



**NEAR EAST UNIVERSITY
INSTITUTE OF GRADUATE STUDIES
DEPARTMENT OF MECHATRONICS ENGINEERING**

**ROBOT PATH
PLANNING:EXPLORING D*(STAR) LITE**

M.Sc. THESIS

Jonathan Elochukwu UZOEGHELU

**Nicosia
January, 2021**

**JONATHAN ELOCHUKWU
UZOEGHELU**

**ROBOT PATH
PLANNING:EXPLORING D*(STAR)
LITE**

MASTER THESIS

2021

NEAR EAST UNIVERSITY
INSTITUTE OF GRADUATE STUDIES
DEPARTMENT OF MECHATRONICS ENGINEERING

ROBOT PATH PLANNING:EXPLORING D*(STAR) LITE

M.Sc. THESIS

Jonathan Elochukwu UZOEGHELU

Supervisor

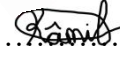
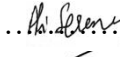

Assist. Prof. Dr. Parvaneh ESMAILI

Nicosia

January, 2021

Approval

We certify that we have read the thesis submitted by Jonathan Elochukwu Uzoeghelu titled **“Robot path planning:Exploring d*(star) lite”** and that in our combined opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Sciences.

Examining Committee	Name-Surname	Signature
Head of the Committee:	Assoc. Prof. Dr. Kamil DIMILILER	... 
Committee Member*:	Assist. Prof. Dr. Ali SERENER	... 
Supervisor:	Assist. Prof. Dr. Parvaneh ESMAİLİ	... 

Approved by the Head of the Department

01/09/2021

.....

Doç. Dr. Hüseyin Hacı

Head of Department

Approved by the Institute of Graduate Studies

20/04/2022

.....

Prof. Dr. Kemal Hüsnü Can Başer

Head of the Institute

Declaration

I hereby declare that all information, documents, analysis and results in this thesis have been collected and presented according to the academic rules and ethical guidelines of Institute of Graduate Studies, Near East University. I also declare that as required by these rules and conduct, I have fully cited and referenced information and data that are not original to this study.



Jonathan Elochukwu Uzoeghelu

08/01/2021

Acknowledgments

I would like to express gratitude for the coaching, guidance and input from several special individual:

Firstly, I would like to express my hearty appreciation to my supervisor Assist. Prof. Dr. Parvaneh ESMAILI for her invaluable suggestion, directions, guidance, advice, correction, constructive criticism and outstanding supervision throughout the course of this project. Next, Special thanks to all my lecturers, all members of staff of the department of Mechatronics Engineering, Near East University for their contribution during the course of my study in the department.

I owe a depth of gratitude to my late father Mr. Uzoeghelu Ochiabuto

Again, my sincere gratitude goes to my dearest and sweetest mother, Mrs. Uzoeghelu Pauline who denied herself of so many things and have toiled day and night to ensure I am who I am today, my prayer is that the Lord will bless you with long life, perfect health and you will reap the fruit of your labour in Jesus name.

I won't forget my lovely Sisters; Princess, Juliet, my nieces: Emmanuella, Jasmine, Ariana, Brenda, Nephews: Joshua, Jeremiah, Jeffrey and Jackson, my beloved Damilola, my ufan mmi Uduak, my wonderful friend Janet, and Daniel Malann and to all others who I can't start mentioning, for all the moral support given during the stressful periods, words cannot convey how much you guys have impacted my life and for your immense contributions to my life. Finally, I wish to thank the leadership and members of Watchman Catholic Charismatic Renewal Movement, North Cyprus. God bless you all. To each and every one of you, thank you very much.

Jonathan Elochukwu Uzoeghelu

To my Mother ...

Abstract

ROBOT PATH PLANNING:EXPLORING D*(STAR) LITE

Uzoeghelu, Jonathan Elochukwu

M.Sc. Department of Mechatronics Engineering

January 2021, 64 pages

Autonomous robot navigation is a key aspect of robotics with different applications such as material handling, surgeries, video games, networking rescue mission, navigation and survey of dangerous and human unfriendly environments, hence path planning needs to be Safe, fast, effective and efficient. Because path(s) quality that is generated greatly affects its application in robotics, it is therefore important the path generated by an algorithm is smooth to a great extent. Typically, finding the shortest path (minimizing the total distance travelled) is the main aim and purpose of the path planning process as it has a direct or indirect effect on the other yardsticks such as the time of computation and the level of consumption of energy.

In this study further improvements have been done on the basic D* lite so that the number of nodes visited(expanded) is greatly reduced and computational time is improved further. This is done by preventing initializing all node during the initialization also by not traversing the entire table when judging if a node is in the priority list or not.

The modified algorithm of D" lite shows improvement in computational time, reduced node expansion and can handle dynamic multiple goals effectively in both static and dynamic environment.

Keywords: path planning, robot, D* Lite (D star lite), dynamic/static environment

ÖZET

Otonom robot navigasyon malzeme işleme, ameliyatlara, video oyunları, ağ kurtarma misyonu, navigasyon ve tehlikeli ve insan düşmanca ortamlarda anket gibi farklı uygulamalar ile robotik önemli bir yönüdür, bu nedenle yol planlama güvenli olması gerekir, hızlı, etkili ve verimli. Oluşturulan yolun kalitesi robotik uygulamasını büyük ölçüde etkilediği için, bir algoritmanın ürettiği yolun büyük ölçüde pürüzsüz olması önemlidir. Tipik olarak, en kısa yolu bulmak (seyahat edilen mesafeyi en aza indirmek), hesaplama süresi ve enerji tüketimi gibi diğer ölçümler üzerinde doğrudan veya dolaylı bir etkiye sahip olduğundan, yol planlama sürecinin temel amacıdır.

Bu çalışmada genişletilmiş düğüm sayısını azaltmak ve hesaplama süresini iyileştirmek için temel D* lite üzerinde daha fazla iyileştirme yapılmıştır. Bu, bir düğümün öncelik listesinde olup olmadığını değerlendirirken tüm tabloyu geçmeyerek da başlatma sırasında tüm düğümün başlatılmasını engelleyerek yapılır.

D* lite'ın değiştirilmiş algoritması hesaplama süresinde iyileşme gösterir, düğüm genişlemesini azaltır ve hem statik hem de dinamik ortamda dinamik çoklu hedefleri etkin bir şekilde işleyebilir.

Anahtar Kelimeler: Yol planlaması; Robot; D* Lite (D yıldız lite); Dinamik/statik ortam

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	i
ABSTRACT.....	iii
ÖZET.....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES.....	vii
LIST OF TABLES.....	ix
CHAPTER 1: INTRODUCTION	
1.1 Background.....	1
1.2 Problem Statement.....	2
1.3 Aim of Study.....	2
1.4 Significance of Study.....	2
CHAPTER 2: LITERATURE REVIEW	
2.1 Environments.....	6
2.2 Environmental Representation/Modelling.....	7
2.3 Goals.....	8
2.4 Search.....	9
2.5 Obstacle Avoidance.....	10
2.6 Path Planning Algorithms.....	11
2.6.1 Types of Path Planning	12
2.6.1.1 Heuristic Approach	13
CHAPTER 3: THE MODIFIED PATH PLANNING ALGORITHM	
3.1 Nodes and Consistency.....	16
3.2 Initialization.....	17

3.3 Grids.....	17
3.4 Modified D* Lite.....	18
 CHAPTER 4: RESULT AND DISCUSSION	
4.1 Map Generation.....	28
4.2 Python Programming Language.....	28
4.3 System Specification Requirement.....	29
4.4 Measurements.....	29
4.5 Results.....	29
4.5.1 Visualisation	35
4.6 Discussions.....	41
 CHAPTER 5: CONCLUSION	
5.1 Conclusion.....	43
 REFERENCES.....	
	444
 APPENDICES	
Appendix 1: Psuedo code.....	49
Appendix 2: Turnitin	51

LIST OF FIGURES

Figure 2.1: Level of path planning (Souissi,et al 2013).....	4
Figure 2.2: Differences between local and global path planning (Raja, 2012)	6
Figure 2.3: path planning chart (Han-ye Zhang, 2018)	12
Figure 3.1: path costs.....	18
Figure 3.2: The goal rhs value is set to zero, and all other rhs and g values are set to infinity	19
Figure 3.2: (b) The goal is added to the open list.....	20
Figure 3.3: (a) The goal is popped of the open list.....	21
Figure 3.3: (b) The goal is expanded and the resulting inconsistent nodes are put the on open list.....	21
Figure 3.4: (a) Remove the minimal node from the list, but do not expand any of the open list's neighbours.....	22
Figure 3.4: (b) Remove the minimal node from the open list and replace it with inconsistent Neighbouring nodes.....	23
Figure 3.5: When an obstacle pops up on the path already planned.....	23
Figure 3.6: (a) Showing outgoing edges to grid label (2, 2)	24
Figure 3.6: (b) Showing Incoming edges to grid label (2, 2)	24
Figure 3.7: Updating nodes.....	25
Figure 3.8: (a) Update the predecessors of (3, 2)	26
Figure 3.8: (b) A new optimal path has been found (3,2)	26
Figure 4.1: Stages of a Static goal in an environment without obstacles.....	30
Figure 4.1b: final results in an unknown(right), known(left) environment.....	30
Figure 4.2: Static goal with obstacle in an unknown environment.....	32
Figure 4.3: Static goal with obstacle in a known environment.....	32
Figure 4.4: Static goal with obstacle in an unknown environment.....	33

Figure 4.5: Static goal with more obstacles.....	33
Figure 4.6: Static and dynamic obstacle in a known environment.....	34
Figure 4.7: multiple obstacles in an unknown environment.....	35
Figure 4.8: multiple obstacles in a known environment.....	35
Figure 4.9: no obstacles in a dynamic environment.....	35
Figure 4.10: multiple obstacles in a known environment.....	37
Figure 4.11: Graph showing number of nodes expanded.....	38
Figure 4.12: Graph showing mean path finding time.....	39

LIST OF TABLES

Table 4.1: Table comparing results of Heuristic planners.....	38
Table 4.2: Table showing results of the modified D* Lite.....	41

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Autonomous robot navigation is a key aspect of robotics with different applications such as material handling, surgeries, video games, networking rescue mission, navigation and survey of dangerous and human unfriendly environments, hence path planning needs to be Safe, fast, effective and efficient. Because path(s) quality that is generated greatly affects its application in robotics, it is therefore important the path generated by an algorithm is smooth to a great extent. Typically, finding the shortest path (minimizing the total distance travelled) is the main aim and purpose of the path planning process as it has a direct or indirect effect on the other yardsticks such as the time of computation and the level of consumption of energy.

For over twenty years scholars have tirelessly worked, studied path planning and several methods and techniques have been developed and improved on. But attention have been focused more on static environment with little been done on dynamic environment. Hence my interest in the subject matter. A robot might be required to work in either a static or changing environment. If the obstacles alter their orientation and/or position with respect to time then the environment is said to be dynamic. It is static if obstacles do not alter their orientation and/or position per time.

D* lite (pronounced D star lite) method of path planning has shown great potential in both dynamic and static environment. Further improvements have been done on the basic D* lite to: reduce memory usage (Kobti, 2008), optimizing the priority in which the nodes are searched, resulting in the avoidance of some unnecessary expansions (Dave Ferguson, 2005) and (Adi Botea, 2004) the complexity of re-planning the initial path is reduced by refining the environment while sacrificing optimality.

1.2 Problem Statement

Research on robot path planning has been extensively carried out because it is a core part of robotics. But from various researches path planning has mostly been on static target/goal and known maps. Few literatures cover dynamic goal in known and unknown environments. So little has been done on expounding methods used in cases of multiple goals, dynamics goal and unknown environment or a combination of any of the aforementioned.

1.3 Aim of Study

- Expound knowledge on D*(star) lite method of path planning
- Produce, Measure, compare and revise D*(star) lite performance in static and dynamic environment.
- Developed D*(star) lite to multiple, static and dynamics goals.

1.4 Significance of Study

This study will help other researchers perform a physical/real life simulation on the proposed method of path planning and if efficient and effective will open ground for increased knowledge acquisition as well as improved application in various robotics sphere for improved human interaction and existence. It will also give more explicit guide in understanding the dynamics and operation of D*(star) lite.

In subsequent chapters path planning is discussed in details considering different methods used in previous studies and proposing a modified version of the chosen method to be understudied.

Chapter 2

LITERATURE REVIEW

Path planning is finding a viable “shortest” (shortest here refers to minimum cumulative edge cost between two (2) vertices in a graph. (Correll, 2020), Transportation (road networks, air networks), delay (in a networking application), Telecommunications (computer networks) or any other yardstick that is important for any other application) path from start to goal configuration. (Koubaa, et al., 2018)

Path planning algorithms are classified based on (Souissi,et al., 2013)

- Constraints (holonomic, non-holonomic, and kinodynamic)
- Nature of the environment (static, dynamic)
- Knowledge of the map (Local/online, Global/offline)
- Completeness (exact, heuristic)

The environment nature in path planning can be broadly categorized into online(local) and offline(global) algorithms (Maram Alajlan, 2003). In online path planning, information about the environment is obtained from the robot's attached local sensor (e.g., vision sensor (camera, GPS), and the robot then constructs a map of the environment using the information fed from the locally attached sensors, which can then be used to autotune the robot's orientation via software. In offline path planning requires agents to have a complete plan before they start executing. Agents must have a full understanding of the map in offline path planning process without the aid of sensors and prior to moving the agent, examine all possible robot position of and the goal in the environment to establish the optimal plan.(Maram Alajlan, 2003)

In a dynamic setting, agents in off-line planning need to know environmental changes in advanced to be able to create a collision-free trajectory. Therefore, for a robot trying to navigate an obstacle filled environment, off-line planning is often insufficient. Planning in a rapidly changing environment such as a metropolis requires an agent to update its plan frequently to respond to the changes around it.

When a robot needs to plan a collision-free path from certain start position to the goal position, this seemingly simple path planning problem is actually computational hard (Reif, 1979).

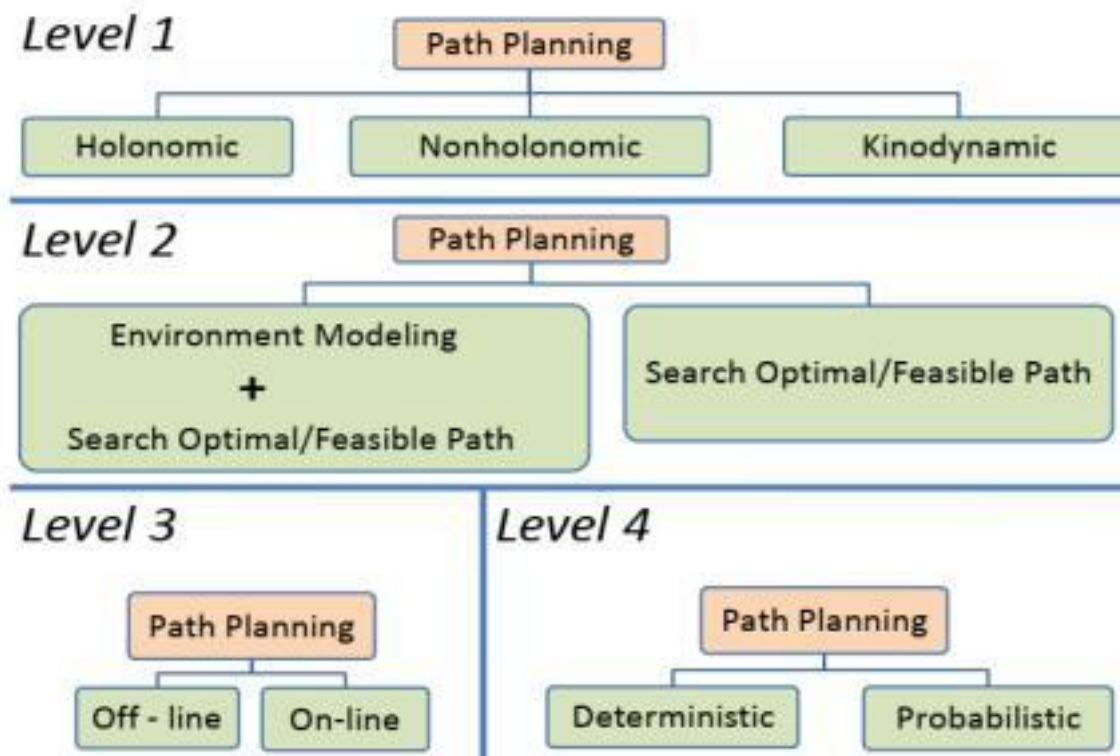


Figure 2.1: Level of path planning (Souissi, et al 2013)

Three key factors should be considered when planning a path: efficiency, precision, and safety. Any robot should be able to find its way in a minimum time possible at the same time using the least quantity of energy possible. An agent should also avoid any barriers in the environment in a safe manner. The agent should accurately follow its ideal and blockage-free course. (Imen Chaari, 2012)

The ability to generate an efficient pathway from a start point to a target in real-time conditions remains a problem challenging robot path planning. It is either optimality is given up for shorter computation times or complex computation has to be done.

To solve the problem of robot navigation, answers must be provided to the following questions: What is the robot vision? Where is the robot? Where is the robot going? How does the robot get there? Perception, localization, mapping, and motion planning are the four core navigation functions, which are determined by the answers to the questions above.

Koubaa, et al.(2008) in robot navigation there are four main pathways. They are namely Perception, localization, cognition and path planning and motion control.

1. Perception: Useful Information required about the environment in question is gotten by the robot by signal analysis and computer vision technologies from sensors.
2. Localization: The robot continuously reminds itself of its orientation and position in the environment by each iteration.
3. Cognition and Path planning: The robot analyses sensor data and takes the appropriate steps in order to determine a path to the goal (s).
4. Motion control: tracking of the optimal path by controlling its motions.

The above-mentioned systems are fused by a controlling unit, so that the robot goal(s) is carried out in a systematic way. The path planning problem can be handled in one or a combination of the following three ways: search-based, sampling-based, or combinatorial.

Once a map is given, the next step is to search it for a path in a known, partially known, or unknown environment, while bearing in mind operation cost, completeness, space complexity, and time complexity.

Norvig (2003) in evaluating path planners some well-known metrics used: completeness, optimality, and complexity.

1. **Completeness** this indicates that from the start to the target/goal if a path exists, the algorithm in use will find it, and if no path exist the planner will pop up a warning or any method indicated. This is important because if a path planner is not complete it is unable to be used in certain circumstances.
2. **Optimality** refers to the ability of the planner to find the shortest path or any other metric specified. This metric can be expanded for non-optimal algorithms to see how sub-

optimal they are. To do this the length of the path generated can be compared to a path generated by an algorithm known to be optimal.

3. **Complexity** This measurement can vary based upon the algorithm and environment. It can be divided into space complexity and time complexity. Space Complexity is the amount of memory requirements for the algorithm to plan a path while the amount of time it takes the algorithm to find a solution if any exist is called time complexity.

2.1 Environments

Maram Alajlan, (2003) path planning can be categorised dependent on the characteristic of the environment namely:

1. Dynamic environment.
2. Static

It is static if obstructions do not alter their orientation and/or position per time. If the obstacles alter their orientation and/or position per time then the environment is said to be dynamic. In a rapidly changing environment, agents frequently do not have time to design a comprehensive strategy for achieving a desired state, and instead must move fast in the face of changing conditions.

DIFFERENCE IN ROBOT NAVIGATION APPROACH

LOCAL PATH PLANNING	GLOBAL PATH PLANNING
<ul style="list-style-type: none">• Sensor-based system• Reactive system• Fast response• Assume incomplete knowledge of the workspace area• Follow path to while avoiding obstacles or objects while moving toward target	<ul style="list-style-type: none">▪ Map based▪ Deliberative system▪ Relatively slower response▪ Assume complete knowledge of the workspace area▪ Obtain a feasible path leading to goal

Figure 2.2: Differences between local and global path planning (Raja, 2012)

2.2 Environmental Representation/Modelling

A map is defined as a world representation such as arrangement and features capable of distinguishing between distinct places in an environment and it is a popular sort of high-level representation. This means, the extent the environment's arrangements and features may be reconstructed from representation. (Lee, 1996) As follows, he explains four representations at various locations along this continuum.

1. Recognizable Places: This is a list of map locations that the robot can reliably recognize. There is no geometric link that can be found.
2. Topological Maps: it takes notes of sites that are linked by pathways that can be transversed in addition to the recognized locations. It is possible to re-establish connectivity between visited locations.
3. Metric Topological Maps: this term refers to maps that include information about distance and angle in addition to the path description. It is possible to recover metric information about previously travelled paths.
4. Full Metric Maps: Object areas are established using a set of rules. Any object in the map can have its metric information retrieved

A detailed and accurate representation of the environment further aids deeper comprehension the factors in the environment, unneeded planning is cut down and computations count is acceptably brought don prior to the robot's global path planning. Free space, cell decomposition, framework space, probabilistic roadmap, and cell decomposition, techniques are all common approaches of environmental representation/modelling. (Otte, 2017).

Nourbakhsh (1997) In path planning, numerous techniques for representing maps (localization) are employed over the years. The following is a list of techniques.

1. Autonomous map building
2. Localization based on a route

3. Cyclic environment
4. Stochastic map technique
5. Environments that are dynamic
6. Beacon systems for positioning
7. Globally unique localization
8. Localization via Markov chains
9. Localization based on a probabilistic map
10. Localization with the Kalman filter.
11. Monte Carlo localization
12. Navigation based on landmarks

2.3 Goals

The path-planner is supposed to fulfil an objective called goal(s)/ target. A single or more goals may be set. To find an available path across C-space starting from the current position of the robot to a goal position given in a map is the most typical target in a path-planning algorithm. Other kind of goals include mapping (gathering as much information as possible about an environment), clandestine movement (moving toward a goal while effectively moving away from other robots), Observation /surveillance trailing an agent at a specified distance while avoiding collision, and visiting a set of locations while at the same time limit the length of movement (traveling salesman problem). It will be of importance to state that the target(goal) can either be single or multi (this is a case where more than one goal exists).

Agent navigation can be divided into two types: stationary target(goal) search (Stentz, 1995), (Sven Koenig M. L., 2002), (Likhachev et al 2005), (Ferguson, 2007), and moving goal search where the goal is moveable on the long run (Ishida & Korf, 1992). (Moldenhauer, 2009a). (Sturtevant, 2009b; Sturtevant, 2009a) Path planning methods for static goal search have been the subject of a lot of research. Algorithms of incremental search utilize data from past searches

to accelerate the present search, allowing them to identify minimal cost solutions for a succession of related search issues much quicker, compared to if each search problem were solved from beginning. (Sven Koenig M. L., 2004).

2.4 Search

While searching from *vstart* to *vgoal*, it is termed forward search, and when searching from *vgoal* to *vstart*, it is called backward search. Bidirectional search begins at both *vgoal* and *vstart* and tries to bring two of the searches together at a mid-way point. When it begins at both the *vgoal* and *vstart* it is multi-directional, also other random points, and tries to connect the various searches so that between *vstart* and *vgoal* a path can be obtained.

If the search algorithm has not yet reached a node, it is considered to be unexplored. If a node has been gotten to, and one of its neighbours in least is not reached yet (forward search), or if it's the neighbour of one node not reached yet, it is considered open (backward search). When a node is gotten to while searching, as well as entirely of the neighbours (search forward), or entirely the nodes it constitutes a neighbour of (search backward), it is referred to have expanded or closed.(Otte, 2017)

2.5 Obstacle Avoidance

During robot mobility, collisions with obstacles must be avoided quickly and safely. Robot navigation refers to the ability of an agent to manoeuvre about its vicinity (unknown/ known) in order to meet a target avoiding collision with various impediments. The agent must figure out a direction to get from where it is now to where it wants to go. A map of the environment is required, as well as a destination location (objective) and the robot's present position (as determined by proper sensory devices or a system in a different location). Detecting collisions between the agent and objects is a basic requirement for smart motion planner in an environment and it must be done quickly enough for the agent to safely adjust its direction or/and halt prior to collision. They involve blockages detection and avoiding obstacle itself. As indicated by (Kunchev, 2006) the algorithms to evade impact may be:

1. Map-based algorithms: These algorithms rely on environmental geometric models or topological maps. The robot has created a model of its surroundings.
2. Mapless-based: These make no attempt to depict the environment explicitly. Sensor systems are used to monitor the environment.
3. Algorithm bug. The fundamental idea behind the bug algorithm is to follow the outline of disturbance in path of the agent and circumvent them.
4. The histogram of a vector field. This method models the environment using a two-dimensional histogram grid that is reduced to 1-D, the polar histogram created with respect to the agent's orientation per given time.
5. Approaches to dynamic windows. The fundamental concept is in the velocity space of the agent selects a control. The agent's path comprises a series of arcs that are circular.
6. A diagram of proximity. A divide-and-conquer strategy is used in the nearness diagram method. To illustrate the position of impediments, the approach divides the work environment into separate sections.

7. Curvature velocity techniques: The curvature velocity approach put into consideration the vehicle's dynamic restrictions, allowing it to travel quickly in a crowded area.

8. Potential field: Agent is regarded as a submerged particle in a potential field formed by the objective and the supposed impediments in the duty environs. Every obstacle provides a potential that is repulsive, while the target creates an attracting potential.

Other algorithms that have been used to avoid barriers/obstacles over time include:

- Elastic Band Concept
- Nearness Diagram
- Dynamic Windows Approach
- Virtual Force Field
- Vector Field Histogram
- VFH+
- Curvature Velocity Method

2.6 Path Planning Algorithms

The majority of current robotic mapping methods are probabilistic in nature. Some algorithms are incremental, allowing them to execute in real time, whilst others take many passes through the data. To construct a map, some methods require precise pose information, while others may do so making use of odometry estimation. A few calculations are prepared to deal with correspondence issues between information recorded at various focuses as expected, though others expect highlights to convey marks that makes them uniquely recognizable. (Kaufmann, 2003)

The path planning problem can be solved by one of the following three categories: search-based, sampling-based, or combinatorial.

Once a graph is given, the next step is to search it for a path in a known, partially known, or unknown environment, while bearing in mind operation cost, completeness, space complexity, and time complexity.

2.6.1 Types of Path Planning

Various approaches have emerged over time in an attempt to solve the problem of robot motion planning. as shown in figure 2.3. Existing algorithms and approaches are divided into four categories, with a fifth added as technology advanced (Francisco Rubio, 2019)

- I. sensor-based planners
- AI. probabilistic methods
- BI. heuristic planners
- IV. classical methods
- V. evolutionary algorithms

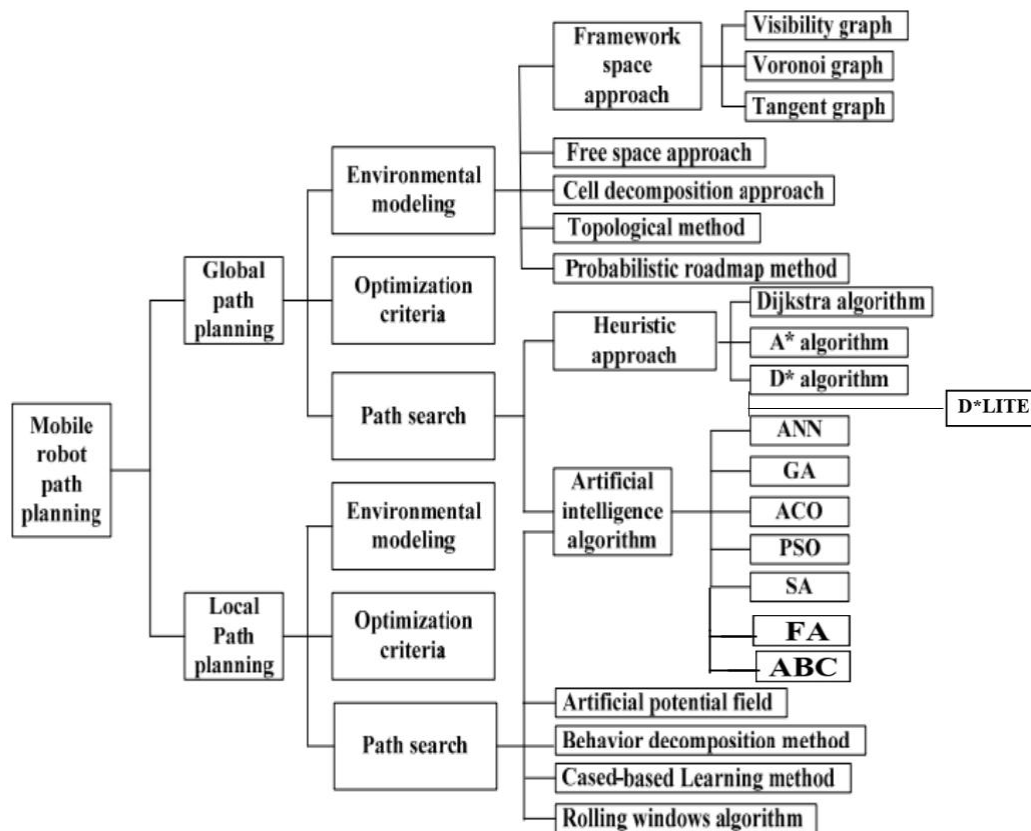


Figure 2.3: path planning chart (Han-ye Zhang, 2018)

2.6.1.1 Heuristic Approach

Heuristic algorithms do not assuredly find a solution, however in the event that they do, are probably going to accomplish such task at a great deal quicker than deterministic strategies. Heuristic planners are also known as search algorithms that are well-informed and they consist of graph search-based algorithms like:

a. Dijkstra Algorithm

It is a developed graph-based algorithm in determining the briefest pathways between nodes, it might be used to describe roadmaps or a discretized workspace, for example. Edsger W. Dijkstra, scientist of computers, proposed the method in 1956.

For solving the shortest path issue in a directed network, it is a standard shortest path algorithm. Its basic premise is that the beginning stage is in the middle and will be stretched out to the conclusion. The two vertices frame each bound of the graph to an ordered element pair. The weight function depicts the estimation of the boundaries. X and Y are the two vertex sets kept by the method. Set X is at first empty. Every moment a vertex in Y is relocated to X, the chosen vertex ensures that the total amount of edge weight from the start to the end is minimized. One major drawback of this method is that it has to transverse more nodes, thereby making the efficiency reduced greatly.

b. A* (star) Algorithm

In 1968, Hart et al put forward the A* algorithm. The Dijkstra type of algorithm was used to construct the A* algorithm. In an environment whose features does not change with time, the A* algorithm is mostly used for global search. A* employs a grid-based search area divided into squares to represent its map. Each square can either be a barrier or a blank space. To find the shortest path, a collision-free path made up of free space squares is calculated (also known as nodes). This algorithm examines all alternative routes to the goal and selects the one with the lowest cost (e.g., the shortest time, shortest distance, etc.). The algorithm selects the pathways that appear to lead to the goal the quickest out of all those discovered. A* works with weighted graphs: it constructs a connect-tree of pathways from an initial point in a graph, in a step wise format, until one of its pathways finishes at the goal position. The current child position

weighted value is updated starting at a specific node, the current position is updated using the child node with the minimum weighted value until all nodes have been traversed. A key to the A* algorithm is determining the function called evaluation $f(n)$, defined as $f(n) = g(n) + h(n)$, $g(n)$ being the actual path cost beginning at the start node to node n , $h(n)$ is defined for the optimal pathway from node n to the goal node in a state space as estimated cost. h value most commonly considered between the two nodes (n) as Euclidean distance. When $g(n)$ does not change, the value of $f(n)$ is mostly influenced by the $h(n)$ value. Close to target node the node $h(n)$ value is small, also, value of $f(n)$ is also little. Because of this, it ensures shortest/minimal path is always found in facing the target location direction. The A* algorithm searches along the goal point of the mobile robot, taking into account the position information. The A* method has a substantially higher trajectory search efficiency than the Dijkstra algorithm.

c. D Algorithm*

In 1994, Stentz proposed the D* algorithm (Stentz, 1995). Using a Dijkstra's search that is changed slightly, this method calculates a trajectory beginning at the target and working backward to the start-up position.

The A* algorithm is mostly used for global search in environments where the characteristics do not vary over time. Nonetheless, in real-world applications, mobile robot trajectory planning is progressively aware of environmental characteristics and changes over time (dynamic). It is mostly used by robots to navigate a path. The problem space of D* algorithm's is represented as a series of positions, where the position reflects the direction of the robot's position. The expense of an arc used to assure the search's direction. Other academics have also looked into the D* method, such as the Theta* algorithm and field D* algorithm

d. Greedy search

The method tackles problems through seeking for the best option available locally for every step while striving to acquire a global best solution. In general, the method of greedy search doesn't provide the exact optimum answer; nevertheless, a greedy process of trial and error may give rise to locally ideal recommendations that are inexactly the same as a global optimum in the shortest time achievable.

In this chapter planning algorithms classification based on different parameters are highlighted also steps involved in robot path planning are discussed, metrics for measuring optimality of a planner are introduced. Different types of environment representation are shown, I also highlighted and explained in details goals and its types, search types. Highlighted in this chapter are methods of obstacles avoidance. Going further in the chapter types of path planning algorithm employed in different researches were also expounded on.

To fix the local minima issue, numerous Heuristic and Meta-heuristic algorithms are utilized in robot trajectory planning. Of all the above-mentioned method used in robot path planning D* lite is chosen for this study because of it is dynamic, flexible and performance because D* Lite algorithm utilizes heuristic to limit the replanning procedure to just states useful for re-evaluating the pathway. This makes it appropriate for both local and global path planning

THE MODIFIED PATH PLANNING ALGORITHM

D* Lite is an extension of A* specifically Lifelong Planning A*(LPA*) that is fundamentally the same with the benefit of faster replanning. Where A* searches from start to goal, D* Lite (Koenig & Likhachev, 2002) searches from goal to start. The root node in D* Lite is required to remain the same for each subsequent search in order to speed up replanning (Koenig,2004) The start node continually changes as the agent moves towards the goal. On the other hand, the goal node is often a fixed location, making it an ideal candidate for the root node. Planning from start to goal would require that the robot detect the changes made throughout the entire path, but planning from goal to start reduces the effects of these changes.

3.1 Nodes and Consistency

When the g and rhs values are equal it is called a consistent node that is

$$\text{Consistency} = (g(x) == \text{rhs}(x)) \quad (3.1)$$

and also, when the g and rhs values differ it is said to be an inconsistent node.

If the value of g is greater than the value of rhs then it is an over-consistent node; that is

$$\text{Over-consistency} = (g(x) > \text{rhs}(x)) \quad (3.2)$$

and if g value is less that of rhs value, it is termed an under-consistent node; that is

$$\text{Under-consistency} = (g(x) < \text{rhs}(x)) \quad (3.3)$$

If a node is inconsistent update all of its neighbours and itself again. Updating nodes will try to make them consistent.

The raise and lower states of D* are practically similar to this concept of consistency. Function h is a heuristic that has a similar relevance to h from A* but differs from h from D*. A directed graph is assumed to be the graph searched. With $c(u, v)$ representing the cost of traversing the

directed edge from origin A to target B . (D^* defined $c(u, v)$ as the path cost from B to A). As a result, the node u 's successors and predecessors are $Succ(u)$ and $Pred(u)$, respectively. The function rhs is given below:

$$rhs(u) = \min_{s' \in Succ(u)} (c(u, s') + g(s')) \quad (3.4)$$

(also known as a priority queue) is an open list U . The key k of the open list is currently listed as a two vector, rather than the previous actual k values. The key is represented by equation 3.5

$$k(u) = [\begin{matrix} \min(c(u, s') + g(s')) \\ \min(c(s', u) + g(u)) \end{matrix}] \quad (3.5)$$

The primary and secondary keys, respectively are the first and second components of the key. If u 's first key is smaller than v 's first key, then mathematically one can write the key as $k(u) < k(v)$. When the first keys become tied, the second keys as a factor are used to break the equality.

3.2 Initialization

Set the goal's rhs value to zero (0), as well as the rhs and g values of all other associated/connected nodes. Because it is inconsistent, in the open list put the goal. The RHS score and G score act as the foundation for D^* Lite path planning and replanning purposes. In the event the D^* Lite algorithm encounters unforeseen obstacles and is forced to replan the path, the RHS score is simply recalculated with the new connectivity cost for all affected nodes. (Choset, et al., 2007)

3.3 Grids

A common and intuitive representation is a grid, which discretizes the environment into many individual nodes. In a 2D environment, these nodes can be 2D grid of squares, triangles, hexagons. Squares are more commonly used for path planning. the successors on square grids are not equidistant, as diagonal successors are separated by a distance of $\sqrt{2} = (1.4)$ and one for vertical or horizontal movement. (Murphy, 2000). The Euclidean distance is given by the below equation

$$\text{Euclidean Distance } (s, s') = \sqrt{(x_s - x_{s'})^2 + (y_s - y_{s'})^2} \quad (3.7)$$

3.4 Modified D* Lite

1. Path Cost

The figure 3-1 shows the path cost for various possible scenarios. Traveling from a blocked cell or to a blocked cell has a cost of infinity. While from one cell that is free to another cell is free the cost is one(1) for non-diagonal transverse and $\sqrt{2} = (1.4)$ for diagonal movements.

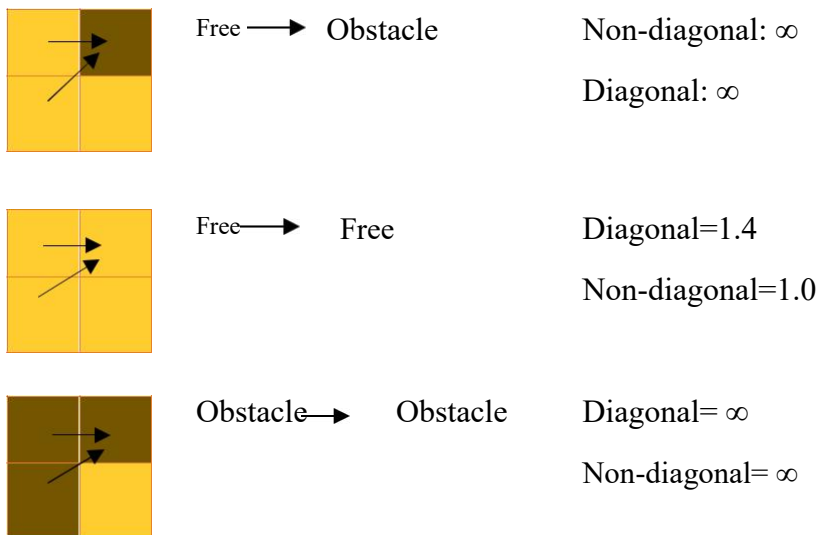


Figure 3.1: path costs

The figures below explain a case scenario of the proposed modified D* lite in grid type called *Eight connected*, with pairwise directed edges between neighbouring cells of the modified D* lite robot path planning algorithm and how they are modifies and updated at each stage.

The first step is where the modification is done instead of initialising all the node only connected nodes are initialised because not all nodes will be searched during the course of path planning.

S denotes the finite set of states of the domain.

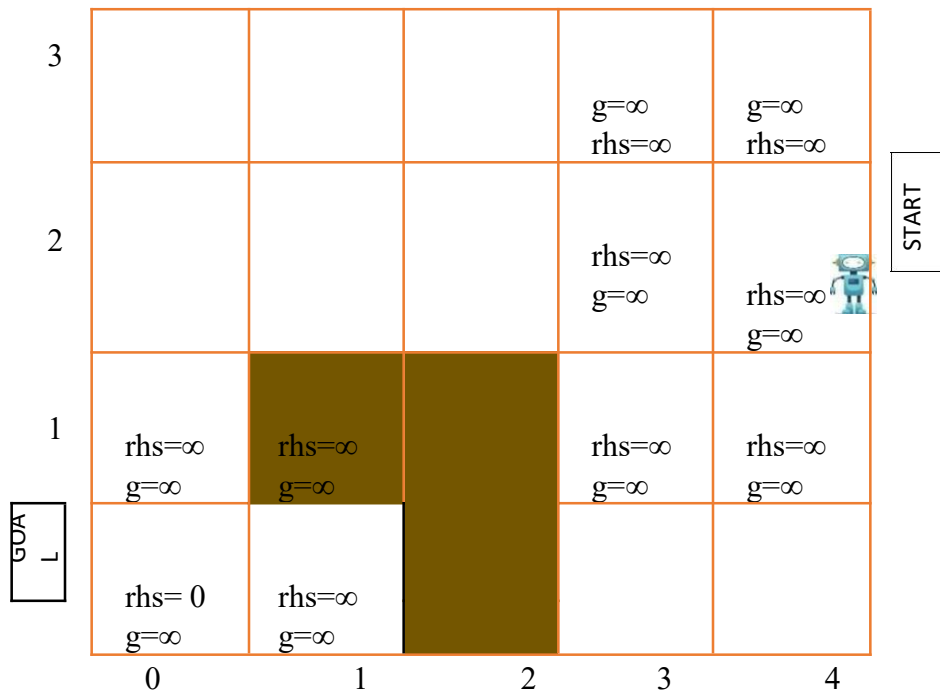
for all $s \in S$ connected to the goal(s) $rhs(s) = g(s) = \infty$;
 for all $s \in S$ connected to the start $rhs(s) = g(s) = \infty$;

else $rhs(s) = g(s) = \text{null}$

$rhs(s_{goal}) = 0$

UpdateState(sgoal)

Initially, as shown in figure 3.2(a) the entirety of associated nodes' value of rhs and g are infinite while the g is infinite, the rhs is zero (0) for the goal node. The aim has been moved to the open list due to its inconsistency. as shown in figure 3.2(b)



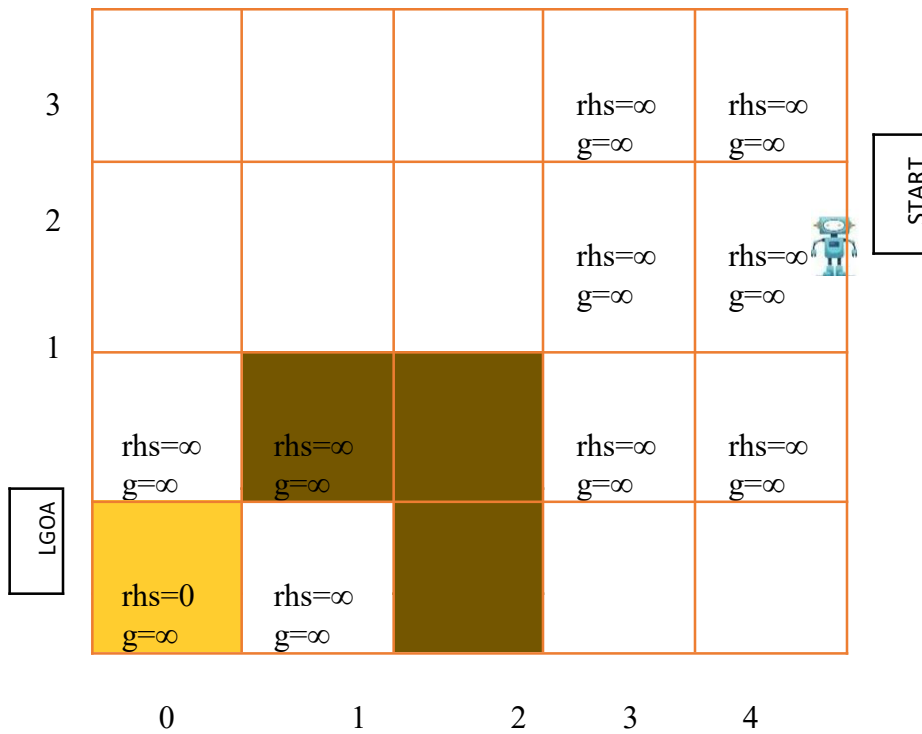


Figure 3.2: (a) set to zero the goal rhs, and set to infinity other rhs and g
 (b) and added to the list open

After the first stage, the goal is put on the open list. The *computeshortestPath* is called and since the goal rhs value is greater than g i.e ($rhs > g$) it is removed from list *open* and made equal to h value as depicted in figure 3-3(a)

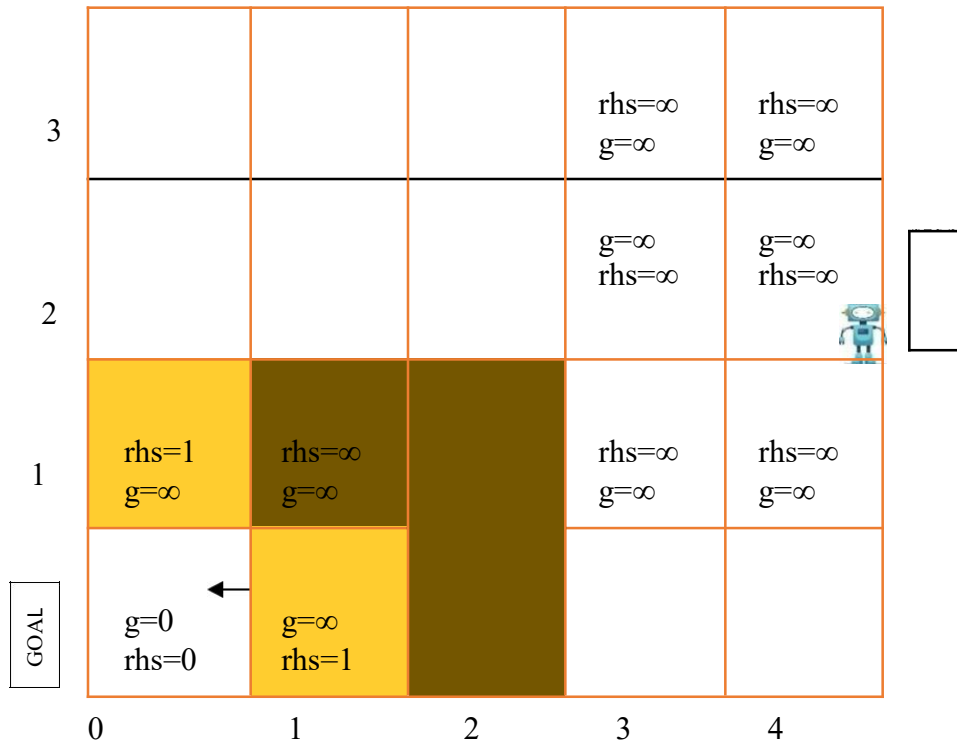
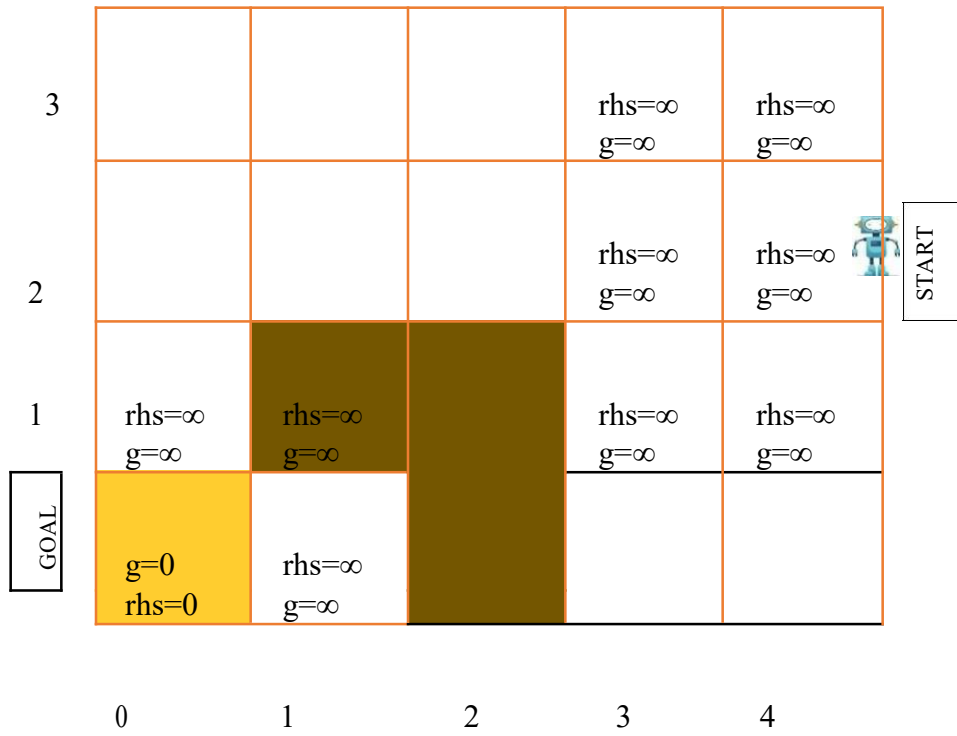


Figure 3.3: (a) The objective is removed from the open list.

(b) target is expanded, and the outcome of inconsistencies are added to the list *open*

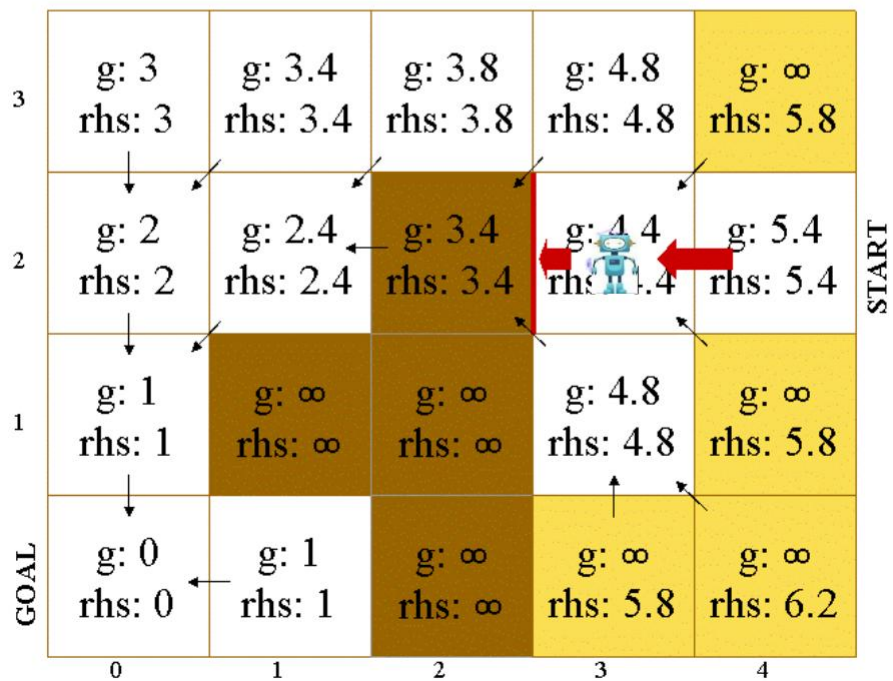
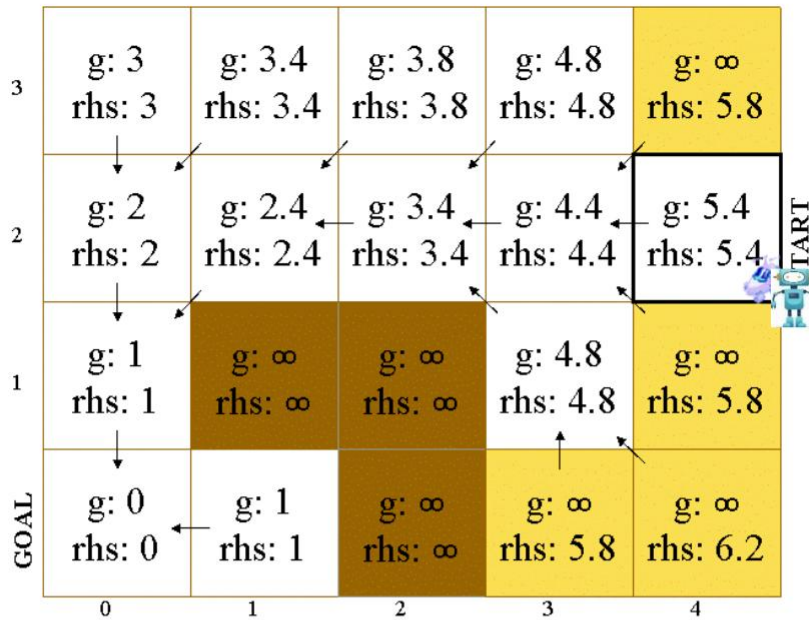


Figure 3.5: When an obstacle pops up in node label (2,2) on the path already planned Figure 3.5(a) the process is repeated till an optimal path is found. In Figure 3.5(b) When an obstacle pops up on the path already planned, a replanning is done though not from the root but from the current node and every other node affected.

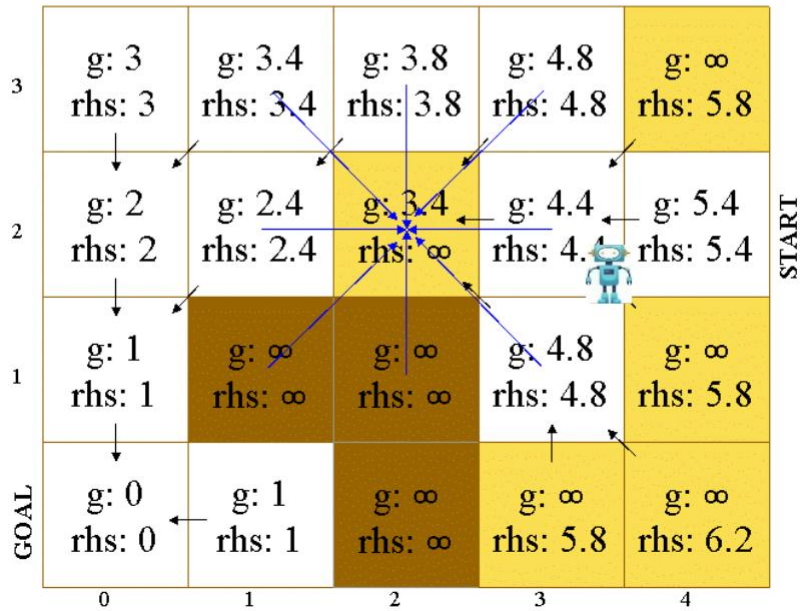
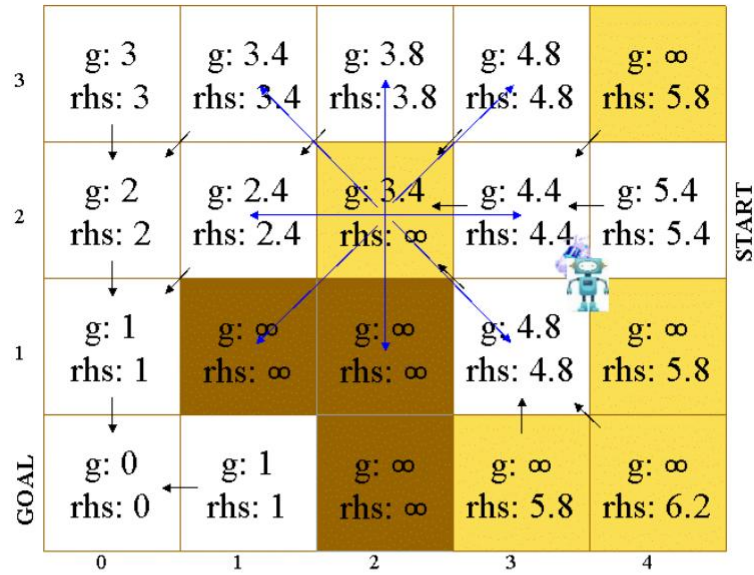


Figure 3.6: (a) Outgoing edges to node label (2, 2)

(b) Incoming edges to node label (2, 2).

when an obstacle pops up in the already found path all affected node will be revisited, recalculated and the process from step one is repeated until another optimal part is found.

Figure 3.6 (a) shows all the outgoing edges of the affected node while figure 3.6(b) shows all the incoming edges of the affected nodes to be revisited and updated.

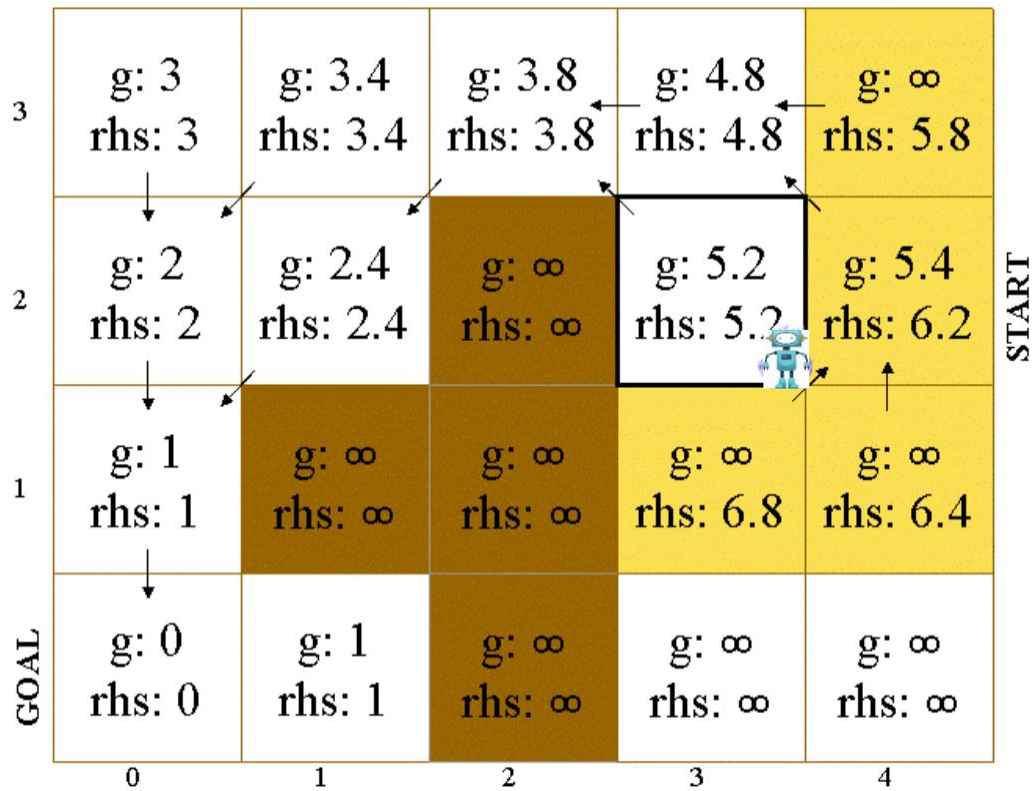


Figure 3.7: updating nodes affected

As shown in figure 3.7 The affected node is placed in the open list first, then explored, the outcome of inconsistent nodes are placed on the list open. The minimal node is removed from the list, but no neighbours are expanded on the open list. Then remove the minimum node from the open list and replace it with inconsistent neighbouring nodes.

Figure 3.8 (b) repeats the *ComputeShortestPath* step until a new optimal solution is discovered. The present robot location must be consistent and have the minimal key value in the list *open* based on all available information for the robot to identify the ideal path.

Note that if $rhs(s) = \infty$, no node will point to s as its minimum cost successor

D* Lite works by keeping track of two scores for each node on a given graph. The first score is the G score. This value keeps track of the cost of arriving at a position on the graph. It can be thought of as the shortest path that connects the starting point to the current point. For the starting point, the G score is zero because the pathfinder is already there. The *rhs* score, on the other hand, is a more useful estimate. This score acts as a one-step look ahead and, in the algorithm, it is used to find the next optimal position for the pathfinder to travel to. If the next position is an obstacle, the connection costs would be set to infinity because the pathfinder can never reach that point. As a result, the *rhs* score is also infinity. Using these two scores, the algorithm propagates a path from the ending point to the start point eventually finding the optimal path from start to finish. (Choset, et al., 2007).

This chapter was dedicated to expounding knowledge on the chosen path planning algorithm which is D* lite. Going further in the chapter, investigation is made on D*lite algorithm which led to modification on D* lite algorithm which in turn lead to the adoption of the name modified D*lite algorithm. Further in this chapter a simple case scenario was used to explain the proposed modifications.

The results are presented in the next chapter. Considering the test cases and validating the results gotten and comparing it with the popular heuristic planning algorithm in both tables and graphs.

Chapter 4

RESULT AND DISCUSSION

The simulation parameters and setups are described in the subsequent paragraph considering the map used, simulation environment, parameters to be measured and how they are measured

4.1 Map Generation

To test our algorithm on various environments quickly, grids is used as small maps. This ensured that we could create my own maps without relying on datasets for testing of path planning algorithm. The size of the world considered is 250 x 250 but was modified for test cases and different other scenarios.

4.2 Python Programming Language

The python programming language has been one of the most preferred, widely used and efficient programming languages in implementing machine learning, deep learning and artificial intelligence algorithms by many researchers across various discipline. Python is used in this work. Python programming language is widely preferred because it is enriched with various readily available library functions making complex problems easier with few lines of codes. Some useful libraries used include:

- Pandas
- Numpy
- Matplotlib
- Math
- Time and collection

4.3 System Specification Requirement

System specification requirement which is further categorized into hardware requirement and software requirement.

a. Hardware requirement

- A modern computer System

b. Software requirement

- Windows 10, Linux, Mac or any other operating system that supports python
- 32-bit / 64-bit Operating system
- Anaconda, PyCharm, Pydev, Eclipse, Jupyter Notebook, Spyder editor

4.4 Measurements

I track and report measures for the complexity of the moving target search problems, including; the number of moves for the hunter until it catches the target (get to the goal(s)), the number of searches. I measured the search algorithms efficiency by tracking and reporting two parameters; the average number of nodes expanded per search and the average time taken to find a path and reach the goal per search in microseconds on a Pentium (R) Dual-core 2.3 2.3 Ghz PC with 6 GB of RAM.

4.5 Results

The results of the research are discussed in detail in this chapter. Various scenario in both known and unknown maps are considered under the following test cases;

Test case 1: Single goal in a map without obstacles

Test case 2: Single goal in a map with obstacles

Test case 3: Multiple goals (static and moving) in a map without obstacles

Test case 4: Multiple goals (static and moving) in a map with obstacles

1. Test case 1

Single goal in a map without obstacles: the map is a 250x250 grid size and the goal is single and there is no obstacle on the map and the environment considered are both static and dynamic.

a. Single goal without obstacle

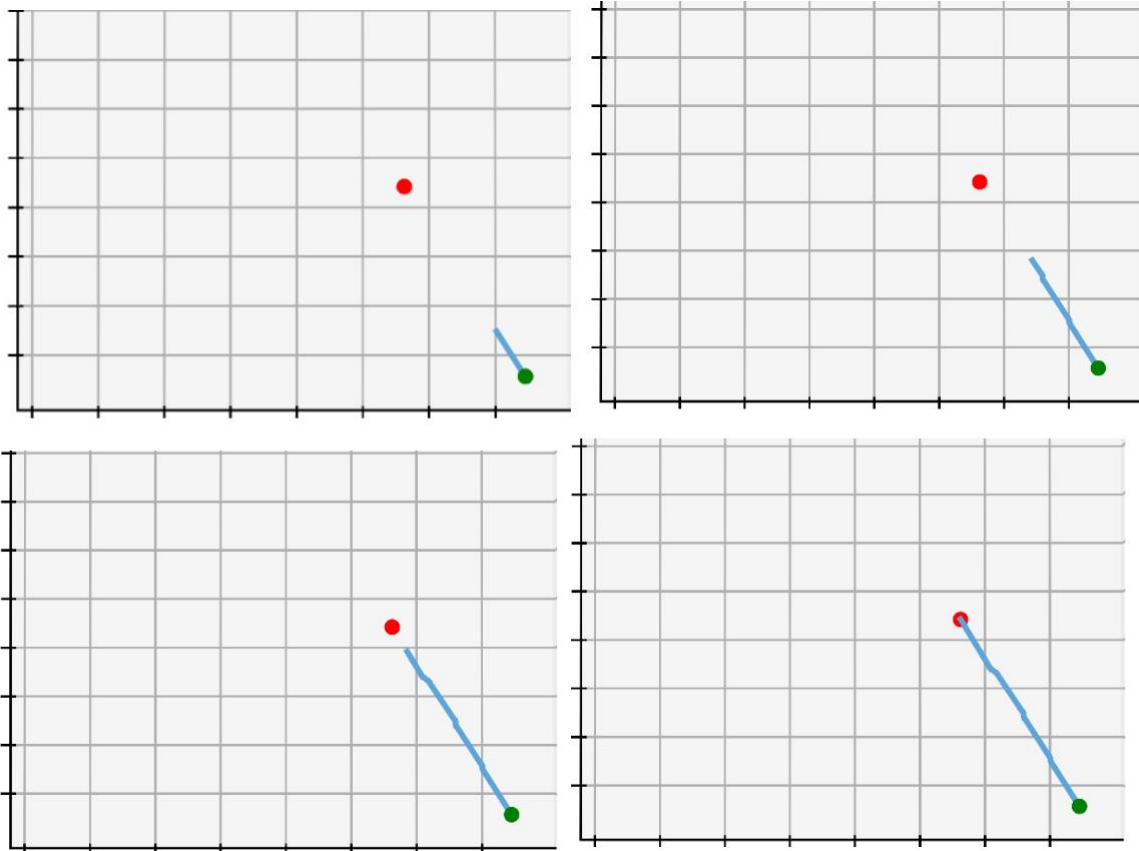


Figure 4.1: Stages of a Static goal in an environment without obstacles

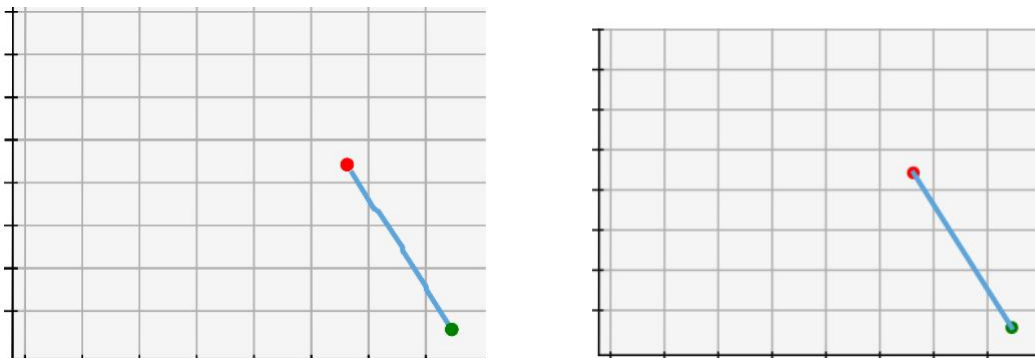
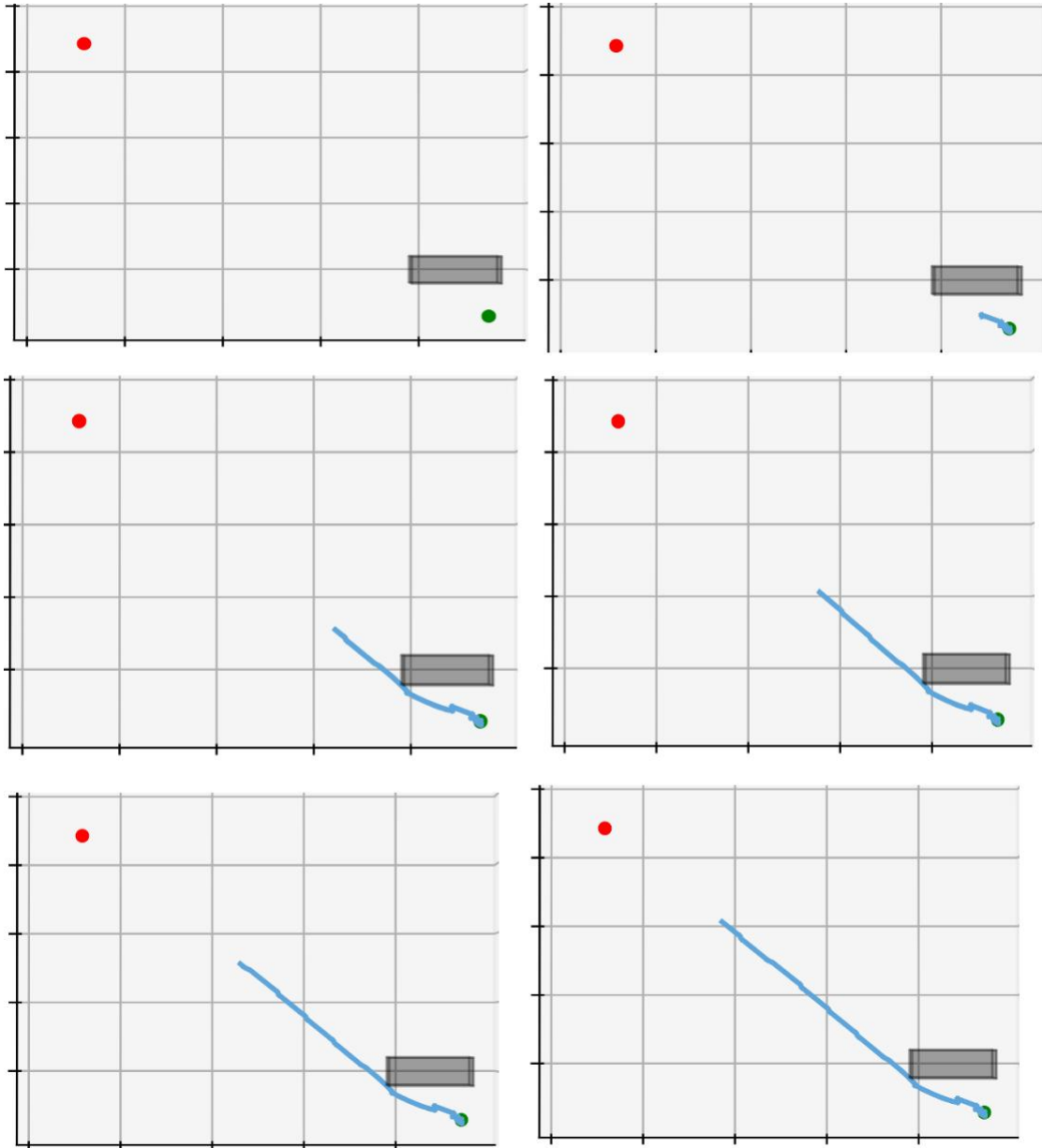


Figure 4.1b: final results in an unknown(right), known(left) environment

2. Test case 2

Single goal in a map with obstacle: the map is a 250x250 grid size and the goal is single and there is a single obstacle on the map and the environment considered are both static and dynamic

b. Single goal with obstacle



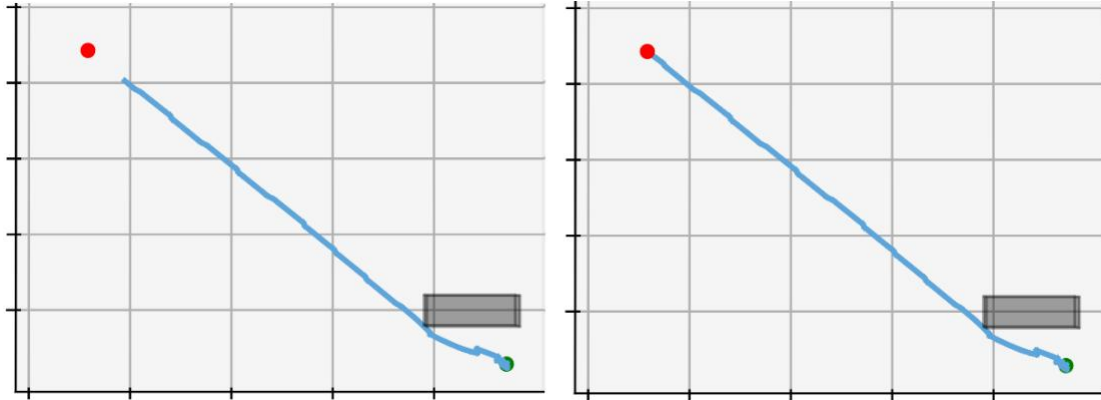


Figure 4.2: Static goal with obstacle in an unknown environment

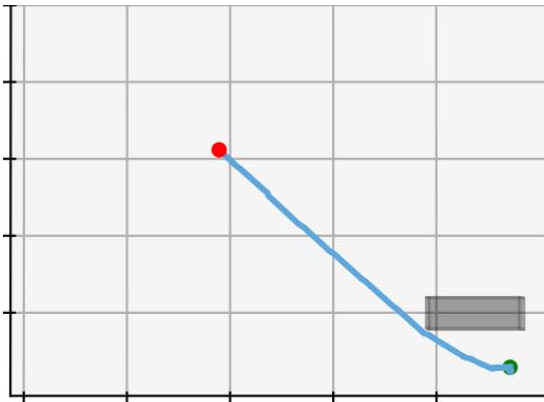


Figure 4.3: Static goal with obstacle in a known environment

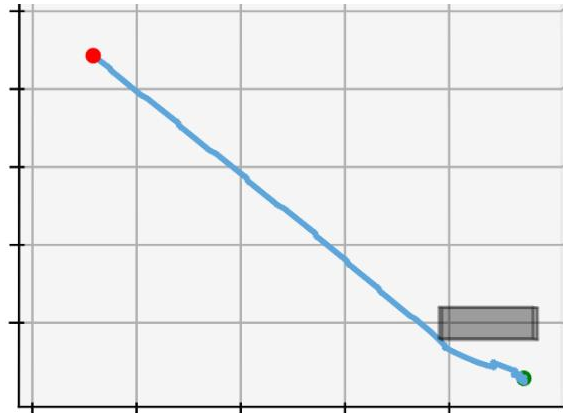


Figure 4.4: Static goal with obstacle in an unknown environment

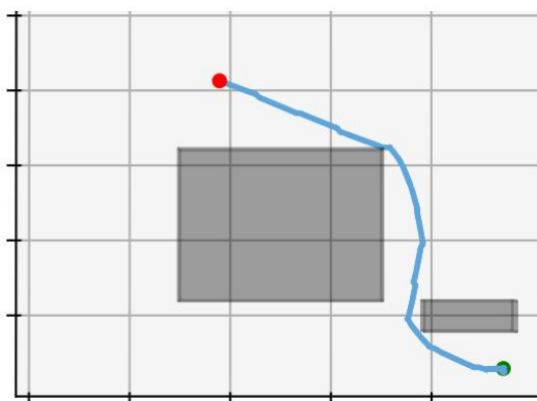
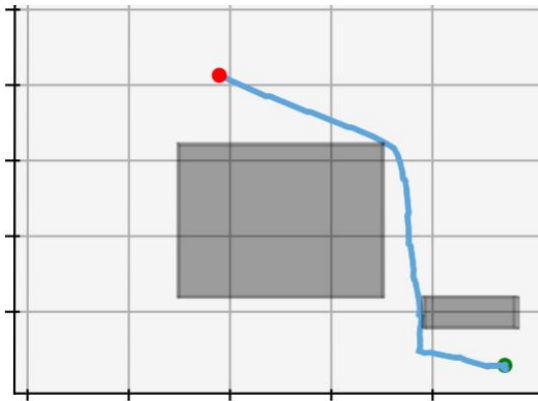
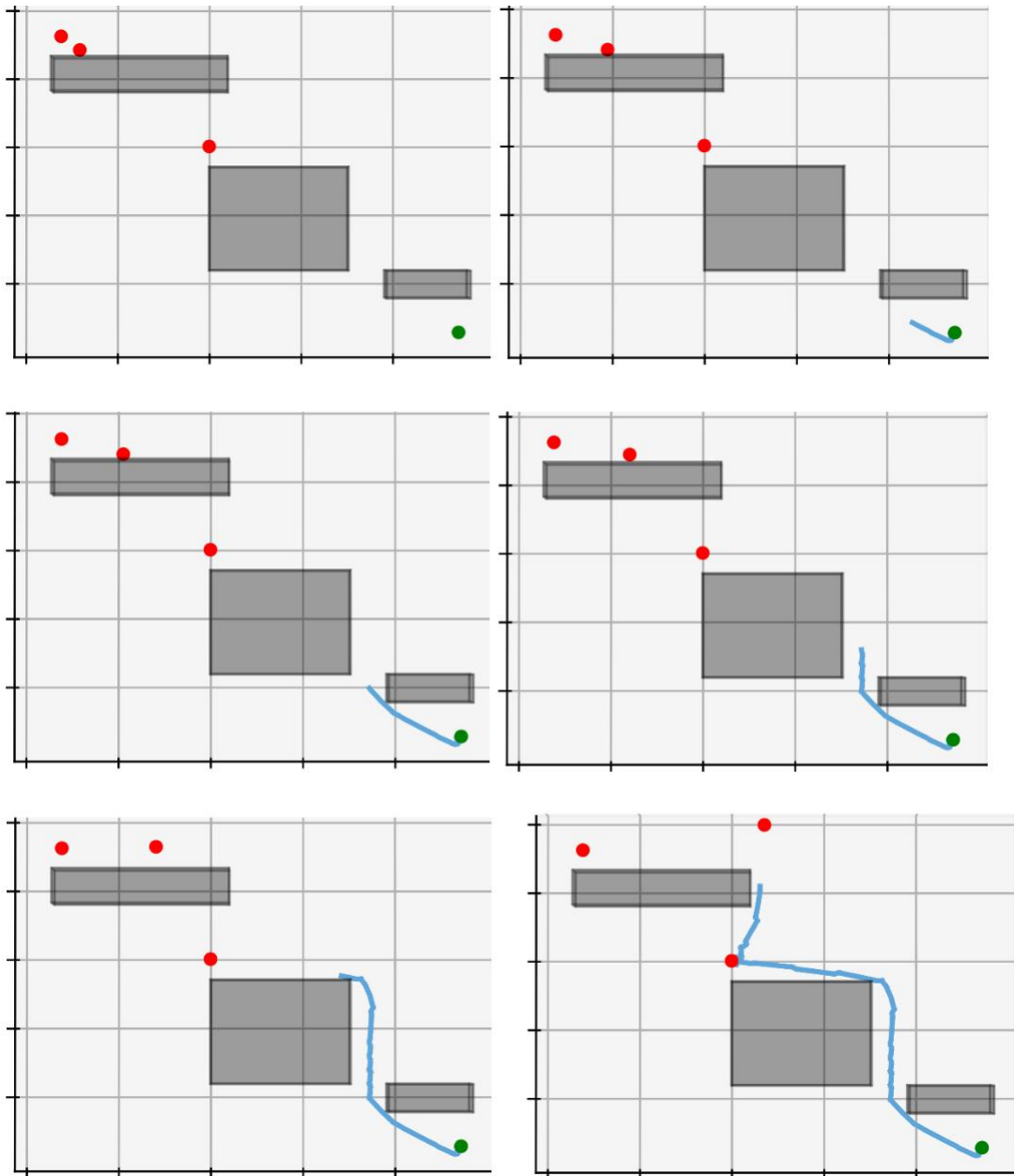


Figure 4.5: Static goal with more obstacles

3. Test case 4

Multiple goals (static and moving) in a map with obstacles: the map is a 250x250 grid size and the goal are three (one moving goal and two static) and there are multiple obstacles on the map and the environment considered are both static and dynamic.

c. Multiple goals (dynamics and static) plus obstacles



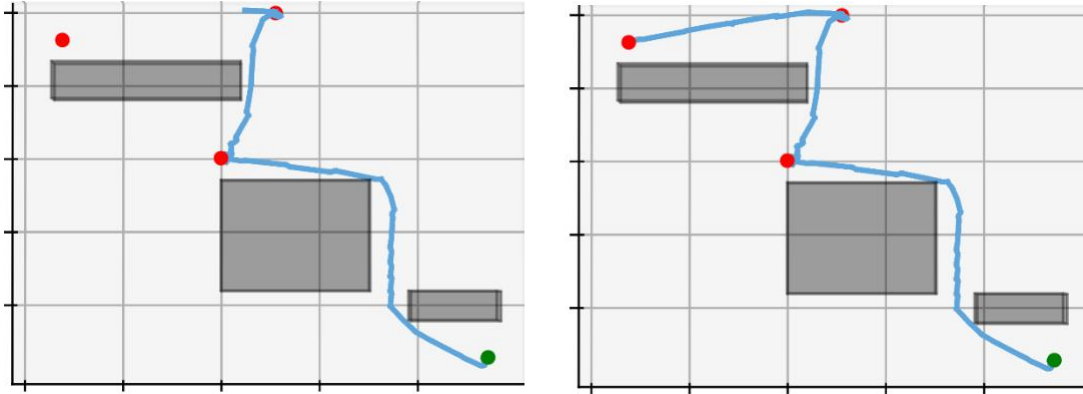


Figure 4-6: Static and dynamic obstacle in a known environment

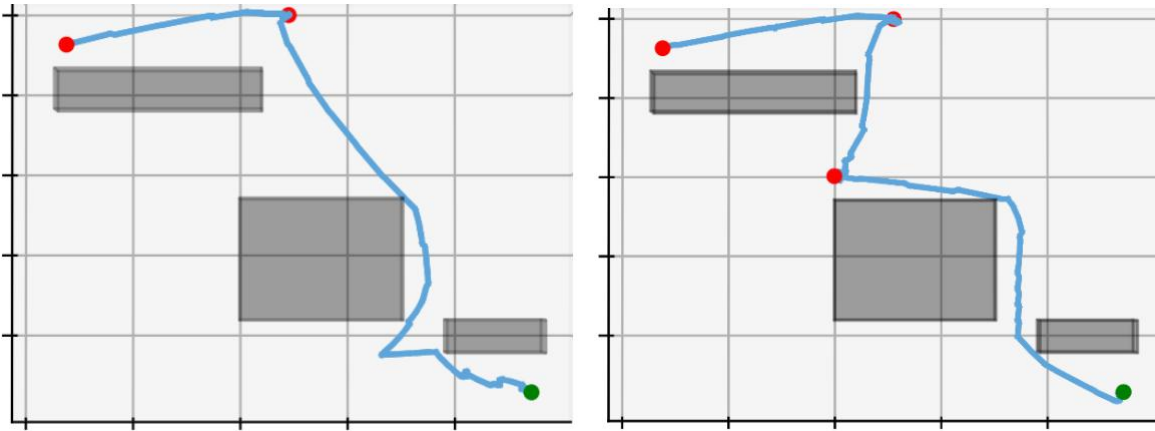


Figure 4.7: multiple obstacles in an unknown environment

Figure 4.8: multiple obstacles in a known environment

4.5.1 Visualisation

A problem that persistently occurs in the algorithm during the process of the robot getting to the goal(s), is that the complete path is not always captured to enhance a complete and more detailed visualization as to its various itinerary. The algorithm could only capture random and non-sequential images, which leads to incomplete knowledge of what goes on at every point in time. By what goes on at every point in time it means one, either or a combination of the following.

- I. The Orientation of the agent.
- II. The Orientation of the goal(s).
- III. The position of the obstacle(s).
- IV. The position of the robot in relationship to the goal(s) and obstacle(s).
- V. The position of the goals in relationship to the obstacle(s).
- VI. The position of the goal in relationship to other goal(s) in case of dynamic goals.

This is what led to the modification in the algorithm to further enhance the visualization output. This will help readers and future research to have more insight and understand better; the movement of the robot, obstacle avoidance and goal(s) or target reaching complexities.

Owing to the above-described problem of random and scattered image generated, there is need for sequential and organised image generation to properly visualise accurately every single step taken by the robot during its course of goal(s) finding. Packages utilised included Matplotlib, Os, moviepy. Matplotlib is used for visualization because it is both fast and provides very quality figure to visualize data in python in diverse format. Os is used to read or write files.

For each step of the robot movement simulated, a plot is made and saved to a file. This is done by specifying a filename, then the *plt.savefig* function is called. Other *savefig* options can be specified such as; dpi, bbox_inches. The saved images are then processed to give an overview of the total journey of the robot by video and gif by the code for better visualization.

Pseudo code

```
ax.scatter(start, colour='g') # plots the robot start position and specifies its colour notation  
ax.scatter(goal, colour='r') # plot the goal(s) position and specifies its colour notation  
makeVideo # this calls the saved plots and merge them into a video. Moviepy is used  
plotname = () # this is the specified filename  
plt.savefig(plotname,dpi=gl.dpi,bbox_inches='tight')  
frames.append(plotname)  
ax1.plot(x, y, linewidth=1, c='r',zorder=2) # zorder specifies where the line is plotted below or above the patch collection  
Plot_folder='imageformat'  
fps=1  
plot_files = [Plot_folder+'/' +img for img in os.listdir(Plot e_folder) if img.endswith(".png")]  
makevideo = moviepy.video.io.ImageSequenceClip.ImageSequenceClip(image_files, fps=fps)  
makevideo.write_videofile('plotvideo.mp4')  
plt.show()  
makegif = VideoFileClip('plotvideo.mp4')  
clip = makegif.subclip(0, 3) # getting specific time duration in  
seconds clip.write_gif('plotvideo.gif') # saving video clip as gif gif =  
VideoFileClip("plotvideo.gif") # loading gif  
gif.ipython_display() # showing gif
```

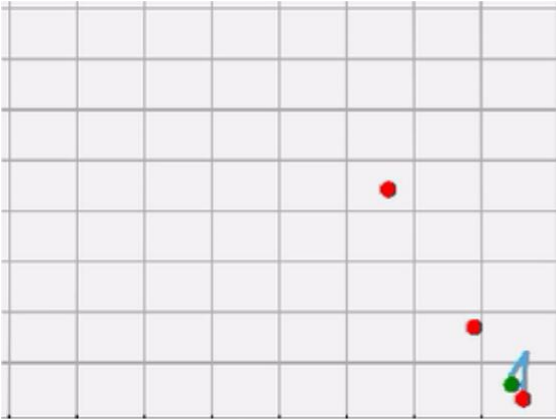


Figure 4.9: No obstacles in a dynamic environment

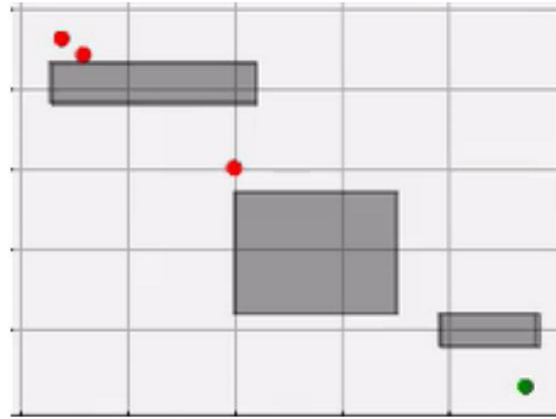


Figure 4.10: multiple obstacles in a known environment

The first step after modifications of the D*lite algorithm henceforth called modified D*lite is to compare the results gotten with previous research.

Table 4.1: Table comparing results of Heuristic planners

Map size	A*star		D*star Lite		D*star Lite Modified	
	Number of nodes expanded	Path finding time(ms)	Number of nodes expanded	Path finding time(ms)	Number of nodes expanded	Path finding time(ms)
10x10	22	63.3	13	23.5	10	20.8731
50x50	287	329.5	90	27.7	80	25.3998
100x100	1036	936.3	265	36.4	257	26.0193
150x150	1433	1527.2	798	41.1	543	30.4860
250x250	2632	4705.9	1485	47.6	1169	38.2316
500x500	4876	21006.1	2355	62.9	1332	55.7358

This table 4.1 compares results obtained from previous researches on well know path planning algorithm such as A* star, D* lite and the modified D*lite (Jeffrey Zhang, 2020)It is seen that the modified D*Lite performs better on the two metrics used to measure optimality which are number of nodes expanded and mean path finding time.

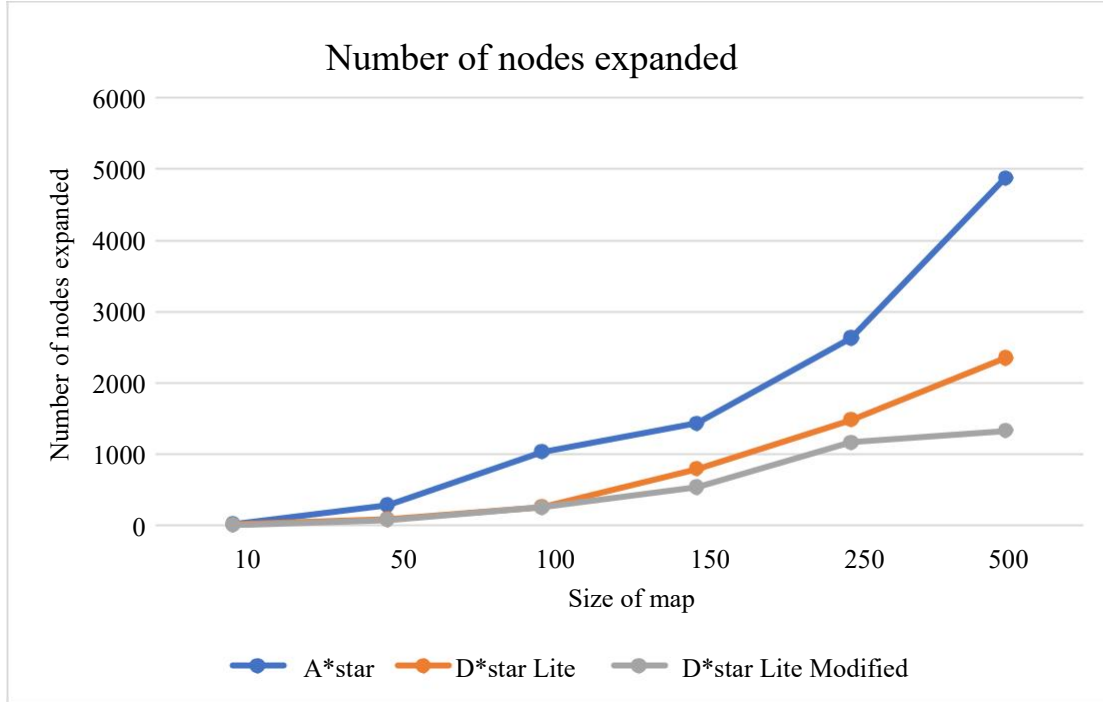


Figure 4.11: Graph showing number of nodes expanded

The graph figure 4.11 above shows a graphical representation of the performance of the three-path planning algorithm whose performances are compared. It is a plot of the number of nodes expanded against the size of the map and clearly Modified D* star lite outperforms both A* star and D star lite both in static and dynamic environment at different sizes of map by maintaining the least number of nodes search during each test case.

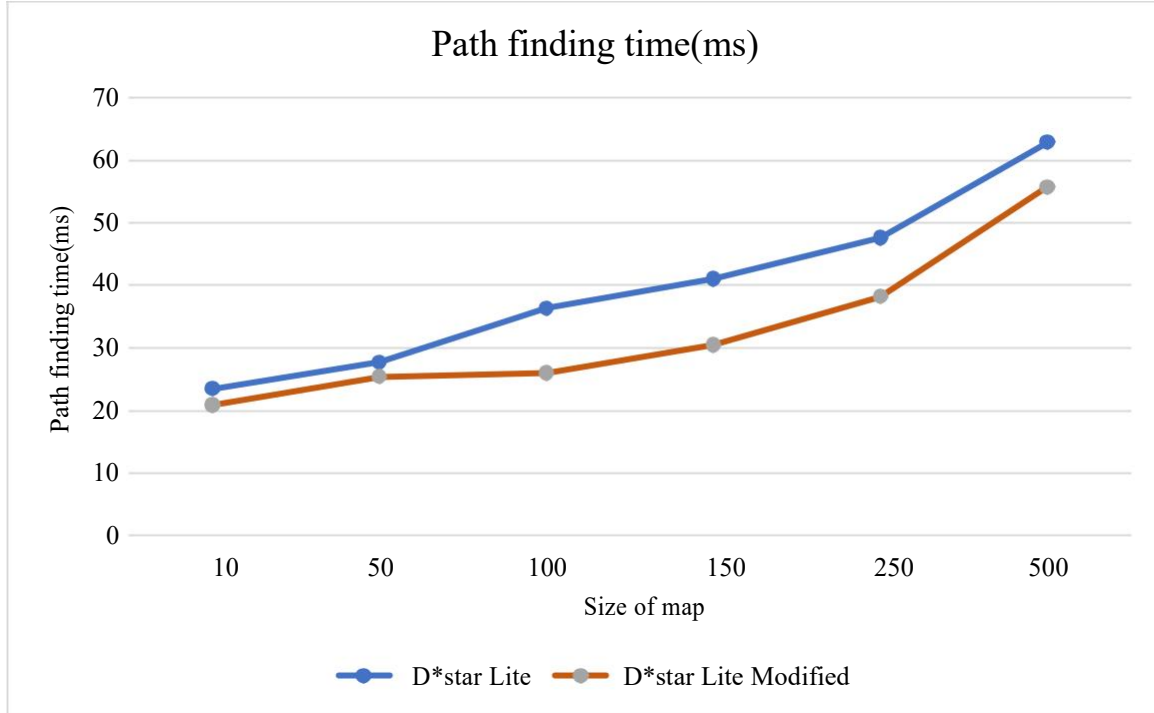


Figure 4.12: Graph showing mean path finding time

The graph figure 4.12 above shows a graphical representation of the performance of the two path planning algorithm whose performances are compared. It is a plot of the mean path finding time against the size of the map and again Modified D* star lite outperforms both D* star lite both in static and dynamic environment in different sizes of map by maintaining the least time pathfinding time.

Table 4.2: Table showing results of the modified D* Lite

Environment	Number of goal(s)	Total Cost	Number of nodes expanded	Path finding time(ms)
Static	1(static)	42.1907	257	23.8912
Dynamic	1(dynamic)	52.13266	257	42.1932
Static	2(1static, 1dynamic)	198.9996	2530	39.2482
Dynamic	2(1 static, 1dynamic)	224.6536	4803	65.7640
Static	3(2 static, 1dynamic)	321.7977	6187	67.9829
Dynamic	3(1 static, 2dynamic)	310.5952	5190	83.3902
Static	3(2 static, dynamic)	461. 3014	8998	112.9337
Dynamic	3(1static, 2 dynamic)	480. 0252	10893	128.6919

The table 4.2 above shows the results of the modified D* lite in static(known), dynamic (unknown) environment as well as its performance when the target(goal) is single or multiple as well as if the goal is static or dynamic.

To further verify and establish the performance of modified D* Lite algorithm and also extend the capability of the algorithm it was tested in both static and dynamic environment while keeping the map size constant at and varying both the number of goals and their attributes (static and dynamic).

4.6 Discussions

It can be seen from table 4.1 and table 4.2 and figures 4.9 and 4.10 that as the environment gets complex from being dynamic to increase in the number and nature of the goals the search time and number of nodes expanded increase and the modified D* lite algorithm outperforms previous planners such as A*, D* lite.

4.6.1 Obstacle densities

It is observed the more complex the environment is (that is the obstacle density), the longer planning takes and the longer the resulting path lengths. Path planning can be sped up by tuning a number of parameters, but his most times results in an increase in path costs

4.6.2 Node expansion

When there is an increase in the number of nodes expanded this can be as a result of various factors. Firstly, the number of goals is a major factor, Again the map size is another factor because the larger the map the larger the number of nodes expanded because this means the targets/goal is further away from the robot start position and there are more nodes to explore to successfully reach the goal. Another factor is that the range of the sensor does not change whether or not the map reduces or increases as map size increases. This limitation will result in less informed paths on larger map, as the robot vision is constrained to a small fraction of the entire map thereby increasing the likelihood of more suboptimal paths and therefore require frequent replanning and more node expansions. Lastly, in a dynamic(unknown) environment and in dynamic (moving) goal problem, the run times and node investigated rose in number, because they update more g-values and consequently every search explored more states and hence it influences the performance of the algorithm.

In this chapter the results are presented from different test cases and their graphical results to comparing heuristic planner to validate and show improvement on modified D* Lite. The results were discussed and the implications of the results presented considering obstacles density and node expansion. The conclusion of the research is presented in the final chapter.

Chapter 5

CONCLUSION

Modified D* lite is a fast reliable and dynamic incremental path planning method and reuse the past search tree and hence explore less states per search.

And in this study, it has been improved D* star lite further to account for dynamic goals, static goals in both dynamic environment and static environment and incorporating single and multiple goals(targets) in an obstacle-free map and in map with obstacle with different level of obstacle density with better visualization for easier understanding of the subject matter.

Different difficulties encountered during the course of testing include irregular results. This was solved by running repeatedly every step of the simulation over a minimum of 50 times and taking the average of the results obtained. Another is programming glitches generating slightly different images at different points.

The results obtained shows a reasonable improvement over D* lite algorithm and it is an indication that it can be improved further for real life application such as rescue mission in disaster zone where human rescue team cannot readily or easily access due to known and unknown obstacle and other related applications.

The work described in this paper can be further extended to account for multiple agents chasing same goal, multiple robots chasing different targets. Again, a possible combination of the method used and UFASTSLAM and any other efficient algorithm can be investigated. Also, in future applications the next steps for further improvement will be to use the mapping of a real-life environments instead of a fictitious map randomly generated to improve the robustness of the applicability. Finally, implementation to a real robot is another future work.

REFERENCES

- Adi Botea , Martin Müller , Jonathan Schaeffer. (2004). Near optimal hierarchical path-finding (HPA*). *Journal of Game Development*, 1, 7-28.
- B.K. Patle, G. B. (2019). A review: On path planning strategies for navigation of mobile robot. *Defence technology*, 15(4), 582-606
- Choset, H., & Burdick, J. (2000). Sensor-based exploration: the hierarchical generalized Voronoi graph. *The International Journal of Robotics*.
<https://doi.org/10.1177/02783640022066770>
- Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., & Thrun, a. S. (2007). *Principles of Robot Motion: Theory, Algorithms, and Implementation*.
- Dave Ferguson, A. S. (2005). The Delayed D* Algorithm for Efficient Path Replanning. *Robotics and Automation*,.in *Proceedings of the 2005 IEEE International Conference*.
- Davidor, Y. (1990). Robot programming with a genetic algorithm. *In Proceedings of international conference on computer systems soft engineering*, 186-191.
- Eusuff M, M. L. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management*, 8(3), 210-215.
- Ferguson, D. a. (2007). Field d*: An interpolation-based path planner and replanner. *Robotics Research Springer*, 239–253.
- Francisco Rubio, Francisco Valero, Carlos Llopis-Albert. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*. <https://doi.org/10.1177/1729881419839596>
- Goerzen C, Z. Kong, B. Mettler. (2010). A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems*, 57, 65- 100.
- Gregor Klančar, Andrej Zdesar, Saso Blazic, Igor Skrjanc. (2017). *Wheeled Mobile Robotics. From Fundamentals Towards Autonomous Systems*. Butterworth-Heinemann.
- Han-ye Zhang, Ei-ming Lin and Ai-xia Chen. (2018). Path Planning for the Mobile Robot: A Review. *symmetry* 10(10), 450
- Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths . *IEEE Transactions on Systems Science and Cybernetics*. 4(2), 100-107

- Imen Chaari, Imen Châari*, Anis Koubâa, Sahar Trigui, Hachemi Bennaceur, Adel Ammar, Khaled Al-Shalfan. (2012). smartPATH: A Hybrid ACO-GA Algorithm for Robot Path Planning. *International Journal of Advanced Robotic Systems*
<https://doi.org/10.5772/58543>
- Ishida, T., & Korf, R. (1992). Moving target search with intelligence. *In Proceedings of the 10th National Conference on Artificial Intelligence*. San Jose, CA.
- Kaufmann, M. (2003). *Exploring Artificial Intelligence in the New Millennium*. San Francisco, CA, United States, Morgan Kaufmann
- Kobti, D. C. (2008). Memory-Bounded D* Lite. *In Proceedings of the Twenty-First International FLAIRS Conference*. (pp 376-378)
- Koenig, S. (2004). A comparison of fast search methods for real-time situated agents. *In Proceedings of the International Joint Conference on Autonomous Agents and multiagents system (AAMAS)*, (pp. 864-871).
- Koenig, S., & Likhachev, M. (2002). Fast replanning for navigation in unknown terrain. *IEEE Transactions On Robotics*,21(3), 354–363.
- Koubaa, A., Chaari, H. B., Ammar, S. T., Mohamed-Foued, Alajlan, S. M., & Javed, O. C. (2018). *Robot Path Planning and Cooperation Foundations, Algorithms and Experimentations*. springer.
- Kunchev. V, L. Jain, V. Ivancevic, A. Finn (2006). Path planning and obstacle avoidance for autonomous mobile robots: a review. *Knowledge-Based Intelligent Information and Engineering Systems*. (pp 537-544) Springer, Berlin, Heidelberg
- L.E. Kavraki, J.-C. Latombe, and M. Overmars. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*,12(4) 566 - 580.
- Latombe, J.C. (1991). Robot motion planning. *Kluwer International Series in Engineering and Computer Science*. Kluwer Academic
- Lee, D. C. (1996). *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped*. Cambridge, United Kingdom: CAMBRIDGE UNIVERSITY PRESS.
- Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., & Thrun, S. (2005). *Anytime Dynamic A**: *The Proofs*. Technical Report CMU-RI-TR-05-12, Carnegie Mellon School.
- Maram Alajlan, M Alajlan, A Koubaa, I Chaari, H Bennaceur, A Amma (2003). Global path planning for mobile robots in large-scale grid environments using genetic algorithms. *In proceeding of International Conference on Individual and Collective Behaviors in Robotics (ICBR) 2003*.

- Masehian, Ellips, D. Sedighizadeh (2007). Classic and Heuristic Approaches in Robot Motion Planning - A Chronological Review. *World Academy of Science, Engineering and Technology* 101-106
- Moldenhauer, C. &. (2009a). Optimal solutions for moving target search. *IJCAI 2009, In Proceedings of the 21st International Joint Conference on Artificial Intelligence*.584-589 Pasadena, California, USA.
- Murphy, R. R. (2000). *Introduction to AI Robotics*. MA: MIT Press, Cambridge.
- Nourbakhsh, I. R. (1997). Interleaving Planning and Execution for Autonomous Robots. In Springer, *The Springer International Series in Engineering and Computer Science book series*, 385, 53-64 . Boston, MA, Springer.
- Otte, M. W. (2017). A Survey of Machine Learning Approaches to Robotic Path-Planning. *In proceeding 2017 International Symposium on Multi-Robot and Multi-Agent Systems*.
- Parker, J. K., & Khoogar, A. R. (1989). Inverse kinematics of redundant robots using genetic algorithms. *In Proceeding International Conference on Robotics and Automation (ICRA)*., (pp. 271-276).
- Passino K, M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control System*, 52-67.
- Raja, P. P. (2012). Optimal path planning of mobile robots: a review. *International Journal of Physical Sciences* 7(9), 1314 - 1320.
- Reif, J. H. (1979). Complexity of the movers problem and generalizations. *In Proceedings IEEE Symposium on Foundations of Computer Science* (pp. 421-427).
- Seda, M. (2007). Roadmap method vs. cell decomposition in robot motion planning. *In Proceedings of 6th WSEAS international conf on signal processing, Robotics and automation*, (pp. 127-132).
- Stentz, A. (1995). The focussed D* algorithm for real-time replanning. *In IJCA Proceedings of the 14th international joint conference on Artificial intelligence*, (pp. 1652–1659)
- Sturtevant, C. M. (2009b). Optimal Solutions for Moving Target Search. *In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, (pp. 1249–1250).
- Sven Koenig, M. L. (2002). D* Lite. *In proceedings of American Association for Artificial Intelligence* 202-215
- Sven Koenig, M. L. (2004, June). Incremental Heuristic Search in AI. *AI Magazine, Volume 25(2)*, (pp 99-112).

Yang X, S., & Deb, S. (2009). Cuckoo search via Levy flights. *In proceedings of Nature and biologically inspired computing, NaBIC 2009, IEEE world congress*, 210.

Yang, X.-S. (2008). Firefly algorithm Nature Inspired Metaheuristic Algorithms. *Sciencedirect*, 79-90.

APPENDICES

APPENDIX 1

Pseudo code

function CalculateKey(s)

return [min(g(s), rhs(s)) procedure Initialize () + h(s, sgoal) + km; min(g(s), rhs(s))]; // to compute the priority that it should have

procedure update_obstacle(s)

M =initialize_obstacle_map; //update map with start positions, initial obstacles and goal

procedure update_obstacles

OPEN = ∅;

km = 0; // cost

procedure Initialize () // rhs value is set, the goal value to zero (0) and all other connected nodes g and rhs values to ∞.

for all s ∈ S connected to the goal(s) rhs(s) = g(s) = ∞;

for all s ∈ S connected to the start rhs(s) = g(s) = ∞;

else rhs(s) = g(s) = null

rhs(sgoal) = 0

sstart = the hunter current state;

sgoal = nearest target () current state

U.Insert (sgoal, CalcKey(sgoal))

function UpdateVertex(u) // vertices for consistency inserts or pop off from the priority queue

function Main () // this is the main program that runs until the goal(s) is/are reached

Initialize ()

ComputeShortestPath ()

while sstart ≠ sgoal do

if $g(s_{start}) = \infty$ then

return no path exists

$s_{start} = \operatorname{argmin}_{s \in \text{succ}(s_{start})} (c(s_{start}, s) + g(s))$

Move to s_{start}

Scan environment for new obstacles if new obstacles exist then

APPENDIX 2

Turnitin

turnitin

Assignments | Students | Grade Book | Libraries | Calendar | Discussion | Preferences

NOW VIEWING HOME > MASTER > JONATHAN UZOEGHELU

About this page
This is your assignment inbox. To view a paper, select the paper's title. To view a Similarity Report, select the paper's Similarity Report icon in the similarity column. A ghosted icon indicates that the Similarity Report has not yet been generated.

Jonathan Uzoeghelu
INBOX | NOW VIEWING. NEW PAPERS ▾

Submit File Online Grading Report | Edit assignment settings | Email non-submitters

<input type="checkbox"/>	AUTHOR	TITLE	SIMILARITY	GRADE	RESPONSE	FILE	PAPER ID	DATE
<input type="checkbox"/>	Jonathan Uzoeghelu -	abstract	0%	--	--		1635661989	25-Aug-2021
<input type="checkbox"/>	Jonathan Uzoeghelu -	chapter 1	0%	--	--		1635661610	25-Aug-2021
<input type="checkbox"/>	Jonathan Uzoeghelu -	conclusion	0%	--	--		1635662176	25-Aug-2021
<input type="checkbox"/>	Jonathan Uzoeghelu -	chapter 4	1%	--	--		1635661831	25-Aug-2021
<input type="checkbox"/>	Jonathan Uzoeghelu -	chapter 2	6%	--	--		1635660449	25-Aug-2021
<input type="checkbox"/>	Jonathan Uzoeghelu -	whole thesis	12%	--	--		1635660166	25-Aug-2021
<input type="checkbox"/>	Jonathan Uzoeghelu -	chapter 3	13%	--	--		1635660630	25-Aug-2021

Assist. Prof. Dr. Parvaneh Esmaili

Windows taskbar: Type here to search | 32°C Sunny | 10:15 AM 8/25/2021