

**A-STAR ALGORITHM IMPLEMENTATION  
FOR ROBOTICS PATH PLANNING  
NAVIGATION**

**EMIRUL RIDZWAN BIN NOR AZMI**

**BACHELOR OF MECHATRONICS ENGINEERING (Hons.)  
(COLLABORATION PROGRAMME WITH KARLSRUHE UNIVERSITY  
OF APPLIED SCIENCE, GERMANY)**

**UNIVERSITI MALAYSIA PAHANG**

## UNIVERSITI MALAYSIA PAHANG

### DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : EMIRUL RIDZWAN BIN NOR AZMI

Date of Birth : 27 AUGUST 1994

Title : A-STAR (A\*) ALGORITHM IMPLEMENTATION  
FOR ROBOTICS PATH PLANNING NAVIGATION

Academic Session : 2017/2018

I declare that this thesis is classified as:

- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)\*
- RESTRICTED (Contains restricted information as specified by the organization where research was done)\*
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

\_\_\_\_\_  
(Student's Signature)

\_\_\_\_\_  
(Supervisor's Signature)

\_\_\_\_\_  
New IC/Passport Number  
Date:

\_\_\_\_\_  
Name of Supervisor  
Date:

NOTE : \* If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

## THESIS DECLARATION LETTER

Librarian,  
Perpustakaan Universiti Malaysia Pahang,  
Universiti Malaysia Pahang,  
Lebuhraya Tun Razak,  
26300, Gambang, Kuantan.

Dear Sir,

### CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name : EMIRUL RIDZWAN BIN NOR AZMI  
Thesis Title : A-STAR ALGORITHM IMPLEMENTATION FOR ROBOTICS PATH  
PLANNING NAVIGATION

Reasons (i)  
  
(ii)  
  
(iii)

Thank you.

Yours faithfully,

---

(UMP Supervisor's Signature)

---

(HsKA Supervisor's Signature)

Date:

Date:

Stamp:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.



## SUPERVISOR'S DECLARATION

We hereby declare that we have checked this thesis/project and in our opinion, this thesis/project is adequate in terms of scope and quality for the award of the degree of Bachelor in Mechatronics Engineering (UMP-HsKA) in cooperation of Universiti Malaysia Pahang with Karlsruhe University of Applied Science, Germany.

---

(UMP Supervisor's Signature)

Full Name :

Position :

Date :

---

(HsKA Supervisor's Signature)

Full Name :

Position :

Date :



## **STUDENT'S DECLARATION**

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

---

(Student's Signature)

Full Name : EMIRUL RIDZWAN BIN NOR AZMI

ID Number : HA13024

Date :

A-STAR (A\*) ALGORITHM IMPLEMENTATION FOR  
ROBOTICS PATH PLANNING NAVIGATION

EMIRUL RIDZWAN BIN NOR AZMI

Thesis submitted in fulfilment of the requirements for the award of the degree of  
Bachelor of Mechatronics Engineering (Hons.) (UMP-HsKA)  
(Dual Award Programme with Karlsruhe University of Applied Science, Germany)

Faculty of Manufacturing & Mechatronics Engineering

UNIVERSITI MALAYSIA PAHANG

MARCH 2018

*Specially dedicated to*

*My father (Nor Azmi Bin Abdul Rahman)*

*My mother (Sarimah Binti Abu Samah)*

*My beloved families*

*and*

*My Lecturers*

*Thank You!*

## **ACKNOWLEDGEMENTS**

Alhamdulillah, all praises belongs to Allah, the Almighty and the most merciful for His blessing to complete this thesis and bachelor degree successfully. Not to forget, Peace and Prayer to the Prophet, Muhammad S.A.W.

First of all, with the most thankful heart, the author would like to thank Dr. Muhammad Aizzat Bin Zakaria (Senior Lecturer at Universiti Malaysia Pahang, Malaysia) for his encouragement, guidance and support from the initial to the final level of this project, which enabling the author to develop an understanding of this project. This thesis would not have been possible without his guidance and suggestions.

Next, the author would also like to express his gratitude towards Prof. Dipl.-Ing. Helmut Scherf, (Lecturer of Karlsruhe University of Applied Science, Germany) for his time and guidance throughout the process of making this project become successful. The author really appreciated of his willingness to monitor the project's progress every month, all the way from Germany in order to make sure that the project is running smoothly without much problems.

Lastly, the author owes his deepest gratitude to his families for their undeniable love, understanding and sacrifice throughout his life. It is also a pleasure to thank author's best friends for their willingness of going through all ups and downs together.



## **ABSTRACT**

This thesis is about the implementation of Astar (A\*) algorithm as path planning algorithm used in robotics navigation. This Astar (A\*) algorithm is a smart algorithm which can produce a pathway with a minimum path score avoiding the obstacles within its way. This algorithm also known as the most famous used algorithm in path planning because of its ability to provide a collision-free pathway with a minimum path score. For this project, the main objective is to design and implement an Astar (A\*) algorithm. With the used of MATLAB software, the algorithm should be able to provide an optimized pathway and the pathway generated in this software should be a collision-free pathway. In order to provide a real-time map, this project is equipped with a single camera to the prototype of the functional area. This single camera will capture an image of the environment and convert to a black and white map for the algorithm to work on. Other than that, the project used random objects as to provide obstacles in the environment (functional area). At the end of this project, the algorithm functions which to provide a shortest pathway and also a collision-free pathway is verified. The factors (light intensity, colors, position) that may affect the behavior of the Astar (A\*) algorithm were identified in the result section in this paper.

## ABSTRAK

Tesis ini adalah mengenai pelaksanaan algoritma Astar (A\*) sebagai algoritma perancangan jalan yang digunakan dalam navigasi robotik. Algoritma Astar (A\*) ini merupakan algoritma pintar yang boleh menghasilkan laluan dengan skor laluan minimum dan mampu mengelakkan halangan dalam perjalanannya. Algoritma ini juga dikenali sebagai algoritma yang paling terkenal digunakan dalam perancangan laluan kerana keupayaannya menyediakan laluan bebas pelanggaran dengan skor jalan yang minimum. Untuk projek ini, matlamat utama adalah untuk mereka bentuk dan melaksanakan algoritma Astar (A\*). Dengan menggunakan perisian MATLAB, algoritma tersebut dapat memberikan laluan yang dioptimumkan dan jalur yang dijana dalam perisian ini harus menjadi jalur bebas pelanggaran. Untuk menyediakan peta yang menandakan keadaan semasa, projek ini dilengkapi dengan kamera tunggal untuk prototaip kawasan berfungsi. Kamera tunggal ini akan menangkap imej persekitaran dan menukar ia sebagai peta untuk algoritma kerjakan. Selain daripada itu, projek itu menggunakan objek rawak untuk menyediakan halangan dalam kawasan sekitar (kawasan fungsian). Pada akhir projek ini, fungsi algoritma yang menyediakan laluan terpendek dan juga laluan-bebas-perlanggaran disahkan. Faktor (intensiti cahaya, warna, kedudukan) yang mungkin mempengaruhi kelakuan algoritma Astar (A\*) telah dikenalpasti dalam bahagian hasil dalam kertas ini.

## TABLE OF CONTENT

<b>DECLARATION</b>	
<b>TITLE PAGE</b>	
<b>ACKNOWLEDGEMENTS</b>	<b>ii</b>
<b>ABSTRACT</b>	<b>iii</b>
<b>ABSTRAK</b>	<b>iv</b>
<b>TABLE OF CONTENT</b>	<b>v</b>
<b>LIST OF TABLES</b>	<b>viii</b>
<b>LIST OF FIGURES</b>	<b>ix</b>
<b>LIST OF SYMBOLS</b>	<b>xi</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xii</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Background of Project	1
1.2 Problem Statement	3
1.3 Objective of The Study	3
1.4 Scope of The Study	4
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>5</b>
2.1 Introduction	5
2.2 Path Planning in Robotics	5
2.3 Obstacle Avoidance	6
2.3.1 Ultrasonic Sensor as an Obstacle Detector	7
2.3.2 Single Camera as an Obstacle Detector	9

2.4	Path Optimisation	10
2.5	Existing Algorithm	11
	2.5.1 Dijkstra's Algorithm	11
	2.5.2 Basic Theta Algorithm	12
2.6	Application of Path Finding Algorithms	13
	2.6.1 Path Finding Algorithms in Games and Virtual Tours	13
	2.6.2 Driverless Vehicles	14
	2.6.3 Path Finding Algorithms for Robot Motion and Navigation	16
<b>CHAPTER 3 METHODOLOGY</b>		<b>17</b>
3.1	Overall	17
	3.1.1 Hardware Development	19
	3.1.2 Software Requirement	21
3.2	System Flow Chart	22
3.3	Building the Initial Map	23
	3.3.1 Identifying the Obstacles on The Map	24
3.4	Astar (A*) Algorithm Working Principle	25
	3.4.1 Step 1: Identifying The Availability of Source and Goal Point on The Map	27
	3.4.2 Step 2: Identifying Workable Adjacent Squares from Neighbouring Cells	28
	3.4.3 Step 3: Choosing The Successor form The Open List Array	29
	3.4.4 Step 4: Constructing Path with Smallest Path Score	29
	3.4.5 Summary of Astar (A*) Algorithm Working Process	30
3.5	Graphical User Interface (GUI)	32
3.6	Conclusion	33

<b>CHAPTER 4 RESULTS AND DISCUSSION</b>	<b>34</b>
4.1 Introduction	34
4.2 Shortest Path Scoring	35
4.3 Obstacle Avoidance	38
4.4 Effect of Complexity of The Maps with Computation Time.	41
4.5 Effect of Light Intensity Towards Astar (A*) Algorithm Behaviour	44
4.6 Effect of Colour Towards Astar (A*) Algorithm Behaviour	47
4.7 Effect of The Goal Position Inside The Obstacle Area	51
<b>CHAPTER 5 CONCLUSION</b>	<b>53</b>
5.1 Introduction	53
5.2 Conclusion	53
5.3 Recommendations	54
<b>REFERENCES</b>	<b>56</b>
<b>APPENDIX A PROJECT GANTT CHART</b>	<b>59</b>
<b>APPENDIX B CODING OF A* ALGORITHM</b>	<b>60</b>

## LIST OF TABLES

Table 3.1: List of item used for the prototype.	19
Table 4.1: List of tests that will be conducted for Astar (A*) algorithm.	34
Table 4.2: Different cases of map with different types of obstacles.	35
Table 4.3: Results of different path scored constructed by the algorithm.	37
Table 4.4: Result of Astar (A*) algorithm's path scored.	38
Table 4.5: Different cases with different source and goal position.	39
Table 4.6: Result of path generated by Astar (A*) algorithm with obstacles avoidance ability.	40
Table 4.7: THREE (3) different cases with different level of map complexity.	41
Table 4.8: Path generated by Astar (A*) algorithm for different level of map complexity.	42
Table 4.9: Result of processing time of Astar (A*) algorithm respected to its complexity of the map.	43
Table 4.10: Result of effect of light intensity to the Astar (A*) algorithm behaviour.	45
Table 4.11: Result of effect of colour to the Astar (A*) algorithm behaviour.	49

## LIST OF FIGURES

Figure 2.1: (a) wave travel in short distance; (b) wave travel in longer distance	8
Figure 2.2: The resulting view of active area of transmitter.	8
Figure 2.3: Searching mechanism of Dijkstra's algorithm	11
Figure 2.4: Searching mechanism of Basic Theta algorithm	12
Figure 2.5: View of 3D path generated in the research by Shafie and Hassan, 2004.	14
Figure 2.6: Framed-Quadtree approach in D* Algorithm.	15
Figure 2.7: Autonomous vehicle used in the D* algorithm trial.	15
Figure 2.8: The vehicle that become champion in DARPA championship in 2005.	16
Figure 3.1: Overall project management flow chart.	18
Figure 3.2: Frame for the prototype of functioning area for the map.	19
Figure 3.3: Final view of the prototype.	20
Figure 3.4: Type of camera used to capture the map.	20
Figure 3.5: The random objects used to create obstacles	21
Figure 3.6: The flow chart of the system.	22
Figure 3.7: (a) The original image capture from camera; (b) The converted black and white image with obstacles spotted.	23
Figure 3.8: (a) Original image with presence of obstacles; (b) Map processing to identify the obstacle area.	24
Figure 3.9: Astar (A*) algorithm flowchart.	26
Figure 3.10: Example of a valid source and goal point	27
Figure 3.11: Pop-up dialog box	27
Figure 3.12: Matrix of a possible connection of the algorithm movement.	28
Figure 3.13: The possible movement of the Astar (A*) searching algorithm	28
Figure 3.14: Pseudocode of Astar (A*) searching algorithm.	29
Figure 3.15: Example of the path generated by Astar (A*) algorithm.	29

Figure 3.16: Development of Graphical User Interface for Astar (A*) algorithm.	32
Figure 3.17: Final look of the Graphical User Interface of Astar (A*) algorithm.	32
Figure 4.1: Possible direction of path score.	36
Figure 4.2: Original image recorded by camera.	38
Figure 4.3: Converted black and white map with a binary value. 1: obstacle-free area; 0: obstacle area.	39
Figure 4.4: Position of the C170 Logitech web camera used in this project.	44
Figure 4.5: Path generated that penetrate through obstacles.	46
Figure 4.6: Dialog box that popped-up to notify an error.	46
Figure 4.7: Image recorded in case 1.	47
Figure 4.8: Image recorded in case 2.	48
Figure 4.9: Image recorded in case 3.	48
Figure 4.10: Goal point that located inside the obstacle area.	51
Figure 4.11: Black and white map shows goal point that located inside the obstacle area.	51
Figure 4.12: Astar (A*) algorithm try to search for the way to reach the goal point.	52



## LIST OF SYMBOLS

$A^*$	<i>Astar Algorithm</i>
$S$	<i>Source point</i>
$G$	<i>Goal point</i>
$F$	<i>Total cost of the path between source and goal point</i>
$g$	<i>Cost of the path from the start to current point node</i>
$h$	<i>Heuristic cost (estimated smaller cost from the current point to goal point)</i>
$v$	<i>Point/node on the graph</i>
$^{\circ}$	<i>Degree</i>
$D^*$	<i>Dynamic Astar Algorithm</i>
$cm$	<i>centimetre</i>
$.bmp$	<i>Bitmap</i>
$.fig$	<i>Figure</i>
$s$	<i>Second</i>
$\approx$	<i>Approximately</i>

## LIST OF ABBREVIATIONS

<i>MIROS</i>	<i>Malaysian Institute Road Safety Research</i>
<i>TOF</i>	<i>Time of Flight</i>
<i>3D</i>	<i>3-Dimension</i>
<i>DARPA</i>	<i>Defense Advance Research Projects Agency</i>
<i>CBAPPA</i>	<i>Cognitive Based Adaptive Path Planning Algorithms</i>
<i>GUI</i>	<i>Graphical User Interface</i>

# CHAPTER 1

## INTRODUCTION

### 1.1 Background of Project

Automation is the technique, method, or system of operating or controlling a process by electronic devices, which reducing human intervention. In Robotics, the autonomous robotic referring to a technology (vehicle, machinery, navigation and etc.) that can functions independently, without help of human. Focusing in navigation area, autonomous navigation refers to the ability of the vehicle to be navigated from its starting point to the directed point accurately and safely without the needs of human driver. In this aspects, accuracy, reliability of automatic navigation and coverage are issues that seriously being centralize.

Studies carried out by the Malaysian Institute Road Safety Research (MIROS) claimed that 80.6% of accidents were attributed to human negligence while driving (Chelvi, 2016) . In the study, it said reckless driving, speeding, inattentive and driving under the influence of alcohol or driving when feeling tired are the main cause for accidents. With awareness of this situation, autonomous navigation is seen as an initiative to minimize the human intervention in navigation, in order to reduce human negligence while driving. This is because, an autonomous technology is something that believes can increase the safety and efficiency not only the driving also the traffics.

In order to provide a secure and effective navigation system, path planning is a technology that highly use in autonomous navigation. According to the definition of path planning by (The RoboRealm , 2006) path planning can be define as a module that has been use to determine a route from one coordinate location to another coordinate location. Path planning is something that requires the automation of mechanical systems that have sensors, actuators and computation capabilities. Generally, path planning is a method that

widely used in many fields, such as artificial intelligence, control theory, and also robotics. In artificial intelligence field, path planning is defined as “a search for a sequence of logical actions that transform an initial robot state onto a desire goal state” (Qidan Zhu, Yongjie Yan, and Zhuoyi Xing , 2006). In the control theory field, path planning deals with many issues such as stability, feedback, and optimality (A. Smith, Ding, Ulusoy, 2012). On top of that, in robotics, path planning playing a major role on how to move a robot vehicle from one point to another point. Path planning is an important primitive for autonomous navigation of the mobile robot which requires the robots to finds the shortest and easiest way (avoiding all the obstacles) between the two points (source and goal).

In order to plan the pathway for navigation, path planning requires the use of map of the covering area in order to search the shortest way to the goals avoiding the obstacles along the way. The most common map is occupancy grid map which the environment is discretized into squares of arbitrary resolution on which obstacles are marked (Correll, 2011). This type of map is widely used in Astar (A\*) algorithm’s application.

Astar (A\*) algorithm is one of the most famous algorithms in path planning. It is widely used in path planning of robotics navigation due to its ability to adopt the distance used, altered or add another distance (Andrej Babinec, 2014). This resulting in wide range of alternating the distance as it is the basic principle of this Astar (A\*) algorithm. This Astar (A\*) algorithm operates by the combination of heuristic searching, and scanning based on the shortest direction. The Astar (A\*) algorithm is define as follows:

$$F(v) = g(v) + h(v) \tag{1.1}$$

Where  $v$  = node on the graph  
 $F$  = total cost of the path between the source point to goal point  
 $g$  = cost of the path from the start node to current point node  $v$   
 $h$  = heuristic cost (estimated smaller cost from the current point to goal point)

## **1.2 Problem Statement**

In autonomous navigation that requires the robot to be operate safely and efficiently, the common issue that the developer face is it failed to avoid the obstacle with the minimum pathway. In some cases, the robot with an obstacle avoidance technology is able to avoid the obstacle but the question that remain in developer's mind is the path founded by the robot is the shortest? This is important to take into account the path that being chosen by the robot is the shortest distance.

As mentioned by Vo Thi Huyen Trang and team, from Astrakhan State Technical University, Russia, they state that, the problem is cause by "operating environment that can be static (fixed obstacles and know in advance) or dynamic (obstacles may not be fixed in advance or do not know)" (Vo Thi Huyen Trang, Tran Quoc Toan, A.A. Sorokin, 2017).

As a step to improve people way of live, this project will design and implement an Astar (A\*) algorithm for autonomous navigation system.

## **1.3 Objective of The Study**

The general purpose of this project is to develop an algorithm which is able to provide the shortest path with ability to avoid any obstacles that presence along its way from any two point (source and goal) of the map. The project specifically aims to implement the set of rules of path finding for the autonomous navigation, which can be implemented into any comparable navigation system. In this regard the specific objectives of this project will include:

- i. To implement the Astar (A\*) algorithm for path planning.
- ii. To analyse the Astar (A\*) algorithm output using simulation and experiment.
- iii. To verify the capability of the Astar (A\*) algorithm for path planning navigation.

## **1.4 Scope of The Study**

This project Astar (A\*) algorithm implementation for robotic path planning, aims to implement a path finding algorithm that able to provide an optimised pathway between two point that can detect and avoid the presence of obstacles along its way. The system is provided with a real time image which act as a map that provide a searching space for the system. Therefore, a small prototype with one single camera will be used in this project. Random object will be used as to provide obstacle. MATLAB Software will be used for the development and simulation of the Astar (A\*) algorithm. This software will also use as an image processing to convert the image taken by the camera into a black and white image in order to provide the map for the pathfinder. The path generated by the simulation of MATLAB software will be the result this project of Astar (A\*) algorithm.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Introduction**

Before designing the application and the system environment, this chapter gives insight about the technique used in this project. Furthermore, general uses of the Astar (A\*) algorithm that related to this project will be specified through the applications of this algorithm.

#### **2.2 Path Planning in Robotics**

Back to late 1940s, since then research in mobile robotics can be traced. However, most of the effort related to path planning is more recent and it has been conducted during the 1980s. In robotics, path planning is something that plays an important role in autonomous robotic navigation. Planning itself refers to a preconceived scheme or method of acting or proceeding. In other words, path planning in robotics can be defines as an operative intelligence which include the set of operations that need to be consider for its navigation. In an algorithm, path planning is done with respect to the functionality of a mobile robot or an autonomous vehicle. It is very useful to design or scheme its routing.

A robot is any machine that featuring a human and mechanical routine tasks automatically upon a command. Since it is a machine, its functionality depends completely upon the set of instructions given to it. These set of instructions programmed for any particular robot, mobile or immobile, will defines its intelligence. In other words, since the robot is a machine, it need set of instruction to tell it what should be done. This will customize it for any specific functionality that the robot should behave.

As the field of mobile robotics expanding, it gives the huge impact to the scope of path planning. This is due to the most of the mobile robots are customized for specific operation. For instance, an industrial robot is generally customized as demanded by the industry which specific to their functionality. Rovers which to be used for exploration on other planets are customized for data collection related jobs. Robots used in medical field are customized for the specific surgery or any similar medical operation (Ghangrekar, 2009). Hence all such robots can be classified to have two parts of development. One part deals with the overall development of the robot, while the other deals with structuring the development as per the required customization of its functionality also its navigation.

Therefore, development of the path planning algorithm in navigation is highly acknowledged as a significant tool to assist towards its desired functions. It creates details need by the robots. This path planning algorithm will determine the logic or scheme need for navigation of the robots.

### **2.3 Obstacle Avoidance**

Obstacle avoidance is a primary requirement of any autonomous mobile robot. An obstacle avoidance robot is design to allow robot to navigate in unknown environment with ability of avoiding collisions. There is a rich literature on the autonomous navigation of a mobile robot with obstacle sensing capability. Basically, a robot with this obstacle avoidance ability will sense the presence of any obstacles in their path, avoid it and resumes its running. All mobile robots feature with some kind of collision avoidance, ranging from primitive algorithms that detect an obstacle and stop the robot in order to avoid a collision. With some sophisticated algorithms, it enables the robot to detour obstacles. There are several methods had been developed in autonomous robot vehicle navigation with collision free function (e.g. wall following method, line following method, edge detection method).

According to research conduct by Kirti Bhagat and her team, it states that a more general and commonly used method for obstacle avoidance is based on edge detection. “A disadvantage with obstacle avoidance based on edge detecting is the need of the robot to stop in front of an obstacle in order to provide a more accurate measurement” (Kirti Bhagat, Sayalee Deshmukh, Shraddha Dhonde, Sneha Ghag, 2016).



Additionally, there are various sensors that can be usefully employed for a mobile robot. These is including:

- Ultrasonic sensor
- Camera as vision sensor

### **2.3.1 Ultrasonic Sensor as an Obstacle Detector**

In order to provide a collision free pathway, the robot vehicle require a wide range of sensors to obtain information about its working area. These sensors will determine the position, velocity, acceleration and behaviour of the object that present at the robot workspace. There are several different sensors used in an autonomous robot vehicle. These sensors will be resulting in various functions of the robot. One of the most common sensor used in rangefinders is the ultrasonic transducer (Polaroid Corporation, 1992). This sensor is widely used in many experiments that requires the vehicle to navigate automatically. It used as a primary means of detecting the boundaries within which the vehicle must operate. An ultrasonic range finder can be built in a low cost but suffers from low angular resolution. It may fail to identify a narrow open space like a doorway if its distance from the sensor is not close.

The working principle of an ultrasonic sensor which will transmits sounds waves and receives sound that reflected from an object. When the ultrasonic waves are incident on an object, it then will diffuse reflection of the energy over a wide solid angle which might be as high as  $180^\circ$  degrees. Thus some fraction of the incident energy is reflected back to the transducer in the form of echoes. The distance between the object and the sensor plays a significant role when using an ultrasonic sensor. This is due to the relationship between the distance of the sensor with the object and its time taken for the reflected wave to be received back by the receiver. If the object is very close to the sensor, the sound waves returns quickly, but if the object is far away from the sensor, the sound waves takes longer to return. As shown in Figure 2.1 the situation in (a) resulting the short time taken for the reflected wave to reach back to the sender compare to the situation in (b).

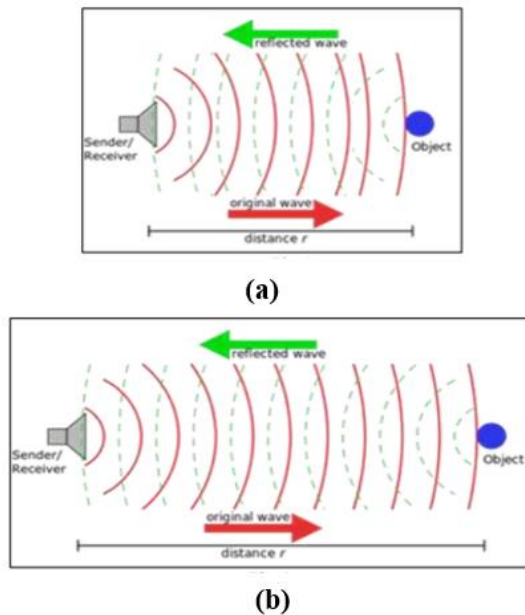


Figure 2.1: (a) wave travel in short distance; (b) wave travel in longer distance.

Source : W.H.Munro. (1990). Ultrasonic Vehicle Guidance Transducers.

W.H. Munro, in his experiment had developed a vehicle guidance system using this type of sensor. In his experiment, the unit used consisting of separate array of transducers and resulting  $60^\circ$  degrees in view of active area of transmitter, as shown in Figure 2.2 (W.H.Munro, 1990).

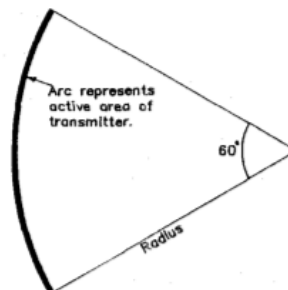


Figure 2.2: The resulting view of active area of transmitter.

Source : W.H.Munro. (1990). Ultrasonic Vehicle Guidance Transducers

The study to investigate the use of ultrasonic sensor as an obstacle detector has been carried out by Intorobotics which then discovering several weaknesses of using ultrasonic as a sensor to detect obstacles.

- Ultrasonic sensors must view a high density surface for good results. A soft surface like foam and cloth has low density and absorb the sound waves emitted by the sensor.
- Could have false responds for some loud noises such as air hoses.

- An ultrasonic sensor has a minimum sensing distance.
- Some changes in the environment can affect the response of the sensor (temperature, humidity, pressure, etc.).

(Intorobotic, 2013)

### **2.3.2 Single Camera as an Obstacle Detector**

Recently the introduction of time-of-flight (TOF) cameras have many advantages for the autonomous navigation of a mobile robot. TOF camera provides 3D information with a real-time frame rate and is much faster (Werner B, Surmann H, Pervolz K, 2006). However, camera calibration and data pre-processing are necessary to get stable measurement. In addition, TOF cameras are still expensive compared to other high tech sensors like laser scanners. Stereoscopic 3D sensing has been a long time topic for perceiving the space around a mobile robot (Chilian A, Hirschmuller H., 2009). This Stereoscopic 3D sensing can be said as a natural way of sensing because most animals and humans obtain information about their surroundings using two eyes. However, this stereoscopy has a difficulty in providing reliable 3D information quickly due mainly to stereo matching problem (Chilian A, Hirschmuller H., 2009).

In this era, the rapid advance in solid state electronics, cameras become smaller and cheaper, and many mobile robots now possess visual sensing capability. Vision sensing has certainly high potential for a robot to perceive its surroundings. When a single camera is mounted on a mobile robot for the autonomous navigation, it is often used for localization by detecting landmarks rather than obstacle detection because detecting obstacles usually requires at least two cameras to obtain depth information. Nevertheless, there are some attempts to find obstacles using a single camera. For example, in a research done by Ulrich and Nourbakhsh, they have tried to detect obstacles based on colour cues. Image pixels that have different colour values to those trained with an empty trapezoidal area in front of the mobile robot are considered to belong to obstacles. This method is simple and can produce a high-resolution binary detection image in real time. However, the method is found quite sensitive to the colours of obstacles (Ulrich , Nourbakhsh, 2000).

When using camera as a vision sensing, there are some aspect need to be consider which is like the position of the camera. This is because, the camera can be fixed position on the wall, or the camera can be mounted on a mobile robot. In research done by Konolige K, in their finding, they found that if a camera is in fixed position, there are several effective and simple techniques to find moving objects in the scene, such as background subtraction. However, if a camera is mounted on a mobile robot, such techniques cannot be used because, in the scene images taken by a moving camera, no entities look static ( Konolige K, 1997).

## **2.4 Path Optimisation**

Path planning is an important primitive for autonomous robots that lets robots find the shortest or optimal path between two points. Otherwise optimal paths could be paths that minimize the amount of turning, the amount of braking or whatever a specific application requires. Known as an intelligent and smart developing technology, path planning appears to have huge attention of researcher in order to make it more reliable and flexible to meet any specification of robotic industry. Thus, many algorithms are being develop to meet this requirement. In path planning algorithm, there are generally two different kinds of planning concerns based on the types of criterion (LaValle, 2006).

- Feasibility – find a plan that causes arrival of the robot from the source point to its goal point.
- Optimality – find a path that optimized performance in some specified manner.

In the study carried out by Martin Florek and his team, he had defined some optimal criterions to compare the performance of several types of algorithms. These criterions are including:

- Computational time
- Path length
- Number of examined cells
- Symmetry of examined environment.

(Martin Florek, Andrej Babinec, Martin Kajan, 2014)

## 2.5 Existing Algorithm

Due to the rapid growth of artificial intelligence industries, path planning appears to have huge attention of researchers. There are large number of researches has been done to make this high technology to be more reliable, more intelligent and become more flexible to fulfil any requirements of the market. Thus, many type of algorithms has been design that give different outcome and acted as a reference to produce a more intelligence algorithm.

### 2.5.1 Dijkstra's Algorithm

Dijkstra's algorithm is one of the earliest and simplest algorithms. It is named after its developer, E. Dijkstra Oliver J. back in 1959. Dijkstra's Algorithm is a chart search algorithm that unravels the single-source shortest path delinquent for a plot with non-negative edge path costs, producing a shortest path tree.

Previously, Dijkstra's algorithm is the most commonly used route finding method for solving the shortest path (Sadeghi-Niaraki, A., Varshosaz, M., Kim, K., and Jung, J, 2011). The Dijkstra's algorithm till enlarges the node that is at the extreme from the initial node, so it finishes up "stumbling" into the goal node. Just like the breadth-first search, it is certain to find the shortest path (Cormen, T., Leiserson, C., Rivest, R., and Stein, C, 2001). Figure 2.3 shows the example of the searching mechanism of Dijkstra's algorithm.

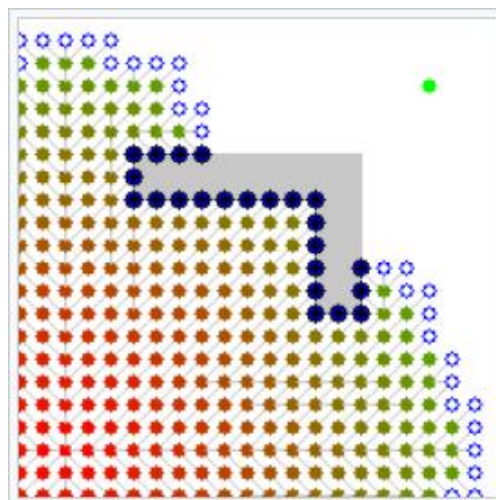


Figure 2.3: Searching mechanism of Dijkstra's algorithm.  
Source: [https://en.wikipedia.org/wiki/Dijkstra%27s\\_algorithm](https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

The Figure 2.3 shows the illustration on how the Dijkstra algorithm is working. This algorithm will enlarge its searching mechanism to the nodes that were extremely far from its initial point until its it reached to the goal nodes. This algorithm may cause in extremely large number of examined nodes which then resulting in a very long computational time (Martin Florek, Andrej Babinec, Martin Kajan, 2014).

### 2.5.2 Basic Theta Algorithm

Basic Theta is an algorithm that has been published by Alex Nash, Kenny Daniel, Sven Koenig and Ariel Felner. This algorithm is basically an extension of Astar (A\*) algorithm. It resides in the test of visibility between cells. In other words, this algorithm will test if the cell has a direct visibility to the cell included in selected sequence, the cell between them will be ignored. Thus, this algorithm making only the cells that robot has to pass will be found.

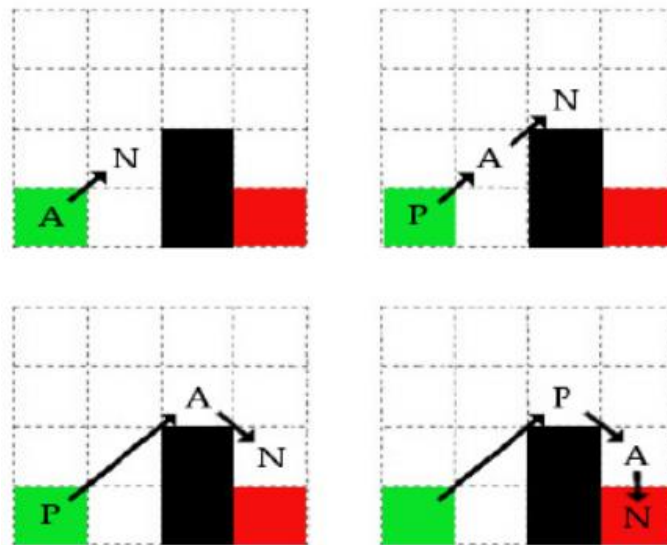


Figure 2.4: Searching mechanism of Basic Theta algorithm.

Source : [https://www.researchgate.net/figure/Basic-Theta-algorithm-Green-cell-is-a-initial-state-red-cell-is-a-goal-state-black\\_fig2\\_270163652](https://www.researchgate.net/figure/Basic-Theta-algorithm-Green-cell-is-a-initial-state-red-cell-is-a-goal-state-black_fig2_270163652)

In Figure 2.4 shows example of Basic Theta\* algorithm. Green cell is an initial point, red cell is a goal point, and the black cells represent the obstacles. In the second step, the cell N has direct visibility to the cell P. That is why the cell A is ignored and the cell P and N are directly connected (Martin Florek, Andrej Babinec, Martin Kajan, 2014).

However, this algorithm is found to have some issue with its computational time. This because, the time of calculation for this algorithm is appear to be long compare to the Astar (A\*) algorithm (Martin Florek, Andrej Babinec, Martin Kajan, 2014).

## **2.6 Application of Path Finding Algorithms**

Path planning or path finding algorithms are very useful in the area of robotic manipulation, as it can be used to control a robot around complicated terrain without the need of human intervention (Carsten, 2007). It is indeed a profitable if a robot on a different planet like Mars, that supposedly should be avoided some topography, but due to the extremely further distance that involved, it makes difficult to guide it through remote control because of too much delay in the radio transmission (Obara T., Yamamoto K., Ura T., Maeda H., Yamato H, 1994).

Besides that, it is also might be useful if the robot is being able to operated underwater by itself, since the radio waves could not get to it. The path finding algorithm might also be used in determining the shortest path to drive between the two positions on a map.

### **2.6.1 Path Finding Algorithms in Games and Virtual Tours**

Path finding is an important part of game programming. In gaming programme, it moves according to the path they (or computer) calculate. There are some algorithms used in the game changing with the complexity and purpose of the path calculation. The most algorithm found currently in games today are the A\* Algorithm. “What makes the A\* algorithm so appealing is that it is guaranteed to find the best path between any initial point and any ending point, assuming that a path exists.” (David M. Bourg, Seeman G., 2004). Users can arrange their visit path and algorithm calculates the route then the virtual tour begins. In the research one by Shafie and Hassan (2004) they had work on this scenario and developed an application capable of doing path planning process. They used an A\* algorithm to find the path (M. Shafie Abd. Latiff, and R. Hassan, 2004). The final path can be seen in a 3D environment in Figure 2.5.

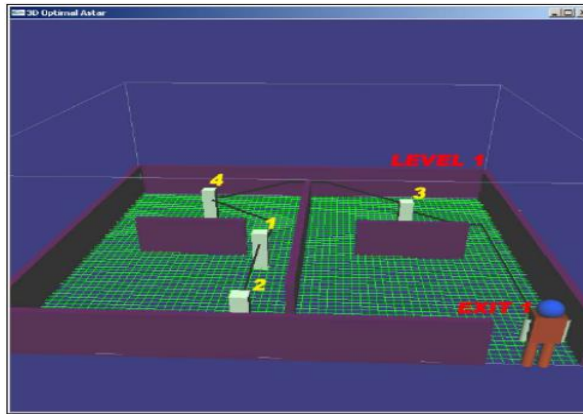


Figure 2.5: View of 3D path generated in the research by Shafie and Hassan, 2004.

Besides that, autonomous triangulation of virtual characters, is also an important task to work on. It is required in order to prevent obstacles and proceed to their paths without any destruction. This is because in a virtual environment not like the robotic navigation, there is no data coming from sensors, there exist the site database only. In the study done by Fröhlich and Kullmann in 2002, they have used Astar (A\*) algorithm to studies on the environmental modelling of the area and resolved that the uniform-sized grid cell methodology is a better choice (T. Frohlich and D. Kullmann., 2002). They also stated that A\* algorithm works well in the environments where the world is flat, which suits our environment well.

### 2.6.2 Driverless Vehicles

Another area of application that involving the path finding algorithms is the programmable vehicles that capable to discover its own way without making any contact with the obstructions. Generally, this driverless vehicles technology is widely used in military industry, but in this modern era, it can be implemented in diverse areas of domestic life.

In year of 2000, Yahja A and his friends has introduced the D\* algorithm which stands for Dynamic Astar (A\*) which in this algorithm, they had use framed-quadtrees to constructing their environment and as well as data structures. Figure 2.6 shows the Framed-quadtrees approach used in their study. They have verified their designed algorithm as a self-directed vehicle as shown Figure 2.7 (Yahja A., Singh S., Stentz A.,



2000). This algorithm continuously transform itself when it gets fresh information about the terrain.

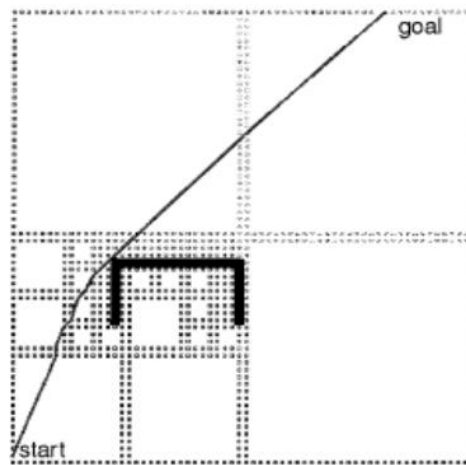


Figure 2.6: Framed-Quadtree approach in D\* Algorithm.



Figure 2.7: Autonomous vehicle used in the D\* algorithm trial.

In United States, there was a race organised by The Defense Advance Research Projects Agency (DARPA) that the contestants were the autonomous ground vehicles came from around the globe. The race was held twice until 2007; one in 2004 where none of the participants were capable to complete the course and the second one was in 2005. During the race in 2005, 4 vehicles completed the 132-mile desert track, and the winner is a VWTouareg which was developed by Stanford Racing Team. This car has a processing system to calculate its route while on the road. On board, computers control the vehicle from start to finish, and there was not any intervention from the race team. The algorithms constantly modified the path according to the information it gets from its

sensors while it moves on the route. Figure 2.8 shown the VWTouareg that become the winner on the race back in 2005.



Figure 2.8: The vehicle that become champion in DARPA championship in 2005.

### 2.6.3 Path Finding Algorithms for Robot Motion and Navigation

Technically, all mobile robot that designed to move in outdoor or indoor environment must have their own programmed and navigational schemes. This is to make sure that these robots are able to find their own pathway towards its desired destination. This scenario makes the path finding algorithm to be inserted on the robot program so that they can move by themselves. This type of technology is an important feature that a robot should have which will make the robot become more reliable with its intelligence. For example, this technology is widely used in military industry, which had been used as a movable robot to detect the hazardous or explosive materials and to discover or investigate the unknown areas.

In 2005, Razavian and Sun has proposed a new algorithm that called Cognitive Based Adaptive Path Planning Algorithms (CBAPPA) and comparing it with Astar (A\*) algorithm. They used it to observe the behaviour of biological units and paid attention on the behaviour of ignoring the irrelevant information from surroundings. This method may not use optimum paths, but the results were efficient (Adam A. Razavian, Sun J, 2005). This algorithm is also prepared for self-processing units, which can be a good example for future works.

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 Overall**

This final year project used three major steps to implement the project starting from planning, developing and testing. The Figure 3.1 shows steps of methodology taken to implement this project.

It starts with planning; defining and understanding the project background and objectives of this project. Then, the summarization from reading journal, book and report paper regarding to project title is done in literature review chapter.

The process is then being continue with design and developing; the development of prototype of the functioning area, which include hardware and software development.

Last but not least, is analysing process, this include testing and analysing the outcome of the project to identify the conclusion. The methodology in this project is implemented step by step in order to achieve its objectives. The Gantt Chart of this project can be referred in Appendix A.

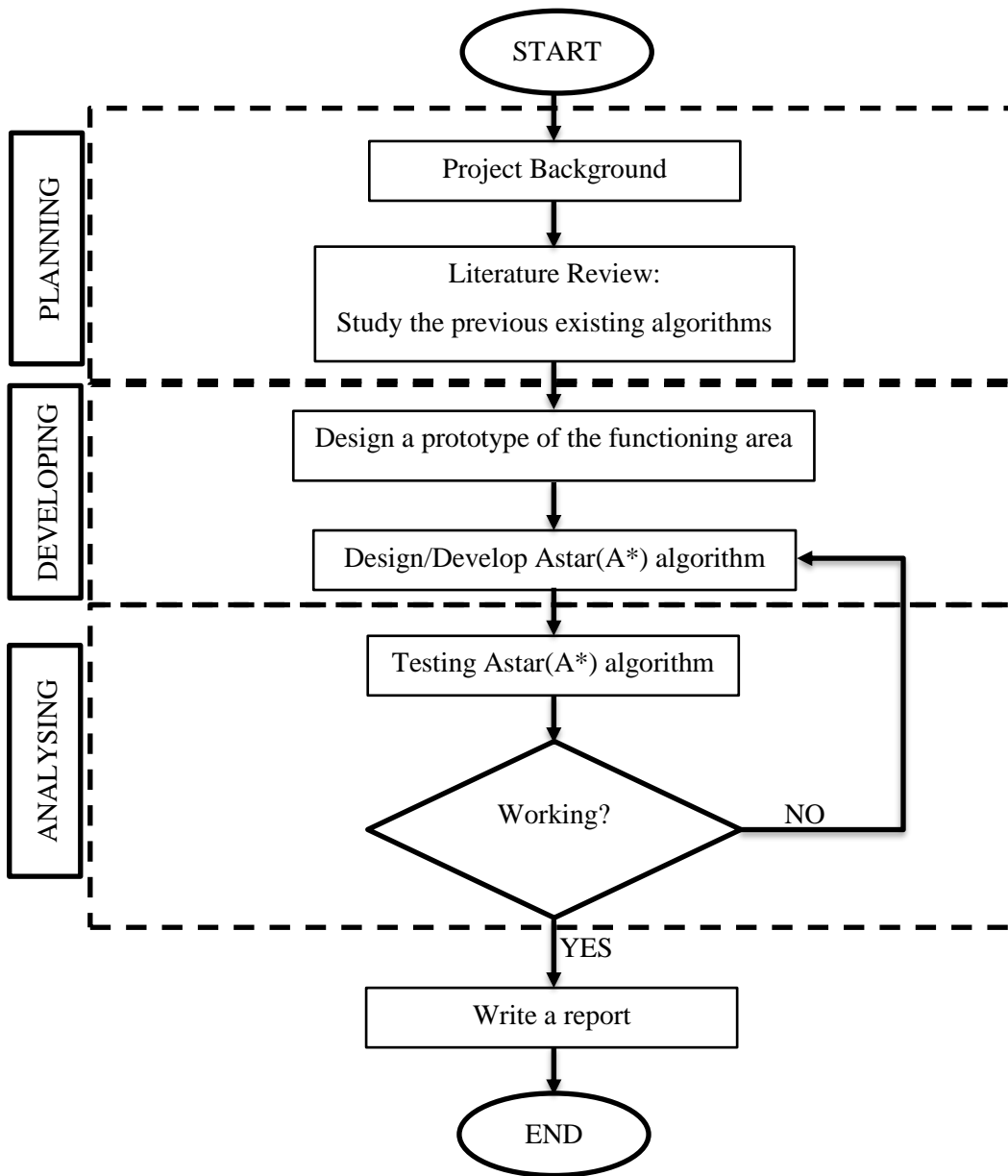


Figure 3.1: Overall project management flow chart.

### 3.1.1 Hardware Development

In order to test the efficiency of this Astar (A\*) algorithm with the real environment, a prototype of the testing area is made. This is to provide a real time situation for the algorithm. Thus, a set of hardware is being developed. Below are the components used for the prototype.




#### 1. Frame

The prototype of the testing area is consisting of Hollow Aluminium profile and Perspex. Figure 3.2 shows the frame of the prototype. The dimension of the frame is ( $h \times l \times w$ :  $30.5cm \times 36cm \times 36cm$ )



Figure 3.2: Frame for the prototype of functioning area for the map.

Table 3.1: List of item used for the prototype.

Item	Quantity	Description
 Al Profile	(12 pcs)	Dimension: $4pcs \times 30.5cm$ $8pcs \times 32cm$
 Perspex	4 pcs	Dimension: $l \times w \times t =$ $37.5cm \times 37.5cm \times 0.3cm$
 Wooden Stick	4pcs	Dimension: $h \times w \times l =$ $1.5cm \times 1.5cm \times 25cm$

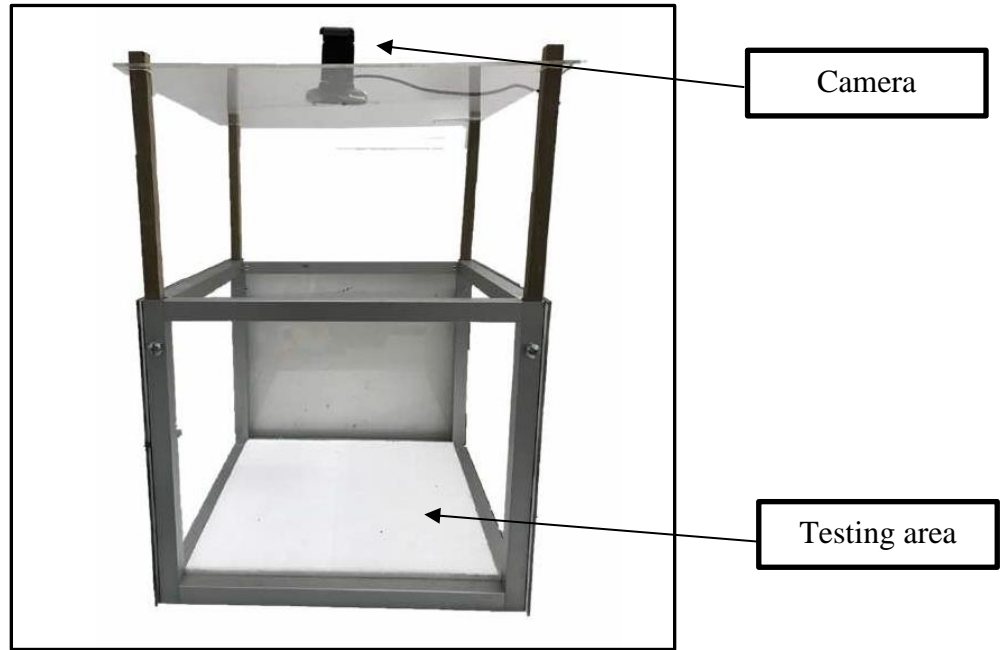


Figure 3.3: Final view of the prototype.

## 2. Camera



Brand	LOGITECH
Model	WEBCAM C170
Resolution	Up to 1024 x 768 pixels
Part No.	PN: 960-000761
Price	RM 80.00
Quantity	1 pcs

Figure 3.4: Type of camera used to capture the map.

### 3. Random Object

In order to provide the obstacle, this project has been using the random object. There are several types of item as shown in Figure 3.5 which has been used in this project such as mini toys, car, and other mini objects.

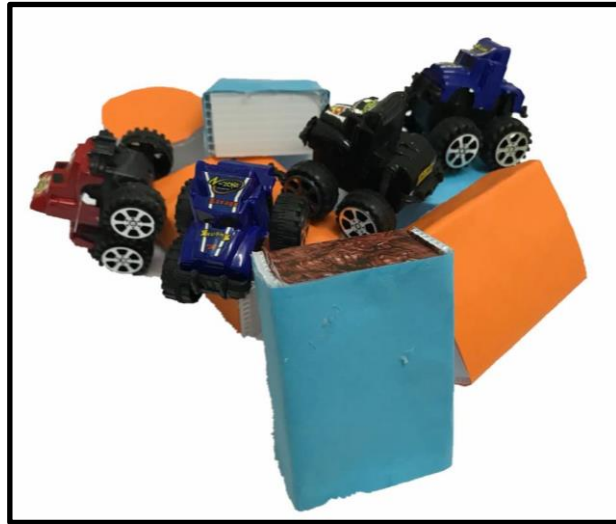


Figure 3.5: The random objects used to create obstacles.

#### 3.1.2 Software Requirement

For software requirement, this project is using MATLAB software. The version used is MATLAB R2017a version. This software is used mainly to design the algorithm. This software is used since it has a Graphical User Interface (GUI) Function which the author familiar with. This software is also used for testing purpose after the algorithm has been developed.

MATLAB software has been selected as simulator since it equipped with the image processing function. Basically, after the image has been captured by the camera, this MATLAB software will process the image and convert it into the black and white .bmp file. This is necessary since the image captured will be used as a map for the system to generate the shortest path with collision free pathway.

### 3.2 System Flow Chart

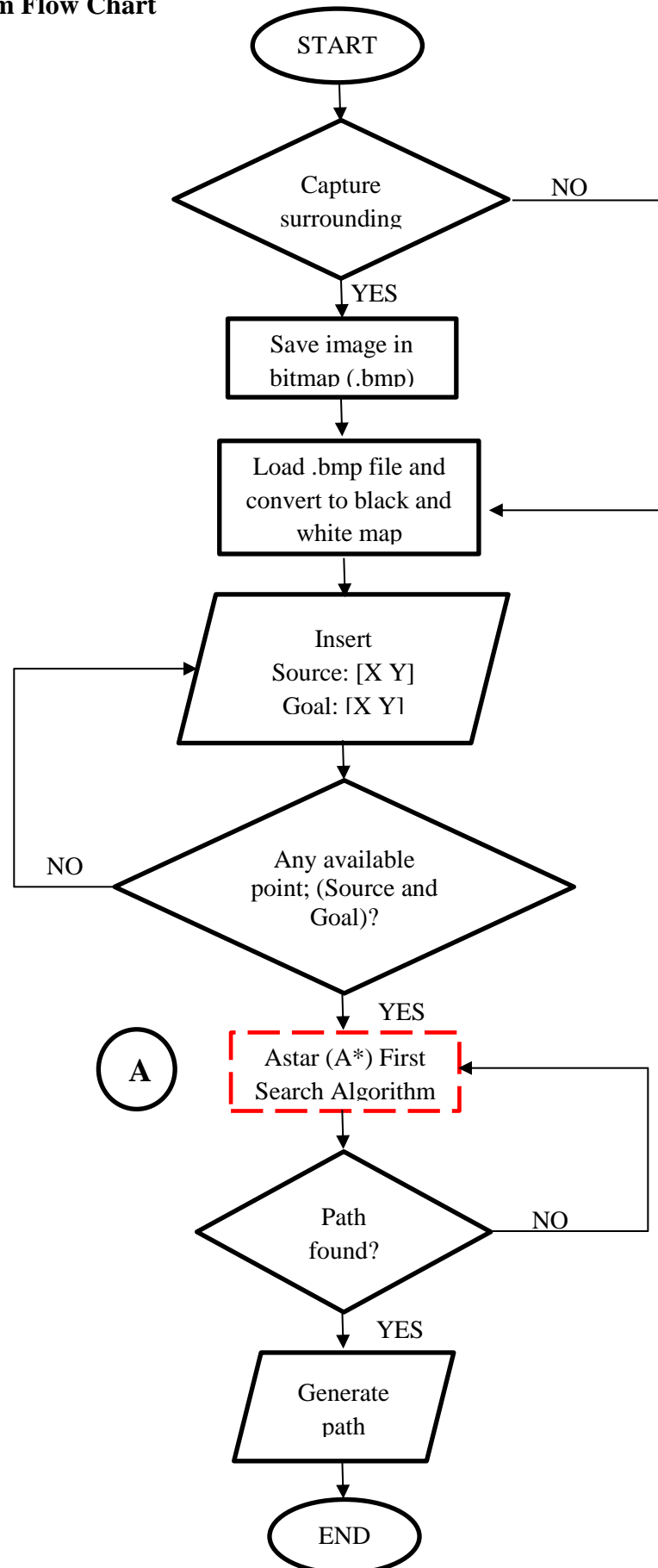


Figure 3.6: The flow chart of the system.



### 3.3 Building the Initial Map

In order to plan a path, it is sometimes necessary to represent the environment in the computer. According to Nikolaus Correll, in his research about path planning in robotics he states that, there are two different complementary approaches of map representation: discrete and continuous approximation. In discrete approximation, the map is sub-divided into block of equal (e.g. a grid map or hexagonal map) or differing sizes like rooms in a building (Correll, 2011).

For this project, the discrete type of map is used. By assuming that all the obstacles are lays on the ground in this Astar (A\*) algorithms, the image captured by the system will provide a top view of the covering area, then be stored in bitmap (.bmp) file. This grid type of image will then be converted into black and white map, which will then be used for the algorithm to provide an optimised path scoring with collision free pathway. Figure 3.7 (a) shows the original image captured from the camera and Figure 3.7 (b) shows the converted black white image file.

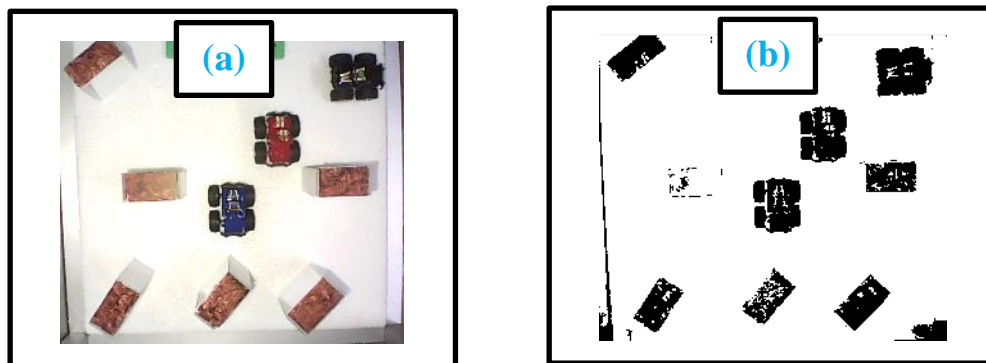


Figure 3.7: (a) The original image capture from camera; (b) The converted black and white image with obstacles spotted.

For this Astar (A\*) algorithm project, the bitmap (.bmp) type of image file is used in this project since it will provide the grid type of map for the system. In this grid map, the environment is discretised into squares of arbitrary resolution (240 X 320 pixels) map size.

### 3.3.1 Identifying the Obstacles on The Map

Obstacle detection is one of the fundamental problems for the navigation of mobile robots. In order to navigate in the real world, it is necessary to detect those portions of the world that are dangerous or impossible to traverse. First and foremost, in order to be able to provide a collision free pathway, the algorithm need to do a scanning process which is to detect the presence and locate the position of the obstacles on the map. Please be noted that in this project, the random objects were used to provide the obstacles. The camera provided will capture the top view of the environment. Then, this process continues with saving the image captured by the camera into bitmap file (.bmp). This step will provide the image in form of grain-form-cell respected to the resolution of saved image. This fine grain array of image file will be act as a map for the algorithm. In this algorithm, the image will be resized to 240 X 320 pixel resolution. Next, the map will be converted into black and white map. This black and white will give a binary value of one (1) or zero (0) to cells appear on the map. Value of one (1) indicate the freeway part of map, where the zero (0) value indicate part of the map that occupied with obstacles. This process can be seen as shown in Figure 3.8.

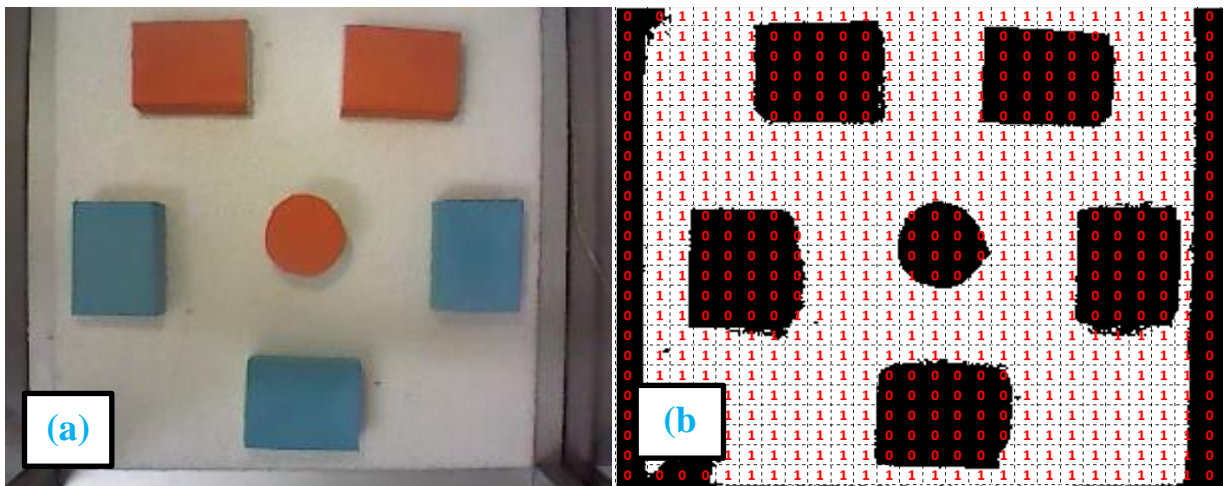


Figure 3.8: (a) Original image with presence of obstacles; (b) Map processing to identify the obstacle area.

From the above map shown in Figure 3.8 (b) , it clearly can be seen that the black spot is where the obstacles lies on the map. These black spot will bring the binary value of zero (0) to the cells on the map. Since the map resolution used for this Astar (A\*) algorithm is (240 X 320 pixels) of Resolution-Y and Resolution-X, that should be an

array of 76800 squares. The cells should be appear smaller than the one in Figure 3.8 (b). The above cells in Figure 3.8 (b) has been magnified so that it gives an array of 624 squares cells, for explanation purpose.

### 3.4 Astar (A\*) Algorithm Working Principle

Astar (A\*) algorithm is one of the most famous algorithms in path planning. It is widely used in path planning of robotics navigation due to its ability to adopt the distance used, altered or add another distance (Andrej Babinec, 2014). This resulting in wide range of alternating the distance, as it is the basic principle of this Astar (A\*) algorithm. Basically, this Astar (A\*) algorithm operates by the combination of heuristic searching, and scanning based on the shortest direction. Astar (A\*) algorithm can be defined by,

$$F(v) = g(v) + h(v) \quad 1.1$$

Where  $v$  = node on the graph  
 $F$  = total cost of the path between the source point to goal point  
 $g$  = cost of the path from the start node to current point node  $n$   
 $h$  = heuristic cost (estimated smaller cost from the current point to goal point)

This Astar (A\*) algorithm work through these two lists; the open list and the closed list. The open list is to hold the best small path score that have not yet been consider. The algorithm will then select the lowest path score (successor) in the open list and put it into the closed list. The node that has been put in closed list will not be reconsider by the algorithm. The heuristic function is the one that estimate the best possible node with smaller value to reach the goal. If the selected node (successor) is not yet reach the goal, it will update all the valid neighbouring nodes onto the open list and repeat the process. All the created nodes in closed list will keep a reference to their parent cell. This make it possible to backtrack back the node to its starting point. The flow chart of this Astar (A\*) algorithm is as shown in Figure 3.9 .

A

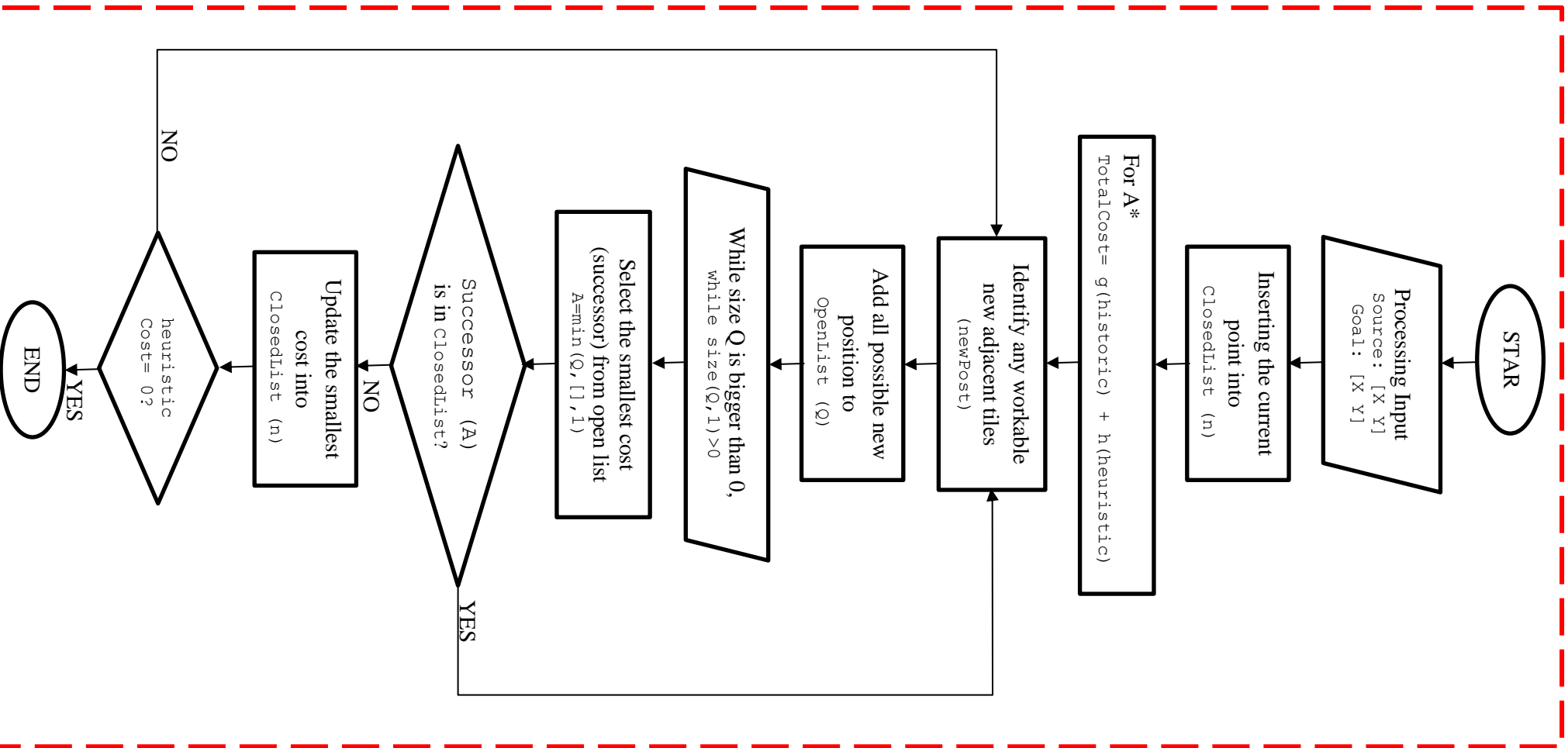


Figure 3.9: Astar (A\*) algorithm flowchart.

From the flow chart of Astar (A\*) algorithm shown in Figure 3.9, it shows the working operation for this algorithm. Since Astar (A\*) is the best first search algorithm, there are several steps need to be consider. The working step of this Astar (A\*) algorithm will be explain in the following section.

### 3.4.1 Step 1: Identifying The Availability of Source and Goal Point on The Map

After providing the map which is in black and white map with a fine-grain array structure, the next step is for the algorithm processing the inputs given by the user. The inputs are; source and goal point, which the source indicate the initial position where the robot lies, and the goal point is the final destination that the robot should be heading to. For these inputs (source and goal), the algorithm will be processing in term of the availability of these points on the map. It should identify whether the source point and the goal point are not located inside the area of obstacles, or they were not situated outside the map area.

If only the points are available on the map and they were not laying on the obstacle, the algorithm will proceed to process of generating the pathway towards the goal. Figure 3.10 shows the example of situation with a valid source and goal point. The (S) sign indicate the source point, and the (G) sign indicate the goal point.

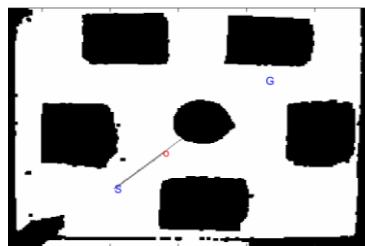


Figure 3.10: Example of a valid source and goal point.

It is different with the situation where the source or the goal point is located on the obstacle area, the system will give a pop-up dialog box to notify the user that the points are not valid to be process. Figure 3.11 shows the pop-up dialog box for this situation.

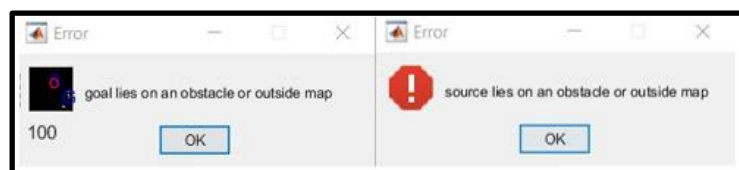


Figure 3.11: Pop-up dialog box.

### 3.4.2 Step 2: Identifying Workable Adjacent Squares from Neighbouring Cells

The next step after identifying the availability of the source and goal point, is the algorithm will then proceed with identifying any workable adjacent cell to the source/current position. Firstly, the source/current point will be inserted to the closed list so that the Astar (A\*) algorithm will not have to reconsider the starting point anymore. From this starting point, all the adjacent cells will be inserted into the open list. Since for this project, it is possible for the algorithm to move diagonally instead of just in up, down, left and right direction. The matrix in Figure 3.12 shows in the possible connection of the robot movement.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

(a) (b)

Figure 3.12: Matrix of a possible connection of the algorithm movement.

From the above possible movement connection of the robot in Figure 3.12, it shows that the value of 2 in the matrix indicate the robot position, value 1 indicate the possible movement of robot, and value 0 tells that the limitation of the robot movement. Thus, and for (a) matrix, it tells the robot that it is possible for the robot to move in diagonal direction, whereas for matrix (b) it's have the limitation towards the robot movement which it can just move in up, down, left and right direction. This can be seen in the Figure 3.13, where the robot in this algorithm can move in diagonal direction.

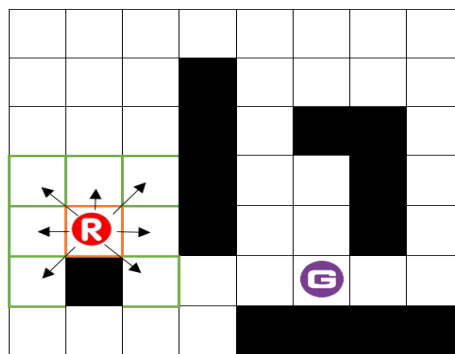


Figure 3.13: The possible movement of the Astar (A\*) searching algorithm.

From the Figure 3.13, the cells with orange border where the initial position of the robot will be inserted into closed list, then all the workable adjacent cell the one with green border will be update into open list array.

### 3.4.3 Step 3: Choosing The Successor form The Open List Array

After the algorithm verified the workable adjacent cells and update the cells into the open list, the heuristic cost (h) which with the smallest estimated score towards the goal will choose the successor from the open list. This successor point (A) will be inserted into closed list. The latest successor point will indicate the current position of the robot. The process will be repeated until it reaches the goal. Each time through the main loop, it examines the vertex (v) that has the lowest  $f(v) = g(v) + h(v)$ .

```
while open_list(Q) is not empty
  current (A) = open_list element with lowest f cost
  if current (A) = goal
    return construct_path(goal) // path found
  remove current from open_list
  add current to closed_list(n)
  for each neighbour in neighbours(current)
    if neighbour not in closed_list
      totalCost (f)=historicCost(g) + heuristicCost(h)
      if neighbour is not in open_list
        add neighbour to open_list
```

Figure 3.14: Pseudocode of Astar (A\*) searching algorithm.

This process as shown in pseudocode in Figure 3.14 will repeating until the heuristic cost (h) is equal to zero (0). This indicate that the algorithm has reach the goal point. Then, the algorithm will be constructing the pathway with the smallest path score.

### 3.4.4 Step 4: Constructing Path with Smallest Path Score

The last step after the algorithm done do the searching process, when the estimated heuristic cost (h) is equal to zero (0), the algorithm will then backtracking the path with the smallest path score. This backtracking process were done by referring to the parent successor node. This parent successor node is the one that exist in closed list. The result of this path constructed by the algorithm can be seen in the example in Figure 3.15.

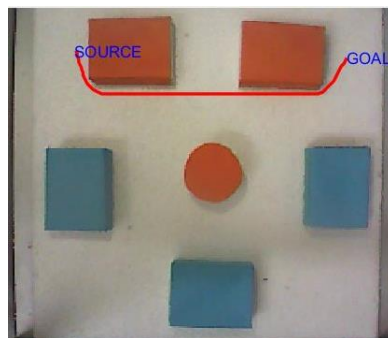
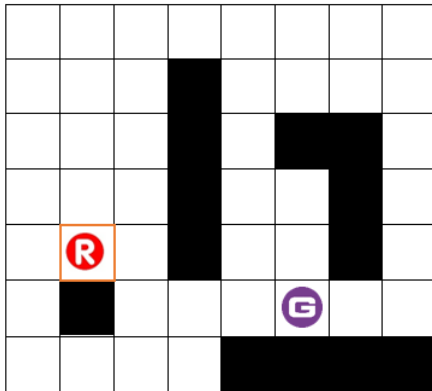


Figure 3.15: Example of the path generated by Astar (A\*) algorithm.

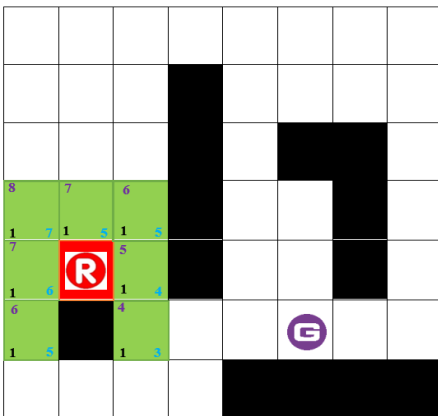
### 3.4.5 Summary of Astar (A\*) Algorithm Working Process

The quick overview on how this Astar (A\*) algorithm is functioning can be seen through the following example of the processes.



Set up the map for the Astar (A\*) algorithm to work on.  
Identify the position of obstacle on the map.

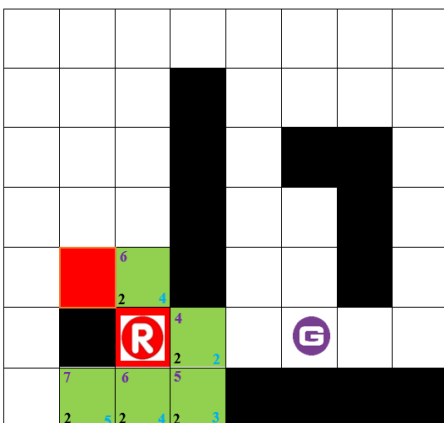
Define the initial position of the Astar (A\*) robot **R** and the goal destination of the robot should be heading to **G**.



Next, identify the workable neighbouring cell on the map from the starting point where the robot lies and put it into the open list. For

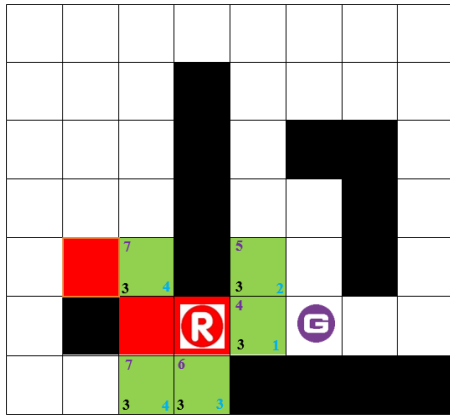
$$F(v) = g(v) + h(v)$$

Choose the cells that have the smallest value of heuristic cost  $h(v)$  and put into the closed list.



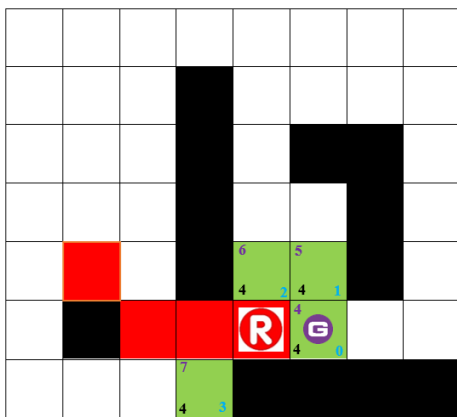
Next, from the previous process, the successor will be the current position of the nodes, in this case, the robot **R** is consider as Astar (A\*) searching vehicle will occupy to this successor point.



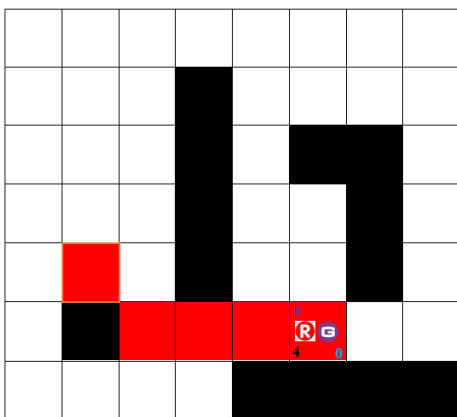


Next, the algorithm will repeat the process of best-first-search towards the goal destination. For every forward action, each cells will be computed with formula

$$F(v) = g(v) + h(v)$$



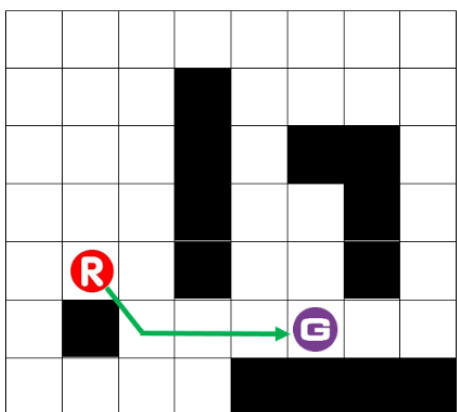
Again, the Astar (A\*) algorithm will choose the cells with the lowest path score and it will continue to iterate until the goal point has been reached.



The algorithm know that the path has been found when the heuristic value  $h(v)$  is equal to zero (0). For this example,

$$F(v) = g(v) + h(v)$$

$$4 = 4 + 0$$



Finally, the algorithm will backtracking all the previous smallest scored nodes in closed list and generate a pathway to navigate the autonomous robot to its goal destination with a shortest pathway.

### 3.5 Graphical User Interface (GUI)

In order to make this algorithm is more user friendly, a graphical user interface (GUI) has been created. This GUI is created using MATLAB software. This will allow the algorithm to be more interactive with the users. Using this GUI, the user can freely decide whether to capture new map or to load the existing map. As shown in Figure 3.16 is the (.fig) file of this Astar (A\*) algorithm project. Meanwhile, the final product look of this Astar (A\*) algorithm project is as shown in Figure 3.17.

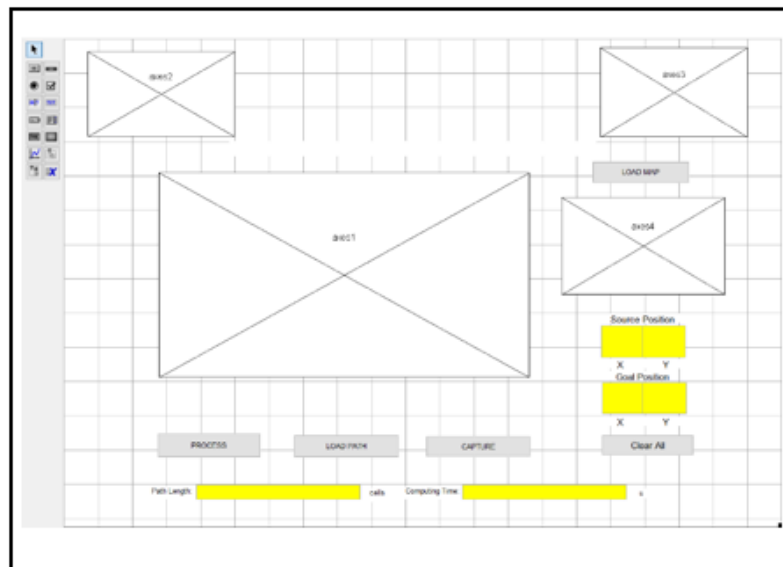


Figure 3.16: Development of Graphical User Interface for Astar (A\*) algorithm.

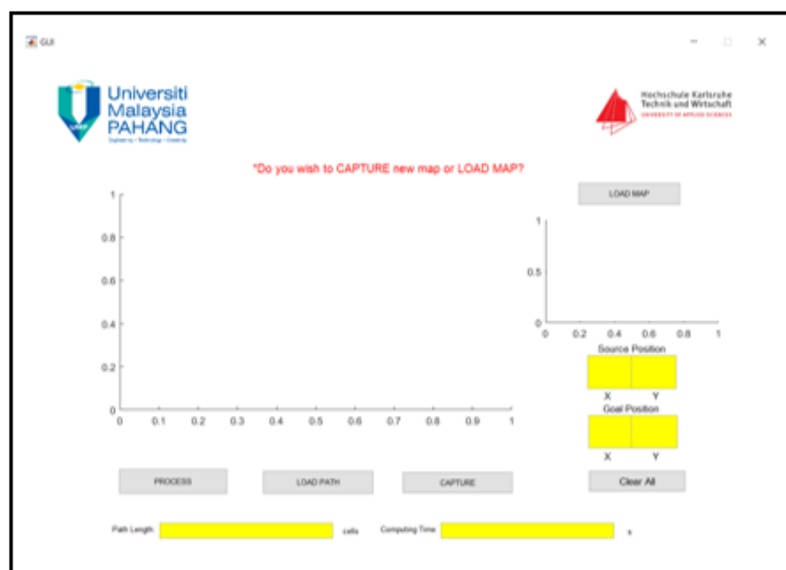


Figure 3.17: Final look of the Graphical User Interface of Astar (A\*) algorithm.

### 3.6 Conclusion

For this Astar (A\*) algorithm project, in order for the algorithm to be workable, there is necessary to provide a map for the algorithm to work on. The map in this project can be provided using a prototype that create an example of surrounding. The camera from this prototype frame will capture the image of environment. The image will then be converted into black and white file to provide a map for this algorithm. Then, from the map, the source and the goal point will be defined by the user. If only the points have been verified by the algorithm, Astar (A\*) algorithm will then start its searching process towards the goal destination. Using this definition of Astar (A\*) algorithm:

$$F(v) = g(v) + h(v) \quad 1.2$$

Firstly, the starting point of the robot will be inserted into the closed list. After that, the algorithm will identify the workable adjacent cells and insert it to the open list. From this open list, the heuristic cost (h) which the node with smallest score will be selected to be a successor. This successor will be then inserted into the closed list. The node that inserted in closed list will not be consider by the algorithm. The process is then repeated until the heuristic cost (h) is equal to zero (0) which tell the algorithm that the goal point has been reached. Then, the algorithm will backtrack the nodes to construct the path with a smallest score (shortest pathway).

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Introduction

In order to validate the ability of this Astar (A\*) algorithm to be implemented as a path planning for robotics navigation, several tests have been conducted. These tests are including the ability of this Astar (A\*) algorithm to provide the minimized pathway, the ability to provide a collision-free pathway, the behaviour of the algorithm towards the different complexity of map, different level of light intensity, different colours of environment, and lastly the effect of the goal's position on the map with the algorithm's behaviour. Thus, this chapter will provide the findings of all these cases. Table 4.1 will explain briefly the determination of these investigations.

*Table 4.1: List of tests that will be conducted for Astar (A\*) algorithm.*

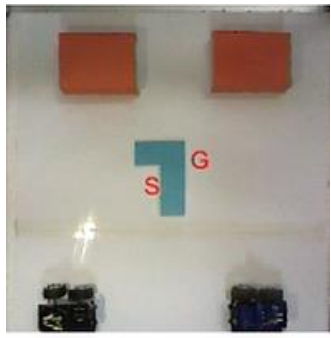


Test	Description
Shortest Path Scoring	To validate the ability of algorithm to provide the shortest pathway.
Obstacle Avoidance	To verify the capability of the algorithm to provide a pathway that avoiding the obstacles along its way.
Effect of Complexity of The Map with Computation Time	This test will investigate effect of map complexity level towards the processing time of the algorithm to reach its goal destination.
Effect of Light Intensity Towards Astar (A*) Algorithm Behaviour	This test will show the behaviour of this Astar (A*) algorithm to the different level of light intensity. Since the map provided for this algorithm is by using single camera, thus, this effect of light intensity to the algorithm's behaviour is important to be analysed.

Effect of Colour Towards Astar (A*) Algorithm Behaviour	This test conducted to investigate the effect of colours to the algorithm's behaviours. This Astar (A*) algorithm is an algorithm that working through map provided by the camera, thus, this test will figure out the relationship of ground's colour and obstacle's colour with this algorithm's behaviours.
Effect of Goal Position Inside The Obstacle Area	In this Astar (A*) algorithm, it's searching behaviour is based on the movable cells on the grid map provided. Thus, there will be some cases where the goal point or source point is located inside the area of obstacles. This test will verify the behaviour of this algorithm when it facing this type of situation.

## 4.2 Shortest Path Scoring

In order to determine whether the path generated by the algorithm is a minimum path score, the following investigation is conducted. This investigation was based on three different cases. Considering that all the cases consists of the same starting point and being directed to the same goal point, this is to analyses the behaviour of the Astar (A\*) algorithm which expected to be able to provide a short pathway connected the source and the goal point. The behaviour of this cases is that they had been differ in the presence and position of the obstacle. The Table 4.2 shows the three (3) different cases.

Table 4.2: Different cases of map with different types of obstacles.

CASE	CASE 1		CASE 2		CASE 3	
						
Source	X	150	Y	131		
Goal	X	186	Y	113		

Consider the position of the source and the goal point is a constant variable, and the position and number of presence obstacles is being manipulated, the result of path generated by the Astar (A\*) algorithm is as shown in Table 4.3.

Supposedly, there are two possible paths can be taken by the algorithm like shown in Figure 4.1 in order to find the shortest pathway to the goal position. There are; path (a) and path (b). This algorithm is smart enough to decide which way should it take that give the best min path scoring. Figure 4.1 shows the possibilities of the path taken by the algorithm.

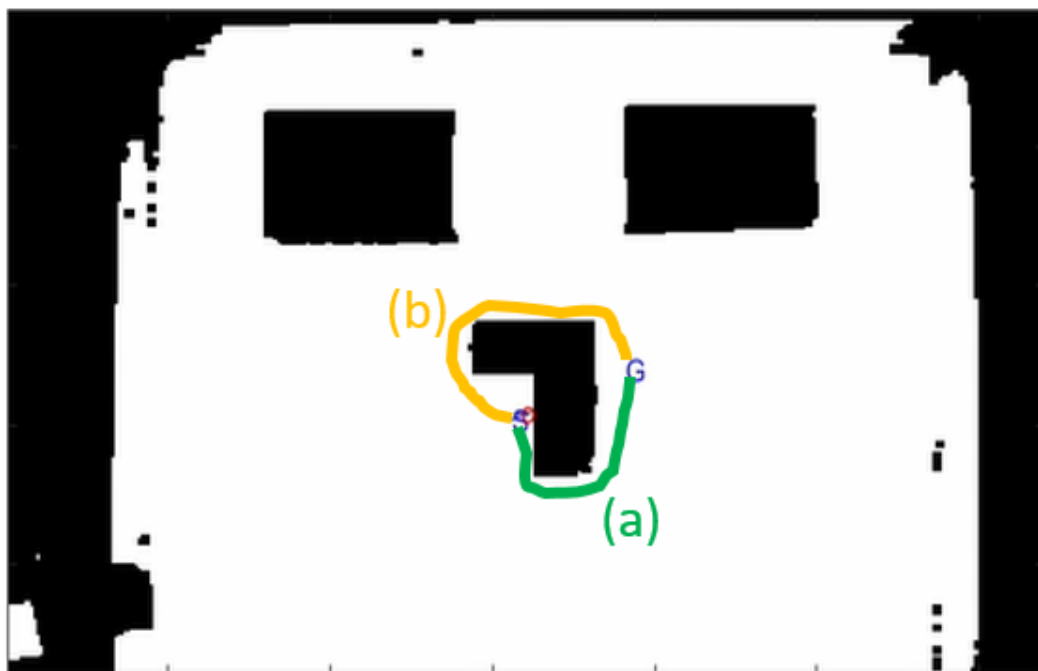


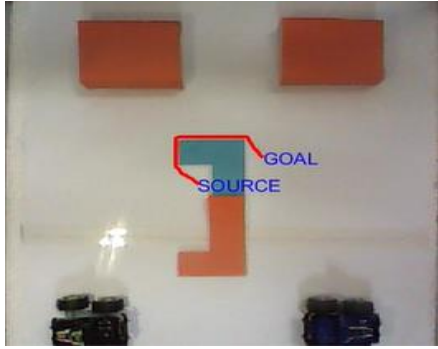





Figure 4.1: Possible direction of path score.

Table 4.3: Results of different path scored constructed by the algorithm.

CASE	PATH GENERATED	
CASE 1		 <p data-bbox="930 551 1361 577"><i>*refer file <a href="#">Shortest Path Case 1</a> from video folder.</i></p>
CASE 2		 <p data-bbox="930 1055 1361 1081"><i>*refer file <a href="#">Shortest Path Case 2</a> from video folder.</i></p>
CASE 3		 <p data-bbox="930 1480 1361 1507"><i>*refer file <a href="#">Shortest Path Case 3</a> from video folder.</i></p>

Referring Table 4.3 in case 1, the number of obstacle presence along the pathway between source point and goal point is one and the shape its appear to be simpler compared to case 2 and case 3. To proof that the algorithm is manage to find its shortest path score towards the goal, it is by comparing the behaviour of the path generated between case 1 and case 2. This is because, case 2 is where the algorithm is force to take the (b) path like shows in Table 4.3, instead of (a) path taken in Case 1.

Table 4.4 shows the results of shortest path scoring. From this result, it is clear that at the fix position starting and goal point, the algorithm is smart enough that it takes path (a) direction with path score of  $\approx 92$  cells. This is shorter if compare to the path (b) direction with  $\approx 94$  cells path score. Thus, this Astar (A\*) algorithm were proven to be able to provide the shortest path scoring.

Table 4.4: Result of Astar (A\*) algorithm's path scored.

CASE	NUM OBSTACLE PRESENCE	PATH TAKE	PROCESSING TIME [s]	PATH LENGTH [cells]
1	1	a	240.113	92.133
2	1	b	228.535	94.095
3	1	b	246.636	94.975

### 4.3 Obstacle Avoidance



Figure 4.2: Original image recorded by camera.

In this algorithm, it is important for the algorithm to have the ability to avoid any obstacle that present along its way. This feature is really important in order to provide a collision free pathway for the robot. Figure 4.2 show the real map with obstacle presence in it. From this real map, it will convert into black and white map and the algorithm will then mark the map with value of 1; movable cell and mark as 0; died cell. Movable cell is where it is accessible for the robot while navigate to the goal destination. Meanwhile, died cell is where the obstacles lie. Figure 4.3 shows the marked map done by the Astar (A\*) algorithm.



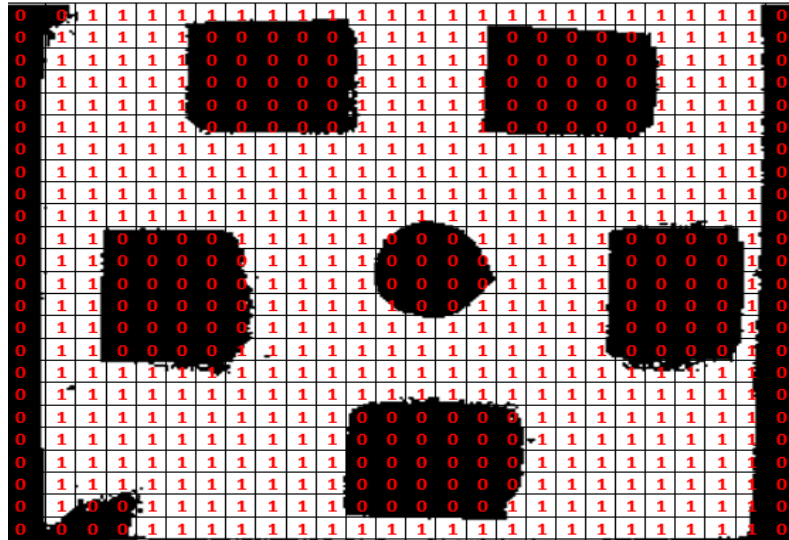


Figure 4.3: Converted black and white map with a binary value. 1: obstacle-free area; 0: obstacle area.

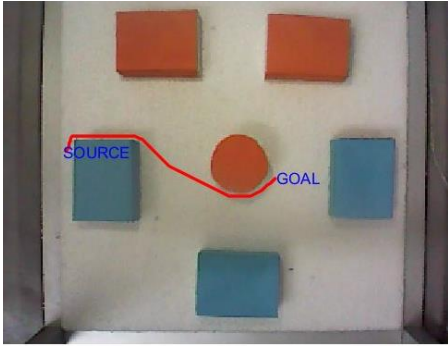
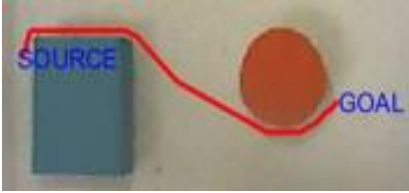


Referring to the map in Figure 4.3, the algorithm will tell the robot to avoid with the obstacle by providing a pathway that have no contact with the obstacle. This is to make sure that the robot will be navigate with a collision free navigation.

Next is to provide the initial or current point of the robot and also the goal destination for the robot. Two cases with different starting and end point were tested in this part. Followings Table 4.5 shows the cases.

Table 4.5: Different cases with different source and goal position.

CASE 1		CASE 2	
Source	(45, 106)	Source	(140, 117)
Goal	(193, 124)	Goal	(195, 117)

Table 4.6: Result of path generated by Astar (A\*) algorithm with obstacles avoidance ability.

CASE	PATH GENERATED	
CASE 1		 <p data-bbox="916 548 1353 611">* refer file <a href="#">Obstacle Avoidance Case 1</a> from video folder.</p>
CASE 2		 <p data-bbox="916 958 1353 1021">* refer file <a href="#">Obstacle Avoidance Case 2</a> from video folder.</p>

According to Maria Isabel Ribeiro, obstacle avoidance function in robotics is methodologies of shaping the robot's path to overcome unexpected obstacles without any contact with the obstacles (Ribeiro, 2005). With the definition of this obstacle avoidance by Maria Isabel Ribeiro, the following result can be verified. The result can be seen in Table 4.6. From the result obtained, it is clear that the Astar (A\*) algorithm manage to generate a path that capable to avoid any obstacles along its way and making no contact with the obstacles.

#### 4.4 Effect of Complexity of The Maps with Computation Time.

In order to investigate the effect of computation time towards the level of complexity of the map, several types of maps with different number of obstacles presence in the map. This is because high number of obstacles presence in the pathway between the source and goal destination, producing a higher complexity level of the map. The Table 4.7 shows the cases with different types of map with different level of map complexity.

Table 4.7: THREE (3) different cases with different level of map complexity.

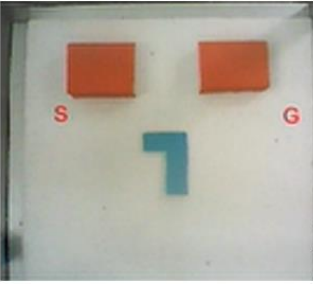
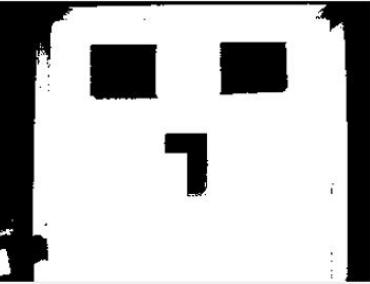
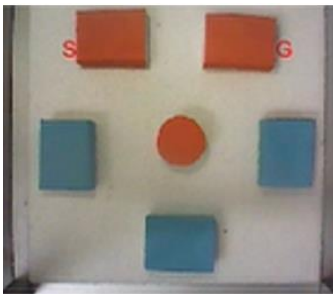
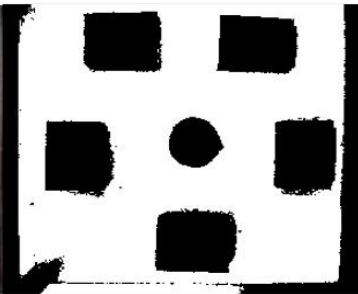

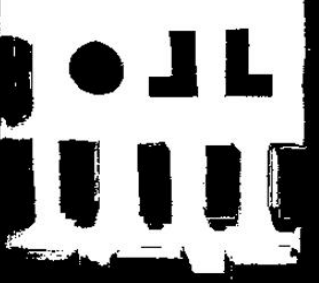
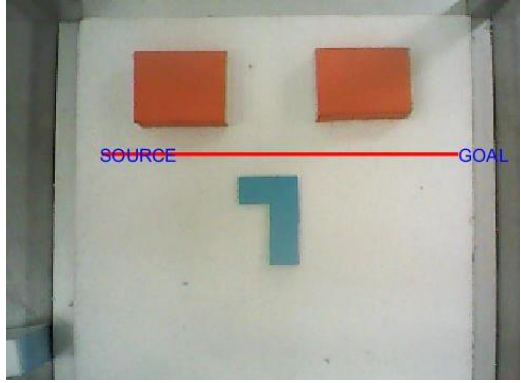
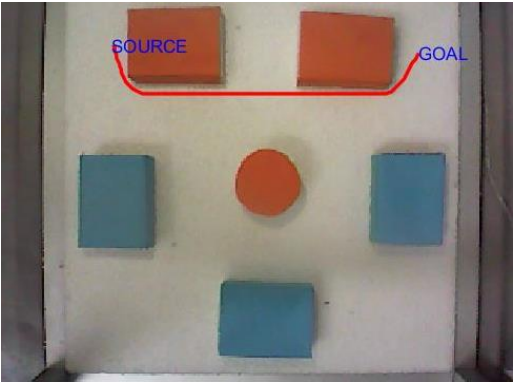
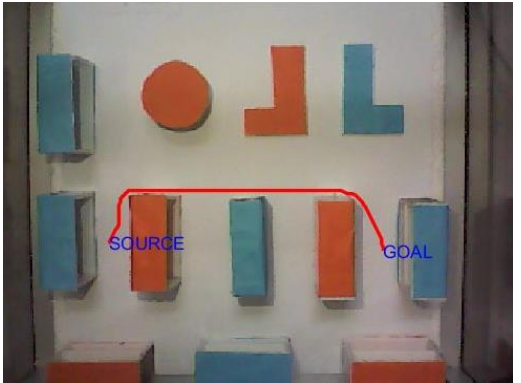
CASE 1			
			
Source	(61, 100)	Goal	(278, 100)
Number of Obstacle in Map			3 Obstacles
Number of Obstacle Along Pathway			0 Obstacle
CASE 2			
			
Source	(72, 30)	Goal	(260, 35)
Number of Obstacle in Map			6 Obstacles
Number of Obstacle Along Pathway			2 Obstacles
CASE 3			
			
Source	(72, 150)	Goal	(240, 155)
Number of Obstacle in Map			12 Obstacles
Number of Obstacle Along Pathway			3 Obstacles

Table 4.8: Path generated by Astar (A\*) algorithm for different level of map complexity.

CASE	PATH GENERATED	
CASE 1		<p><i>*refer file <a href="#">Effect Map Complexity 1</a> from video folder.</i></p>
CASE 2		<p><i>*refer file <a href="#">Effect Map Complexity 2</a> from video folder.</i></p>
CASE 3		<p><i>*refer file <a href="#">Effect Map Complexity 3</a> from video folder.</i></p>

From the pathway generated by this algorithm as shown in Table 4.8, the effect of map complexity to the computation time can be determined. As it can be seen from path generated in Table 4.8, if the number of obstacle present in between the source point and the goal point is high, which lead to complex map, thus, the time taken for the algorithm to reach the goal point is longer. But, as the complexity of the map is at lower level, the time taken for the algorithm to reach to its goal point is shorter. The result of this investigation can be seen in Table 4.9. From the results, its clear that as for the case 1 which have the lowest level of complexity of the map gives the shortest processing time (30.807 seconds) compare to case 3 with highest complexity level which give processing time of (1370.550 seconds). This proven that regardless how long the path length generated by the algorithm, if it is involving the high level of map's complexity, it will take longer computation time.

*Table 4.9: Result of processing time of Astar (A\*) algorithm respected to its complexity of the map.*

CASE	NUM OBSTACLE PRESENCE	MAP COMPLEXITY LEVEL	PROCESSING TIME [s]	PATH LENGTH [cells]
1	0	low	30.807	217
2	2	high	991.542	220.284
3	3	highest	1370.550	216.514

#### 4.5 Effect of Light Intensity Towards Astar (A\*) Algorithm Behaviour

Astar (A\*) algorithm is an algorithm that work by scanning the surrounding through the map provided. In this project, the map is being provided by the image taken by the camera. The type of camera that has been used in this project is a web camera model C170 from Logitech. The pixel value for this camera is only up to 5 megapixels with software enhanced (Logitech, 2018). Since the camera is put on the top of the prototype to get the top view of the surrounding, it is really dependent of the light intensity. Figure 4.4 shows the position of the C170 Logitech web camera used in this project.

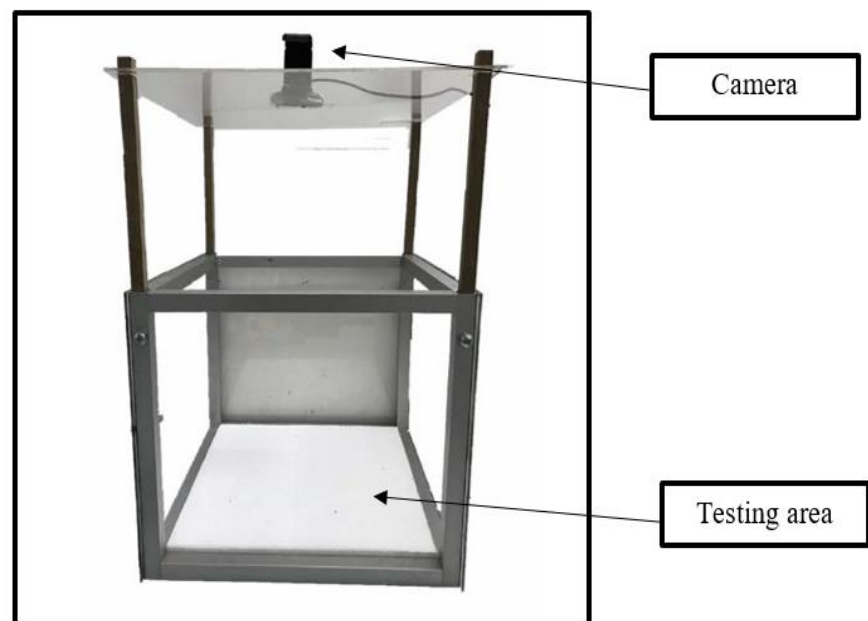

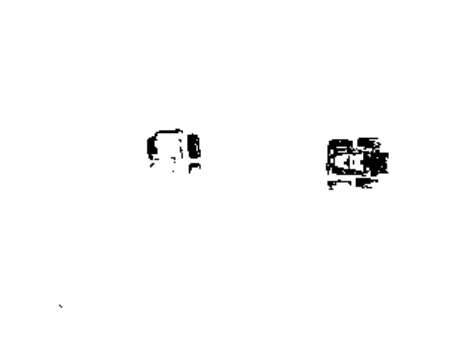
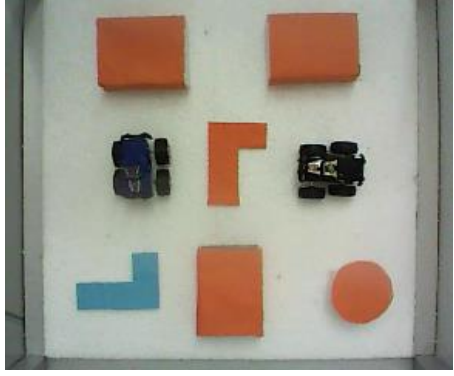
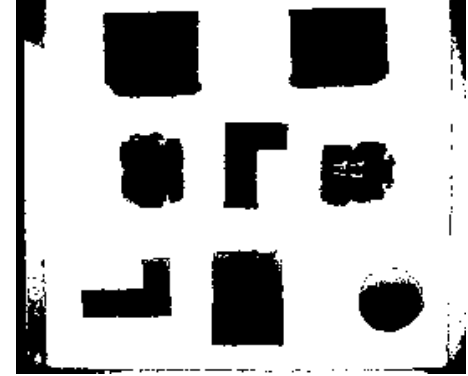
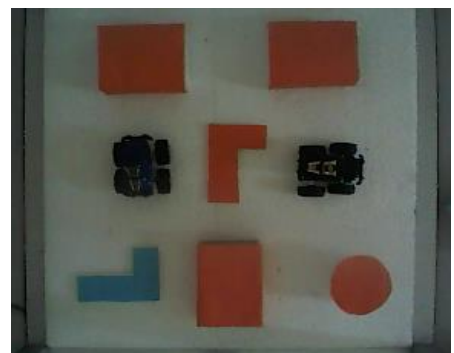



Figure 4.4: Position of the C170 Logitech web camera used in this project.

According to the position in Figure 4.4 it makes the light intensity as one of significant aspect for this project. This is because, any excessive or deficiency amount of light will affect the behaviour of this Astar (A\*) algorithm. If there are excessive amount of light detected by the camera, the algorithm will not be able to detect the presence of obstacles. This is because, the obstacles will appear to be a white region when the map is converted into black and white map. The situation is contradictorily when the camera is receiving small amount of light intensity. This will make the map to appear as a black region on the map. Thus, the algorithm will claim that the source or goal point is lay on the obstacle. The result can be seen in the Table 4.10.

Table 4.10: Result of effect of light intensity to the Astar (A\*) algorithm behaviour.

LEVEL OF LIGHT	ORIGINAL MAP	CONVERTED B/W MAP
EXCESSIVE		
MEDIUM		
LOW		

As shown in Table 4.10, the result of the effect of light intensity towards Astar (A\*) algorithm behaviour obtained. It is clear that for the excessive level of light case, the map is appearing to be white even there are obstacles appear on the map. This is something that called fatal error for the program since the path still can be generated even though there is obstacle lies in between the source and goal point. The algorithm seems not to be able to detect the presence of obstacles along its way. As a result, this may cause huge collision between the robot and the obstacle. Thus, the robot navigation become harmful whether to itself also to its surroundings. Figure 4.5 shows the situation where the path generated just simply go through the obstacles.

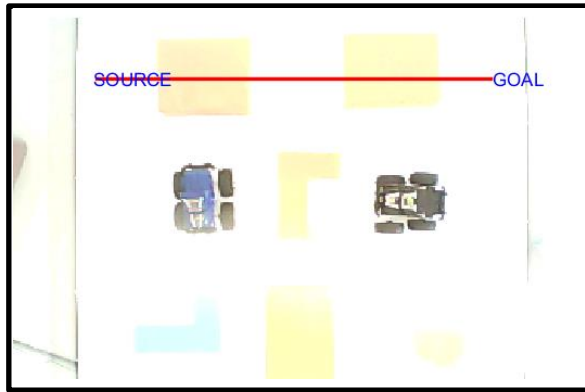


Figure 4.5: Path generated that penetrate through obstacles.

Contrarily with the situation with the low level of light intensity is applied to the algorithm, this will resulting in the pop-up notification appear from the system. This notification is from the algorithm to notify the user that the source or goal were lies on the obstacles. This is due to the fact the map is appear to be black region (refer to Table 4.10 at low light case) that indicate there is the area of obstacle which impossible for the robot to move through it. Figure 4.6 show the situation where the pop-up notification is appeared due to this situation.

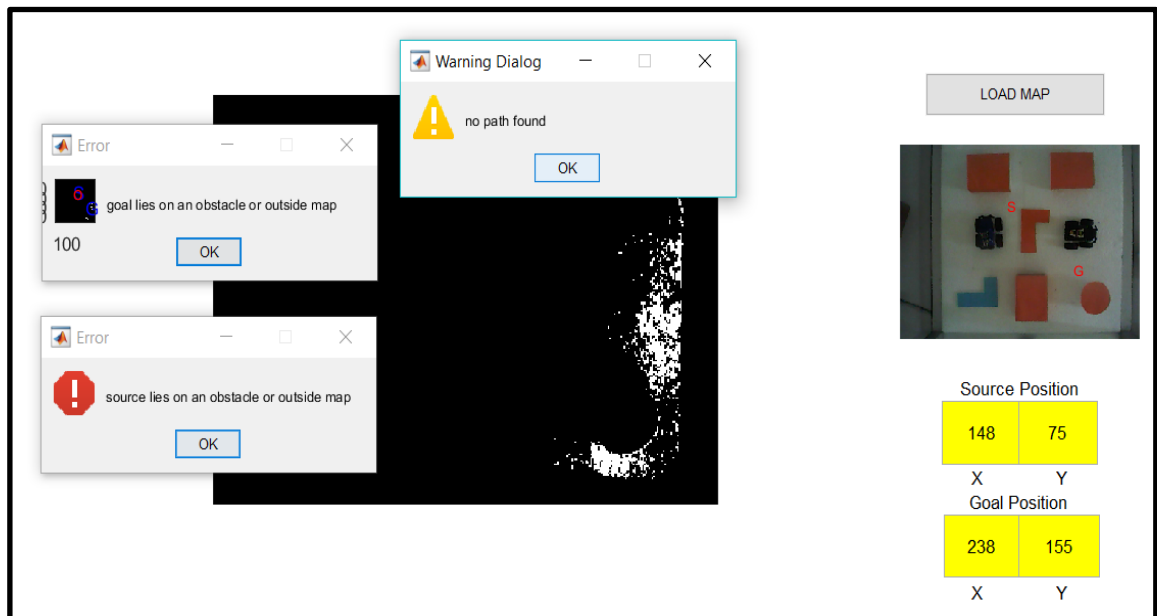


Figure 4.6: Dialog box that popped-up to notify an error.



#### 4.6 Effect of Colour Towards Astar (A\*) Algorithm Behaviour

In every image captured by the camera, besides lighting, colour also is a something that will affect the quality of image taken. This is due to the fact that colour of the image recorded by the camera is highly dependent with lighting condition. Although it is something that depending to each other, in this project, colour is something that will give an impact towards the result. This is because, in order to mark the position of the obstacles on the map, it is depending on the colour saturation of the random object. This colour saturation is important to be comparable to each other in order to be make sure that the conversion of image file to the black and white map is flawlessly.

Since the Astar (A\*) algorithm is a collision free pathway provider, thus it is compulsory for the system to be able to distinguished between the obstacles and the non-obstacles objects on the map. In this context, colour will plays a significant role for the algorithm to be able to locating the obstacles. There are two important aspects for this investigation:

- Aspect 1: The colour of ground in the map layout, and
- Aspect 2: The colour of random objects used to provide obstacles.

These can be verified through the following cases:

##### CASE 1



Figure 4.7: Image recorded in case 1.

In this case, the colour used for ground in the map is same as the colour of random objects used. The Figure 4.7 shows the image captured by the camera for this case.

## CASE 2



Figure 4.8: Image recorded in case 2.

For this second case, the colour of the ground in the map were different with the colour of the random objects used in this project. The Figure 4.8 shows the image capture by the camera for this case.


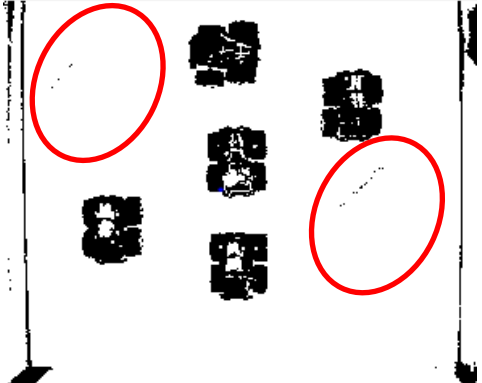

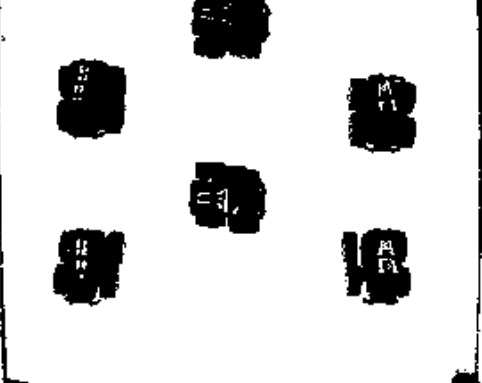

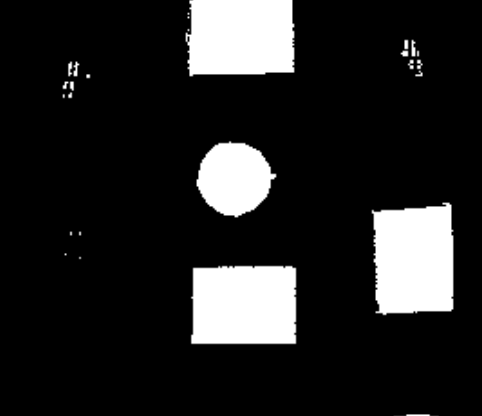
## CASE 3



Figure 4.9: Image recorded in case 3.

Lastly, for the third case, the colour of the ground layout is darker compared with the colour of the objects used in this projects. This third case is to investigate whether the algorithm is able to distinguish the free path and the obstacles area in the map if it a darker colour of ground. The Figure 4.9 shows the image capture by the camera for this case.

Table 4.11: Result of effect of colour to the Astar (A\*) algorithm behaviour.

CASE	ORIGINAL IMAGE	BLACK AND WHITE MAP
1		
2		
3		

From the result shown in the Table 4.11, there are three different cases. For case 1, the colour of ground used is blue which is same as the colour of obstacle. Referring back to the table of result for case 1, the red-circled mark shows that the missing obstacle when the map is converted into black and white image. This is due to the same colour of the obstacle with the road make it look transparent with the ground. Thus the algorithm may read that there are no obstacles in that area.

Besides that, for case 2, the situation is that the ground colour is different with the random object used. The result for this case can be seen by referring back to the result shows on Table 4.11, where for this case, the image processing in this algorithm is able to locate all the obstacles appear on the map. Note that the colour used for the ground colour of this case is light yellow.

Next, for the case 3, this situation is same as situation on case 2, but instead of using the light colour of ground skin, a brown colour of ground has been used. Technically, this brown colour ground is more saturated colour compared to the light yellow colour used in case 2. Unfortunately, the result obtained by using this colour is as shown in Table 4.11 for case 3. When it converted into black and white map, the image processing in this system do not consider the ground as a roadway for the algorithm, but instead it considers the ground as obstacle area, and the light colour of obstacle is consider as an obstacle-free area. This situation can be seen in the table of case 3.

As a conclusion, for this investigation of the effect of colour to the Astar (A\*) algorithm behaviour, it can be said that the algorithm will able to work accordingly if the map provided is containing the different colour of the ground and obstacles, also when the colour of the ground is lighter. This is because, the system will provide the map for the algorithm to work on by converting the recorded image file by the camera into a black and white binary map file. Thus, if darker part in the object will be consider as a black area by the system. That's why the result of situation in case 3 shows that the algorithm is confuse between obstacle-free area with the area that contain obstacle.

#### 4.7 Effect of The Goal Position Inside The Obstacle Area

In Astar (A\*) algorithm, there are some cases that resulting the algorithm didn't manage to find any shortest path towards the goal. One major example of the case is when the goal point is placed within the obstacle area. Although the obstacle practically will show the binary zero value at the area of obstacle on the map, but, some cases that the obstacle area itself contain area that available for the algorithm to go through. For an example, the fence-type of obstacle that shows the obstacle area is only at outer-side of obstacle and at inner side, the area appears to be a non-obstacle area. This situation can be seen as shown in Figure 4.10.

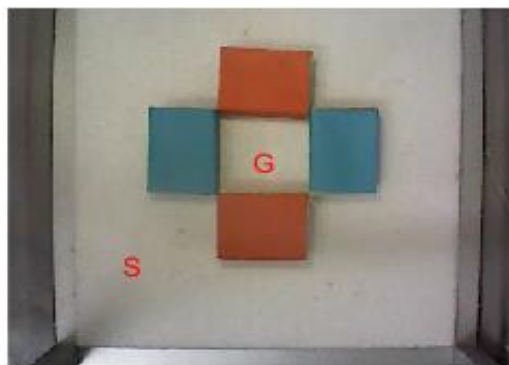


Figure 4.10: Goal point that located inside the obstacle area.

As shown in the Figure 4.10, the source position is outside the obstacle area, but the position of the goal point is within the obstacle area. Note that, although it is an obstacle area, it still has the available terrain for the algorithm to reach the goal, but because of the goal point is surrounding by the obstacles, it makes it impossible for the algorithm to reach the goal point. The situation in a map view can be seen in Figure 4.11 where it shows that the goal point is surrounded by the black area which is fence-type of obstacle.

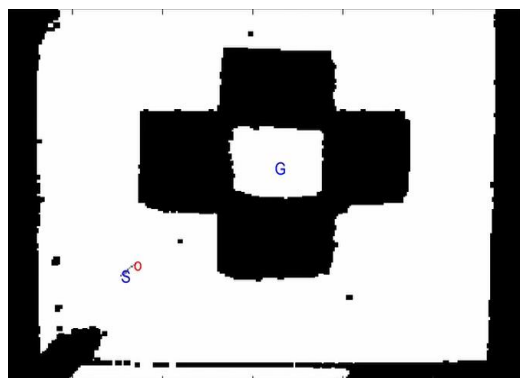


Figure 4.11: Black and white map shows goal point that located inside the obstacle area.

From this situation as shown in Figure 4.11, it can clearly seen that there is no available pathway for the algorithm to penetrate into the obstacle area to reach the goal point. As the result, the algorithm will get stuck and the searching process is then stop. This is because, the algorithm can't find any available way to reach the goal. The Figure 4.12 shows the result for this situation where the algorithm got stuck and stop it process. Thus, for this situation, it is considering as a limitation to the Astar (A\*) algorithm as it not be able to identify that the goal point is within the obstacle area.

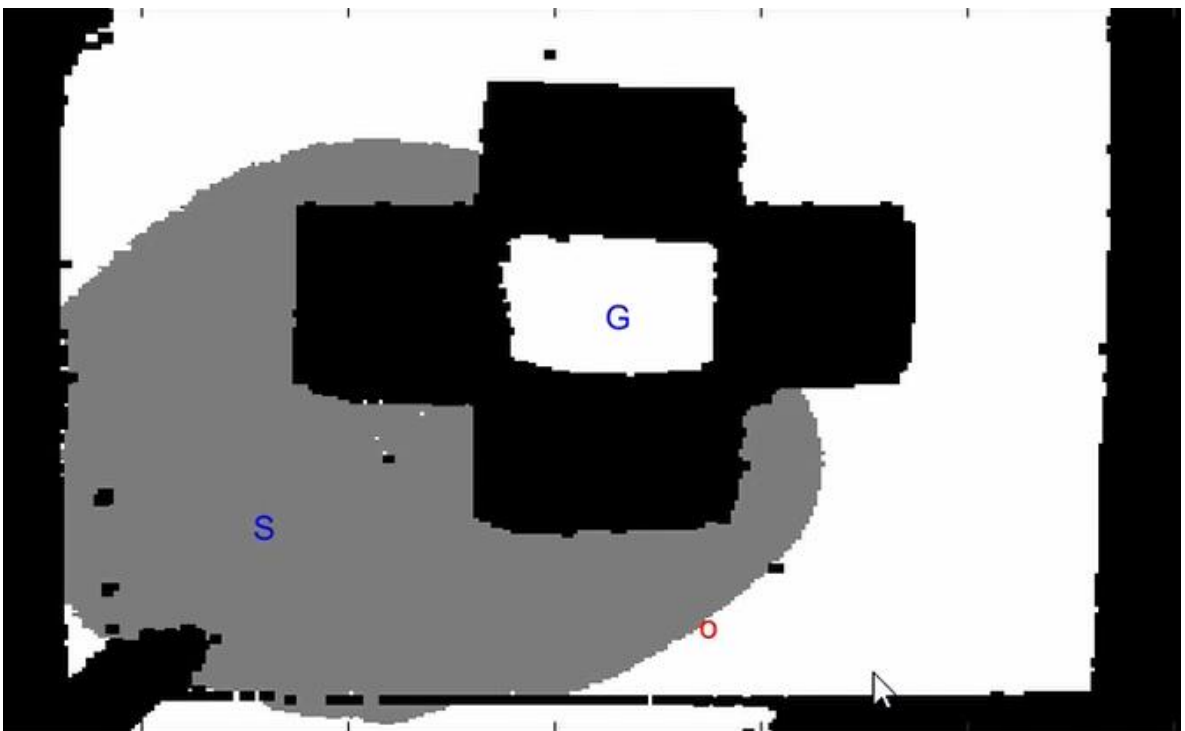


Figure 4.12: Astar (A\*) algorithm try to search for the way to reach the goal point.

*\*refer file [Goal in Obstacle Area](#) from video folder.*

## **CHAPTER 5**

### **CONCLUSION**

#### **5.1 Introduction**

This chapter will give a short and briefly summary for overall of the project and also some recommendations for the future research.

#### **5.2 Conclusion**

As a conclusion, the main objectives of this project which to implement an Astar (A\*) algorithm as a path planning for robotics navigation had been achieved. Basically, this project which has been done at Universiti Malaysia Pahang. After a few discussion sessions with UMP's supervisor as well as HsKA's supervisor, eventually the Astar (A\*) algorithm is functioning accordingly.

According to the real functions of this Astar (A\*) algorithm, it is supposedly should be able to provide a collision-free pathway with an optimised distance towards the goal point. In the end of this project these functions have been tested and the result is positive for both functions which reflected to the objectives of this algorithm. There are two main functions which has been achieved by this Astar (A\*) algorithms.

First and foremost, the Astar (A\*) algorithm is able to provide a pathway with the minimum path score. This function is successfully tested by using MATLAB software. For this function, the algorithm managed to generate a pathway and find its way with a shortest distance to the goal point.

Secondly, the main function of this Astar (A\*) algorithm is to be able to provide a collision-free pathway. In other word, the algorithm should be able to avoid any obstacles that appear along its way. This function is tested by providing a real time obstacles using any random objects. This situation with random objects as obstacle is then captured using a single camera and the image recorded is converted as a black and white map. The obstacles will be appeared as a black spot in the map, and this Astar (A\*) algorithm is manage to pass-by the obstacles with no contact with the obstacles.

However, despite of positive response of both function, there are also several factors that need to be consider in order to implement this Astar (A\*) algorithm. The factors like light intensity, colours of environment, and position of the goal and source point compulsory to be consider in this project. This is because, without considering to these factors, it may result a faulty to the Astar (A\*) algorithm searching behaviour.

To conclude, this project Astar (A\*) Algorithm Implementation for Robotics Path Planning Navigation which has been carried out within three (3) months at Universiti Malaysia Pahang is successfully done. This path planning algorithm is proven to be safe to be implemented in robotics navigation.

### **5.3 Recommendations**

Although the project's objectives were successfully achieved, but it is still has several ways could be done in order to improve the result's accuracy. Thus, several recommendations can be considered for the good of future study. Due to the lack of time and techniques, this project only focused on implementation of algorithm for robotic navigation.

For the first recommendation, in the future work of this project, the project scope should be improvised, where the researcher can be applying to any robots. This is to testify the ability of the robot to move follows the path that generated from this project. If there are any issues appear during the process of applying this algorithm, the further analysis can be conducted to solve the issues. This is because, there are some possibility that the robot cannot follow the path generated by the algorithm in this project.



In order to test whether the robot is able to follow the line generated in this project, the researcher could also do some improvement to the algorithm so that it can include the tolerance when considering the obstacles area. This is because, if we look at the way the line is generated in this project when it's avoiding the obstacles, it is really near with the obstacle's wall. This situation is a little bit sweating, because although the path generated of this obstacle is successfully avoiding the obstacles, but, in the future when this algorithm is being apply to any robot, the researcher should include the size of the robot itself because the robots may be difference in size, sometimes bigger and sometimes smaller.

Other than that, for the next recommendation, in the future, the researcher can also include the sensors to the system, this is because, beside of the existing camera is taking a top view of the system, the sensors also can be synchronized to the system where it can detect the presence of the obstacles in a three dimensional (3D). This will make the system to not to rely to the camera that very sensitive with the light and colours, it will then detect the presence of the obstacle using waves (light or sound). As for example, the type of sensors that can be use is laser sensor or ultrasonic sensor.

## REFERENCES

- Chilian A, Hirschmuller H. (2009). Stereo camera based navigation of mobile robots on rough terrain. In H. H. Chilian A, *Stereo camera based navigation of mobile robots on rough terrain* (pp. 4571-4576). St. Louis, USA: IROS.
- Konolige K. (1997). *Small vision systems: hardware and implementation*. Hayama, Japan: Int. Symp. Robotics Research,.
- A. Smith, Ding, Ulusoy. (2012). Robust multi-Robot Optimal Path Planning With Temporal Logic Constraints. *Robotics and Automation (ICRA)*.
- Adam A. Razavian, Sun J. (2005). Cognitive Based Adaptive Path Planning Algorithm for Autonomous Robotic Vehicles. *Southeast Con*, 8-10.
- Andrej Babinec. (2014). Path Planning With Modified Astar Algorithm for a Mobile Robot. *ScienceDirect*, 61.
- Carsten, J. (2007). Global Path Planning on board the Mars Exploration Rovers. *IEEE Aerospace Conference*.
- Chelvi, S. T. (14 FEB, 2016). *Road accidents mainly caused by human error, study reveals*. Retrieved from The Sun Daily: <http://www.thesundaily.my/news/1692065>
- Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). *Single-Source Shortest Paths :Introduction to Algorithms*. 2nd ed. Cambridge: 581-635.
- Correll, N. (4 September, 2011). *Introduction to Robotics #4: Path-Planning*. Retrieved from Correll Lab CU Computer Science: <http://correll.cs.colorado.edu/?p=965>
- David M. Bourg, Seeman G. (2004). *AI for Game Developer*. O'Reilly.
- Ghangrekar, S. Y. (2009). A PATH PLANNING AND OBSTACLE AVOIDANCE ALGORITHM FOR AN AUTONOMOUS ROBOTIC VEHICLE. 14.
- Intorobotic. (20 November, 2013). *Intorobotic*. Retrieved from Into Robotics: <https://www.intorobotics.com/types-sensors-target-detection-tracking/>

- Kirti Bhagat, Sayalee Deshmukh, Shraddha Dhonde, Sneha Ghag. (2016). Obstacle Avoidance Robot. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 1-2.
- LaValle. (2006). *Planning Algorithms*. New York, USA: Cambridge University Press.
- Logitech. (2018). C170 Web Camera by Logitech. Malaysia.
- M. Shafie Abd. Latiff, and R. Hassan. (2004). *An Efficient Virtual Tour- A Merging*. Kuala Lumpur.
- Martin Florek, Andrej Babinec, Martin Kajan. (2014). Path Planning with Modified Astar Algorithm for Mobile Robot. *ScienceDirect*, 61.
- Obara T., Yamamoto K., Ura T., Maeda H., Yamato H. (1994). Development of an Autonomous Underwater Vehicle R1 with a Closed Cycle Diesel Engine.
- Polaroid Corporation. (1992). *Ultrasonic Ranging Systems*. Cambridge Massachusetts. Retrieved from Cambridge Massachusetts.
- Qidan Zhu, Yongjie Yan, and Zhuoyi Xing . (2006). Robot Path Planning Based on Artificial Potential Field. *College of Automation, Haerbin Engineering University*.
- Ribeiro, M. I. (2005). Obstacle Avoidance. 1-2.
- Sadeghi-Niaraki, A., Varshosaz, M., Kim, K., and Jung, J. (2011). Real world representation of a road network for route planning in GIS.
- T. Frohlich and D. Kullmann. (2002). *Autonomous and Robust Navigation for Simulated Humanoid Characters in Virtual Environments*. First International Symposium on Cyber Worlds (CW'02).
- The RoboRealm . (2006). *The RoboRealm Vision for Machines*. Retrieved from Path Planning: [http://www.roborealm.com/help/Path\\_Planning.php](http://www.roborealm.com/help/Path_Planning.php)
- Ulrich , Nourbakhsh. (2000). Appearance-based obstacle detection with monocular color vision. *AAAI*.

Vo Thi Huyen Trang, Tran Quoc Toan, A.A. Sorokin. (2017). Using modification of visibility-graph in solving the problem of shortest path for robot. *2017 International Siberian Conference on Control and Communications (SIBCON)*, 1.

W.H.Munro. (1990). Ultrasonic Vehicle Guidance Transducers. *ieeexplore*, 349.

Werner B, Surmann H, Pervolz K. (2006). 3D time-of-flight cameras for mobile robotics. In *3D time-of-flight cameras for mobile robotics* (pp. 790-795). Beijing, China: IROS.

Yahja A., Singh S., Stentz A. (2000). *An Efficient on-line Path Planner for Outdoor Mobile Robots*. Robotics and Autonomous Systems.

## APPENDIX A PROJECT GANTT CHART

No	PROJECT ACTIVITY	P-Plan A-Actual	NOV					DEC				JAN				
			W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
			(1-3)	(6-10)	(13-17)	(20-24)	(27-1)	(4-8)	(11-15)	(18-22)	(25-29)	(1-5)	(8-12)	(15-19)	(22-26)	(29-2)
<b>1.0</b>	<b>CHAPTER 1: PROJECT BACKGROUND RESEARCH</b>															
1.1	Define project background	P A														
1.2	Define project problem statement & objectives	P A														
1.3	Define project scope of study	P A														
<b>2.0</b>	<b>CHAPTER 2: LITERATURE REVIEW</b>															
2.1	Search through the sources & collect materials of previous work	P A														
2.2	Identify and understanding the algorithm based on previous	P A														
<b>3.0</b>	<b>CHAPTER 3: METHODOLOGY</b>															
3.1	Experimental set up	P A														
3.2	Design and develop algorithm using Astar based	P A														
3.3	Implement the new concept system	P A														
<b>4.0</b>	<b>CHAPTER 4: RESULT, ANALYSIS AND DISCUSSION</b>															
4.1	Testing	P A														
4.2	Compare the result with the previous system	P A														
<b>5.0</b>	<b>CHAPTER 5: CONCLUSION AND RECOMMENDATION</b>															
5.1	Identify the future works	P A														
5.2	Conclude the project	P A														
<b>6.0</b>	<b>REPORT DOCUMENTATION</b>															
6.1	Report & Presentation	P A														

## APPENDIX B CODING OF A\* ALGORITHM

```
function varargout = GUI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @GUI_OpeningFcn, ...
    'gui_OutputFcn',  @GUI_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT
function varargout = GUI_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;

% --- Executes just before GUI is made visible.
function GUI_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;
imageOriginal=imread('logo2.jpg');
axes(handles.axes2);
imshow(imageOriginal);
axes(handles.axes3);
imshow('logo3.jpg');
set(handles.instruction,'string','*Do you wish to
CAPTURE new map or LOAD MAP?','FontSize',12);
guidata(hObject, handles);
```

```

% --- Executes on button press in processload.
function processload_Callback(hObject, eventdata,
handles)
set(handles.edit7, 'String', '');
set(handles.edit8, 'String', '');

global source; global goal; global map; global path;
global n; global Q;
global closedList; global mapOriginal; global
resolutionX;
global resolutionY; global newPos; global A; global
filename;

mapOriginal=im2bw(imread(filename));
resolutionY=240;
resolutionX=320;
axes(handles.axes1);
imshow(mapOriginal);

startX= get(handles.startX, 'string');
startX=str2double(startX);
startY= get(handles.startY, 'String');
startY=str2double(startY);
source=[startY, startX];
goalX= get(handles.goalX, 'String');
goalX=str2double(goalX);
goalY= get(handles.goalY, 'String');
goalY=str2double(goalY);
goal=[goalY, goalX];

%=====connection matrix=====
% robot (marked as 2) can move left, right, up and
down (all 1s).

conn=[1 1 1 1 1; % another option of conn
      1 1 1 1 1;
      1 1 2 1 1;
      1 1 1 1 1
      1 1 1 1 1];

%=====connection matrix=====

display=true; % display processing of nodes
%%%%%%%%%% parameters end here %%%%%%%%%%%

```

```

mapResized=imresize(mapOriginal,[resolutionX
resolutionY]);
map=mapResized; % grow boundary by a unit pixel
for i=1:size(mapResized,1)
    for j=1:size(mapResized,2)
        if mapResized(i,j)==0
            if i-1>=1, map(i-1,j)=0; end
            if j-1>=1, map(i,j-1)=0; end
            if i+1<=size(map,1), map(i+1,j)=0; end
            if j+1<=size(map,2), map(i,j+1)=0; end
            if i-1>=1 && j-1>=1, map(i-1,j-1)=0; end
            if i-1>=1 && j+1<=size(map,2), map(i-
1,j+1)=0; end
            if i+1<=size(map,1) && j-1>=1, map(i+1,j-
1)=0; end
            if i+1<=size(map,1) && j+1<=size(map,2),
map(i+1,j+1)=0; end
        end
    end
end
source=double(int32((source.*[resolutionX
resolutionY])./size(mapOriginal)));
goal=double(int32((goal.*[resolutionX
resolutionY])./size(mapOriginal)));
set(handles.instruction,'string','A* Algorithm is
processing. Press CLEAR ALL to stop!','FontSize',12);

if ~feasiblePoint(source,map)
    errordlg('source lies on an obstacle or outside
map','Error');
    set(handles.instruction,'string',
'ERROR!','FontSize',12);
end

if ~feasiblePoint(goal,map)
    errordlg('goal lies on an obstacle or outside
map','Error');
    set(handles.instruction,'string',
'ERROR!','FontSize',12);
end
if length(find(conn==2))~=1, error('no robot specified
in connection matrix'); end

```



```

%**Part of these code is took from R. Kala(2014)*****%
%Code for Robot Path Planning using A* algorithm,
Indian Institute of Information Technology Allahabad
%structure of a node is taken as startY, startX,
historic cost, heuristic cost, total cost, parent
index in closed list (-1 for source)
Q=[source 0 heuristic(source,goal)
0+heuristic(source,goal) -1]; % the processing queue
of A* algorihtm, open list
closed=ones(size(map)); % the closed list taken as a
hash map. 1=not visited, 0=visited
closedList=[]; % the closed list taken as a list
pathFound=false;
tic
counter=0
colormap(gray(256));
grid;

set(0,'userdata',0)
while size(Q,1)>0
    pause(.01)
    if get(0,'userdata')
        return;
    end
    [A, I]=min(Q,[],1);
    n=Q(I(5),:) % smallest cost element to process
    Q=[Q(1:I(5)-2,:);Q(I(5)+1:end,:)]; % delete
element under processing
    if n(1)==goal(1) && n(2)==goal(2) % goal test
        pathFound=true;break;
    end
    if checkPath(n(1:2),newPos,map) %if path from
n to newPos is collission-free
        if closed(newPos(1),newPos(2))~=0 % not
already in closed

historicCost=n(3)+historic(n(1:2),newPos);
        heuristicCost=heuristic(newPos,goal);
        totalCost=historicCost+heuristicCost;
        add=true; % not already in queue with
better cost
        if length(find((Q(:,1)==newPos(1)) .*
(Q(:,2)==newPos(2))))>1
            I=find((Q(:,1)==newPos(1)) .*
(Q(:,2)==newPos(2)));
            if Q(I,5)<totalCost, add=false;
            else Q=[Q(1:I-1,:);

```

```

                Q(I+1:end,:);];
            add=true;
        end
    end
    if add
        Q=[Q;newPos historicCost
heuristicCost totalCost size(closedList,1)+1]; % add
new nodes in queue
    end
end
    end
    closed(n(1),n(2))=0;closedList=[closedList;n]; %
update closed lists
***Part of these code is took from R. Kala(2014)***%

    if display
        image((map==0).*0 + ((closed==0).*(map==1)).*125
+ ((closed==1).*(map==1)).*255);
        counter=counter+1;

text(source(2),source(1),strcat('S'),'color','b');
    text(n(2)-1,n(1),strcat('o'),'color','r');
    text(goal(2),goal(1),strcat('G'),'color','b');
    M(counter)=getframe;
end
end

if ~pathFound
    warndlg('no path found!')
end
if pathFound
    warndlg('Operation Done. Press LOAD PATH to
continue.');
```

```

    set(handles.instruction,'string','*Successful.
Press LOAD PATH to continue.','FontSize',12)
end
set(handles.edit8, 'String', toc);
```

```

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
set(handles.instruction,'String','');
global path; global pathLength; global prev; global n;
global closedList;
global mapOriginal; global resolutionX; global
resolutionY; global source; global goal;
global filename;
path=[n(1:2)]; %retrieve path from parent information
prev=n(6);
while prev>0
    path=[closedList(prev,1:2);path];
    prev=closedList(prev,6);
end
path=[(path(:,1)*size(mapOriginal,1))/resolutionX
(path(:,2)*size(mapOriginal,2))/resolutionY];
pathLength=0;
for i=1:length(path)-1,
pathLength=pathLength+historic(path(i,:),path(i+1,:));
end
fprintf('Path Length=%d \n\n',pathLength);
set(handles.edit7, 'String', pathLength);
axes(handles.axes1);
imshow(filename);
line(path(:,2),path(:,1),'linewidth',2,'color','r');
text(source(2)-
1,source(1),strcat('SOURCE'),'color','b');
text(goal(2),goal(1),strcat('GOAL'),'color','b');

% --- Executes on button press in capture.
function capture_Callback(hObject, eventdata, handles)
set(handles.instruction,'string','');
webcamlist
cam = webcam('Webcam C170')
cam.AvailableResolutions
cam.Resolution = '176x144'
preview(cam)
cam.Resolution = '320x240'
img = snapshot(cam);
axes(handles.axes1);
imshow(img)
imgname = sprintf('a_%s.bmp', datestr(now,'mm-dd-yyyy
HH-MM-SS'));
imwrite(img,imgname)

```

```

% --- Executes on button press in LoadImage.
function LoadImage_Callback(hObject, eventdata,
handles)
global filename;
set(handles.instruction,'string','');
cla(handles.axes1,'reset');
filename = uigetfile({'*.jpg;*.bmp;*.png;*.gif','All
Image Files';...
    '*.*','All Files' },'mytitle')
axes(handles.axes4);
imshow(filename);
set(handles.instruction,'string','*Please select your
SOURCE point.','FontSize',12);
[x,y] = ginput(1);
h1 = text(x,y,'S', ...
    'HorizontalAlignment','center', ...
    'Color',[1 0 0], ...
    'FontSize',8);
set(handles.startX,'String',round(x(1,1)));
set(handles.startY,'String',round(y(1,1)));
set(handles.instruction,'string','*Please select your
GOAL point.','FontSize',12);
[x,y] = ginput(1);
h2 = text(x,y,'G', ...
    'HorizontalAlignment','center', ...
    'Color',[1 0 0], ...
    'FontSize',8);
set(handles.goalX,'String',round(x(1,1)));
set(handles.goalY,'String',round(y(1,1)));
set(handles.instruction,'string','*Press PROCESS to
continue.','FontSize',12);

% --- Executes on button press in ClearAll.
function ClearAll_Callback(hObject, eventdata,
handles)
set(0,'userdata',1)
set(handles.instruction,'string','*Do you wish to
CAPTURE new map or LOAD MAP?','FontSize',12)
cla(handles.axes1,'reset');
cla(handles.axes4,'reset');
set(handles.startX,'String','');
set(handles.startY,'String','');
set(handles.goalX,'String','');
set(handles.goalY,'String','');
set(handles.edit7,'String','');
set(handles.edit8,'String','');

```

```
function edit7_Callback(hObject, eventdata, handles)
global path; global pathLength;
for i=1:length(path)-1,
pathLength=pathLength+historic(path(i,:),path(i+1,:));
end
fprintf(pathLength);
```