doReplanning$
7      End
8  End
```

3.2.3 The replanning procedure

The Replanning function calculates the new path by first invalidating the nodes where the random obstacle appeared (line 1 of algorithm 6).After that a local goal is chosen, this goal is not the final goal that the robot has to reach, but it is a sub-goal that is located on the existing optimal path and that is not part of the obstacles configuration.

It is worth mentioning that the optimal path may be blocked with many obstacles other than the one the robot is looking at for instance, and for this scenario, the sub-goal location can not exist on the optimal path that was already defined by the robot, in this case, the nodes that are closer to the initial optimal path and not part of any occupied configuration are going to be candidates to be a sub-goal .

The third step is to modify the current search tree through choosing a new sampling area and the *Rewire()* function in order to avoid the current random obstacle. Each node in the sampling area is rewired in such a way that current location of the robot becomes the parent of that node.

When the search tree is modified, path execution is done in order to determine the best path to the sub-goal location, from there the robot will resume executing the optimal path to the actual goal location.

Algorithm 6: $doReplanning$

```
1  InvalidateNodes()
2  ChooseSubGoal()
3  SetReplanningArea (),
4  Rewire()
```

```
5  RRT*(qrobot)
6  SettingreplanedPath()
7  End
```

3.3 Simulation

All the computations were performed on a laptop with an Intel Core i5 processor clocked at 1.8 GHz equipped with 4Go of RAM.

The used environment map is that of a storage house sized 100×80 . where the positions on the map are represented as three coordinates $[x\ \ y\ \ \boldsymbol{\theta}]$, “ x” and “ y” represent the cartesian coordinates along the X and Y axes respectively, and “$\boldsymbol{\theta}$ “represent the orientation of the robot at the specific position.

For demonstration purposes, the Map is generated as a binary occupancy map that assigns a “ true” value to the occupied spaces and a “ false” value to the free spaces.

While planning the path, the map is converted to an occupancy matrix map that converts the boolean values of the binary occupancy map into floating point values. This conversion is necessary since sensors acquire information as floating numbers and not as exact numbers.

The Sensor reading ranges are shown in blue, and the robot is considered as a rigid body, this requires an inflation of the used map. The robot in the illustration is shown as a small triangle.

The tree expansion is shown in orange and the initial planned path is shown in bold red. While the replanned path after the robot encountered a random obstacle is demonstrated as a thin red continuous line. and the red dotted line represent the robot attempts to calculate the feasible path while locally moving towards the goal.

The starting position is shown as a blew circle, while the goal position as a green circle.

All the previously explained functions and algorithms were implemented using MATLAB2020a.

3.3.1 The first scenario

- one random obstacle crossing the robot path to the goal.
- Detection range : from 2 to 30 meters
- The detection angle goes from $-pi$ to pi .

As shown in Figure 3.1, RRT* is launched at 10000 nodes , the shortest path from start to goal is initially generated through the global tree expansion.

The robot path encountered an obstacle edge blocking its path as shown in Figure 3.2, where the replanning process started as demonstrated in Figure 3.3.

The robot was able to replan a new path using the same tree during 63.10 seconds , and the and the path length was 111.50 .

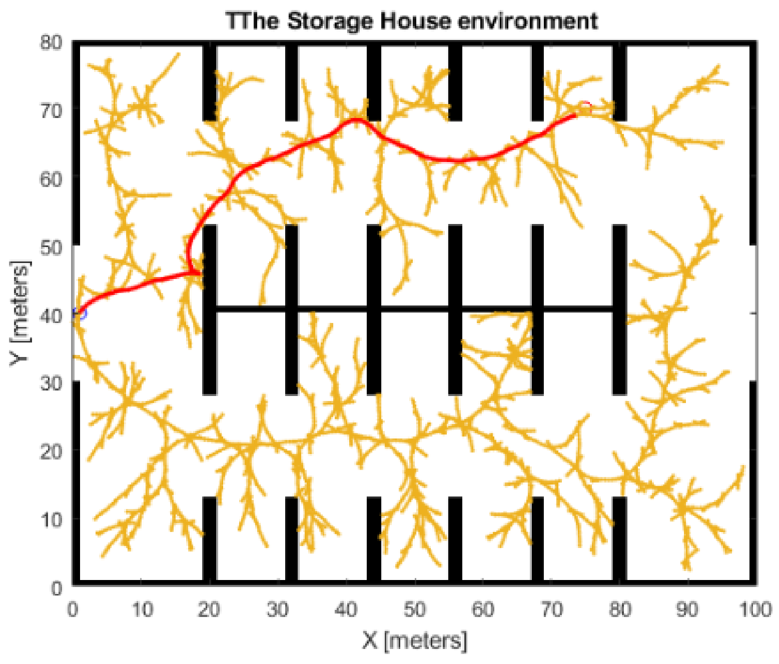

Figure 3.1: The initial planned path for RRT* for the first scenario.

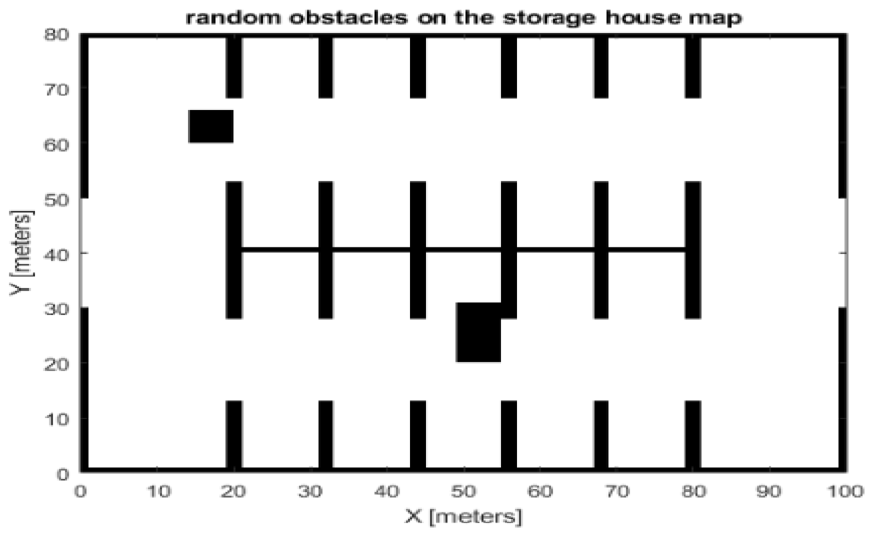

Figure 3.2: The path blocking obstacle placements

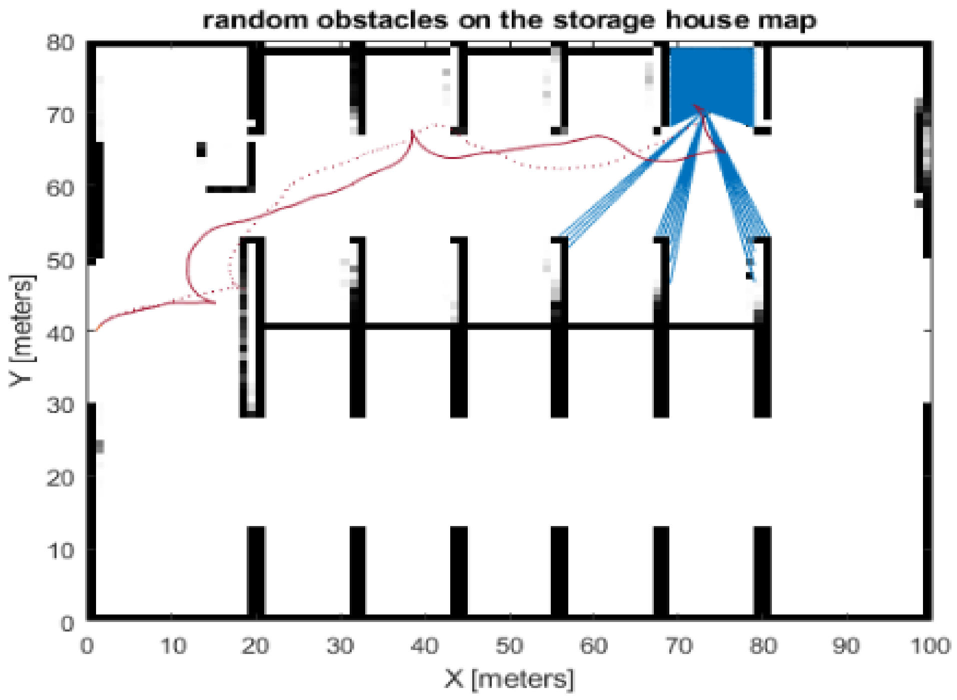

Figure 3.3: The replanned path demonstrated as thin red line .

As shown in Figure 3.4, the start and goal locations are changed , and two random obstacles blocking the robot's path are considered instead of one .

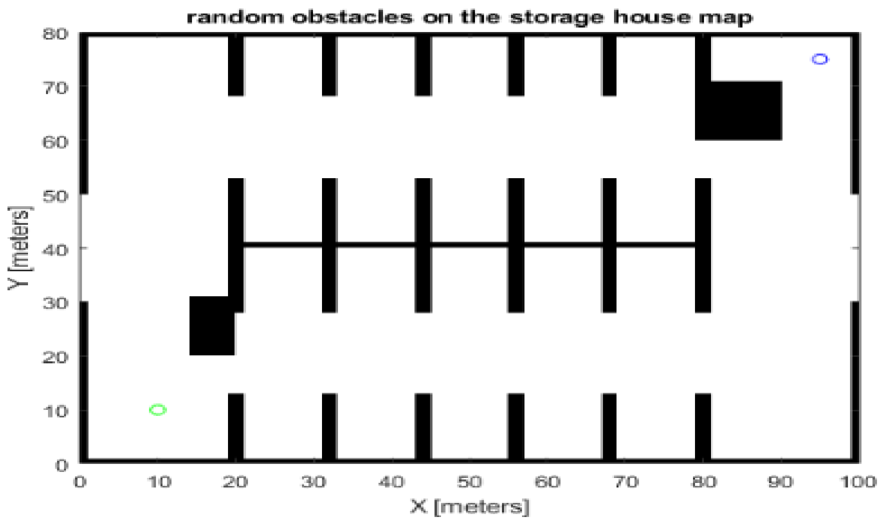

Figure 3.4: Two random obstacles added to the path and the new start (in blue circle) and goal(in green circle) locations

The new starting location was [95 75 45], and the goal location [10 10 0], the robot planned a 150.50 meters long initial path within 70 .54 seconds , which was more than the replanning process time that took 60.24 seconds, and generated a replanned path of 147.50 meters long. Less than the previous one . Figure 3.5 demonstrates the initial path, while Figure 3.6 demonstrates the replanned new path .

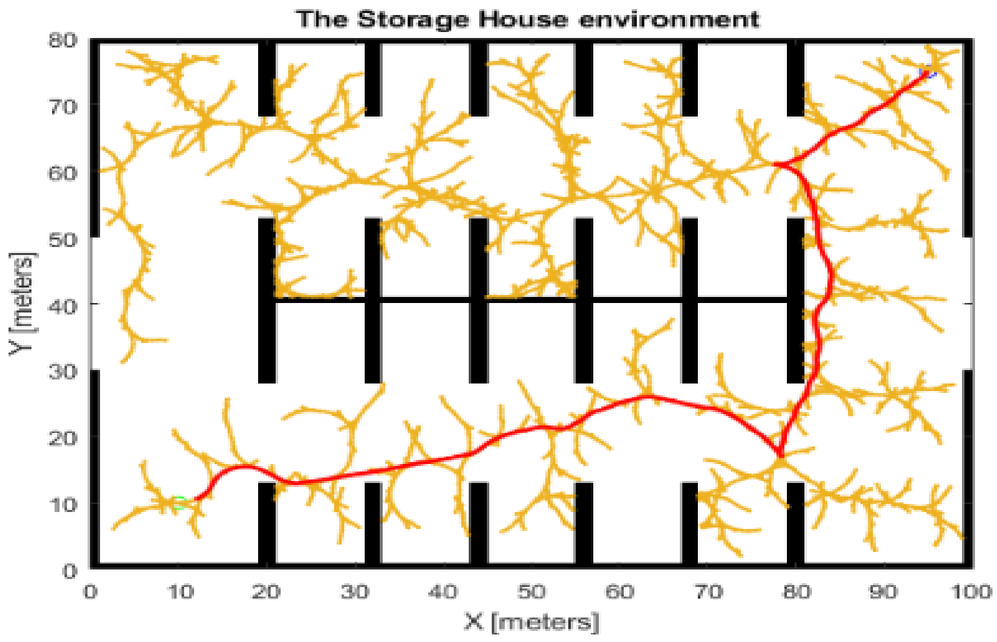

Figure3.5: The initial planned path

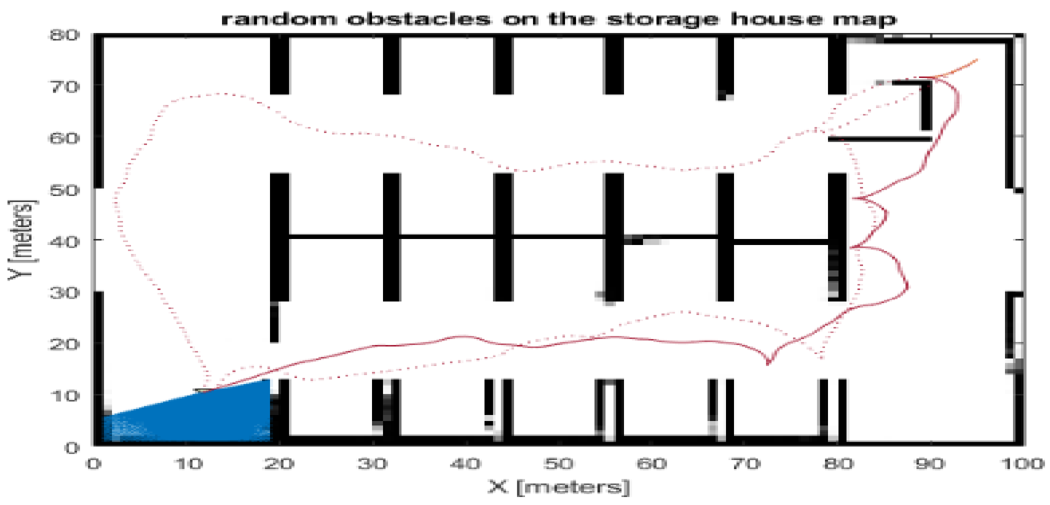

Figure 3.6: The replanned path shown in red continuous line

3.3.2 The Second scenario : changing the detection range

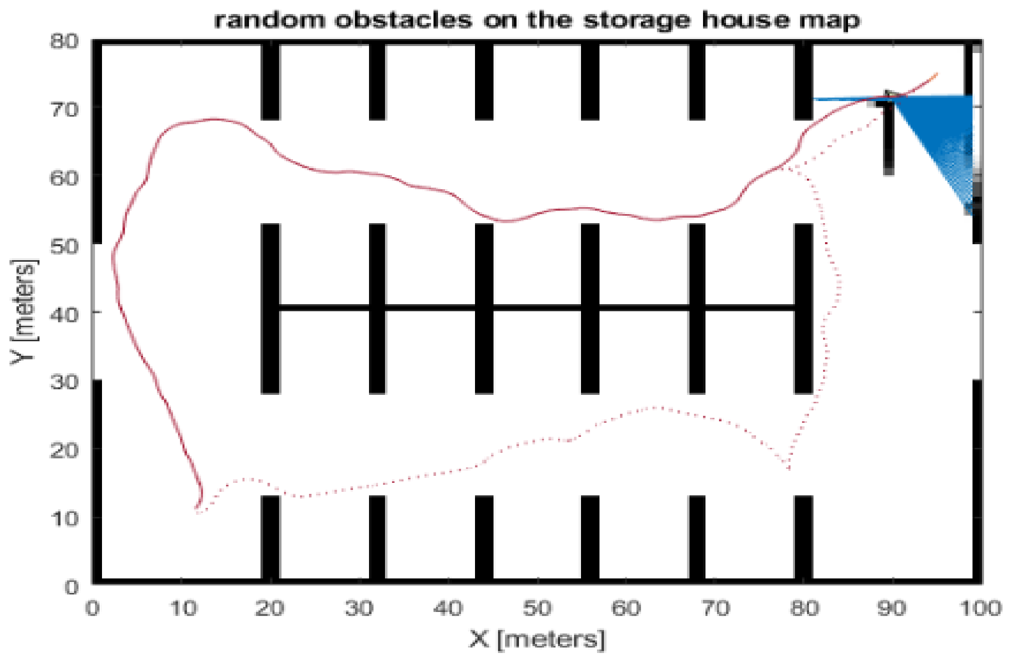

Figure 3.7: The replanned path due to a shorter detection range.

Using the last scenario with two obstacles, we decreased the detection range to go from 0 to 20 meters. At first, the robot was able to see only one side of the first obstacle, and it calculated a first replanned path that was longer than the initial one, as shown in Figure 3.7.

While the robot kept moving towards the replanned path , it detected the other unseen obstacle sides, however by then it was late to make a replanning and avoid the obstacle.

3.3.3 The Third Scenario: Changing the detection angle

- **From - pi to 0 :** Changing the detection angle to go from -pi to 0 resulted in the same result as in the previously used detection range as show in Figure 3.8. However, the time to replan the path was 66,93 seconds, which is longer than that of the first used detection angle.

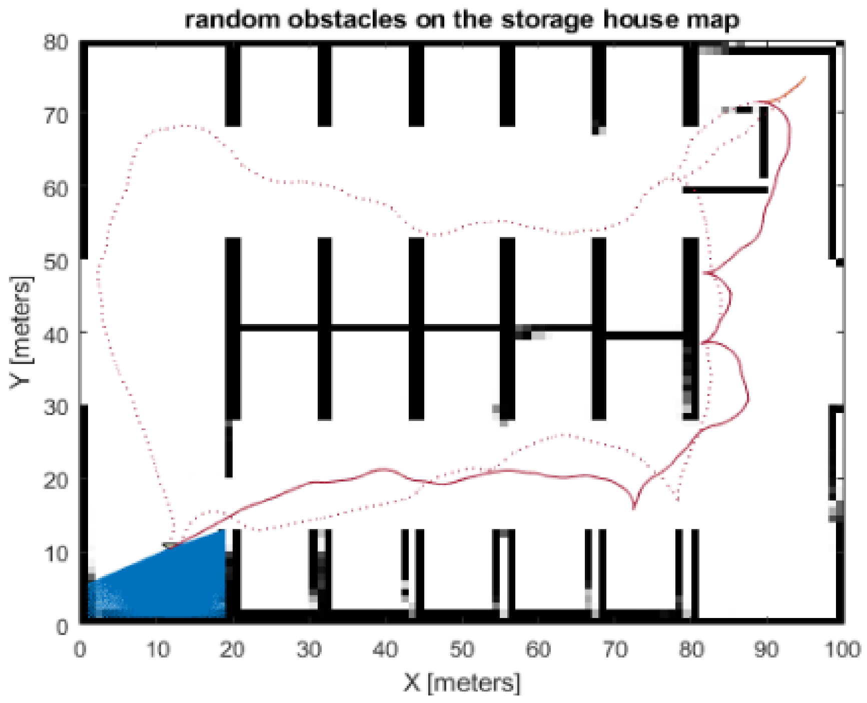

Figure 3.8: the replanned path after changing the detection angle to change from -pi to 0

- **From 0 to pi :** the robot was ale to replan a path in this case, However, the replanned path was long in terms of length 144.00 meters , and in terms of time 74.29 seconds. In addition , the replanned path crossed the obstacle and the robot want through which unlikely to happen in reality. Therefore this was a false unvalidated path.

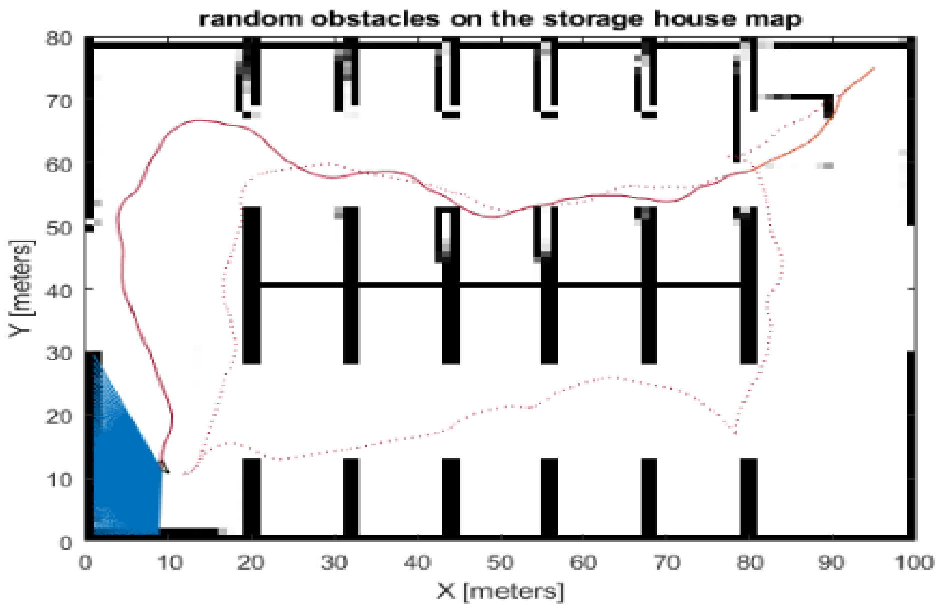

Figure 3.9: The replanned path after changing the detection angle to change from 0 to pi

3.3.4 More than two obstacles

Given more than two obstacles blocking the robot path and a more difficult starting and goal positions as shown in Figure 3.10, the robot was able to plan a global path of length 119.00 meters as demonstrated in Figure 3.11, and perform a replanning that generated a 124.00 meters in 78.53 seconds as shown in Figure 3.12.

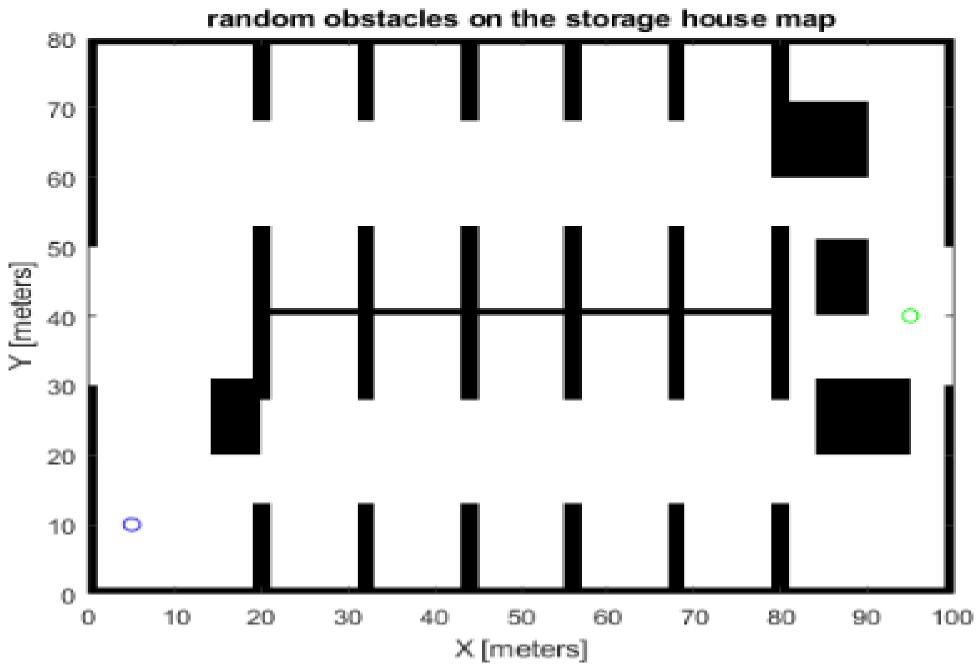

Figure 3.10: More than two obstacles blocking the robot path

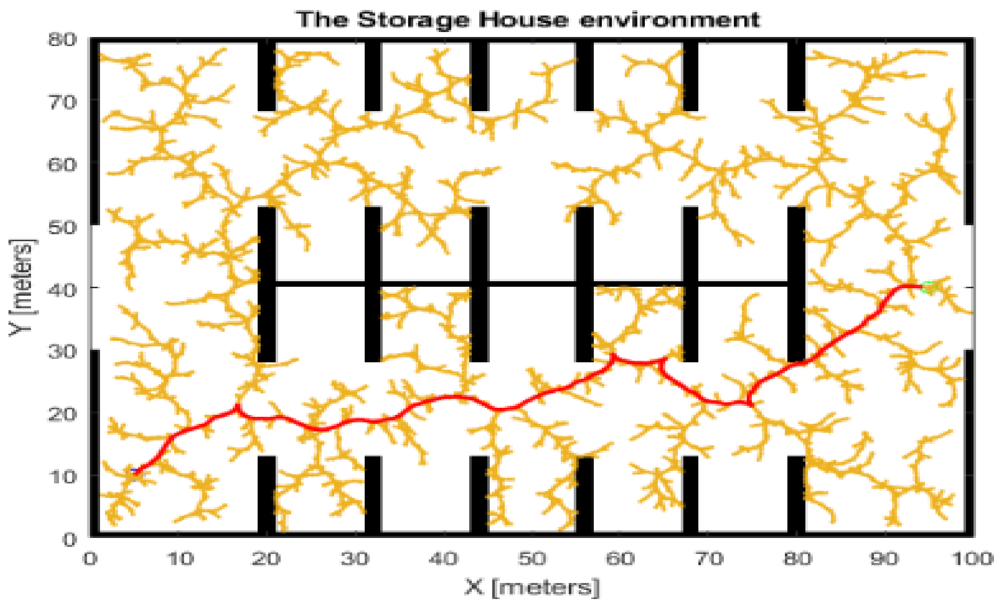

Figure 3.11: The initial planned global path

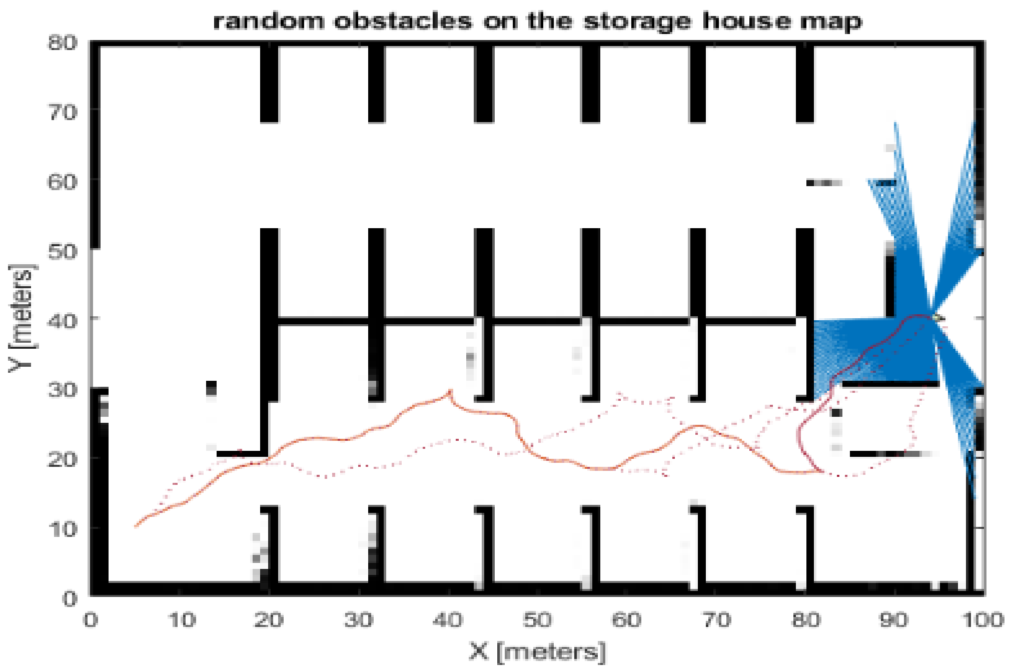

Figure 3.12: The local replanned path in the presence of three obstacles blocking the robot path

Considering five obstacles blocking the robots path as shown in Figure 3.13, resulted in a replanned path of length 166.00 in 219.00 seconds as demonstrated in Figure 3.14

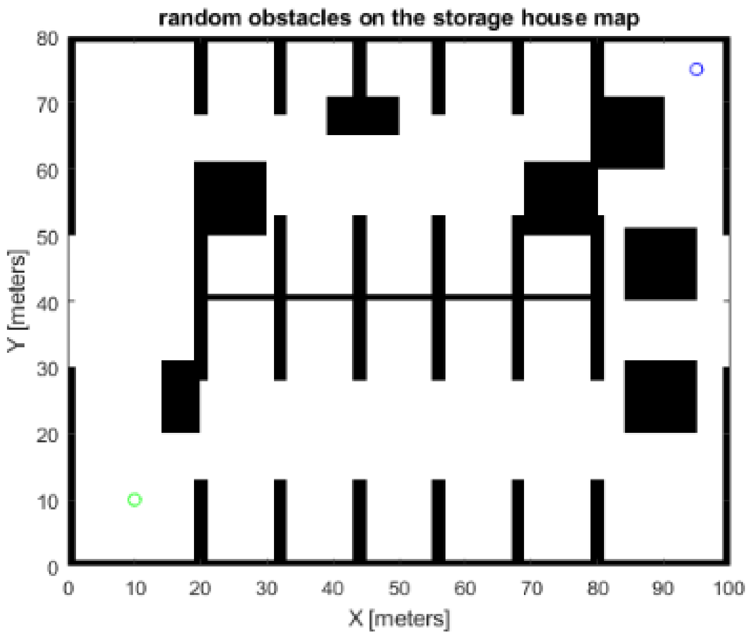

Figure 3.13: Five obstacles on the path

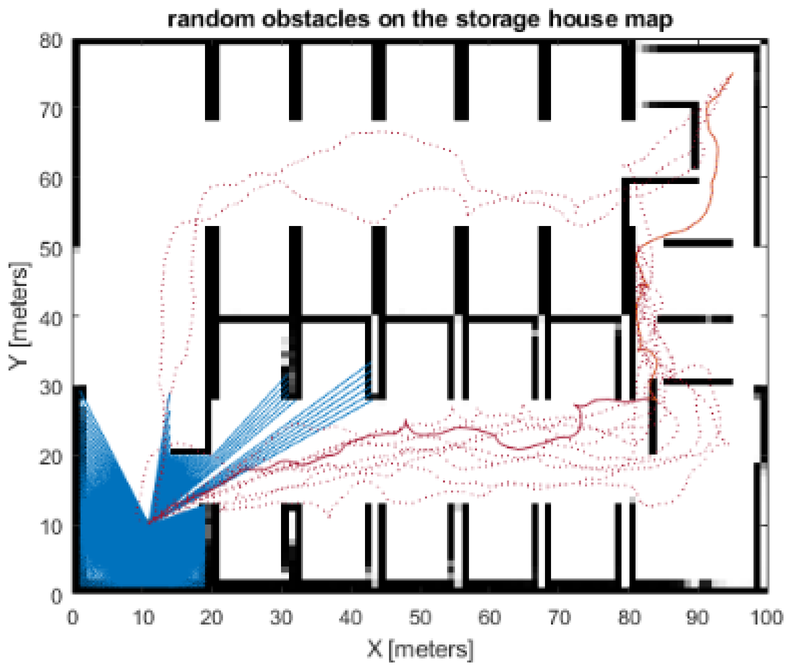

Figure 3.14: The new replanned path in the presence of five obstacles

Table 3.1 summarizes the previously driven results. From the table it becomes clear that the replanning time depends on the obstacles size and density and on their placements and the used detection range.

The number of obstacles	The Detection Range (meters)	The Detection Angle (radians)	The Replanned Path Length (meters)	The Replanning Time(seconds)
One	2 to 30	$-pi$ to pi	111.50	63.10
Two	2 to 30	$-pi$ to pi	147.50	60.24
	0 to 20	$-pi$ to pi	unvalidated	unvalidated
	2 to 30	$-pi$ to 0	147.50	66.93
	2 to 30	0 to pi	unvalidated	unvalidated
Three	2 to 30	$-pi$ to pi	124.00	100.53
Five	2 to 40	$-pi$ to pi	166.00	219.00

Table 3.1: Simulation results for different obstacles and different detection ranges and angles.

3.4 Discussion

Many relevant observations are to be conducted from the simulation results:

- While locally planning the path, the algorithm did not need to regenerate a new tree each time it found a new obstacle, instead, it used the same tree nodes to regenerate a new route, which is a positive factor that saves time , since there is no need to increase the number of nodes even when the environment is big

- Decreasing the detection range of sensors on the robot will result in a decrease of the algorithm's efficiency and time, since the robot will either be unable to see all the obstacle sides, or in case it did, then it will not have time to react faster to replan a path. Therefore it is desirable to always keeping the detection range two times higher than the robots' velocity.

- The detection angle plays a major role during the replanning process, since assuring a half circle [-pi pi] detection angle makes the robot more predictive about the different angle and sides of the obstacle, therefore the replanned path will be feasible and the algorithm run time will be much faster this is more important when the map environment is larger .

- Planning the path globally then locally helps the robot to easily invalidate the nodes in the obstacle region while replanning. However, if the environment is larger and the goal is farther, it may add time because of the increased number of computations.

3.5 Conclusion

Replanning a feasible path using RRT* is efficient in terms of usage of nodes numbers, along with the right detection angles and ranges choice.

In the next chapter, a general conclusion is gathered about the overall work. In addition to a proposed future work.

General Conclusion and Future Work

General Conclusion and Future Work

In this work we approached the path planning problem for a ground wheeled mobile robot with a method that combined global and local path planning .

The proposed algorithm should be valid for all types of wheeled mobile robots. The assumptions that have been made throughout this project where:

- The robot is aware of the overall environment but not aware of the obstacles placements.
- The obstacles are considered as random but non-moving obstacles and take a rectangular shape.
- No specific sensors where used, instead an overall detection range was chosen.

The objective was to find a feasible path from starting to ending location, and dynamically reacting to the random obstacles that appear in the robots' range of sight. Our contribution approach uses RRT* and dynamic replanning to choose the right path. The proposed approach saves the same number of nodes that are used initially even while replanning the path, through invalidating the nodes in obstacles regions while replanning and connecting them in a valid manner to generate a feasible path. This process saves memory and time .

As long as the robot is moving around the environment, it calculates many paths and takes only the most feasible path. The proposed algorithm was tested using MATLAB , and the obtained results achieved the objectives that were set at the beginning of this project related to performing a complete replanning process using RRT* and achieving the feasible path in the presence of random obstacles.

Some corner cases where also tested and commented on , related to the choice of the detection range and angles, since they allow the robot to plan a more accurate path when planned properly.

Future Work

Since research work in this area is always a subject of enhancement , future extended work in this scope should mostly be concerned with:

- **Considering unknown moving obstacles** : since the purpose of developing robotics applications and performance is to make it as close to the reality as possible. Future work considering the same approach with obstacles that are moving in addition of being random is very crucial, since it will take into consideration the obstacles velocities and moving angles, as well as those of the robot.
- **Considering multiple robots in the environment**: in todays' applications, multiple robots should interact in the same environment, therefore planning feasible paths and being able to differentiate between robots is a work area to be considered in the future.

- **Extending the environment to outdoor ones**: this aligns with considering moving obstacles, however considering outdoor environments can make the robot react to obstacles of different densities, such as pedestrians.

Bibliography

[1] Rubio F, Valero F, Llopis-Albert “. *A review of mobile robots: Concepts, methods, theoretical framework, and applications*,” International Journal of Advanced Robotic Systems. March-2019

[2] Disha Chandrakant Shah , “ *path planning for mobile robots using Potential Fields Method “,* University of Texas, Arlington, May 2018

[3] Jia Zhu, Shili Zhao, Ran Zhao. *“Path Planning for Autonomous Underwater Vehicle Based on Artificial Potential Field and Modified RRT”,* International Conference on Computer, Control and Robotics (ICCCR°, June 2021

[4] Khatib O. *“ Real time obstacle avoidance for manipulators and mobile robots”.* In: IEEE international conference on robotics and automation, Missouri, vols. 25-28; Mar 1985

[5] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, “*Probabilistic roadmaps for path planning in high-dimensional configuration spaces*,” IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, Aug 1996.

[6] S. Karaman and E. Frazzoli, “*Sampling-based Algorithms for Optimal Motion Planning*” in 49th IEEE Conference on Decision and Control (CDC), Dec 2010, pp. 7681–7687.

[7] J. Bruce and M. Veloso, “*Real-time randomized path planning for robot navigation*,” IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, 2002, pp. 2383–2388 vol.3.

[8] D. Ferguson, N. Kalra, and A. Stentz, “*Replanning with rrts*,” in Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., May 2006, pp. 1243–1248.

[9] Park, Kevin M. Lynch and Frank C., *Modern Robotics, Mechanics, Planning and Control*, Cambridge university Press, May 2017.

[10] Koubaa Anis , Bennaceur, Hachemi , Chaari, Imen, Trigui, Sahar , Adel , Ammar , Sriti , Mohamed-Foued , Alajlan, Maram , Cheikhrouhou, Omar , Javed, Yasir , "*Robot Path Planning and Cooperation, Foundations, Algorithms and experimentations "*, Springer International Publishing, 2018.

[11] G. Klancar, A. Zdesar, S. Blazic, and I. Skrjanc, "*Wheeled mobile robotics: from fundamentals towards autonomous systems"*. Waltham, MA: Elsevier, 2017.

[12] R. Raja and A. Dutta, "*Path planning in dynamic environment for a rover using A∗ and potential field method,*" 18th International Conference on Advanced Robotics (ICAR), 2017.

[13] G. Carbone and F. Gomez-Barvo, *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*. Dordrecht: Springer, 2015.

[14] B.K. Patle , Ganesh Babu L, "*A review: On path planning strategies for navigation of mobile robot "*, Elsevier publishing , April 2019

[15] Rostami, S.M.H., Sangaiah, A.K., Wang, J. *et al.* *"Obstacle avoidance of mobile robots using modified artificial potential field algorithm"*. J Wireless Com Network 2019.

[16] H. Wang, B. Yan, X. Li, X. Luo, Q. Yang and W. Yan, *"On optimal path planning for UAV based patrolling in complex 3D topographies"* 2016 IEEE International Conference on Information and Automation (ICIA), 2016, pp. 986-990.

[17] Roland Siegwart, Margarita Chli, Juan Nieto, Nick Lawrance , *" autonomous mobile robots course, sampling based motion planning ", ETHzürich , 2018*

Bibliography

[18] *IRB 6640 A strong robot for numerous applications*. [online] Available at: <https://new.abb.com/products/robotics/industrial-robots/irb-6640> [Accessed 2 september 2021]

[19] “ *KR 4 AGILUS: Compact, flexible robot for various applications in the electronics industry”*, KUKA-2021. [Online]. Available: https://www.kuka.com/en-de/products/robot-systems/industrial-robots/kr-4-agilus. [Accessed: 02- Sep- 2021]

[20] "Atlas is an R&D platform pushing the bounds of robotics forward", *Boston Dynamics*, 2021. [Online]. Available: https://www.bostondynamics.com/atlas. [Accessed: 02- Sep- 2021]

[21] "Outdoor Security & Inspections robots for business application RaaS turn-key robotic solutions", *SMP Robotics - Autonomous mobile robot*, 2021. [Online]. Available: https://smprobotics.com/. [Accessed: 02- Sep- 2021].

[22] *"Aerial Robot for Sewer Inspection", eurecat*, 2021. [Online]. Available: https://eurecat.org/en/portfolio-items/aerial-robot-for-sewer-inspection/. [Accessed: 02- Sep- 2021].