UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF MECHANICAL ENGINEERING

Path planning algorithms for autonomous robotic systems

by

IOANNIS LOIZOU

Submitted in partial fulfillment of the requirements for the degree of Diploma in Mechanical Engineering at the University of Thessaly

Volos, 2022

UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF MECHANICAL ENGINEERING

Path planning algorithms for autonomous robotic systems

by

IOANNIS LOIZOU

Submitted in partial fulfillment of the requirements for the degree of Diploma in Mechanical Engineering at the University of Thessaly

Volos, 2022

Approved by the Committee on Final Examination:

Advisor	Dr. Konstantinos Ampountolas, Associate Professor, Department of Mechanical Engineering, University of Thessaly
Member	Dr. Dimitrios Pantelis, Professor, Department of Mechanical Engineering, University of Thessaly
Member	Dr. Georgios Saharidis, Associate Professor, Department of Mechanical Engineering, University of Thessaly

Acknowledgements

I would like to give my sincere gratitude to my thesis advisor, Associate Professor Dr. Konstantinos Ampountolas for his valuable help and guidance during the work of this thesis. I also want to thank the members of the committee Professor Dr. Dimitrios Pantelis and Associate Professor Dr. Georgios Saharidis for examining my work.

Furthermore, I am deeply thankful to my family and friends for their support during my studies and this work. Finally, I want to thank my partner which was by my side all the time encouraging me to give my best and push forward.

Path planning algorithms for autonomous robotic systems

LOIZOU IOANNIS

Department of Mechanical Engineering, University of Thessaly, 2022

Supervisor: Dr. Konstantinos Ampountolas

Associate Professor in Control Engineering

Abstract

Autonomous robotic systems are being used more and more in the last years, in many applications such as delivery, search and rescue, transport, and others. Each application has its unique characteristics, thus different solutions are needed. One of the most important aspects of autonomous systems is path planning. In this thesis, different path planning algorithms such as A*, RRT, PRM and APF are explained and compared by doing a comparative quantitive case study on a map of the campus of the University of Thessaly and concluded on the strength and weaknesses that each one posse. Finally, the best-suited algorithm is proposed to be used for an in-house robot that may be developed at the university which is PRM.

Keywords: Path Planning, Motion Planning, Autonomous robots, A*, RRT, PRM, APF

Αλγόριθμοι σχεδιασμού διαδρομής για αυτόνομα ρομποτικά συστήματα

ΛΟΙΖΟΥ ΙΩΑΝΝΗΣ

Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας, 2022

Επιβλέπων Καθηγητής: Δρ. Κωνσταντίνος Αμπουντώλας,

Αναπληρωτής Καθηγητής Συστημάτων Ελέγχου

Περίληψη

Τα αυτόνομα ρομποτικά συστήματα χρησιμοποιούνται όλο και περισσότερο τα τελευταία χρόνια, σε πολλές εφαρμογές όπως την διανομή, την έρευνα και διάσωση, τις μεταφορές και πολλές άλλες. Κάθε εφαρμογή έχει τα δικά μοναδικά χαρακτηριστικά και για αυτό χρειάζονται διαφορετικές λύσεις. Ένας από τους πιο σημαντικούς τομείς είναι ο σχεδιασμός διαδρομής. Σε αυτή την διπλωματική εξηγούνται και συγκρίνονται διάφοροι αλγόριθμοι σχεδιασμού διαδρομής όπως A*, RRT ,PRM, και APF και γίνετε ένα πείραμα πάνω χάρτη του προαύλιου του πανεπιστημίου Θεσσαλίας που καταλήγει στα πλεονεκτήματα και τα μειονεκτήματα που έχει ο καθένας. Κλείνοντας προτείνετε ο πιο ταιριαστός αλγόριθμος για ένα ρομπότ το οποίο μπορεί να δημιουργηθεί στο πανεπιστήμιο και είναι ο PRM.

Λέξεις-κλειδιά: Σχεδιασμός διαδρομής, Σχεδιασμός κίνησης, Αυτόνομα ρομπότ , A*, RRT ,PRM, APF

Table of Contents

1 INTRODUCTION 1

1.1 MOTIVATION 1
1.2 THESIS ORGANIZATION 1
1.3 ROBOT SYSTEM PROCEDURES 2
1.4 WHEELED MOBILE ROBOTS CATEGORIES 4
1.5 REAL-WORLD MOBILE ROBOT APPLICATIONS 5
1.6 GLOBAL AND LOCAL PLANNING 7

2 LITERATURE REVIEW 9

2.1 MODELLING OF WORKSPACE IN PATH PLANNING 10
2.2 ENVIRONMENT CELL DECOMPOSITION AND ROADMAP GRAPHS 10

3 METHODOLOGY (DESCRIPTION OF PROPOSED PATH PLANNING ALGORITHMS) 13

3.1 MAP CREATION 13
3.2 A* SEARCH 16
3.3 RRT 19
3.4 PRM 20
3.5 ARTIFICIAL POTENTIAL FIELDS 22

4 APPLICATION/CASE STUDY 27

4.1 APPLICATION OF A* 27
4.2 APPLICATION OF RRT 29
4.3 APPLICATION OF PRM 33
4.4 APPLICATION OF ARTIFICIAL POTENTIAL FIELDS 36

5 CONCLUSIONS 42

5.1 DISCUSSION 42
5.2 SUGGESTIONS FOR FURTHER STUDY 46

REFERENCES 47

APPENDIX 51

List of figures

Figure 1.1 Path for action (Caron, 2016) 2
Figure 1.2 A lidar sensor in the left and in the right the result taken from lidar 3
Figure 1.3 Diagram that demonstrates Turtlebot 3 and its components (Robotis, 2015) 4
Figure 1.4 (Left) Conventional wheel, (Center) Omniwheel, (Right) Mecanum wheel (Lynch & Park, 2017) 5
Figure 1.5 Starship robot doing the delivery (Insider, 2021) 6
Figure 1.6 Path planning global and local classification 8
Figure 2.1 Path planning categorization (Sánchez-Ibáñez et al., 2021) 9
Figure 2.2 Elevation map (5 Free Global DEM Data Sources – Digital Elevation Models, 2022) 12
Figure 2.3 Types of grids and roadmaps (Sánchez-Ibáñez et al., 2021) 12
Figure 3.1 Google maps University of Thessaly campus dimensions (Google, 2022) 13
Figure 3.2 Google Maps measurements of mechanical engineering building (Google, 2022) 14
Figure 3.3 Grid map created with zeros and ones 15
Figure 3.4 (Left) Directed weighted graph (Center) Undirected weighted graph (Right) Directed unweighted graph (Lynch & Park, 2017) 16
Figure 3.5 A roadmap in 2 dimensions (Lynch & Park, 2017) 21
Figure 3.6 Obstacle that creates a local minima on the left and two obstacles that they overlap in the right (Triharminto et al., 2016) 23
Figure 3.7 Gradient Descent example (Pandey, 2022) 25
Figure 3.8 (Top left) The 2d grid map has three obstacles and goal point which is denoted a +. (Top right) The summed attractive and repulsive potentials which results in this mountain like view (Bottom left) A Contour plot of the potential (Bottom right) Force vector field (Lynch & Park, 2017) 26
Figure 4.1 Scenario 1 A* path 28
Figure 4.2 RRT expansion tree with high randomness 30
Figure 4.3 RRT expansion tree with low randomness 30
Figure 4.4 RRT path results 31
Figure 4.5 60 sample points on the map in pink 33
Figure 4.6 Creation of the connections on 60 sample points (Roadmap) 33

Figure 4.7 Creation of the connections on 100 sample points (Roadmap) 34
Figure 4.8 Optimal path of the algorithm calculated on 60 sample points.......................... 34
Figure 4.9 Collision check example (Khokhar, 2021) 36
Figure 4.10 Attractive potential with K = 0.002 37
Figure 4.11 Attractive potential with K = 0.000833 37
Figure 4.12 Repulsive potential with k = 8000 38
Figure 4.13 Repulsive potential with k = 2000 38
Figure 4.14 Gradient decent algorithm visualized with a ball rolling down 39
Figure 4.15 Special Scenario APF extracted path 40
Figure 4.16 APF extracted path with vector field and... 40
Figure 5.1 Scenario 1 Path length results 42
Figure 5.2 Scenario 2 Path length results 43
Figure 5.3 Scenario 3 Path length results 44
Figure 5.4 Scenario 2 A* path.. 51
Figure 5.5 Scenario 3 A* path.. 51
Figure 5.6 Scenario 2 PRM path (1 of the 100) 52
Figure 5.7 Scenario 3 PRM roadmap 52
Figure 5.8 Scenario 3 PRM path (1 of the 100) 53
Figure 5.9 Scenario 2 RRT tree.. 53
Figure 5.10 Scenario 2 RRT path (1 of the 100) 54
Figure 5.11 Scenario 3 RRT tree.. 54
Figure 5.12 Scenario 3 RRT path (1 of the 100) 55
Figure 5.13 Scenario 3 APF path 55

List of Tables

Table 3.1 A* pseudo code (GeeksforGeeks, 2022) 18
Table 3.2 RRT pseudocode (Lynch & Park, 2017) 19
Table 3.3 PRM pseudocode (Lynch & Park, 2017) 21
Table 3.4 APF pseudocode 26
Table 4.1 A* results 28
Table 4.2 RRT results 31
Table 4.3 PRM results 35
Table 4.4 Trial and error factors 39
Table 4.5 APF results 40
Table 5.1 Criteria ranking for algorithms 45

Symbols and Abbreviations

APF	Artificial Potential Fields
B	Obstacle
$\mathbf{C_{free}}$	Free Configuration space
$\mathbf{C_{space}}$	Configuration space
$\mathbf{d_{range}}$	Range of influence of the obstacles
$\mathbf{F_{Bi}}$	Repulsive force
$\mathbf{F_{goal}}$	Attractive force
K	Constant scaling parameter (Attractive)
k	Constant scaling parameter (Repulsive)
P	Potential field
$\mathbf{P_b}$	Repulsive potential
$\mathbf{P_{goal}}$	Attractive potential
PRM	Probabilistic Roadmap
$\mathbf{q_{start}}$	Start point in Configuration space
$\mathbf{q_{goal}}$	Goal point in Configuration space
RRT	Rapidly-exploring random tree
$\mathbf{U_B}$	Repulsive potential
$\chi_{\mathbf{free}}$	Free state space
$\chi_{\mathbf{start}}$	Start point in state space
$\chi_{\mathbf{goal}}$	Goal point in state space
$\chi_{\mathbf{sample}}$	Sample point in state space

1 Introduction

1.1 Motivation

Robotics in the last years are having exponential growth in production and use. As algorithms progress and computational power increases robots are becoming smarter by the day which results in increased use in all sectors of life. While mechanically robots have remained the same or with small improvements for the last years, their intelligence had seen massive changes. Artificial intelligence allows robots to become more autonomous and less dependent on human intervention.

That's why robots as they become more autonomous, they have to be also more reliable. Path planning is a big part of it and at the same time one of the most challenging because there is never one fit solution. Path planning selection can be a quite difficult task.

The goal of this thesis is to do a quantitative comparative study on four path planning algorithms such as A*, RRT, PRM and APF and compare their performance on a map that represents the University of Thessaly campus which is located in Pedion Areos. Then conclude on which algorithm is the best for use in a possible in-house-produced robot.

1.2 Thesis organization

This thesis is divided into five chapters:

In Chapter 1 an introduction on what is a robot is covered and the main information that is needed to proceed in the next chapters

In Chapter 2 a literature review is being presented that explains what type of path planning algorithms exist and how they are classified

In Chapter 3 the creation of the map is explained and all the path planning algorithms that are used in this thesis

In Chapter 4 the application of the previously explained algorithms is shown with a detailed explanation of the results

In Chapter 5 which is the final, all the results are being compared and a proposal is made for which is the best-suited algorithm

1.3 Robot system procedures

When human walking is analyzed, a clear pattern occurs. First, it uses its eyes to see the environment, the nose to smell it, the feet, or hands to feel it and the ear to hear it. These are the sensors needed to collect all the possible data and then fuse it together to decide the state that is currently in. This is called *perception*. This is the first important step for the human to help him to walk. When the state Is decided then the next step is to decide how to proceed to achieve its goal. For example, if a human wants to walk from the bedroom of the house to the kitchen, he needs to consider the map that he is remembering of his house and plan its route optimally. Sometimes it needs to go straight to the goal or to do a subgoal before reaching its destination. Every occasion is different and depends on a lot of factors. When humans want to go as fast as possible without covering unnecessary distances, they calculate the optimal path in their minds before starting. This is what called *planning*. When the path to the goal is decided, the human must walk to reach its destination otherwise the goal is unachievable. To do this human uses its feet one at a time to cover the distance needed to complete the path. This is called *action*. Now that the three main phases are explained the problem can be seen dynamically. The human senses his environment and his deciding its state or otherwise called in this case localizing. Then uses his mind with his memory to decide the best possible path that needs to be taken to reach the goal and then start walking toward it. But as the human is walking the state is subject to change and it needs to be assessed again. Then the planning may also be subject to change if the state forces the plan to change because of unexpected changes to the environment. And walking is not always precise to the point so the action may need correction to correctly follow the path. All this is in a constant loop until the desired goal is met. This procedure is what all robots use with sensors for perception, microprocessors, and memory for planning, and motors for action.

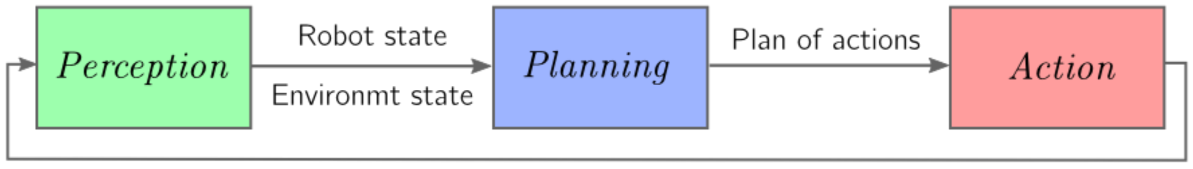


Figure 1.1 Path for action (Caron, 2016)

One of the most used sensors used for perception in robotics is the light detection and ranging sensor (LiDAR). LiDAR uses laser beams that are safe for the eyes to help the computer and the robot to see the 3d environment and have an accurate representation of the world. In a typical lidar sensor, the surroundings are impacted by pulsed light pulses. These pulses return to the sensor after bouncing off nearby objects. The sensor determines the distance traveled by each pulse by measuring the time it takes for it to return to the sensor. A precise, real-time 3D map of the environment is produced by repeating this procedure millions of times per second. A point cloud is a name given to this 3D map. The lidar point cloud can be used by an onboard computer for secure navigation. Below an open-sourced robot is analyzed as an example

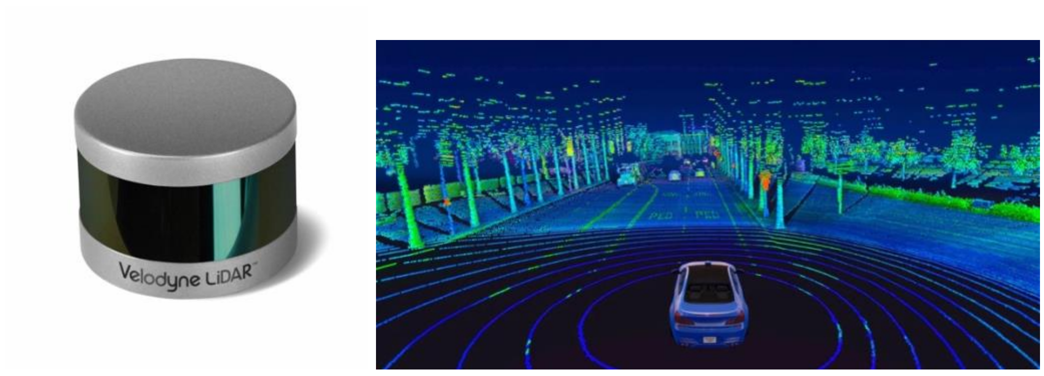


Figure 1.2 A lidar sensor in the left and in the right the result taken from lidar (Velodyne Lidar, 2018)

TurtleBot3 example

TurtleBot3 is a small, cheap, programmable mobile robot that is open-sourced for use in education, research, hobby, and product prototyping. The goal of the turtleBot 3 is to be accessible to anyone who wants to apply or improve its robotics skills while testing in a real-world environment. Because it is also modular, different sensors and microprocessors are available to be used.

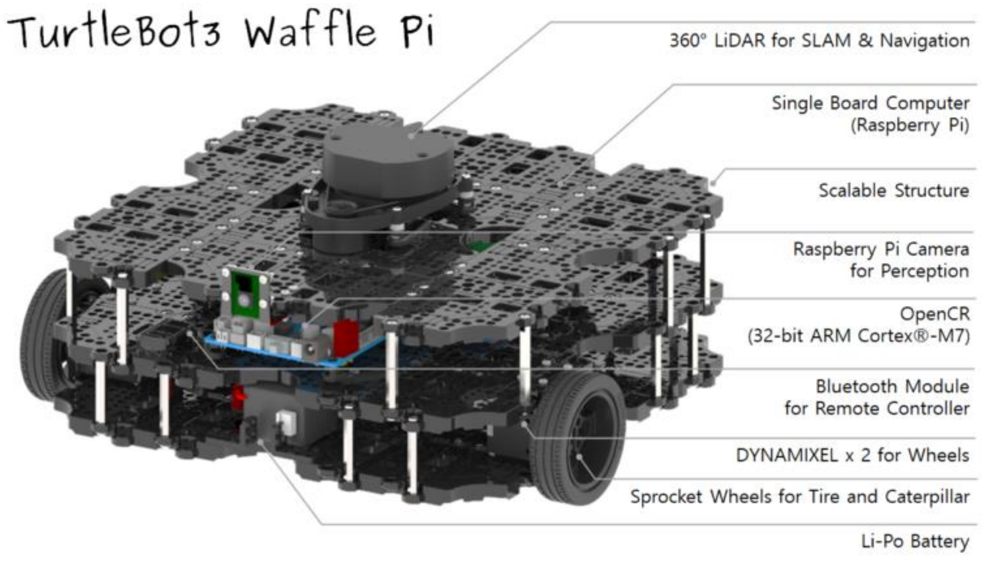

TurtleBot3 Waffle Pi
360° LiDAR for SLAM & Navigation
Single Board Computer (Raspberry Pi)
Scalable Structure
Raspberry Pi Camera for Perception
OpenCR (32-bit ARM Cortex®-M7)
Bluetooth Module for Remote Controller
DYNAMIXEL x 2 for Wheels
Sprocket Wheels for Tire and Caterpillar
Li-Po Battery

Figure 1.3 Diagram that demonstrates Turtlebot 3 and its components (Robotis, 2015)

For the perception phase, the following sensors are used 360 LiDAR for SLAM and navigation, Camera, and IMU but because each sensor provides different data, a fusion technique has to be applied to make the data usable. Then for the planning phase, it has a Raspberry Pi 4. It is a single-board computer with electronic gates to get the data from the sensors and provide action to the actuators. Finally, the motors DYNAMIXEL (XL430-W250-T) are used for the action which is moving the robot from one place to another. While this is a simple and cheap example most robots use these parts. What changes are the specification of the parts depending on the requirements of the mission that the robot needs to handle. While the parts are mostly the same there is one thing that can change dramatically from one robot to another. Many claim that are the most important factor of all, the software. If you take two mechanically same robots with just different software, it is possible to make one work like a kid's toy and the other like a state-of-the-art machine. That's why algorithms used in robots are crucial for their capabilities.

1.4 Wheeled mobile robots categories

As mentioned by (Lynch & Park, 2017) wheeled mobile robots are separated into two types, which are omnidirectional and Non-holonomic. Omnidirectional robots can move in any straight way possible on a plane, while non-holonomic cannot move sideways and they have constraints on the way they can move due to their wheels. To determine if the robot is omnidirectional or nonholonomic, the design of the wheels is decisive. Nonholonomic robots use the conventional wheels that are used everywhere (e.g. car wheels): The wheel rotation axle is parallel to the ground and if it needs to be steered it is changing the rotation axle degree but staying parallel to the ground and not slipping sideways. The omnidirectional robots wheeled robots usually use omniwheels or mecanum wheels. These wheels are only driven forward and backward, and they do not steer, because of their design it is possible to slip sideways which is used instead of turning with the conventional method. The omniwheels use rollers that are aligned straight and mecanum wheels rollers have some amount of tilt. While the omnidirectional robots may seem the best choice because of the moving capabilities they can only work best on flat and hard ground.

The category of the robot influence directly the selection of the algorithms used for planning and control and that's why when wheels are decided also the selection of algorithms that the robot can work with is changing. While some path planning algorithms are used in both omnidirectional and nonholonomic robots, a lot of other path planning algorithms can't work with the differential constraints that the convectional wheels have.

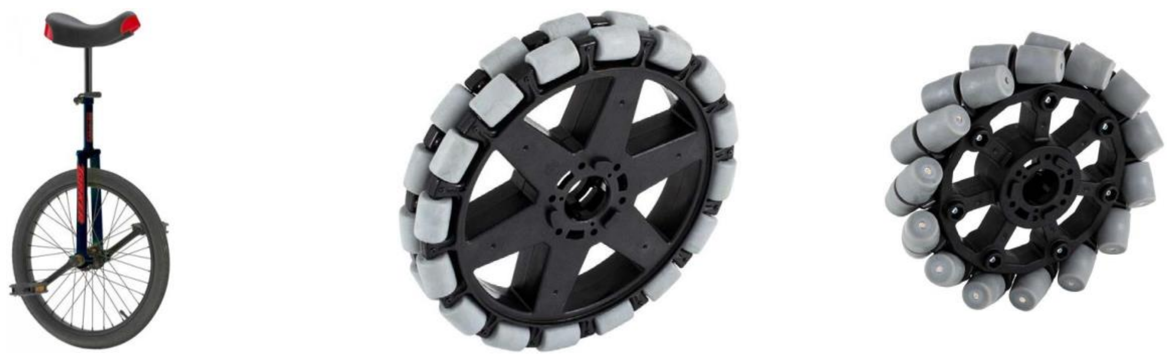


Figure 1.4 (Left) Conventional wheel, (Center) Omniwheel, (Right) Mecanum wheel (Lynch & Park, 2017)

1.5 Real-world mobile robot applications

As mentioned by (Wikipedia contributors, 2022) An Estonian startup called Starship Technologies is working on automated delivery robots. The company has its main office in San Francisco, California, and has engineering facilities in Helsinki, Finland, and Tallinn, Estonia. Additionally, Starship maintains offices in Mountain View, California, Washington, DC, Germany, and London, UK. Starship received up to $100 million in funding in January and February 2022 from the European Investment Bank and venture capitalists. This money will be used for R&D and to add 1,700 more robots to the company's fleet. Since its founding in 2014, the company has raised over $202 million.

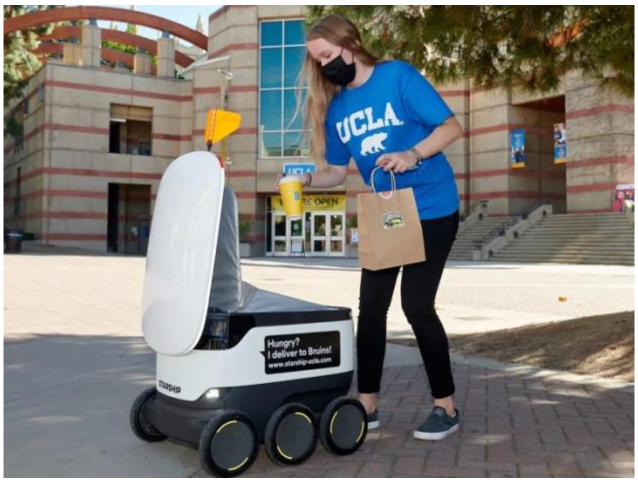


Figure 1.5 Starship robot doing the delivery (Insider, 2021)

Starship creates and manages last-mile delivery robots. These robots are electrically powered, and they don’t use the road only the sidewalk at a speed like pedestrians, which is a max speed of 6 km/h (3.7 mph). It is only used for short-delivery ranges and while it is autonomous, in case of emergency a remote operator can take the control of the robot, to assist it to get back on track. By having this option, the downtime of the robots reduces significantly. The robot uses computer vision and image processing to detect edges and mapping techniques to decide if the terrain is accessible to navigate. The weight of the robot is 25 kg and can carry up to 9.1 kg of food delivery. Robots' average battery life is around 18 hours, and the robot can travel on average each day around 40 km. The sensors that the robot is carrying are ultrasonic sensors, radar, cameras, GPS, IMU, and possibly some more. The interesting fact is that no lidar sensor is present.

Speakers are present on the robots to be able to communicate with the humans when needed. Usually, the orders take place through mobile apps on IOS and Android devices. When the robot arrives, the client is notified from the app, and to be able to receive the package the client must verify its identity through biometric security. The service was trialed in more than 100 cities and 20 nations before going into operation commercially. The corporation has operations in Germany, Finland, Estonia, the United States, and the United Kingdom. Starship will start concentrating heavily on providing delivery services on college campuses in 2019. With the aim of reaching one million students, it has announced intentions to expand its service to 100 college campuses in the US. In order to aid with the delivery driver shortage during the COVID-19 epidemic, Starship increased the amount of grocery delivery robots employed in both the UK and the US.

Starship claims that its robots are “powered by zero-carbon electricity, and for the average delivery the electricity that is used as much as boiling a cup of tea”. Research was carried out by Starship Technologies and Milton Keynes Council to evaluate the effects of zero-emission robots in cities over a period of three and a half years. The study report states that the Starship fleet has avoided 280,000 automobile trips and more than 500,000 miles of car travel, saving 137 tons of CO2 and 22 kg of NOx throughout the course of the study.

1.6 Global and local Planning

Path planning is considered one of the most important research problems in robotics that a robotics engineer has to solve. A great deal of problems is solved by choosing an appropriate path planning algorithm. It has been used to direct the robot toward a specific objective from a simple trajectory planning to a complicated sequence of actions. Since a priori knowledge of the global environment is not always available, path planning cannot always be created in advance. Path planning can be used in partially organized and unstructured situations by presenting a suitable method. In order to keep the robot moving from the start location to the goal location through various intermediate stages, a proper trajectory is constructed that is covering all the in-between locations. Every choice made by path planning algorithms is determined by the information that is currently accessible, as well as by other factors such as the Euclidean distance computation's calculation of the shortest distance to the target point. From the start state to the target location, there may be more than one path. However, there are several circumstances where there is no way to get to the goal location. For better results, the best path must travel the fewest distances, be free of obstacles and collisions, and take the least amount of time to arrive at the desired state. The chosen trajectory must also be smooth and devoid of sharp twists because a robot may be subject to a variety of motion restrictions, like the nonholonomic condition (Klancar et al., 2017). According to (Hoy et al., 2014) path planning can be divided into two categories. The first is global path planning. In this case, the environment is static, and the control design considers global information that is known a priori. This strategy costs a lot to deploy but has received a lot of attention in the literature so far. The second method is local path planning, in which the robot's movement generates the path based on information from its sensors. In order to react to a new environment, the robot needs to generate a new path. Although the design of this system is more complex, it is more practical. In a path planning algorithm, there are four primary factors that must be (Teleweck & Chandrasekaran, 2019). Optimizing is the first. This criterion guarantees that the chosen solution is the best in terms of distance, time, cost, and other factors. The second requirement is completeness, which guarantees that the path planning algorithm will provide a solution if it is feasible. The next is precision and accuracy. To move from the origin to the destination state, this criterion is essential. The execution time is the final requirement. Because robots are real-time systems the time needed for computing is very limited for practical uses.

1

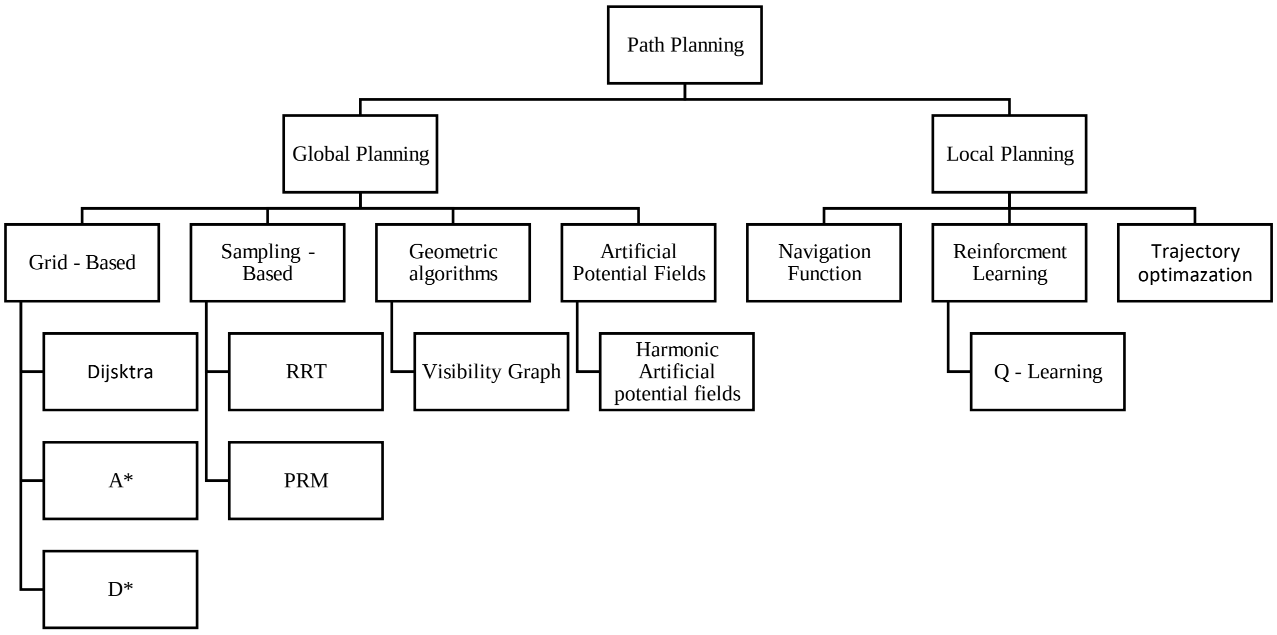


Figure 1.6 Path planning global and local classification

[1] APF can also be consider a local planning algorithm

2 Literature review

When mobile robots gain autonomous capabilities, they become more useful. It enables the removal of human operators, which may be advantageous from an economic and safety standpoint. To create autonomy most of the time path planning algorithms need to be selected and implemented to make the robot move correctly from a start point to an end goal. Because of the vast selection of path planning algorithms that are available in literature many can find this task challenging.

The four path planning categories are discussed below in Figure 2.1, with each category broken down into two subcategories. This classification is based on the underlying ideas and methods that create and return a path. Later a more in-depth explanation of these categories and the reason behind their arrangement will be presented.

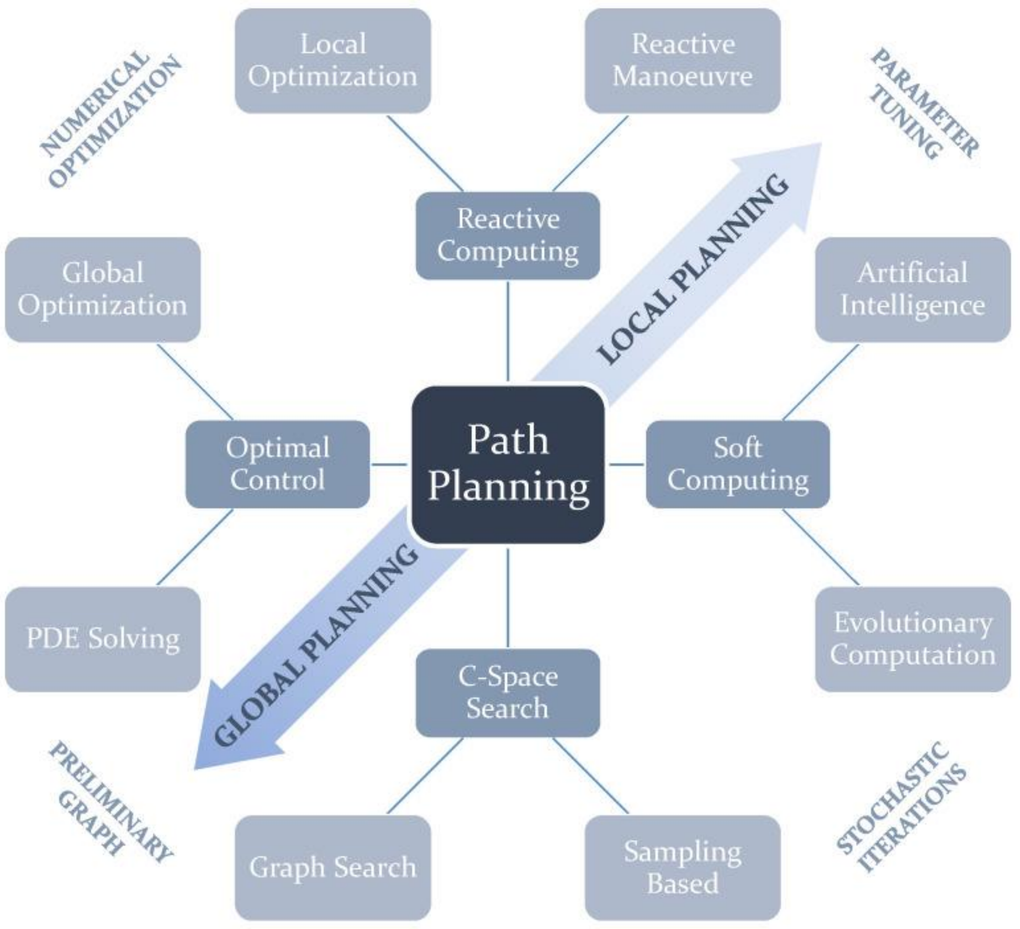


Figure 2.1 Path planning categorization (Sánchez-Ibáñez et al., 2021)

The following classification is based on how algorithms work. Many literature reviews split it into two main categories as stated by (Sánchez-Ibáñez et al., 2021). Which is, if the environment is static or dynamic. When the environment is static the map does not change when time passes. On the other hand, in case the map changes its path in real-time, it is considered as dynamic. The second biggest classification is the one discussed above. Global and local planning. The global planning is usually associated with offline planning and local with online planning. One issue with this classification is that some algorithms are considered both local and global algorithms (e.g. APF). A reactive computing algorithm is best used in local planning, but it is possible also to be used in global planning. Another classification by (Vagale et al., 2021) is between classic algorithms which require a complete map beforehand to work. Such classic algorithms usually are graph search algorithms. The other is Advanced which contains sampling-based algorithms and soft computing. (SOUISSI et al., 2013) offers a number of distinct and logical path planning classifications. When the robot model is the main aspect path planning algorithms are classified to (non-holonomic, holonomic, kinodynamic). When the map is the main aspect (required or not required before the planning). When replanning is the main aspect offline or online. Finally, the last classification proposes is when the algorithm provides always the same solution and it is divided into deterministic and probabilistic.

2.1 Modelling of workspace in path planning

Information about the environment should be fed in the path planner. It can be either the obstacles that are in the environment or features of the surface that are important for the planning. For example, the minimum distance may not be the primary factor. It can be the minimization of the energy that the robot needs to complete the mission based on the terrain or the wind. This type of optimization was very important for the mars rover because of the limited energy that it had available.

2.2 Environment cell decomposition and roadmap graphs

Surface mobile robots traverse a specific area of space by driving from one location to another. Therefore, it is important to think about how the path planner will handle this surface and how the locomotion model will interact with it. For example, some algorithms need the

creation of a graph that is a simplified view of the environment that the robot is moving. There are many ways to build a graph.
This work (Sánchez-Ibáñez et al., 2021) classifies it in Cell decomposition and roadmaps.
The first of these is dividing the surface into cells. Grids that are regular or irregular. Figure 2.2 a–c demonstrates how grid cells are shaped: squares, triangles, and hexagons can be used to construct regular grids. Its key benefit is the straightforward indexation of each node, which enables instant access to any of them and an efficient method of memory storage. But it is possible to provide inferior pathways. Irregular grids, like the one shown in Figure 2.2 d, allow for better adaptability of the grid to terrain features with varied values of resolution.
According to (Nash, 2013), other types of cell decomposition include navigation meshes and circle-based waypoint graphs. Roadmaps are used to describe the environment in a different way, as was already mentioned. A roadmap is a network composed of nodes connected by edges. Each edge shows how to get from one state to another while each node represents a potential state for the robot. Some examples of roadmaps include Visibility graphs, State-Lattice graphs, and Voronoi graphs (Figure 2.2e-f). When using Graph Search methods, like (Likhachev & Ferguson, 2009) work, the latter method involves creating the edges based on the motion constrains, ensuring that the final path is possible given the robot's mobility limitations.
These graphs' cells or nodes can store information about the surface where they are located as static or dynamic elements. For instance, this could be data about elevation. An electronic elevation map (DEM).

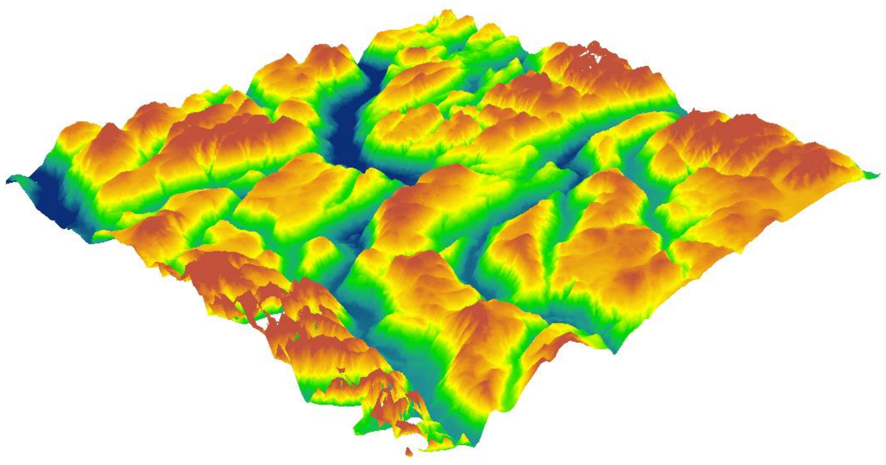


Figure 2.2 Elevation map (5 Free Global DEM Data Sources – Digital Elevation Models, 2022)

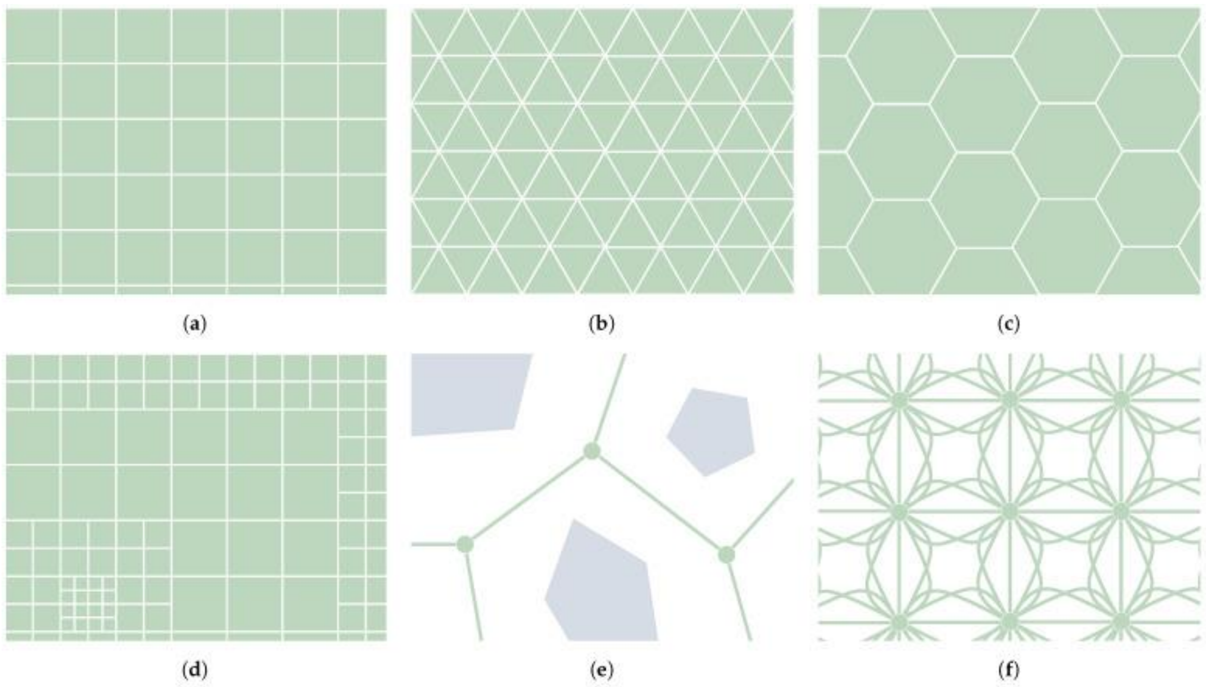


Figure 2.3 Types of grids and roadmaps (Sánchez-Ibáñez et al., 2021)

3 Methodology (Description of proposed Path Planning algorithms)

3.1 Map creation

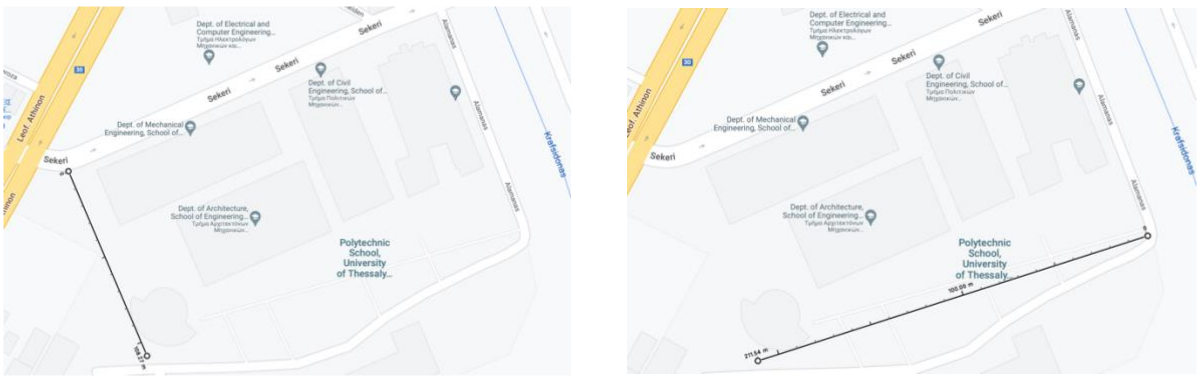
For this case study, the map of the university of Thessaly campus in Pedion Areos will be used. The robot to get the grid map has two options, one is to explore the area completely at least one time to be able to map the university campus, or the other one to load it on the robot because it is known, in this case study the map will be extracted from the google maps using the measuring tool provided in the maps.

To be able to perform the case study as simply as possible to compare the path planning algorithm some assumptions and simplifications must be made. It will simplify the problem but not change it.

Assumptions and simplifications:

- The map is assumed that it does not have unreachable paths and what is presented on the map as not a building is an area where the robot can move freely
- The difference in height in certain areas is not considered
- Buildings are simplified to rectangles and circles
- The distances have an error of ± 1 meter

Map size: 108m * 212m



Dept. of Electrical and Computer Engineering...
Sekeri
Leof. Athinon
Dept. of Mechanical Engineering, School of...
Dept. of Civil Engineering, School of...
Dept. of Architecture, School of Engineering...
Polytechnic School, University of Thessaly
Kratedonos

Figure 3.1 Google maps University of Thessaly campus dimensions (Google, 2022)

Buildings size and location

Mechanical engineering measurements:

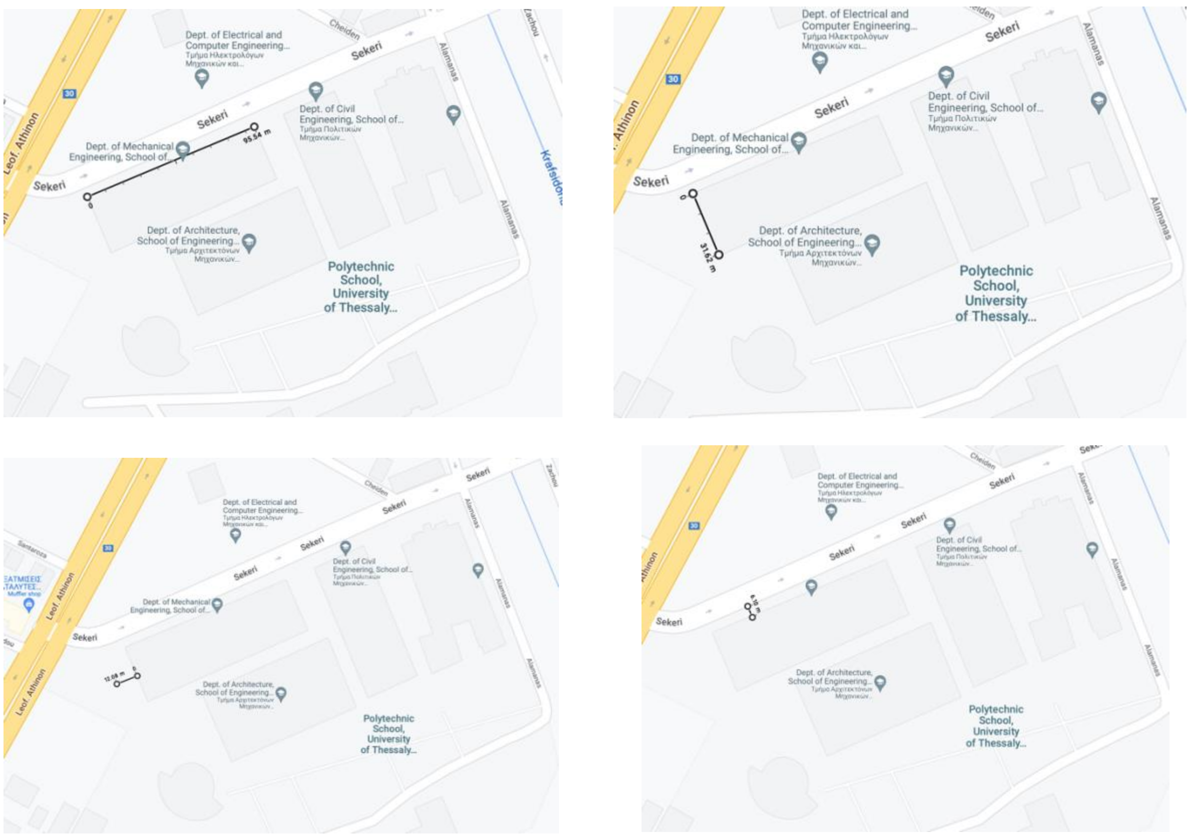

Dept. of Electrical and Computer Engineering...
Dept. of Mechanical Engineering, School of...
Dept. of Civil Engineering, School of...
Dept. of Architecture, School of Engineering...
Polytechnic School, University of Thessaly...
Sekeri
Leof. Athinon
Alamanas
95.54 m
31.52 m
12.58 m

Figure 3.2 Google Maps measurements of mechanical engineering building (Google, 2022)

The same methodology is being followed for measuring the other buildings of the campus like Architecture engineering, Civil engineering, and Urban engineering and the amphitheater.

The map will be a grid map with dimensions equal to the width and height in meters and each grid point represents a square of area 1m^2. The free space will be represented with zeros and the obstacles with ones. By placing ones in the right place of the matrix the shape of the obstacle forms and then you have a grid map that can be used with path planning algorithms. Below is an example with a rectangle obstacle in the centre.

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.1)$$

When the obstacle has the shape of a circle the equation of the circle can be used to find which grid points are inside the obstacle and equal them with 1.

$$t = (x - 13)^2 + (y - 13)^2 \leq 11\text{^}2 \quad (3.2)$$

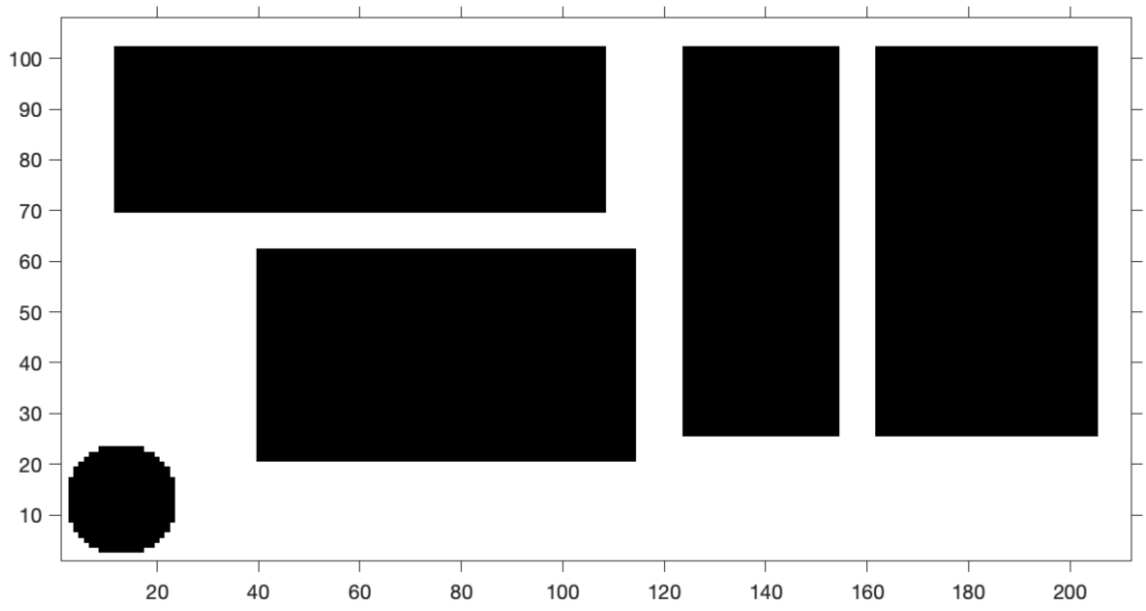


Figure 3.3 Grid map created with zeros and ones

Graph and Trees

According to (Lynch & Park, 2017) frequently path planning algorithms explicitly or implicitly represent the C-space as a graph. The graph is made with a number of nodes and several edges that connect two nodes. The node represents a point in space and the edge a path that connects the two nodes without passing through an obstacle or to have violated a constraint.

The graph it is possible to be directed or undirected. In a directed graph every edge is one-way but two different edges can be present between nodes in opposite directions. In an undirected graph, each edge is bidirectional.

There are weighted and unweighted graphs. Every edge in a weighted graph has a cost that corresponds to a positive number which represents something like meters or the cost of gas. In an unweighted graph, each edge has the same cost so there is no reason to have weights for each edge. For example, each edge has a cost of 1.

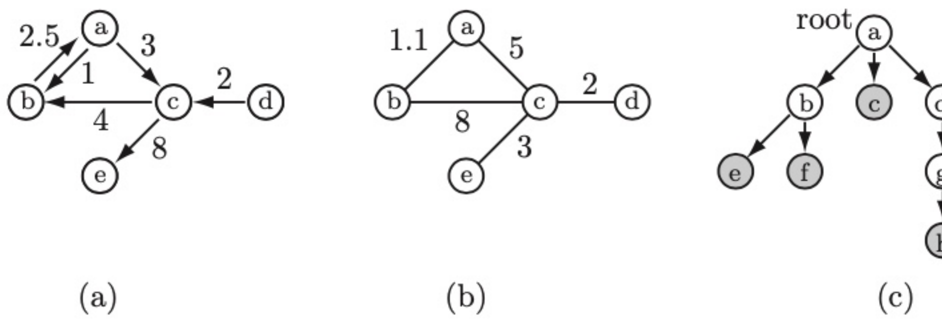


Figure 3.4 (Left) Directed weighted graph (Center) Undirected weighted graph (Right) Directed unweighted graph (Lynch & Park, 2017)

Graph search

When a search algorithm possesses the optimality property, it indicates that the path is guaranteed to be the most optimal. When a search algorithm possesses the completeness property, it guarantees that the algorithm will always discover the path. This is stated by (Roy, 2021)

3.2 A* search

One of the best search methods for locating the shortest path between graphs is A-star, commonly known as A*. It uses information about path cost and heuristics to find the path. That’s why it is also called an informed search algorithm. It is an optimal and complete algorithm.

A* uses a formula each time it visits a node to find the cost related to that node and then until the desired point is reached, it enters the node with the lowest value..

$$f(n) = g(n) + h(n) \tag{3.3}$$

- g(n) is the distance that the node has from the starting point , by following the path that the algorithm created until that step
- h(n) is the heuristic, which is the estimated distance from the node to the goal node without taking into account obstacles.

That's why heuristics are estimated guesses. As the algorithm doesn't really know the exact map before it calculates the g(n) cost. Many argue that A* falls into the category of artificial intelligence because it is like a thinking brain.

To calculate distance and create a cost for each node there are two distinct ways. The Manhattan distance which is going only in straight lines and the Euclidean which the diagonal way is possible

$$D_{manhattan} = |x_1 - x_2| + |y_1 - y_2| \tag{3.4}$$

$$D_{euclidean} = ((x_1 - x_2)^2 + (y_1 - y_2)^2)^{\frac{1}{2}} \tag{3.5}$$

By choosing the type of formula for the distance cost you also choose how the robot can behave in the map. For example, if the Manhattan distance is chosen then the robot cannot move diagonally. But if Euclidean distance is chosen then the robot can move diagonally.

Table 3.1 A pseudo code (GeeksforGeeks, 2022)*

```
Algorithm: A*
 1: Create the open list
 2: Create the close list
 3: Insert the starting node in the open list with f = 0
 4: while the open list != empty do
 5:   find the node with min(f) in open list and make it q
 6:   Drop q from open list
 7:   Create q eight successors and set their parents to q
 8:     for each successor
 9:       If successor == goal
10:         stop search
11:       else calculate g and h for successor
12:       end if
13:       if a node position == successor position is in the open list
14:       which  a lower f than successor, ignore this successor
15:       end if
16:       if  a node position == successor position is in the closed list which
17:       was a lower f than a successor, ignore this successor,
18:       else add the node to the open list
19:       end if
20:     end for
21: end while
```

Sampling methods

All grid-based methods give an optimal solution based on the discretization of the map. That leads to increased computational complexity when the degree of freedom increases which makes them almost useless for applications that have more degrees than $\mathcal{R}^2$. A different approach for planners is sampling methods that use random or deterministic functions to select a sample in the C-space or state-space as mentioned by (Lynch & Park, 2017). A function to check if the sample that was created is in X_{free} . A function to find prior free-space samples in the vicinity, and a basic local planner to link to or move toward the new sample. These routines are used to create a graph or tree that represents the robot's possible moves. Sampling methods in general they trade resolution accuracy of the solution to gain computational speed for finding satisfying solutions. Most sampling procedures are probabilistically complete: as the number of samples grows to infinity, the likelihood of discovering a solution, if one exists, approaches 100%.

3.3 RRT

The basic RRT algorithm uses a single tree that is expanding toward the x_{goal} from the x_{start} while also avoiding the obstacles in the way.

Table 3.2 RRT pseudocode (Lynch & Park, 2017)

```
Algorithm: RRT
 1:  create search tree T with $x_{start}$
 2:  While T < max(Tree size) do
 3:    Create $x_{samp}$
 4:    find $x_{nearest}$ in the tree to $x_{samp}$
 5:    Use a local planner to create a path from $x_{nearest}$ to $x_{new}$ in
       the direction of $x_{samp}$
 6:    If path does not cross through an obstacle then
 7:      add $x_{new}$ to the tree by creating and edge from $x_{nearest}$ to $x_{new}$
 8:      if $x_{new} == x_{goal}$ then
 9:        return 0
10:      end if
11:    end if
12:  end while
13:  return 1
14:  Find path with A*
```

Usually, the sampler at (line 3) chooses x_{samp} nearly uniformly distributed with a slight bias towards the x_{goal}. The $x_{nearest}$ node which is close to the search tree T (line 4) is the node minimizing the Euclidean distance from x_{samp} . A simple local planner (straight line motion) (line 5) finds a path that connects $x_{nearest}$ to x_{new}. If the path does not cross any obstacle is added to the search tree T. The total effect of the slight bias on the nearly uniform distribution sample is that it pushed the tree towards them, resulting in fast exploring x_{free} .

The algorithms are adjustable in three main parts: how to sample from X, how to select the " nearest" node in T, and how to find the path to develop the tree towards the x_{samp}. Even a slight modification to the sampling method, for example, can result in a completely different running time of the method. There are a big variety of planners based on RRT like: RRT* , Bidirectional RRT etc.

RRT*: An improved version of RRT which results in a more optimal path because of the continues improvement of the tree

Bidirectional RRT: A faster exploring tree starting from the start and the end point at the same time and meeting at the middle

3.4 PRM

Before addressing any specific queries, the PRM employs sampling to create a roadmap representation of C_{free}. The roadmap is an undirected graph, which means that any direction of the edge can be taken by the robot. As a result, PRMs are most useful for kinematic issues in which an accurate local planner can discover a path (not considering obstacles) from any q_1 to any q_2. The simplest example is when there is no kinematic constraints for the robot a straight line planner is used.

After the undirected graph is completed, a q_{start} node is added and connected with the closest nodes in the graph. A similar approach is present with the q_{goal} node. When the graph is fully completed with the start and end node a graph search algorithm is used to find the most optimal path. Typically, A* is used to find the path optimally and efficiently.

An important topic in PRM roadmap-construction algorithm is how to sample C_{free} . The most common is sampling in a uniform distribution on C and deleting samples that are in obstacles. It is known that sampling more densely near the obstacles increases the possibility to go through narrow passages which results in higher efficiency of the algorithm but it quite complicated to implement. One more option is deterministic multi-resolution sampling.

Table 3.3 PRM pseudocode (Lynch & Park, 2017)

```
Algorithm: PRM
 1:  for i = 1, . . . , N do
 2:    Create sample point q_i
 3:    Insert sample point in roadmap list R
 4:  end for
 5:  for i = 1 , . . . , N do
 6:    N(q_i) = j closest neighbors of q_i
 7:    for each q ∈ N(q_i) do
 8:      if path does not cross obstacle from q to q_i
         And no previous edge exist then
 9:        insert an edge from q to q_i into roadmap R
10:      end if
11:    end for
12:  end for
13:  return Roadmap
14:  Find path with A*
```

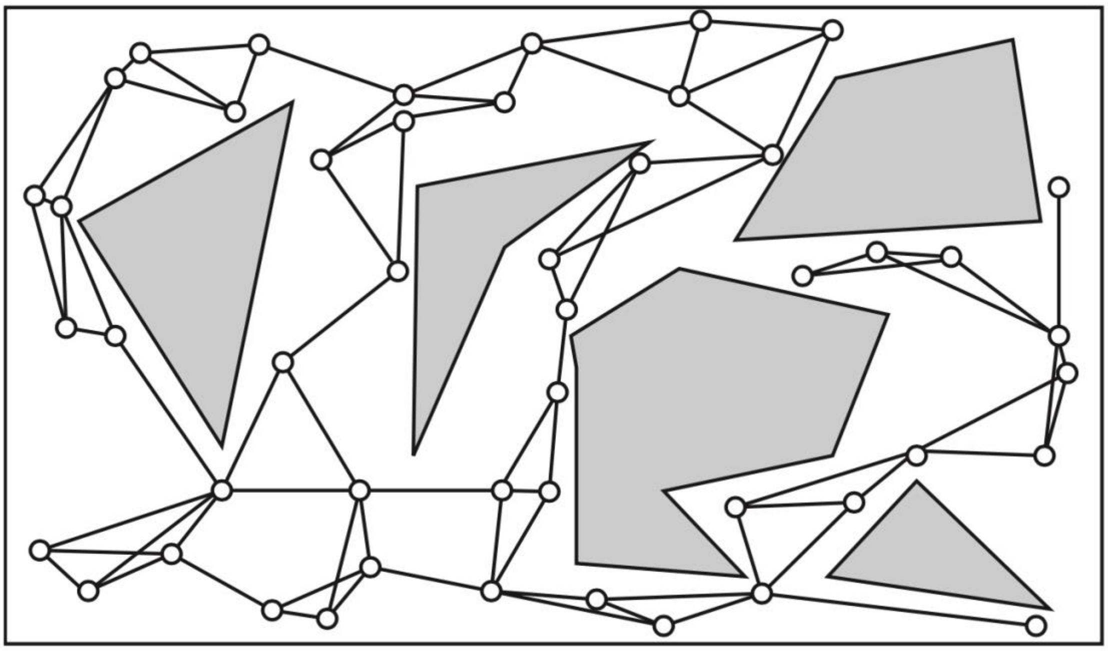


Figure 3.5 A roadmap in 2 dimensions (Lynch & Park, 2017)

3.5 Artificial potential fields

Artificial potential fields are influenced from potential energy fields that exist in nature, like gravitational and magnetic fields. A potential field $P(q)$ defined over C produces a force $F = -\frac{dP}{dq}$ that pushes an object from high to low potential. If the gravitational potential field is taken as an example with $g = 9.81\ m/s$ and an object with mass m at a height of h then the potential energy is $P(h) = mgh$ and results in an acting force which is $F = -\frac{dP}{dh} = -mg$. The mass will plummet to the Earth's surface due to the force.

In robot path planning, the obstacles which are defined in the map are getting assigned a high virtual potential and the q_{goal} a low virtual potential which results in a potential difference and a force is created equivalent to the negative gradient of the artificial potential which is pushing the robot into the goal and while avoiding the obstacles. The artificial potential field differs from other planning algorithms that exist. Because the field’s gradient can be calculated very fast It allows the planning to be calculated in real-time instead of calculating it in advance. The approach can even deal with obstacles that move or appear unexpectedly if adequate sensors are used.

The basic method has the issue of causing the robot to become stuck in a local minima that is created even if the goal has a feasible path. Furthermore, two more problems arise from this approach. First, when obstacles are too close to each other they tend to close the path due to the overlapping negative potentials which result in an “unfeasible path” that in reality, the robot can easily pass through. Second, when you have an obstacle next to the goal the repulsive potential doesn’t allow the robot to reach its goal. These problems are under investigation with many researchers trying to provide their solutions on the matter like this one is proposed here (Yujiang & Huilin, 2017).

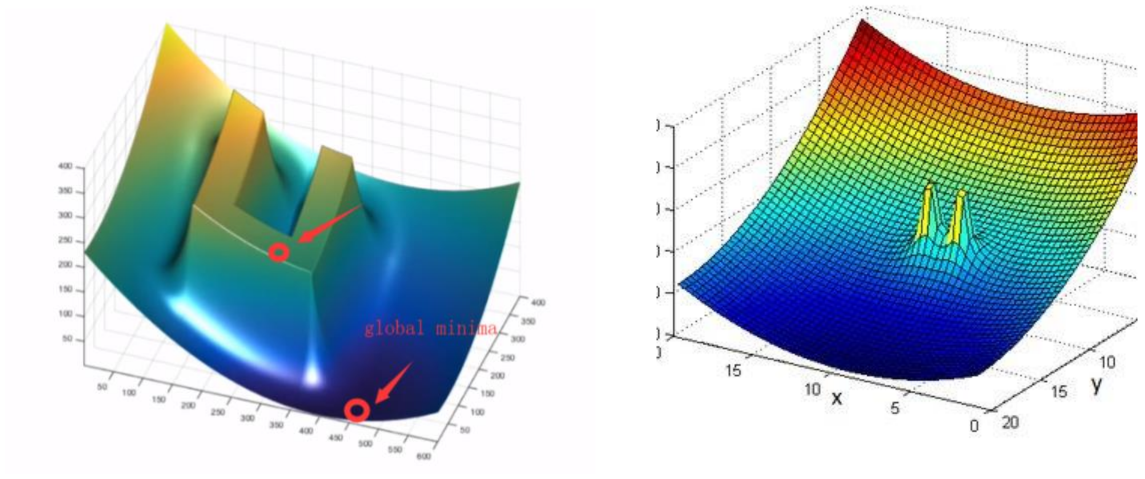


Figure 3.6 Obstacle that creates a local minima on the left and two obstacles that they overlap in the right (Triharminto et al., 2016)

Let's select a point robot in C-space. A q_{goal} is applied in a quadratic potential energy "bowl" with zero energy assigned at the goal,

$$P_{goal}(q) = K\left\|q - q_{goal}\right\|^2 \tag{3.6}$$

where K is simply a constant scaling parameter.
The force created by the potential is

$$F_{goal}(q) = -\frac{\partial P_{goal}}{\partial q} = 2 * K(q_{goal} - q) \tag{3.7}$$

an attractive force potential in proportion to the goal's distance.

The repulsive potential created by C – obstacle B can be calculated from the distance $d(q, B)$ to the obstacle

$$P_B(q) = \frac{k}{2d^2(q, B)} \tag{3.8}$$

where k>0 is a scaling factor. Only for points outside the obstacle is the potential properly defined $d(q, B) > 0$. The obstacle creates the following force

$$F_{b(q)} = -\frac{\partial P_B}{\partial q} = \frac{k}{\partial^3(q,B)} * \frac{\partial d}{\partial q} \tag{3.9}$$

The total potential is calculated by adding the attractive goal and repulsive obstacle potentials together.

$$P(q) = P_{goal}(q) + \sum_i \left(P_{Bi}(q)\right) \tag{3.10}$$

creating a total force of

$$F(q) = F_{goal}(q) + \sum_i F_{Bi}(q) \tag{3.11}$$

It is important to know that the combination of the attractive and the repulsive potential does not guarantee that will result in a minimum (zero force) exactly at q_{goal} .

Also, because the simple obstacle potential would normally provide unlimited potentials and forces near the edges of obstacles, a maximum potential is tactic that is frequently used.

By using the simple obstacle potential, obstacles that are far away from the robot have also influence. To eliminate this and add speed up the algorithm the distance is defined where if it exceeded the obstacle will be ignored. A range of influence of the obstacles $d_{range} > 0$ so that the potential is zero for all $d(q,B) \geq d_{range}$

$$U_B(q) = \begin{cases} k\left(\frac{1}{d(q,B)} - \frac{1}{d_{range}}\right)^2 & if\ d(q,B) < d_{range} \\ 0 & otherwise. \end{cases} \tag{3.12}$$

Gradient descent

Gradient descent is an iterative first-order optimization algorithm that is used to find the min and the max of a function. It is widely used in mechanical engineering, control engineering and computer games. Gradient descent has two strict limitations that must be taken into account. The function needs to be differentiable and convex.

To calculate the gradient for a univariate function a simple first-order derivative is calculated at the given point. If the function is multivariate then it is a vector of derivatives in all the axes.

$$\nabla U(q) = \begin{pmatrix} \frac{du(q)}{dx_1} \\ \frac{du(q)}{dx_2} \\ \cdots \\ \frac{du(q)}{dx_n} \end{pmatrix} \tag{3.13}$$

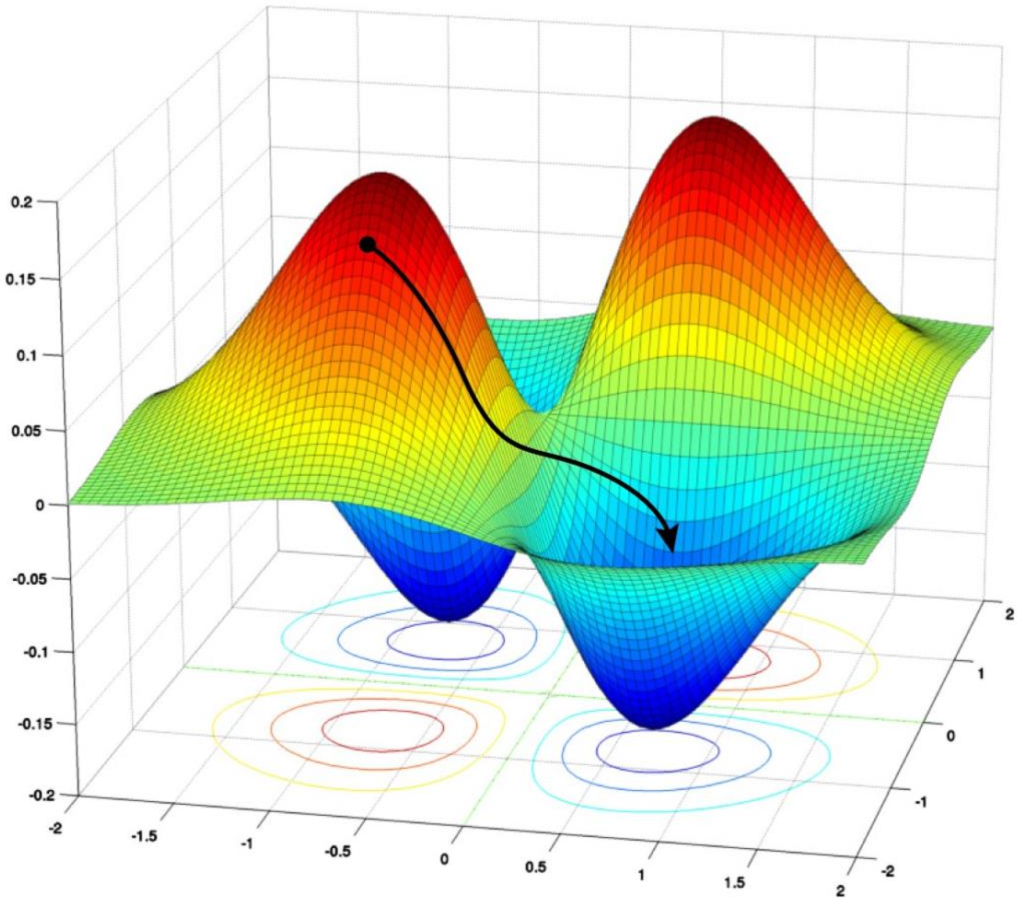


Figure 3.7 Gradient Descent example (Pandey, 2022)

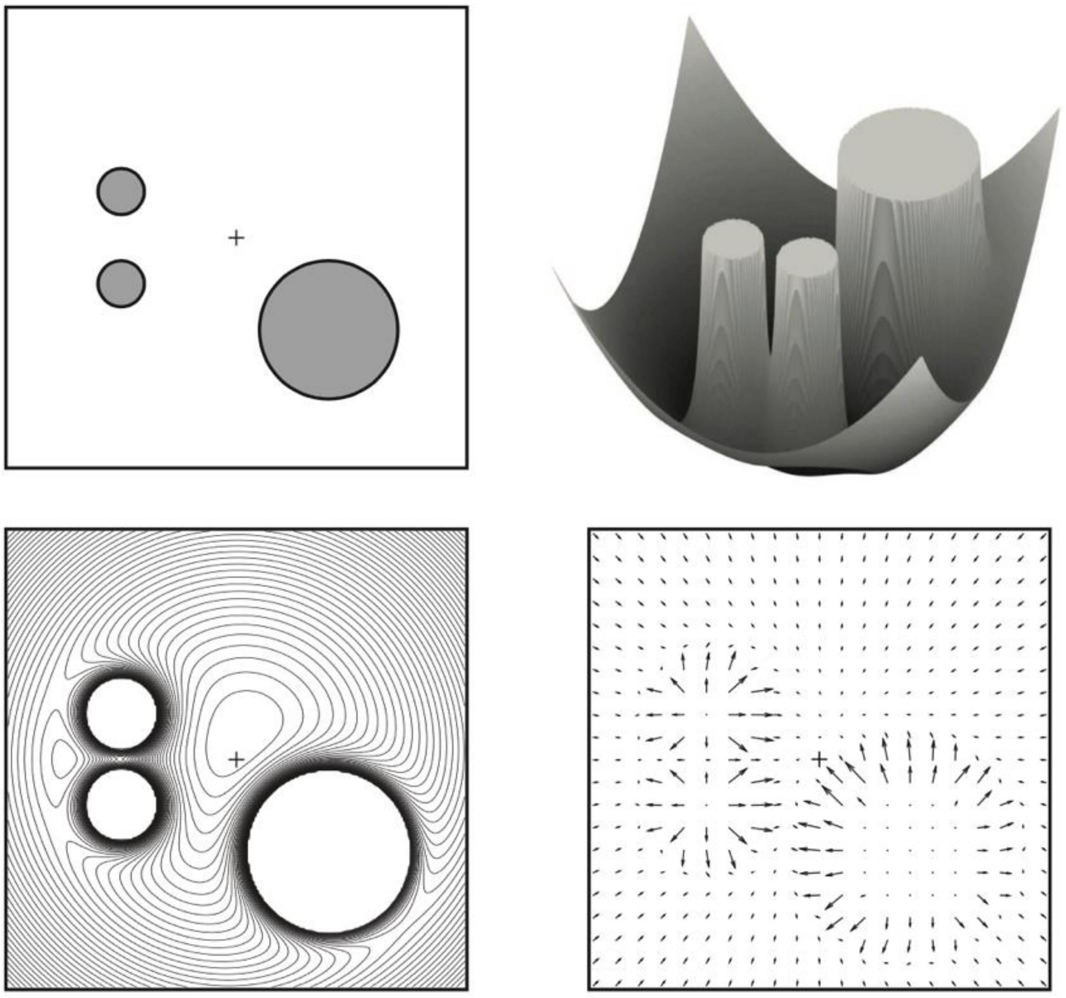


Figure 3.8 (Top left) The 2d grid map has three obstacles and goal point which is denoted a +. (Top right) The summed attractive and repulsive potentials which results in this mountain like view (Bottom left) A Contour plot of the potential (Bottom right) Force vector field (Lynch & Park, 2017)

Table 3.4 APF pseudocode

```
Algorithm: Artificial Potential fields
1:   While position != goal then
2:       Determine distance from the obstacle
3:       Measure distance from the goal
4:       Calculate attractive potential
5:       Calculate repulsive potential
6:       Calculate total potential
7:       Calculate Gradient
8:   end
```

4 Application/Case Study

For this thesis, the three scenarios will be studied:

1. On university campus with a starting point at (5,80) and a goal point at (118,45)
2. On university campus with a starting point at (160,90) and a goal point at (35,35)
3. On a simple map with a circle obstacle with a starting point at (5,80) and a goal point at (200,20)
4. A special scenario in the university campus for APF in which local minima does not occur, with a starting point at (5,80) and a goal point at (200,20)

The reason that a special scenario is created for APF is that no other starting point and goal point can provide a feasible path due to reasons that will be explained later. The other algorithms are not applied in this scenario because the APF although it works it does work as it is intended to. The comparison happens in scenario 3 which APF can have acceptable results.

Scenario 2 and 3 results can be found in the appendix. Only scenario 1 will be compared tottaly

4.1 Application of A*

As mentioned before A* has some factors that need to be decided. First, the distance calculation needs to be chosen. Since the robot needs to move diagonally to have more options in the grid map a Euclidean distance formula is chosen for the calculation of the cost for a transfer from each node. Then by applying the A* on the grid map the following results are obtained

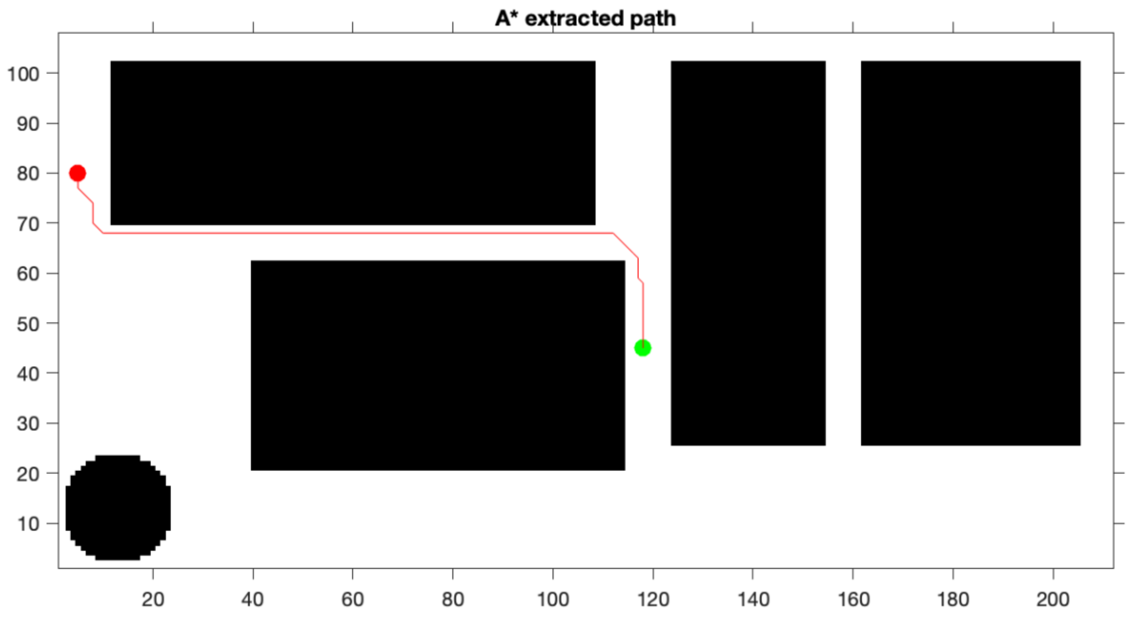


Figure 4.1 Scenario 1 A path*

Table 4.1 A results*

Scenario	Path length [m]	Time to process [s]
1	141.55	0.27
2	194.52	0.476
3	219.85	1.35

Advantages and Disadvantages of the method

One of the main advantages of the A star is that it will always provide the most optimal path for a given start point and a goal point. By having this advantage, it provides confidence in a robotics application that the path planning algorithm will provide the best possible path for the given environment.

Another advantage is that A star except for providing the most optimal path also never fails to execute due to the deterministic nature of the algorithm. That can be quite important in search and rescue missions where robots need to always work properly.

While having these upsides it also has its disadvantages one of which is that the algorithm is not able to replan fast without reconstructing the whole map. This is a big disadvantage when the environment is dynamic, and obstacles are moving. For this specific problem, a new improved version of A star is there which is called D star. Where it can replan fast in dynamic environments

One of the biggest disadvantages of A star and generally graph search methods is that they are exponentially slower when the degrees of freedom are increased and when the griding needs to be finer. That results in limiting the use of A star in high degree of freedom environments. That’s where sampling methods are the best and preferred.

4.2 Application of RRT

RRT's fundamental idea is actually fairly simple. Random Points are created and connected to the nearest node that is accessible. A verification that the vertex lies outside of an obstacle must be done each time a vertex is generated. Chaining the vertex to its nearest neighbor must also avoid obstacles, as well. When a node is produced within the goal region or a limit is reached, the algorithm is finished. One of the most important parameters of the algorithm is randomness which greatly affects the results.

When the randomness factor has a high value then the exploring of the map is more and that allows the algorithm to be able to replan fast when needed by using the existing tree. That comes with the computational cost of the initial exploration but with the reduced cost for replanning. On the other hand, when the randomness factor has a low value the initial exploration computational cost is low but with the expense of limited replanning capabilities. In the following figure, it can be clearly seen the effect that the randomness factor has. Because the method is not deterministic a statistical analysis is required to obtain the results in a meaningful way.

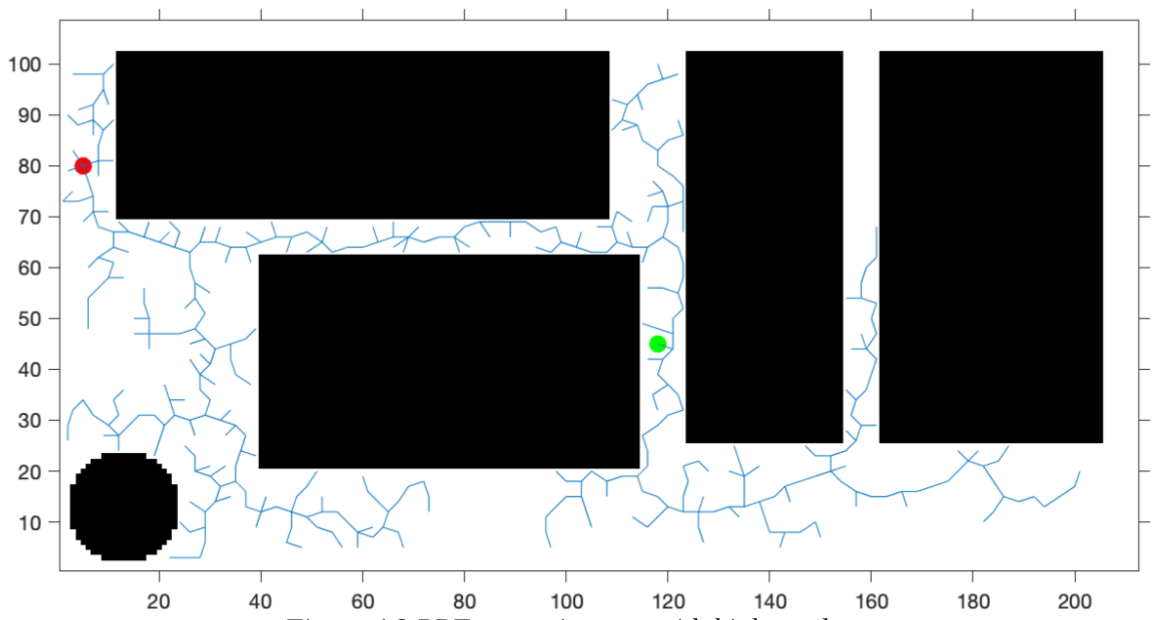


Figure 4.2 RRT expansion tree with high randomness

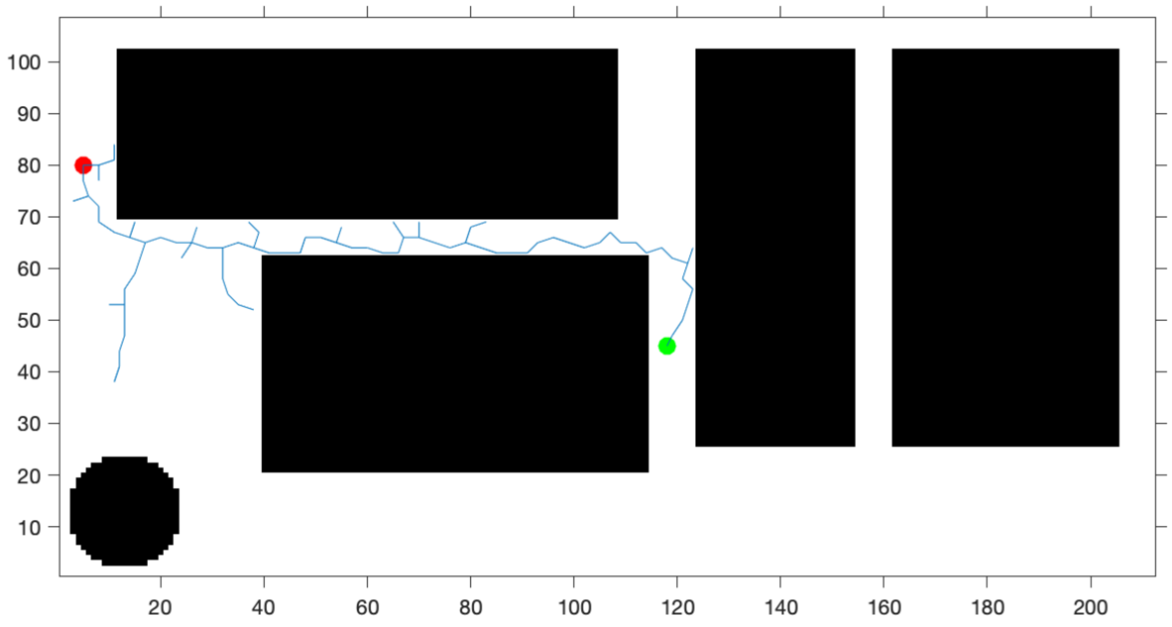


Figure 4.3 RRT expansion tree with low randomness

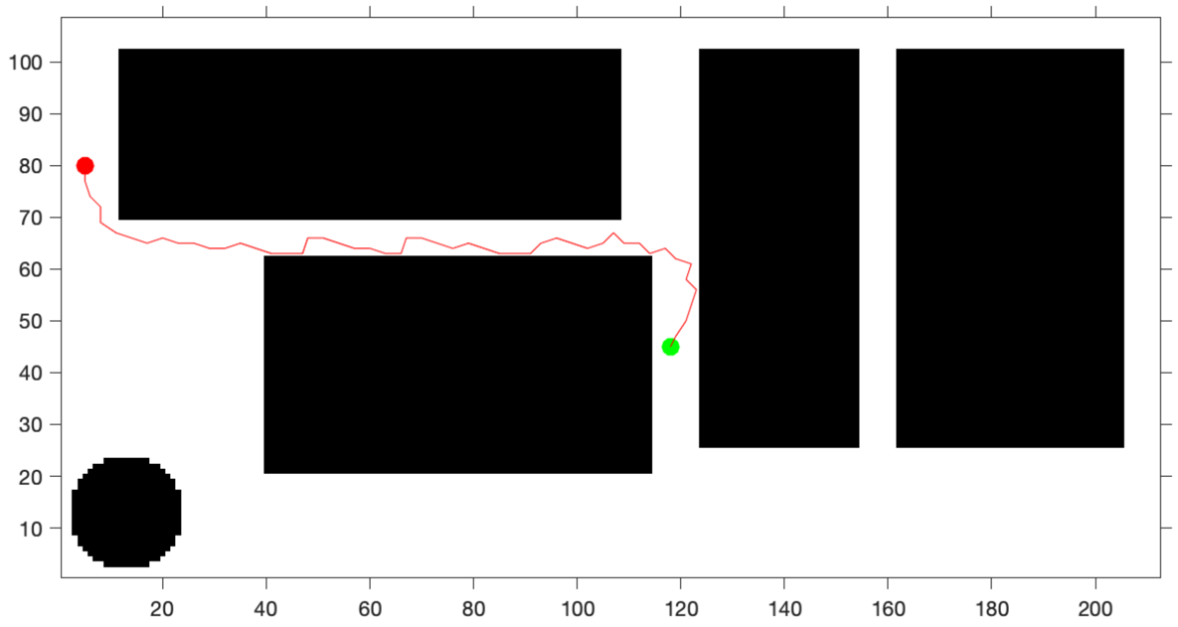


Figure 4.4 RRT path results

To acquire the path length and time to process off the algorithm, 100 runs will take place and the mean and standard deviation will be calculated.

Table 4.2 RRT results

Scenario		**Path length [m]**	**Time to process [s]**
1	Mean	160.18	0.026
	Standard deviation	12.11	0.046
2	Mean	225.20	0.033
	Standard deviation	24.27	0.032
3	Mean	257.89	0.064
	Standard deviation	12.49	0.064

Advantages and Disadvantages of the method

One of the most important advantages of the RRT algorithm is the low processing time to process a path. Due to its probabilistic nature, the algorithm has a significantly reduced computational cost which is one of the best of path planning algorithms.

It is worth mentioning that RRT can work exceptionally well in a high degree of freedom environments compared to graph search algorithms. That’s why RRT and general sampling methods are preferred when the degrees of freedom of the environment are high.

Although RRT has some drawbacks. The path that is produced from the algorithm is not guaranteed that it will be optimal. This must be taken into consideration if the optimality of the path is a top priority. RRT star is created to address this issue.

Due to the probabilistic nature of the algorithm, it is possible that sometimes will fail. It will not be able to provide any path. That’s why for critical robotics missions this may not be the best option

4.3 Application of PRM

By applying the PRM algorithm that is presented above, a roadmap is created for the proposed case studies. This will happen by choosing a number of 60 points to be sampled in the free grid map and all nodes connect to all the other nodes that can create feasible connections. A roadmap of 100 sample points will be shown also. Then A* algorithm computes the optimal path. Because the method is not deterministic a statistical analysis is required to obtain the results in a meaningful way.

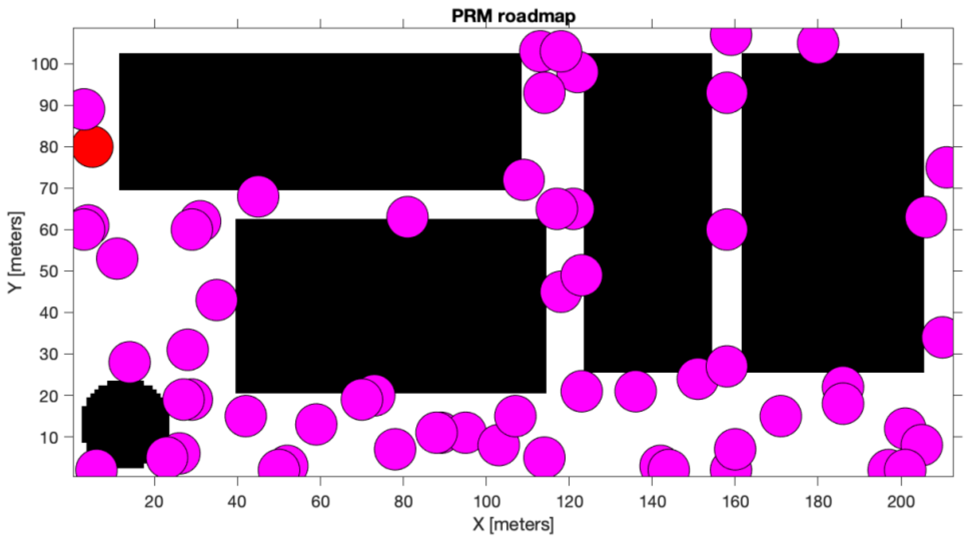


Figure 4.5 60 sample points on the map in pink

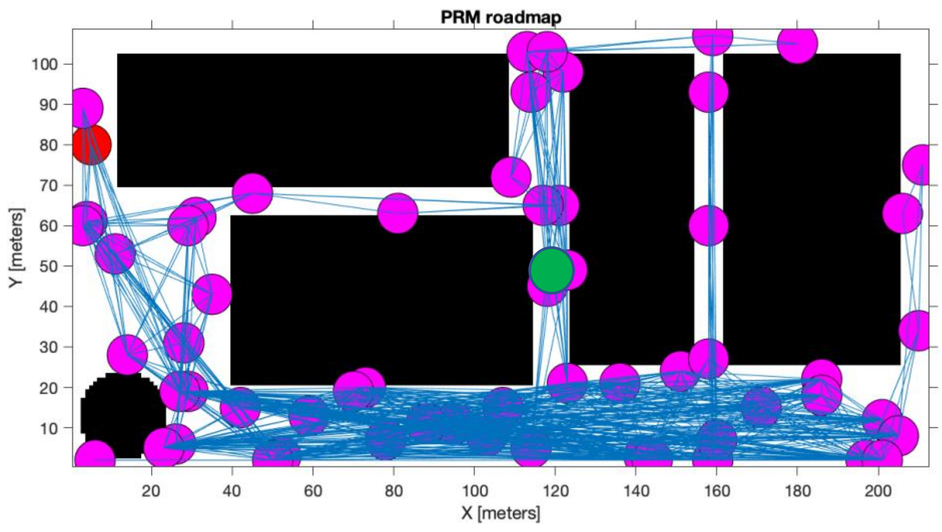


Figure 4.6 Creation of the connections on 60 sample points (Roadmap)

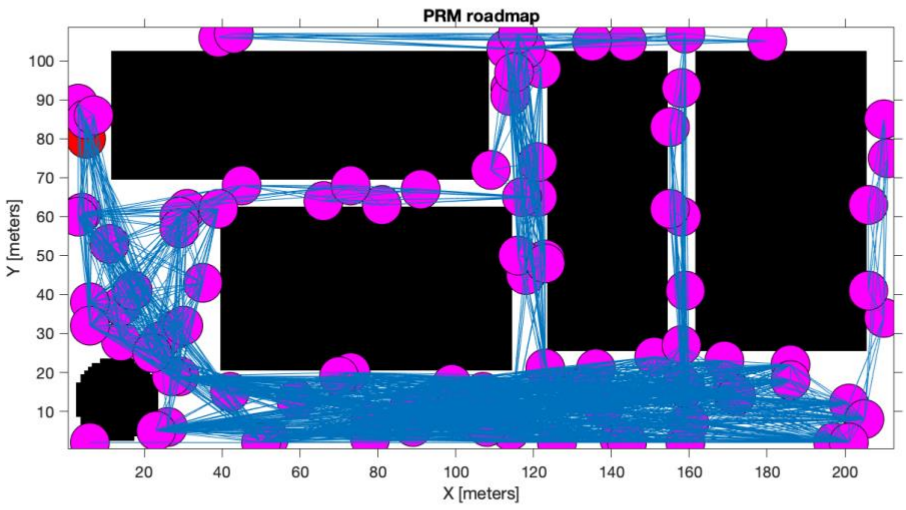


Figure 4.7 Creation of the connections on 100 sample points (Roadmap)

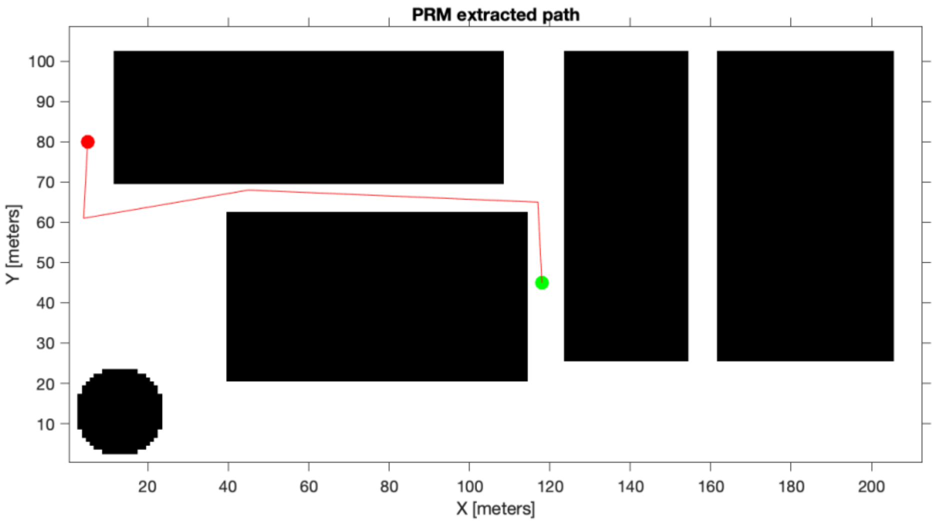


Figure 4.8 Optimal path of the algorithm calculated on 60 sample points

Statistical analysis of the results

To acquire the path length and time to process off the algorithm, 100 runs will take place and the mean and standard deviation will be calculated.

Table 4.3 PRM results

Scenario		Path length [m]	Time to process [s]
1	Mean	172.6	0.0608
	Standard deviation	19.6	0.02
2	Mean	221	0.065
	Standard deviation	11.6	0.0131
3	Mean	206.33	0.175
	Standard deviation	0.73	0.027

Advantages and Disadvantages of the method

One of the most important advantages of this method is that because is very simple and uses sampling it can compute extremely fast which is making it able to run the algorithm in real-time online on a robot.

The roadmap that is created by this method for a specific map that is static, allows the robot to change easily the start point and goal point and calculate a new path extremely fast without the need to rebuild a roadmap. That's why this algorithm is very efficient when you need a lot of replanning in the same map

On the other side when the map has a lot of narrow passages even with a very high number of sample points it is possible that will not be able to find a path because of the collision checking with the obstacles according to (Khokhar, 2021).

Another downside is that the algorithm is not always guaranteeing an optimal path which may result in unnecessary movements of the robot

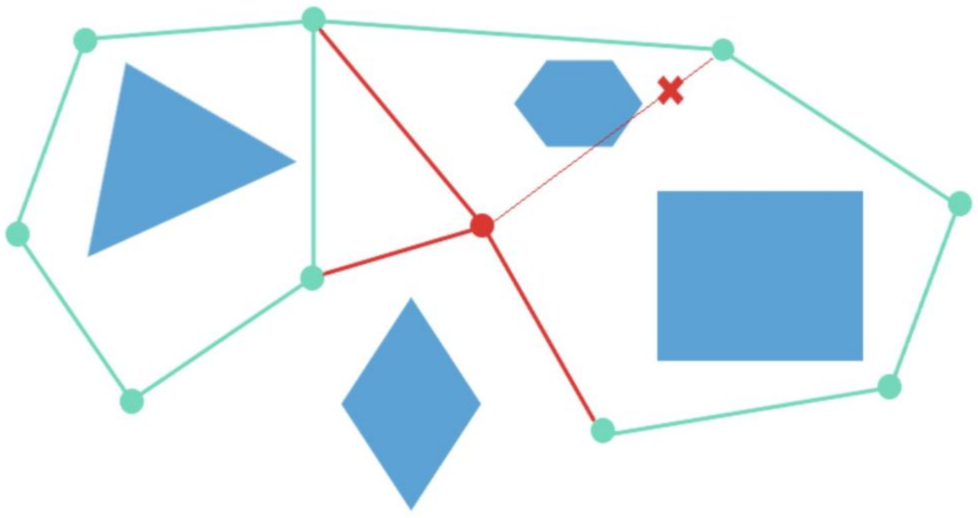


Figure 4.9 Collision check example (Khokhar, 2021)

4.4 Application of Artificial Potential fields

To be able to apply this algorithm to the case study all the parameters should be decided to give the best possible result given the environment. This is a highly complicated task because many factors should be considered. Usually, there are two different methods to achieve that. The first one is choosing parameter values by trial and error and adjusting accordingly based on the results you get. The second method is to use optimization to decide the parameters of the problem based on the constraints you have (Li et al., 2011). For this case study trial and error, method is used.

The first parameter that will be selected to manipulate its value is K which is the scaling factor of the attractive potential. By increasing this factor, you increase the slope of the potential which results in greater “force” on the robot. By decreasing it you achieve exactly the opposite. It may seem at a first glance that increasing the artificial potential as much as possible is a good strategy but if the attractive potential is bigger than it should then it would lead to overcoming the obstacle’s repulsive potential and leading to an unfeasible path. Furthermore, it can introduce periodic phenomena which result in bad path smoothness.

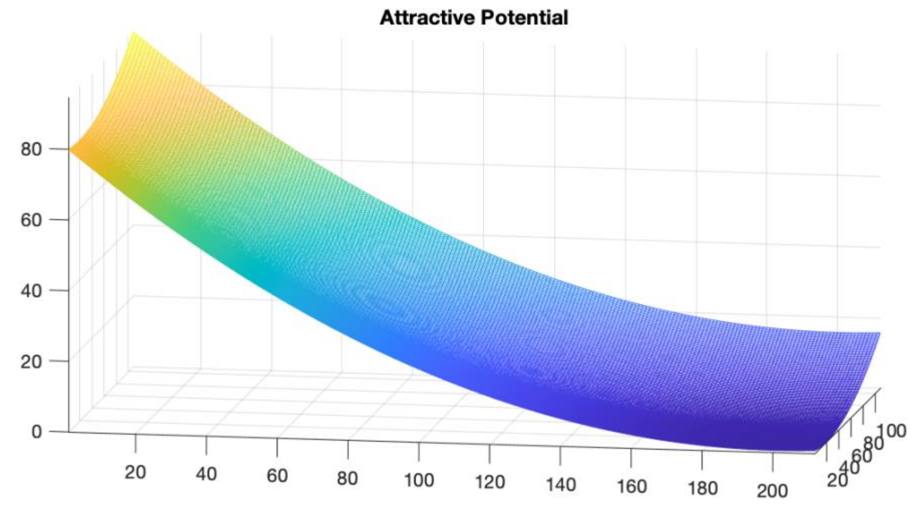


Figure 4.10 Attractive potential with K = 0.002

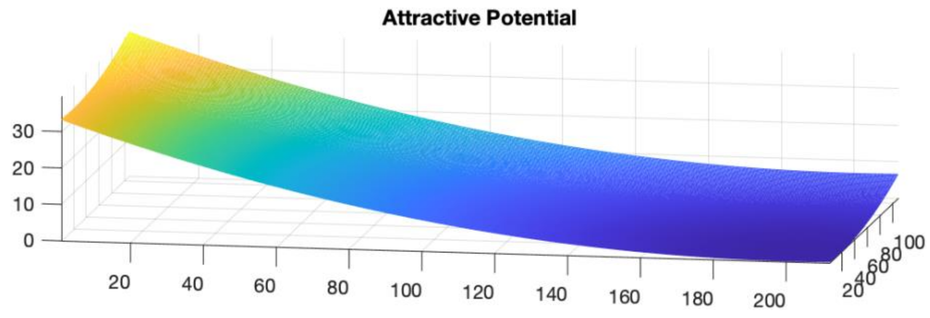


Figure 4.11 Attractive potential with K = 0.000833

The next parameter that will be studied is k which is the scaling factor of the repulsive potential. When this factor is increasing then also the repulsive potential increases that's why bigger values lead to 'taller obstacles and smaller values lead to 'shorter obstacles'. The main goal here Is to find the most optimal value so the robot can move freely without having any artificial problems. Firstly, it is very important to don't make the value too small because that leads to a very small repulsive potential which means that the obstacles don't have the required 'height' to repel the robot and it can go through them as in the case with the highly attractive potential factor. This should not lead to the conclusion that the factor should take very high values because if the values are higher than the optimal then the path will lose its smoothness due to the steeper slope that it will be created at the base of the obstacle.

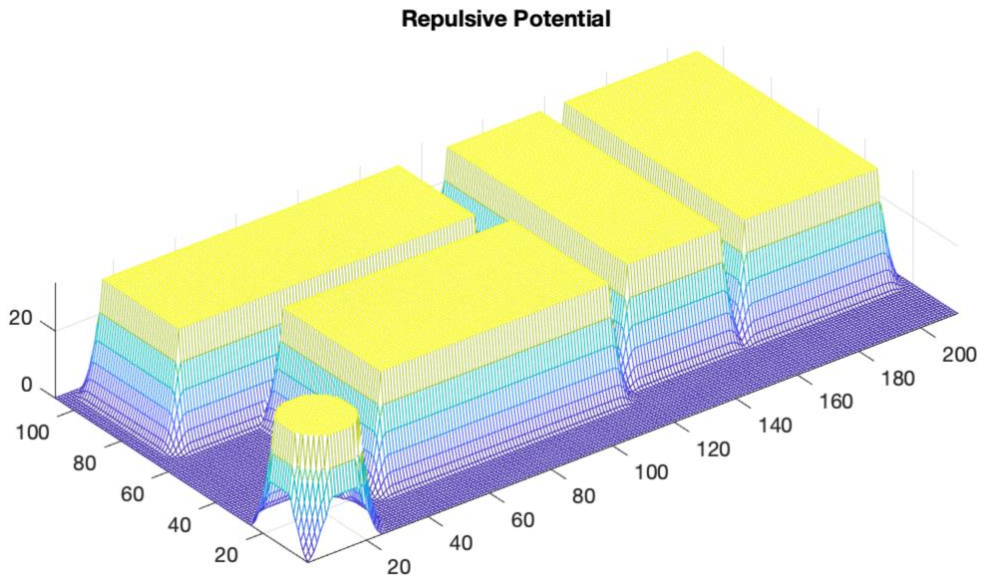


Figure 4.12 Repulsive potential with k = 8000

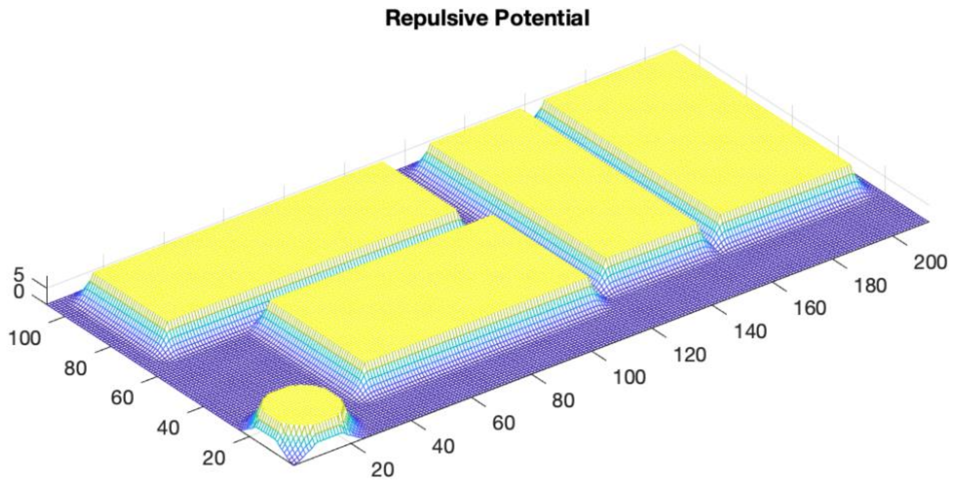


Figure 4.13 Repulsive potential with k = 2000

The last parameter that will be studied is the d_{range} which is the distance that affects the width of the repulsive potential. By having large values means that the obstacle repulsive potential will have a wider affect and while it may give a smoother path it is possible to overlap the repulsive potential in a close gap area and block the entrance. By having small values, the repulsive potential is steeper and if it is too steep the path it can have very bad smoothness. The goal again is to find the optimal value that will result in a good path. The parameters that give the best results for the given start and end goal after the trial and error procedure is :

Table 4.4 Trial and error factors

K	0.002
k	8000
d_{range}	1.07

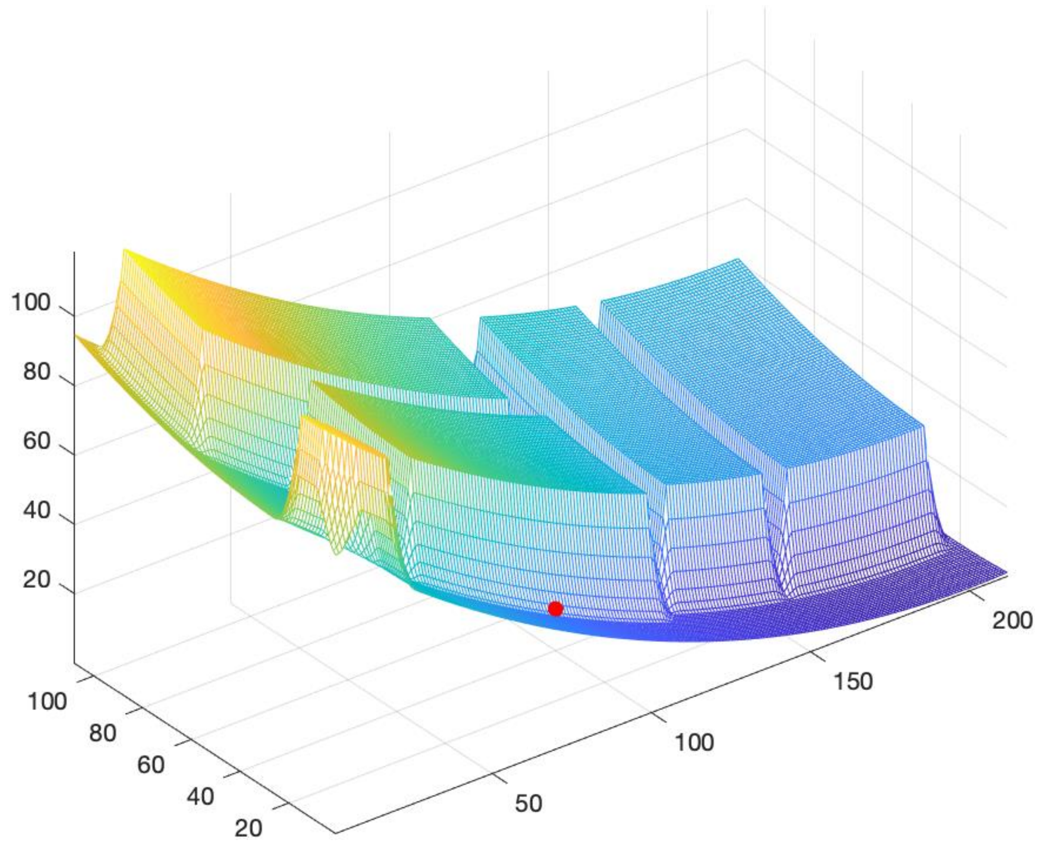


Figure 4.14 Gradient decent algorithm visualized with a ball rolling down

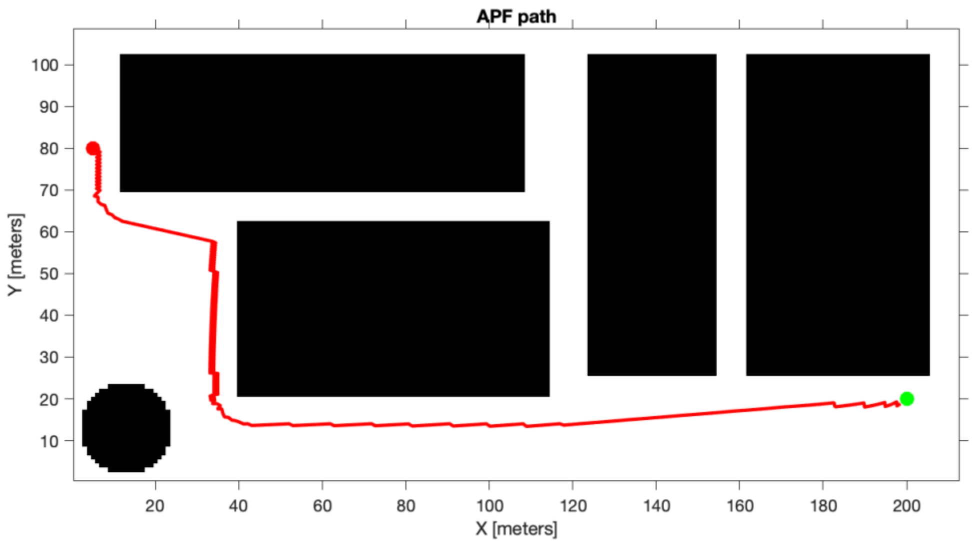


Figure 4.15 Special Scenario APF extracted path

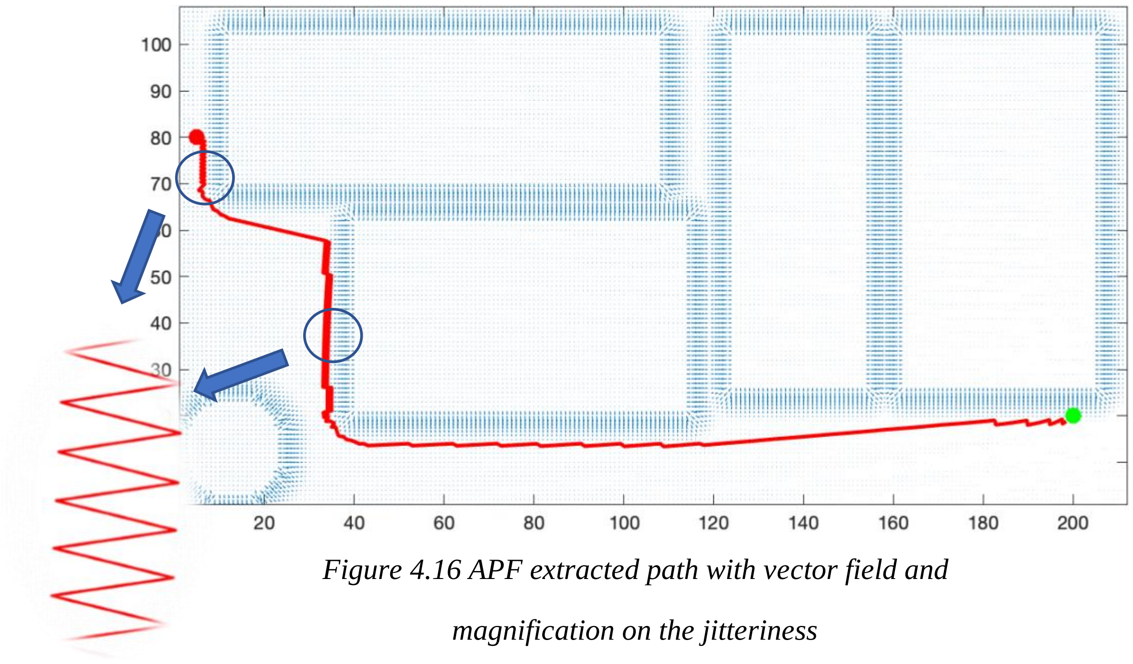


Figure 4.16 APF extracted path with vector field and magnification on the jitteriness

Table 4.5 APF results

Scenario	Path length [m]	Time to process [s]
1	-	-
2	-	-
3	235	0.0246
Special	300	0.0243216

Results discussion

The length of the path is quite long. This is a consequence of the jittering of the path near the obstacles. To avoid local minima this jittering was necessary. If a smoothing algorithm is being applied on the path, then it can be useful, but it will not be the most optimal.

Advantages and Disadvantages of the method

The computation time needed for this algorithm is neither fast nor slow, it is somewhere in the middle. While the computation time is simply good the replanning capabilities of this algorithm are its strong point. It can work well in dynamic environments but with a cost.
The algorithm do not provide the most optimal path. And often it can be trapped in what is called local minima. That leads to the algorithm not being used now for real-life applications

5 Conclusions

5.1 Discussion

Concluding with this thesis all the results from the case studies will be discussed and analyzed. All the algorithms will be compared A*, RRT, PRM, and APF and the best one for our needs will be proposed. Each algorithm will be commented on its path length, process time, smoothness, and replanning capabilities. Below are the results, for scenario 1. They will be presented in a stacked column graph:

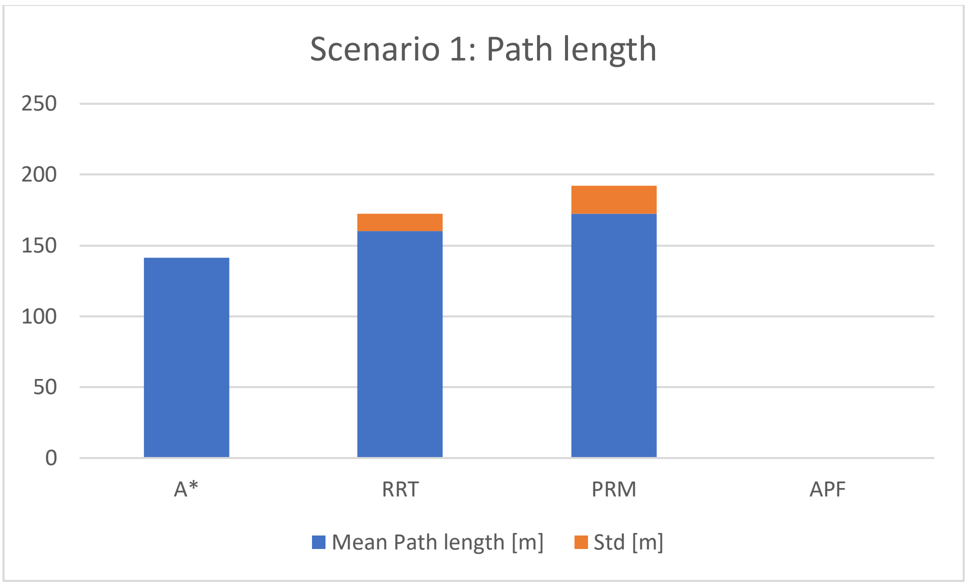
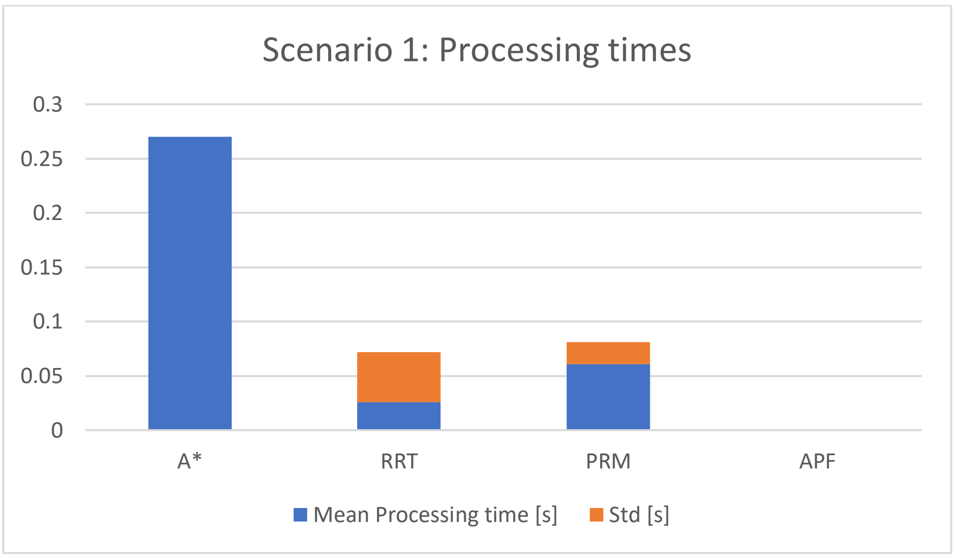


Figure 5.1 Scenario 1 Path length results



Starting of APF the results that were not promising. The algorithm for scenario 1 was getting stuck in a local minima and it could not provide a path. Then as expected A* gave the shortest path of all 3 but with the longest process time. PRM and RRT had similar results in path length and process time with a notable difference for the RRT processing time being almost half of the PRM but with a higher standard deviation. Below are the results for scenario 2. They will be presented in a stacked column graph:

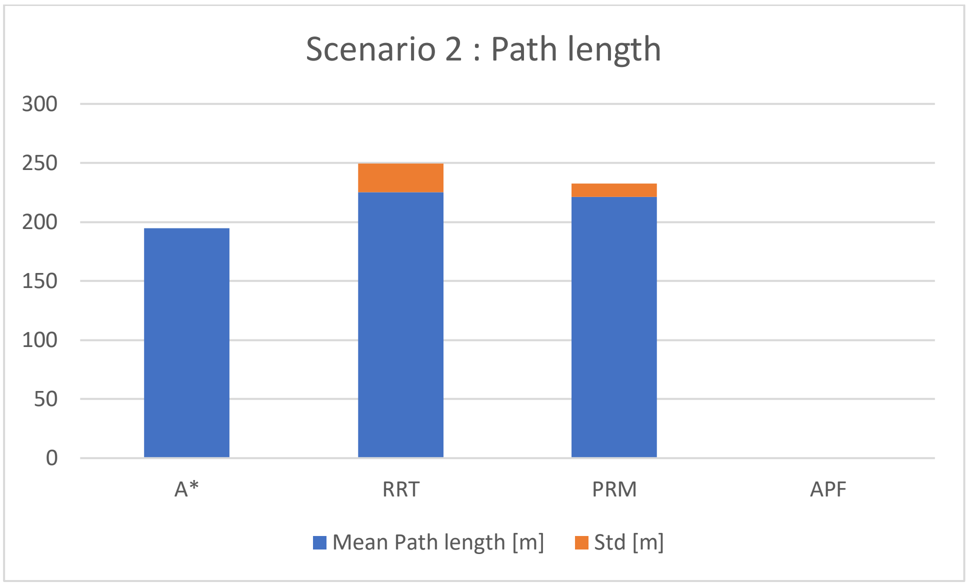
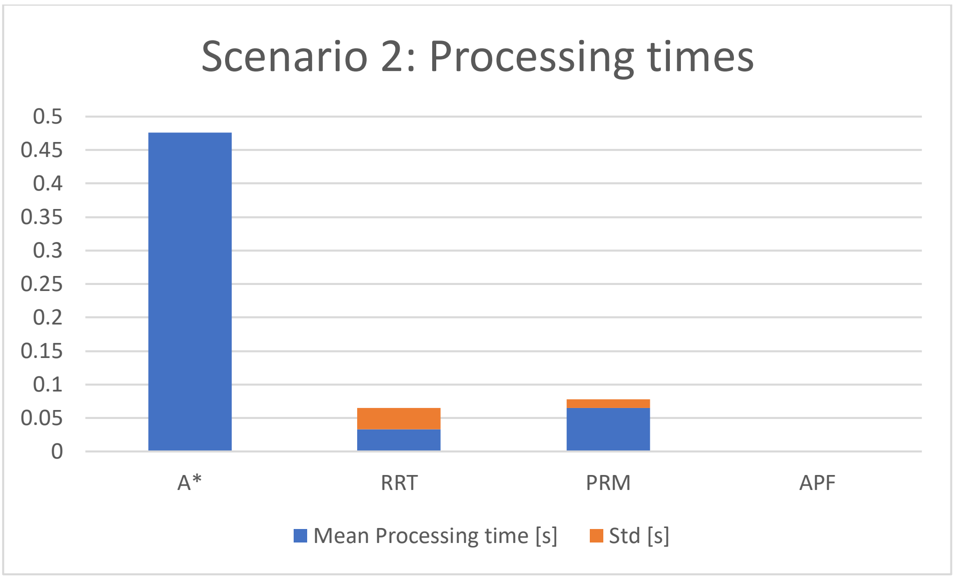


Figure 5.2 Scenario 2 Path length results



Here there is similar pattern to scenario 1 where A* gives the shortest path and longest process time. Also, APF does not conclude to a path. RRT and PRM are having again similar results in both metrics but RRT has less process time. Below are the results for scenario 3, will be presented in a stacked column graph:

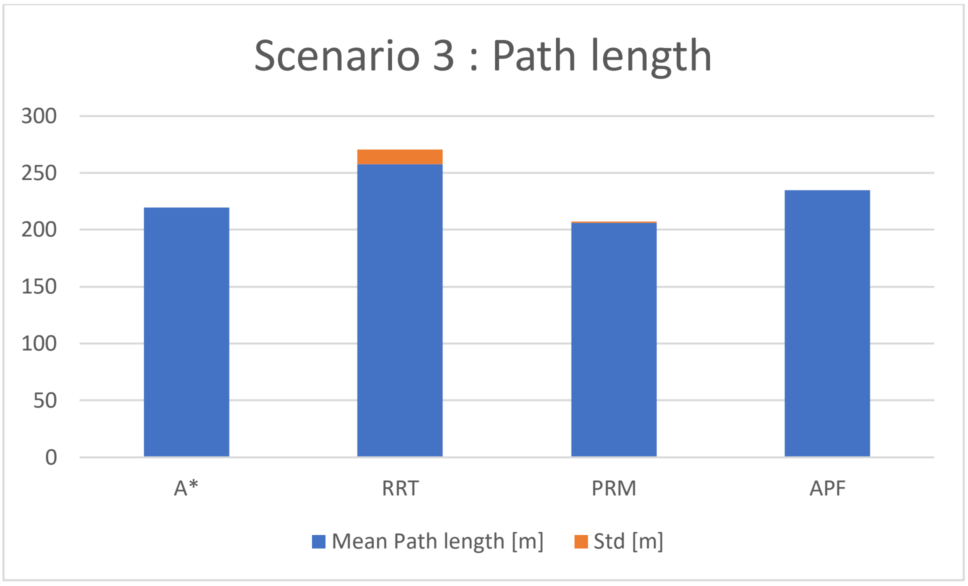
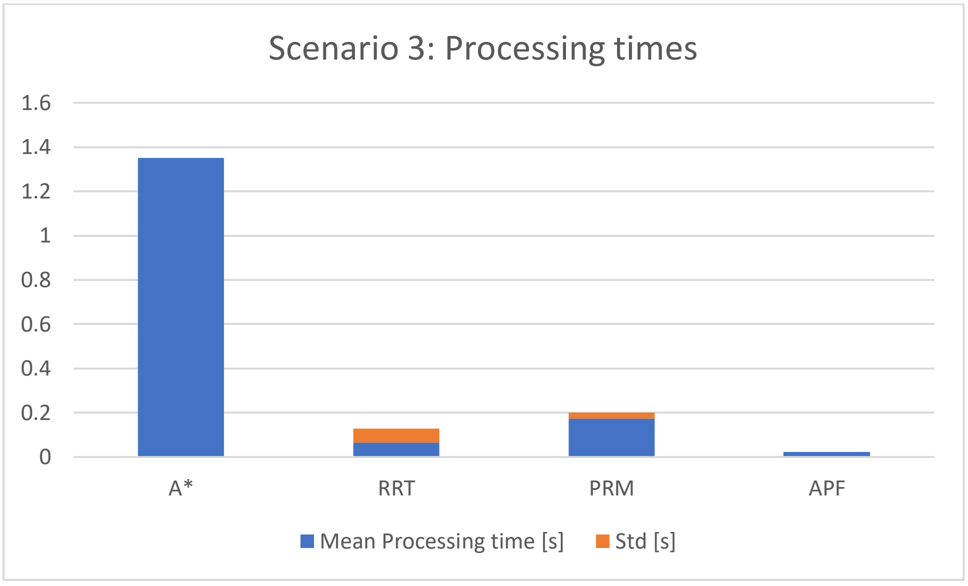


Figure 5.3 Scenario 3 Path length results



For scenario 3 things have changed. Now A* and PRM have similar path lengths with a bit shorter for PRM, and RRT has the longest path length with APF having a similar path length. For the processing times, things have stayed the same with A* having the bigger process time and RRT the shortest with PRM being a close second. Something that cannot be unnoticed is the extremely fast process time of APF which is faster than anything else. It shows that if APF finds a modification that provides a solution to the local minima problem then it will be one the best path planning algorithms.

Now to be able to conclude which algorithm is the best for our specific case study it is necessary to take a look at the other two criteria that define the quality of the algorithm which is smoothness and replanning capabilities. Below we can see a table that descrebie each algorithm criterias for this case study.

Table 5.1 Criteria ranking for algorithms

(Ranking follow this order: Bad, Good, Great, Excellent)

	A*	**RRT**	**PRM**	**APF**
Path Length	Excellent	Good	Great	Good
Process time	Bad	Great	Good to Great	Excellent
Smoothness	Excellent	Good	Great	Bad[2]
Replanning	Bad	Good	Great	Excellent

As we can see no algorithm is perfect and each one has its strong points. By taking into consideration all the criteria at the same time while being a good fit for our case, PRM is the best-suited algorithm to be used on a robot at the university campus. It excels at nothing but is great on almost all of them. Having an almost optimal path length, low process times, great smoothness, and replanning capabilities it is making it a great choice for our case.

[2] Generally, APF has one of the best smoothness on path planning algorithms when the parameters can take the optimal values. In this case because obstacles were overlapping with the optimal values it was necessary to choose non optimal parameters.

5.2 Suggestions for further study

In this thesis the application of different path planning algorithms has been studied on a static map with no moving obstacles. A possible future work is to test the replanning capabilities of the algorithms that were mentioned above and see how the planning algorithm reacts to unexpected changes.

All the planning here was analyzed on a global level which is good when you know the map but also doesn't take into consideration the small details that may appear along its path. That's where local planning (Trajectory optimization) is useful. That's why trajectory optimization could be studied next.

While planning is crucial to be able to plan it is needed for the robot to the surrounding environment and to be able to localize where it is. This can be done with the method of simultaneous localization and mapping (SLAM).

Finally, the robot has to determine with what speed and acceleration it has to move along its path. Here is where control theory is important and PID or Model Predictive control (MPC) needs to be studied further.

With everything above mentioned all the necessary part of the brain of the robot will be almost ready for a real-life application of an in-house produced robot at University of Thessaly.

References

5 Free Global DEM Data Sources – Digital Elevation Models. (2022, May 29). GIS Geography. Retrieved September 25, 2022, from https://gisgeography.com/free-global-dem-data-sources/

Caron. (2016, July). Robot procedures. Scaron. https://scaron.info/ntu-2016/

GeeksforGeeks. (2022, May 30). A* Search Algorithm. Retrieved September 25, 2022, from https://www.geeksforgeeks.org/a-search-algorithm/

Google. (2022). Pedion Areos Volos. Retrieved April 10, 2020, from https://www.google.com/maps/@39.3591985,22.9298102,17z

Hoy, M., Matveev, A. S., & Savkin, A. V. (2014, March 4). Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. Robotica, 33(3), 463–497. https://doi.org/10.1017/s0263574714000289

Insider. (2021, August). https://i.insider.com/60129f736dfbe10018e006a6?width=1136&format=jpeg

Khokhar, A. (2021, December 30). Probabilistic Roadmap (PRM) for Path Planning in Robotics. Medium. Retrieved September 25, 2022, from https://medium.com/acm-juit/probabilistic-roadmap-prm-for-path-planning-in-robotics-d4f4b69475ea

Klancar, G., Zdesar, A., Blazic, S., & Skrjanc, I. (2017, January 24). Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems (1st ed.). Butterworth-Heinemann.

Li, Q., Wang, L., Chen, B., & Zhou, Z. (2011, July). An improved artificial potential field method for solving local minimum problem. 2011 2nd International Conference on Intelligent Control and Information Processing. https://doi.org/10.1109/icicip.2011.6008278

Likhachev, M., & Ferguson, D. (2009, June 26). Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles. The International Journal of Robotics Research, 28(8), 933–945. https://doi.org/10.1177/0278364909340445

Lynch, K. M., & Park, F. C. (2017, July 7). Modern Robotics: Mechanics, Planning, and Control (1st ed.). Cambridge University Press.

Nash, A. (2013, September 18). Any-Angle Path Planning | AI Magazine. Retrieved September 25, 2022, from https://ojs.aaai.org/index.php/aimagazine/article/view/2512

Pandey, N. (2022, June 14). What is Gradient Descent? - FAUN Publication. Medium. Retrieved September 25, 2022, from https://faun.pub/what-the-hell-is-gradient-descent-97a75cc5cc4e

Robotis. (2015, January). TurtkeBot3. https://emanual.robotis.com/docs/en/platform/turtlebot3/features/

Roy, B. (2021, December 12). A-Star (A*) Search Algorithm - Towards Data Science. Medium. Retrieved September 25, 2022, from https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb

Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., & García-Cerezo, A. (2021a, November 26). Path Planning for Autonomous Mobile Robots: A Review. Sensors, 21(23), 7898. https://doi.org/10.3390/s21237898

Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., & García-Cerezo, A. (2021b, November 26). Path Planning for Autonomous Mobile Robots: A Review. Sensors, 21(23), 7898. https://doi.org/10.3390/s21237898

SOUISSI, DUVIVIER, & ARTIBA. (2013, October). Path planning: A 2013 survey. IESM CONFERENCE. https://www.researchgate.net/publication/282054384_Path_planning_A_2013_survey

Teleweck, P. E., & Chandrasekaran, B. (2019, April). Path Planning Algorithms and Their Use in Robotic Navigation Systems. Journal of Physics: Conference Series, 1207, 012018. https://doi.org/10.1088/1742-6596/1207/1/012018

Triharminto, H. H., Wahyunggoro, O., Adji, T. B., Cahyadi, A. I., & Ardiyanto, I. (2016, December 1). A Novel of Repulsive Function on Artificial Potential Field for Robot Path Planning. International Journal of Electrical and Computer Engineering (IJECE), 6(6), 3262. https://doi.org/10.11591/ijece.v6i6.11980

Vagale, A., Oucheikh, R., Bye, R. T., Osen, O. L., & Fossen, T. I. (2021a, January 29). Path planning and collision avoidance for autonomous surface vehicles I: a review. Journal of Marine Science and Technology, 26(4), 1292–1306. https://doi.org/10.1007/s00773-020-00787-6

Vagale, A., Oucheikh, R., Bye, R. T., Osen, O. L., & Fossen, T. I. (2021b, January 29). Path planning and collision avoidance for autonomous surface vehicles I: a review. Journal of Marine Science and Technology, 26(4), 1292–1306. https://doi.org/10.1007/s00773-020-00787-6

Velodyne Lidar. (2018, April). https://www.businesswire.com/ /Velodyne-LiDAR-Announces-Puck-Hi-ResTM-LiDAR-Sensor-Offering-Higher-Resolution-to-Identify-Objects-at-Greater-Distances and in the right the result taken from lidar https://store.clearpathrobotics.com/blogs/blog/how-to-choose-a-lidar

Wikipedia contributors. (2022, August 14). Starship Technologies. Wikipedia. Retrieved September 25, 2022, from https://en.wikipedia.org/wiki/Starship_Technologies

Yujiang, Z., & Huilin, L. (2017, December 31). Research on mobile robot path planning based on improved artificial potential field. Mathematical Models in Engineering, 3(2), 135–144. https://doi.org/10.21595/mme.2017.19520

Software:

A brief description of the used software. «The development and execution of the algorithms has taken place on my personal laptop MacBook Pro 2017. The specifications of the laptop are the following:

- Processor: Dual-Core Intel Core i5CPU 2.3GHz,
- Memory: 8,00 GB,
- OS: macOS Monterey v12.6,

The software used was MATLAB R2019b with a foundation code of R. Kala (2014) Code for Robot Path Planning and Robotics: Computational Motion Planning

Appendix

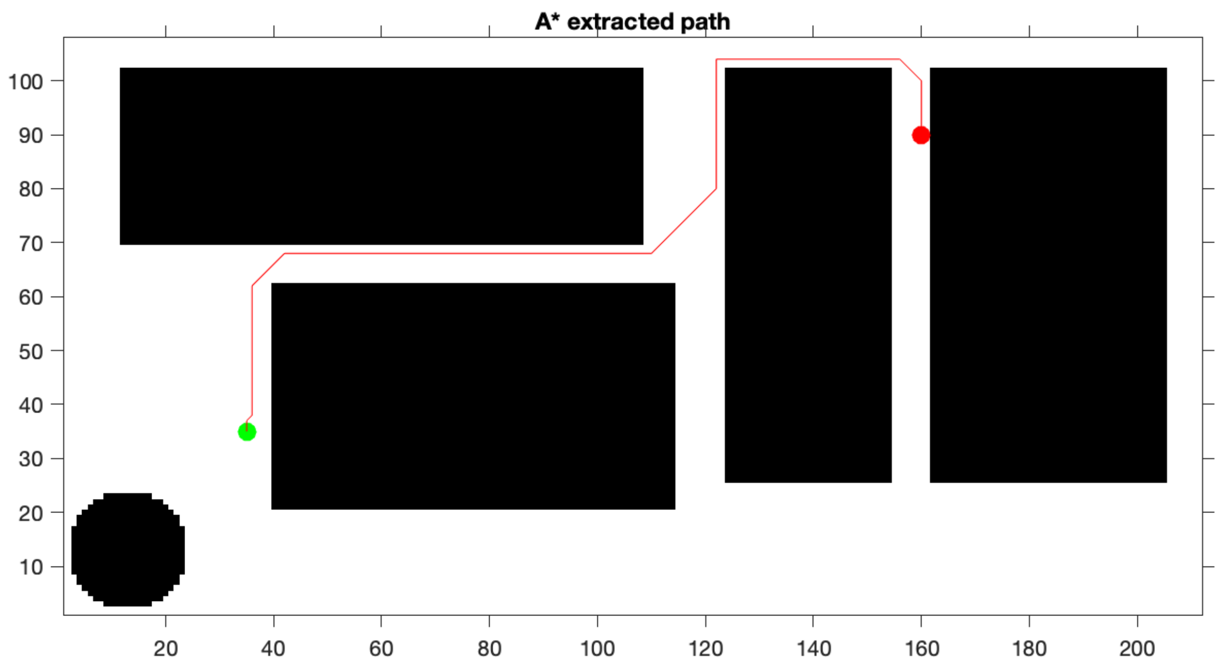


Figure 5.4 Scenario 2 A path*

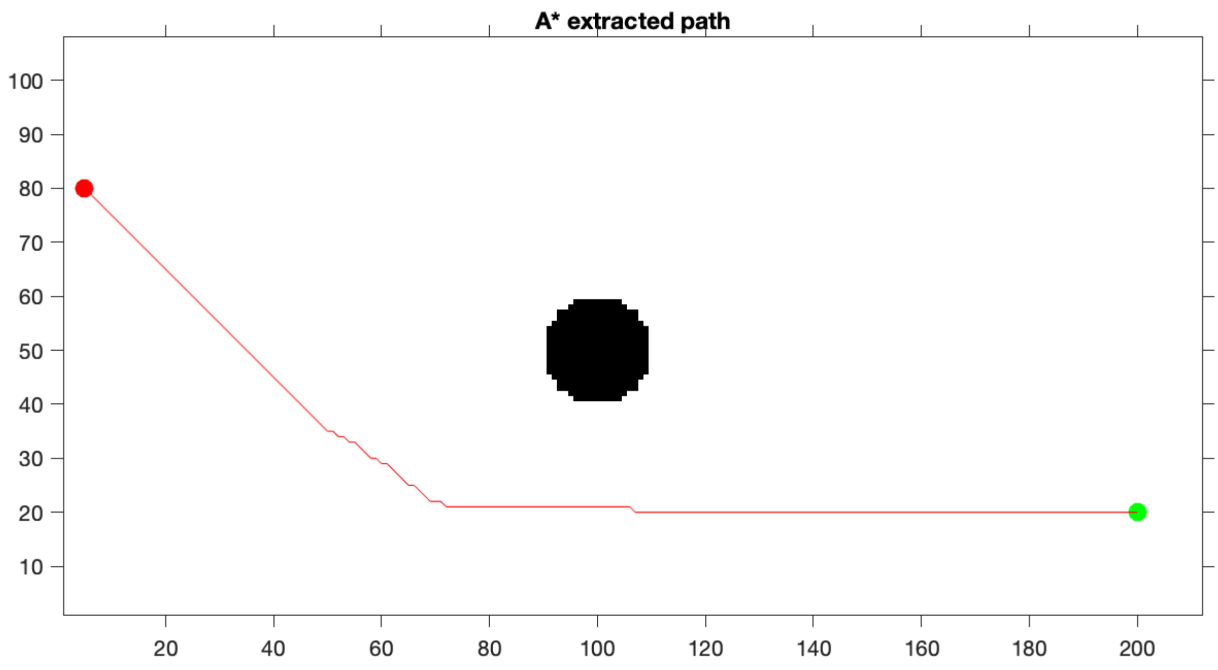


Figure 5.5 Scenario 3 A path*

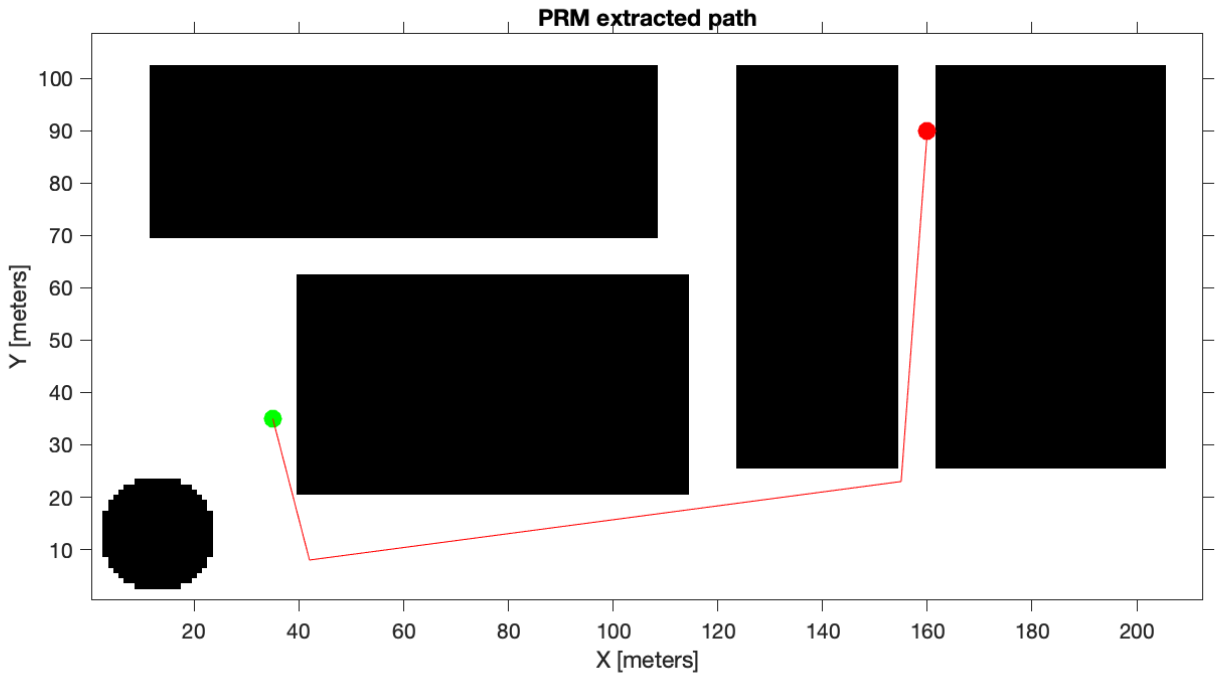


Figure 5.6 Scenario 2 PRM path (1 of the 100)

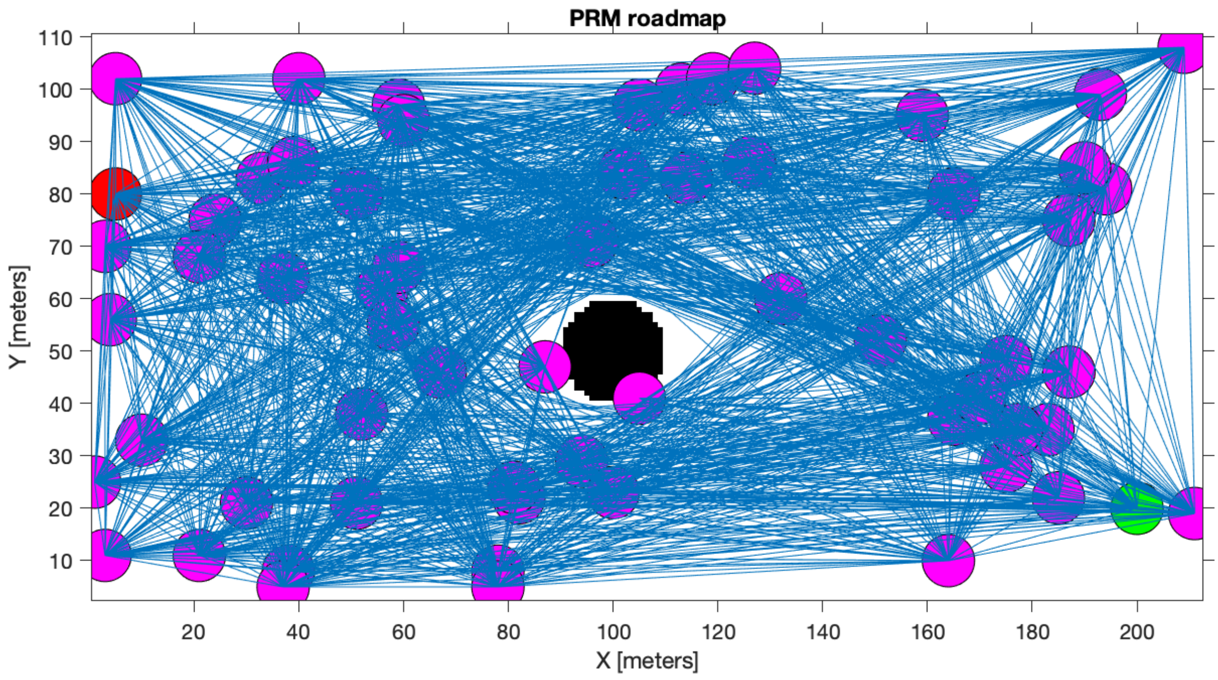


Figure 5.7 Scenario 3 PRM roadmap

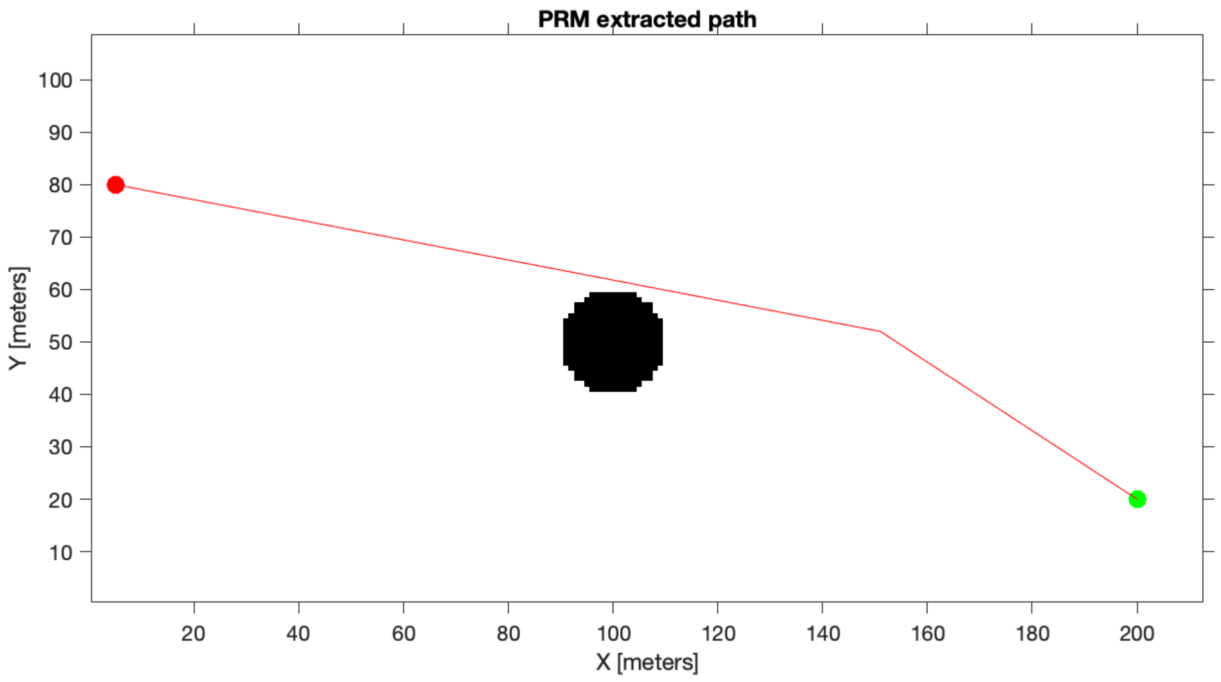


Figure 5.8 Scenario 3 PRM path (1 of the 100)

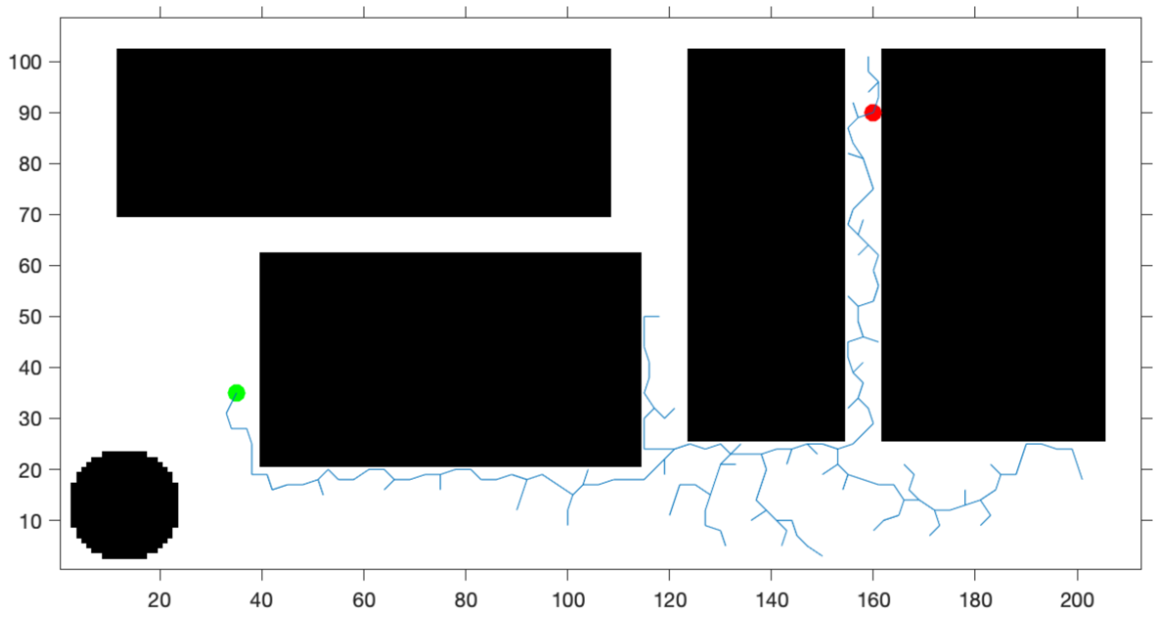


Figure 5.9 Scenario 2 RRT tree

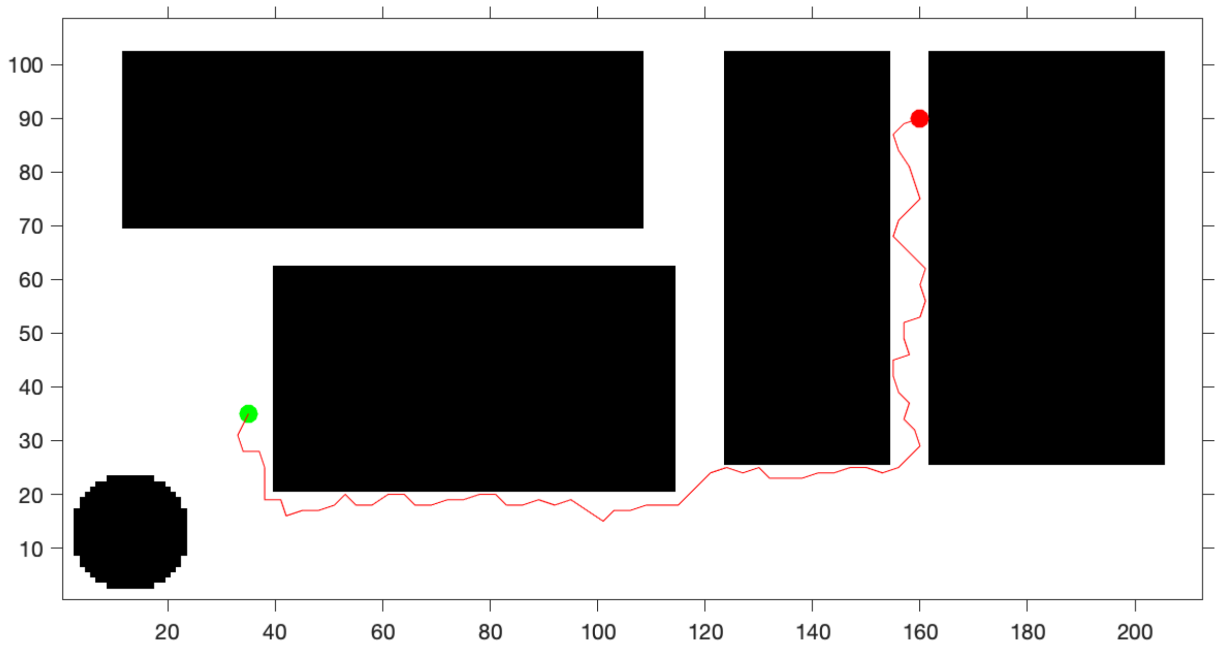


Figure 5.10 Scenario 2 RRT path (1 of the 100)

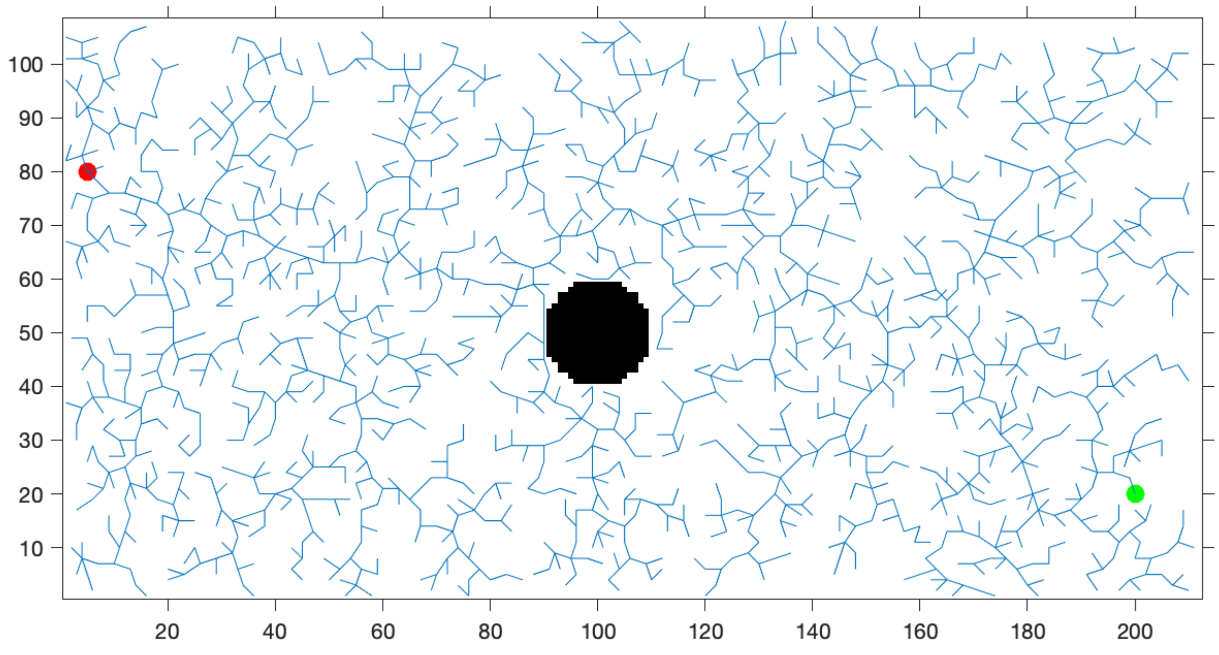


Figure 5.11 Scenario 3 RRT tree

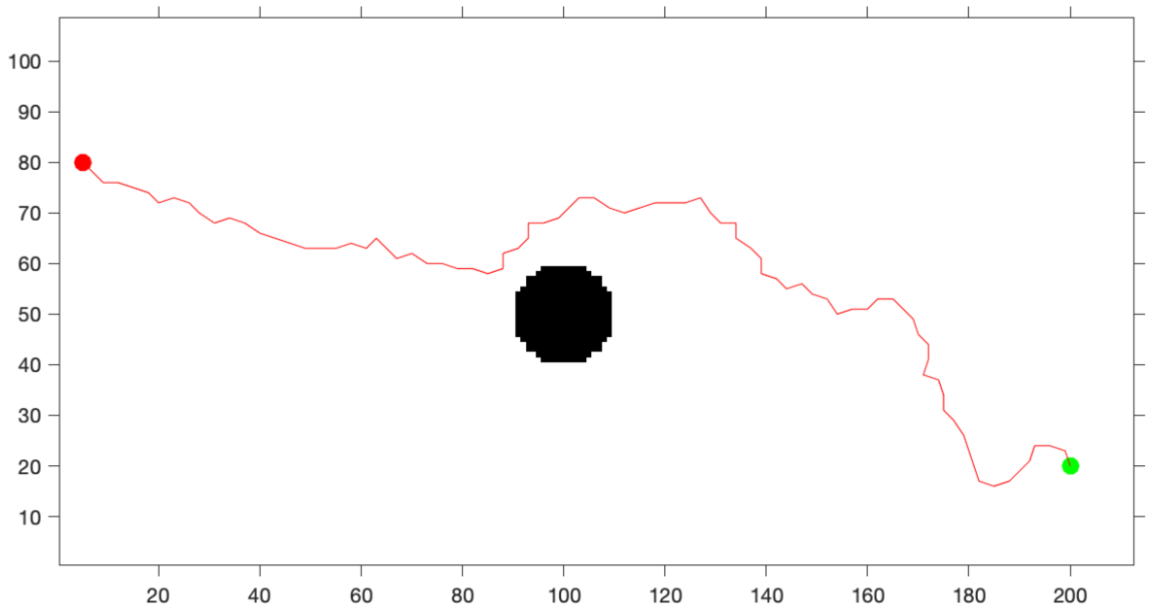


Figure 5.12 Scenario 3 RRT path (1 of the 100)

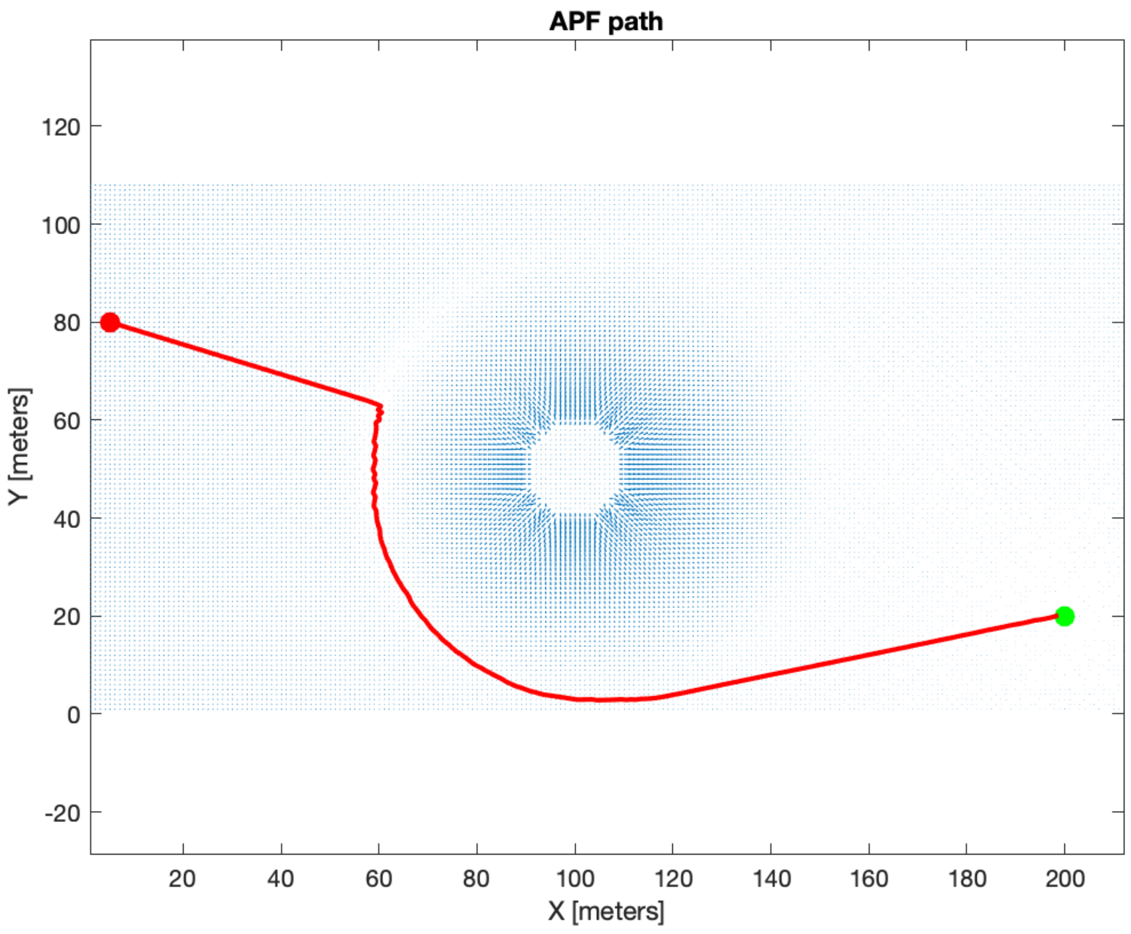


Figure 5.13 Scenario 3 APF path