OPTIMAL PATH PLANNING ALGORITHM FOR SWARM ROBOTS USING BAT ALGORITHM WITH MUTATION (BAM)

LIM PEI YEE

Bachelor Of Electrical Engineering
(Electronics) with Honours

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	LIM PEI YEE
Date of Birth	:	28 SEPTEMBER 1997
Title	:	OPTIMAL PATH PLANNING ALGORITHM FOR SWARM OF ROBOTS USING BAT ALGORITHM WITH MUTATION (BAM)
Academic Session	:	2020/2021 II

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:

[illegible] [illegible]

(Student's Signature)	(Supervisor's Signature)
970928-03-6578	DR. DWI PEBRIANTI
New IC/Passport Number	Name of Supervisor
Date: 11 FEBRUARY 2022	Date: 11 FEBRUARY 2022

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

THESIS DECLARATION LETTER (OPTIONAL)

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
Thesis Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

[illegible]

(Supervisor's Signature)

Date: 11 FEBRUARY 2022

Stamp: DR. DWI PEBRIANTI, ENG. TECH.
SENIOR LECTURER
COLLEGE OF ENGINEERING
UNIVERSITI MALAYSIA PAHANG
LEBUHRAYA TUN RAZAK
26300 GAMBANG, KUANTAN, PAHANG
TEL : +609-549 2239 FAX : +09-549 2689

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

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Electrical Engineering (Electronics) with Honours.

[illegible]

____

(Supervisor's Signature)

Full Name	: DR. DWI PEBRIANTI
Position	: SENIOR LECTURER
Date	: 11 FEBRUARY 2022

____

(Co-supervisor's Signature)

Full Name	:
Position	:
Date	:

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

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.

[illegible]

(Student's Signature)

Full Name	: LIM PEI YEE
ID Number	: EA18002
Date	: 11 FEBRUARY 2022

OPTIMAL PATH PLANNING ALGORITHM FOR SWARM ROBOTS USING BAT ALGORITHM WITH MUTATION (BAM)

LIM PEI YEE

Thesis submitted in fulfillment of the requirements
for the award of the Bachelor of
Electrical Engineering (Electronics) with Honours

College of Engineering
Department of Electrical & Electronics Engineering
UNIVERSITI MALAYSIA PAHANG

FEBRUARY 2022

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Dr. Dwi Pebrianti for the continuous support throughout my whole project, for her patience, motivation, enthusiasm and immerse knowledge. Her guidance helped me in all the time of the project and the writing of this thesis.

Besides, I would like to thank my fellow coursemates for the suggestions, opinions and help given to me. It is very helpful especially during the bottleneck of my project.

Last but not least, I would like to thank my family and my friend, Own Shan Ju for the support and motivation. Thank you for spending time to listen to me whenever I need to speak and having faith in me that I would be able to complete this project.

ABSTRAK

Dalam navigasi robot, perancangan laluan sentiasa menjadi masalah paling penting di mana robot sepatutnya boleh bergerak dari kedudukan permulaan ke kedudukan gol tanpa sebarang halangan. Ini kerana robot tidak dapat merancang laluan optimum dalam persekitaran yang diketahui dan halangan yang ada dalam persekitaran tersebut meningkatkan kesukaran untuk robot bergerak mengikut laluan yang dirancang dalam persekitaran. Pemyelidikan semasa dalam navigasi robot adalah untuk melaksanakan algoritma pengelakan halangan kepada robot mudah alih untuk merealisasikan perancangan laluan robot mudah alih. Walau bagaimanapun, masih terdapat ruang untuk penambahbaikan seperti melaksanakan algoritma pengelakan halangan ke dalam robot swarm. Objektif kajian ini adalah untuk mencadangkan 'Bat Algorithm with Mutation (BAM)' bagi menyelesaikan masalah pengelakan halangan robot mudah alih. Projek ini disiapkan dengan mencipta robot mudah alih beroda di mana robot itu menggunakan pengawal P. Seterusnya, robot dilatih untuk bergerak dari satu titik ke titik lain dan dimasukkan ke dalam persekitaran maya dengan halangan static. Algoritma pengelakan halangan kemudiannya dilaksanakan kepada robot. Akhir sekali, ia dapat dilihat bahawa robot tersebut mampu bergerak dlam laluan yang dirancang tanpa bertembung dengan halangan yang ada di dalam persekitaran.

ABSTRACT

In robot navigation, path planning is always the most crucial problem where robots should be able to move from starting position to goal position without colliding into any obstacle. This is because robot is unable to plan an optimum path in a known situation and obstacles available increases the difficulty for robot to move according to the planned path in an environment. The current research in robot navigation is to implement an obstacle avoidance algorithm to a single mobile robot to realize the path planning of a mobile robot. However, there is still room for improvement such as implementing the obstacle avoidance algorithm into swarm robot. The objective of this study is to propose Bat Algorithm with Mutation (BAM) for solving the problem of obstacle avoidance of mobile robots. This project is completed by creating a wheeled mobile robot where the robot uses a P controller. Next, robot is trained to travel from one point to another point and inserted into a virtual environment with static obstacle. The obstacle avoidance algorithm is then implemented to the robot. Lastly, it can be seen that the robot is able to move in the planned path without colliding with the obstacle in the environment.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BAM	Bat Algorithm with Mutation
BA	Bat Algorithm
DE	Differential Evolution
PID	Proportional-Integral-Derivative
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization
AS	Ant System
GA	Genetic Algorithm
WLAN	Wireless Local Area Network
EA	Evolutionary Algorithm
WMR	Wheeled Mobile Robot
MISO	Multiple Input Single Output

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Project Background

Path-planning is a basic feature of autonomous mobile robots that allows them to find the shortest – or otherwise best – path between two points. Otherwise, ideal paths may be those that minimise the amount of turning, accelerating, or anything else a particular application necessitates.

Mobile robots take more and more parts in human's daily life. Humans tend to rely on robots more as robots do ease human's life. In some condition, robots help to perform task better and safer than human, for example environment monitoring in an earthquake disaster site or nuclear disaster site. In order for the robots to able to perform the assigned tasks, they have to know their locations within the environment and comprehend what the world around them looks like. Even if robots have the information about the environment, they still need to choose an optimal path from the starting point to the destination point to achieve their target. [3] In this project, Bat Algorithm with Mutation (BAM) is chosen to be implement on the swarm robots.

Bat Algorithm with Mutation (BAM) is a variant of the Bat Algorithm (BA) that incorporates a mutation operator. The presence of loudness and pulse emission rate values in the updating value of BA parameter's equations distinguishes BA from other swarm intelligence optimization techniques. The implementation of BA as an optimization algorithm was motivated by the design of micro bats, which communicate using the echolocation principle.

Bat algorithm (BA) relies solely on random walks for its search, therefore a quick convergence cannot be assured. Hence, a major change is done to the mutation operator of BA, including two smaller changes with the intention of speeding up the convergence, making the approach is more feasible for a wide range of applications without losing the characteristics of the original technique. The first modification is using a fixed frequency and loudness in BAM compared with using various frequencies and loudness in BA. Next modification is adding a mutation operator in the algorithm as an attempt to increase population variety in order to improve search efficiency and accelerate the convergence to the best solution. By implementing BAM in swarm robots, robots are able to compute the best solution by generating a new solution continuously after a solution is generated.

1.2 Problem Statement

Path or route planning is defined as a problem that an agent moves from starting to end point by avoiding obstacles in order to reach the target at minimum cost. [3] Therefore, path planning is one of the most crucial research problems in robotics as many problems in various field can be solved by proposing path planning, such as rescue operations of natural disaster.

Obstacles available increases the difficulty for robot to move according to the planned path in an environment. In case of mobile swarm robots to manoeuvre autonomously in a known or unknown environment collision avoidance includes avoiding collision with random environmental elements and other robots. [1]

1.3 Objectives

1. To develop a system that a robot is able to manoeuvre autonomously in a known environment.
2. To develop a robot that is equipped with optimal path planning algorithm.

1.4 Scope of Project

1. This project will only be done in simulation.
2. Mobile robots will implement the obstacle avoidance algorithm in a known environment

3. Only fixed obstacle is present in the known environment.

1.5 Thesis Layout

There are five chapters available in this thesis. Chapter 1 will be discussing about the project background, problem statement, objectives of the projects and scope of project.

Chapter 2 discuss about the literature review on optimal path planning, swarm robots and Bat Algorithm with Mutation. Chapter 3 introduce the methodology of the project including the mobile robot construction, environment creation, the obstacle avoidance algorithm and the details of the system.

Chapter 4 review on the result obtained after simulating the system and Chapter 5 is the conclusion of the project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter discussed on the literature review on the keypoints of the project, which are optimal path planning, swarm robots and Bat Algorithm with Mutation (BAM).

2.2 Optimal Path Planning

Path planning [3] is defined as a problem that an agent (robot) moves from starting point to the ending point by avoiding obstacles in order to reach target at the minimum cost. The cost to find the optimal path varies under different circumstances (time, distance, energy etc). There are a few types of algorithm used for optimal path planning, such as Particle Swarm Optimization Technique, Ant Colony Optimization and Modified Genetic Algorithm.

2.2.1 Particle Swarm Optimization Technique (PSO)

PSO is a stochastic population-based search method inspired by animal social behaviour like bird flocks and fish schools. It aims to find the global optima by seeking the best optima among the group repeatedly. Each person in PSO is referred to as a particle. In a swarm, each particle adjusts its location based on two factors: its personal optimal position and the swarm's overall best position.

PSO starts with particles that are randomly scattered throughout the problem space. Within the range (Vmin – Vmax), the particles have a velocity in a random direction. Following this distribution, each particle assesses its fitness based on its position, with the goal of determining which particle has the best position (Leader of the swarm). The algorithm now iterates over movement steps, with each movement particle updating its velocity using equation (2.1), which is based on the personal best position and the global best position.

$$V_{i+1} = w \text{ x } V_i + c_1 \text{ x } r_1 \text{ x } (P_i - X_i) + c_2 \text{ x } r_2 \text{ x } (P_g - X_i) \tag{2.1}$$

$$X_{i+1} = X_i + V_{i+1} \tag{2.2}$$

where vi and xi are particle i's velocity and position vectors, Pi is particle i's best local position, and Pg is the best global position discovered in the whole population. The two parameters c1 and c2 are positive constants known as learning factors; c1 represents how strongly a particle is drawn to its best location, while c2 represents the global position. The weight w determines how dependent the current velocity is on the prior velocity. The stochastic part of the procedure is provided by the uniform random variables r1 and r2. After determining the velocity, particles update their location. The equation updates the position of the particles (2.2). [1]

Both the PSO and the collision detection operate in a single loop. PSO starts with a random variable and then moves on to the swarm approach after a few iterations. After calculating the PSO velocity and location, the collision from a dynamic obstruction is estimated. In order to detect a static barrier, eight infrared sensors are placed around the robot's torso, as illustrated in Fig.4. The sensor data are checked in each iteration, and if the sensors detect a static obstacle, the sub-routine for static obstacle avoidance is repeated until the impediment is avoided. Then the regular PSO resumes. Figure 2-1 depicts a flowchart of the entire system.

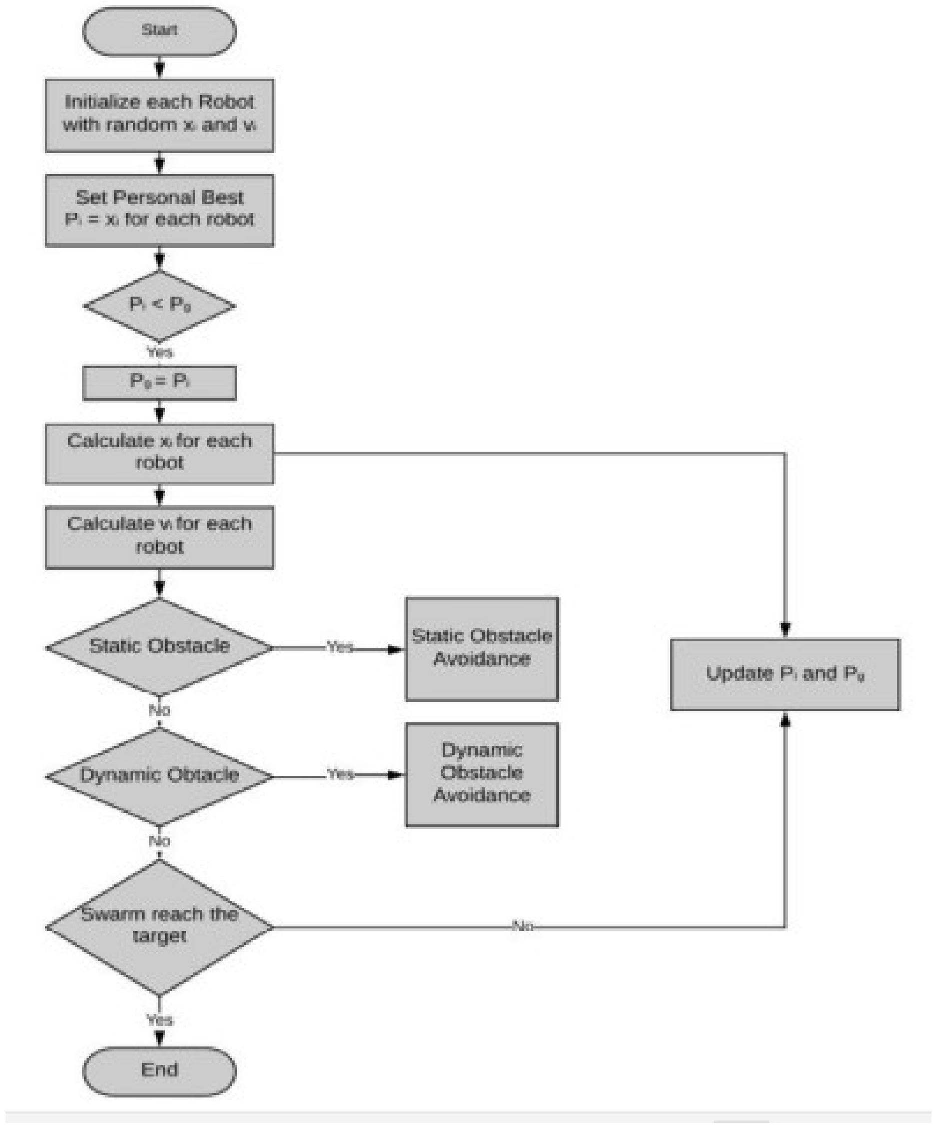

Start
Initialize each Robot with random x_i and v_i
Set Personal Best $P_i = x_i$ for each robot
$P_i < P_g$
Yes
$P_g = P_i$
Calculate x_i for each robot
Calculate v_i for each robot
Static Obstacle
Yes
Static Obstacle Avoidance
Update P_i and P_g
No
Dynamic Obtacle
Yes
Dynamic Obstacle Avoidance
No
Swarm reach the target
No
Yes
End

Figure 2-1 Particle Swarm Optimization Flow Chart

2.2.2 Ant Colony Optimization

State transition rules, local update rules, and global update rules are the three main rules that govern the design of AS for robot route planning. After each building stage, all of the ants do a local pheromone update. It is only applied by each ant to the last edge traversed:

$$\tau_{ij} = (1 - \phi) \cdot \tau_{ij} + \phi \cdot \tau_0 \qquad (2.3)$$

where $\phi \in (0,1]$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone.

At first, ants would use state transition rules based on heuristics and pheromones set down by the ants to decide the next node to be visited.

The ants with a greater likelihood of choosing that node will acquire an accurate distance estimate using a heuristic equation and a larger amount of pheromone from that node. With these criteria in place, ants may strike a balance between exploration and exploitation using the relatives' coefficient.

Using the update local rules formula, the pheromone will be lowered locally by the provided evaporation rate while the route is being built. After all of the ants have completed the road to the target, the global updating procedure is used, in which the ants deposit their pheromone based on the path distance, as shown in equation (2.4) below.

$$\tau_{ij} \leftarrow \begin{cases} (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} & \mathit{if}\ (i,j) \mathit{belongs\ to\ the\ best\ tour,} \\ \tau_{ij} & \mathit{otherwise} \end{cases} \qquad (2.4)$$

The pheromone concentration will be adjusted until it draws more ants from the following generation to take the shorter route. Finally, as illustrated in Fig. 2-2, the ideal robot path is discovered utilising the notion of ant behaviour. [7]

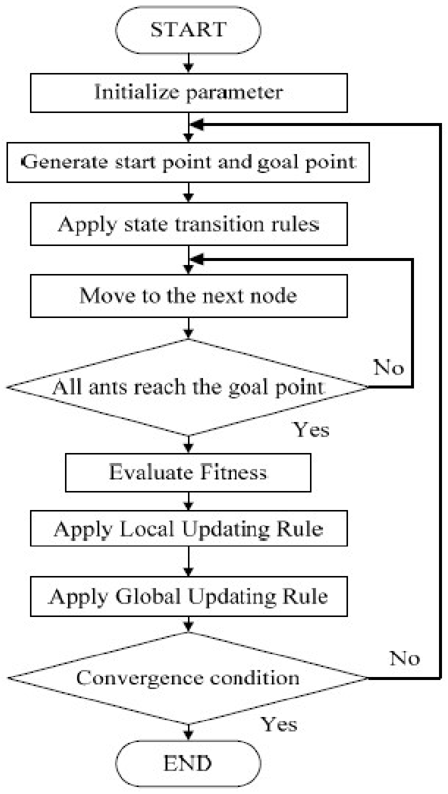

START
Initialize parameter
Generate start point and goal point
Apply state transition rules
Move to the next node
All ants reach the goal point
No
Yes
Evaluate Fitness
Apply Local Updating Rule
Apply Global Updating Rule
Convergence condition
No
Yes
END

Figure 2-2 Ant Colony Optimization Algorithm Flowchart

2.2.3 Modified Genetic Algorithm

The Genetic Algorithm's key benefit is its excellent search quality[5]. However, there are several significant drawbacks to this strategy. The original population may have impassable pathways, and the following generation may face the same problem, converting the search environment to a graph[9]. Initialization, fitness function, Selection, and other elements should be defined based on the original genetic algorithm.

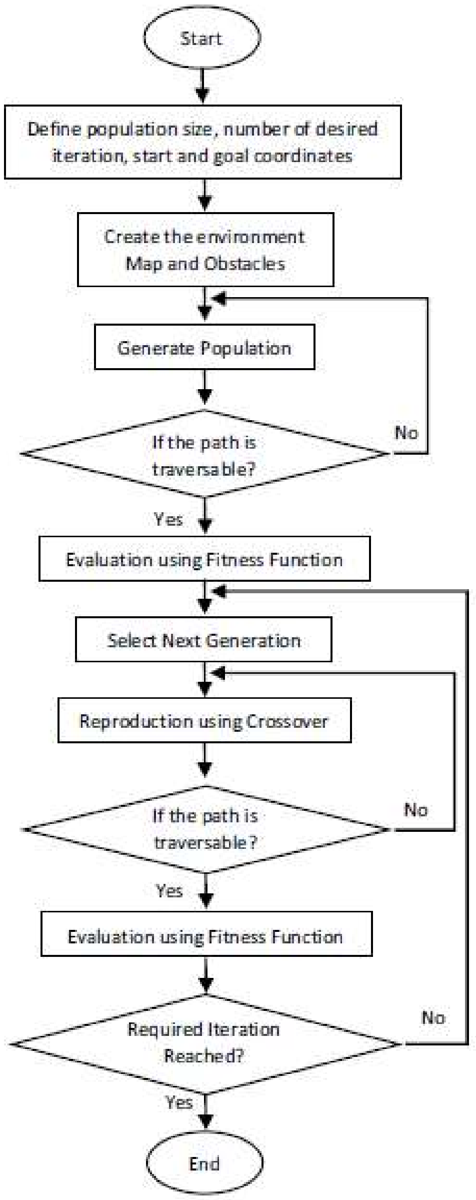

Start
Define population size, number of desired iteration, start and goal coordinates
Create the environment Map and Obstacles
Generate Population
If the path is traversable?
No
Yes
Evaluation using Fitness Function
Select Next Generation
Reproduction using Crossover
If the path is traversable?
No
Yes
Evaluation using Fitness Function
Required Iteration Reached?
No
Yes
End

Figure 2-3 Modified Genetic Algortihm Flowchart

The suggested algorithm's flowchart is shown in the diagram above. The initial population comprises the number of chromosomes to be generated in the first stage (initialization). Individual GA chromosomes, or chromosomes, are solutions to the current situation. In reality, each individual in the environment represents a path between the start and destination produced by the path planning algorithm. After startup, the population may be represented as follows:

$$p_1, p_2, p_3, p_4, \ldots, p_n \tag{2.5}$$

Each individual could have any of the following coordinates which can be shown in the form of:

$$c_1, c_2, c_3, c_4, \ldots, c_n \tag{2.6}$$

where c1 denotes the starting place and cn denotes the destination. The fitness function's job is to determine the length of each chromosome's route. As a result, the superior outcome has a shorter route length. In the genetic algorithm approach, the fitness function is extremely significant. The following formula is used to determine the distance of each explorable path:

$$L_{(i)} = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{2.7}$$

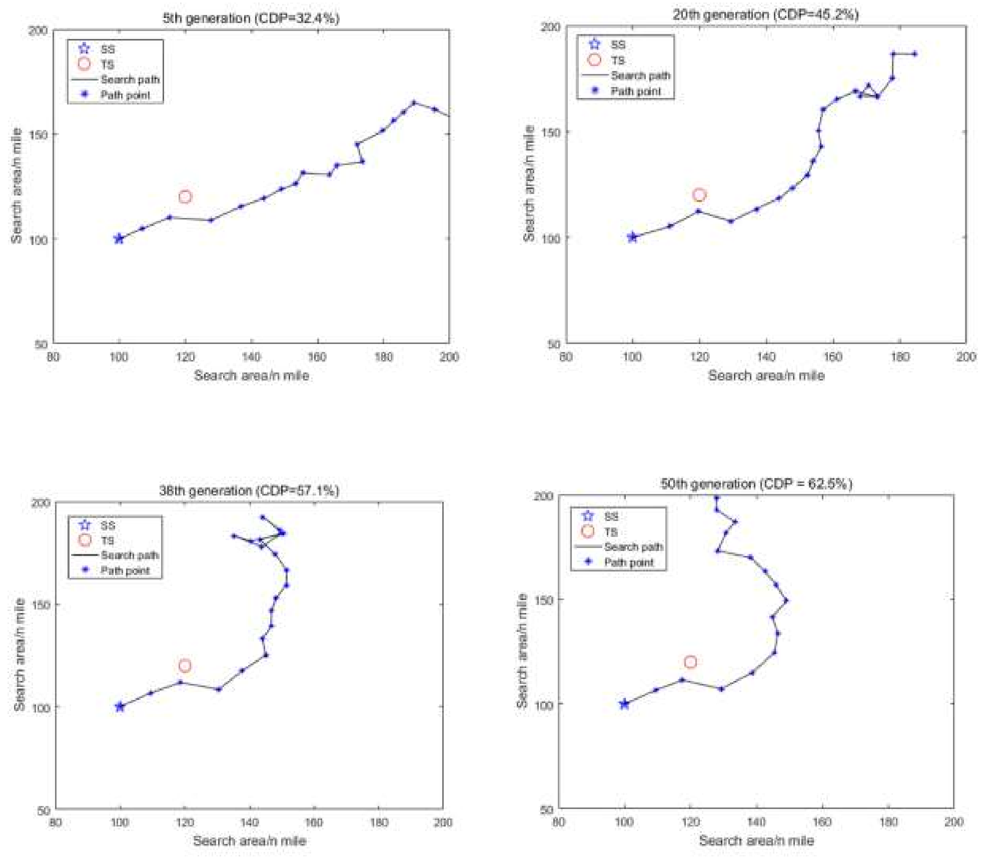
where *xi, yi* are the robots current coordinates and the $x(i-1)$, $y(i-1)$ its robot's previous position. The length of the whole path can be computed by this equation:

$$\text{Length} = \sum_{i=1}^{n} L_{(i)} \tag{2.8}$$

When n is the number of cells in the selected path and L(i) denotes the distance covered by the n cells, the Length will be the exact length of the whole path. The entire route length will be determined by the fitness function's findings, and the best chromosome will be the one with the lowest fitness value.

The viable path for the robot will be examined in the selection stage of the algorithm, which means this function will examine the created path (individually) to see if all of the cells in the path are traversable or if there is an obstacle in the robot's path. If the robot cannot follow the course, the person will be eliminated from the population, and a new population will be created to replace the destroyed one.

At the conclusion of the day, everyone's fitness would be updated. Individuals with lower fitness values have a higher likelihood of appearing in the following generation. To guarantee that the chromosome with the lowest fitness does not undergo any mutations or crossovers, this person will skip the crossover function and proceed directly to the next generation. The best member of each generation will always be preserved, and the performance of the Genetic Algorithm will increase. After that, the leftovers will be sorted according to their fitness value and then sent via the crossover function. The algorithm creates a number and assigns it to each person depending on their fitness value in the crossover. Candidates for the crossover function are selected based on the allotted number. The crossover function will then establish a new generation and the kid that will replace the parent chromosome. All newly formed chromosomes will be checked for traversal to ensure that none of the newly created members have an infeasible route. If a person is discovered to be infeasible, it will be removed from the group, and the crossover function will be used to replace them.



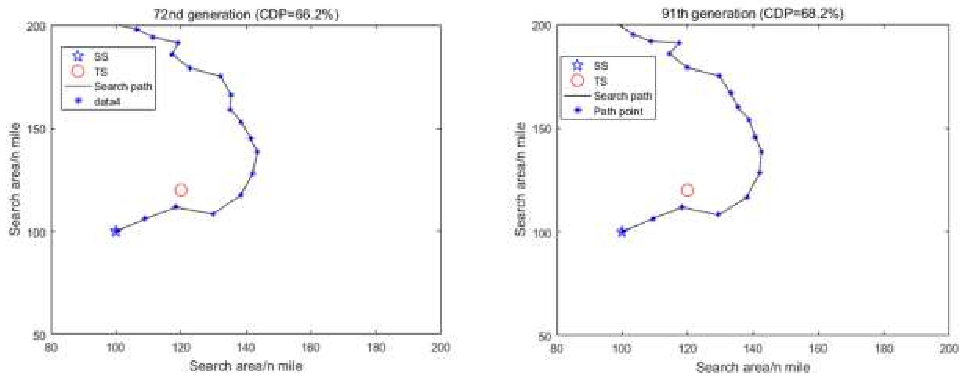


Figure 2-4 Optimal search path of selected generation

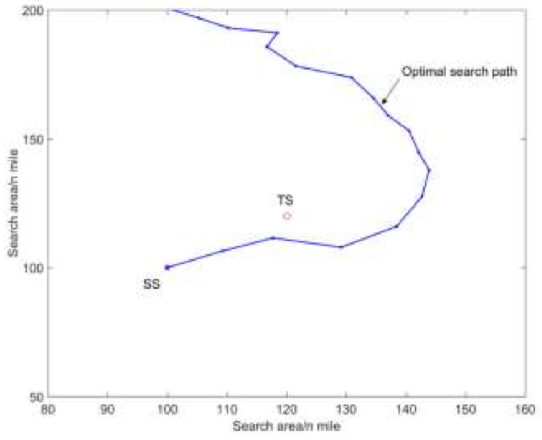


Figure 2-5 Optimal search path calculated by MGA

2.3 Swarm Robots

Swarm robotics[10] is a concept based on the social behaviour of animals or insects to develop a robust robotics system utilising vast numbers of similar robots. Individual robot interactions and robot interactions with the environment are the sources of collective robot behaviour [11]. It is simple to do activities that are difficult to complete with a single robot using this strategy. Sensor technology, motor technology, power supply technology, telecommunications technology, control technology, and artificial intelligence technology are all being researched for robotics. Self-organization and emergence are terms used in swarm robotics to describe cooperative task solving capabilities. Self-organization refers to swarm organisation that emerges from the system itself, whereas emergence refers to the necessity for local contact between individual robots that emerges in a decentralised manner[12]. Different coordination mechanisms, such as job allocation, self-configuration, and pattern creation, have been documented for regulating individual robot motions. Instead of focusing on a single robot system, researchers are investigating the coordination of multi-robot/swarm systems, as multi-robot systems have various benefits and applications. Efficiency, flexibility, fault-tolerance, scalability, and so on are examples. Environmental monitoring, surveillance, dispersed sensing tasks, oil cleanup, underwater localisation, and many more applications are all possible with a multi robot system. The purpose of a sensing system is to detect and measure the existence of items. Neighboring robots, barriers, and targets are examples of objects. The technical challenge is to create and deploy actual mobile robots at an affordable price.

Communication between robots is necessary for the interaction of many robots to carry out specified tasks, such as one robot delivering commands or updates to other robots [13]. It is now feasible to connect one gadget to another thanks to advancements in wireless communication technologies. The benefit of robot communication is that the work may be completed more quickly. For communication between sensors and electrical equipment, WLAN, which is based on IEEE 802.11 standards, and WPAN, which uses technologies such as infrared, wireless USB, Bluetooth, and ZigBee, is used.

2.3.1 Swarm Intelligence

Complexity theory[14] tries to explain how seemingly basic individual creatures may act intelligently as a group. Although there is no formal definition of complexity, the overall concept is that a single unit can generate a large number of nonlinear connections with its surroundings. Foragers in the animal realm, such as bees and ants, communicate with one another in the quest for food, resulting in emergence, a magnificent show of teamwork. Individual insect behaviour is extremely basic, despite the enormity of this collective survival plan. Emergence is a system's behaviour in which order is established by a self-organizing unit capable of adjusting to changes in its surroundings through feedback (e.g., an ant colony searching for food). Swarm intelligence is an emergent idea in which, while individual individuals exhibit basic behaviour, as a collective, they demonstrate intelligence capable of solving difficult issues.

2.4 Bat Algorithm with Mutation

2.4.1 Bat Algorithm

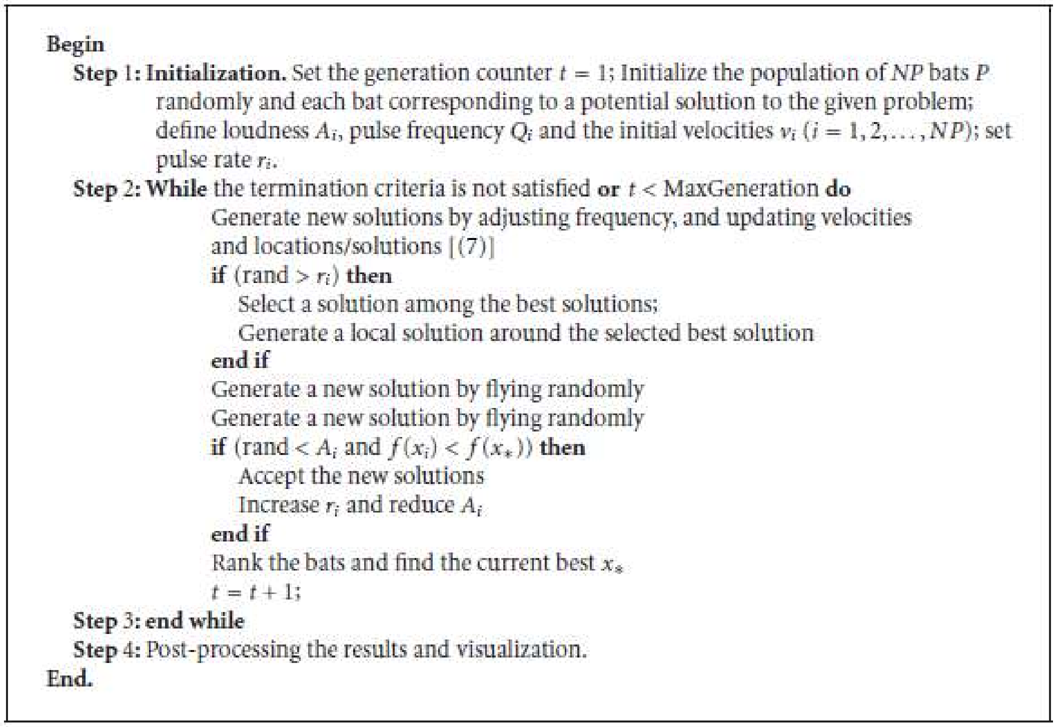
Bat algorithm[15] is based on idealizing some of the echolocation characteristics [20] of bats, which are following approximate or idealized rules.

(1) All bats apply echolocation to sense distance, and they always “know” the surroundings in some magical way.

(2) Bats fly randomly with velocity vi and a fixed frequency $f$min at position xi, varying wavelength λ, and loudness A0 to hunt for prey. They can spontaneously accommodate the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target.

(3) Although the loudness can change in different ways, it is supposed that the loudness varies from a minimum constant (positive) Amin to a large A0.

Based on these approximations and idealization, the basic steps of the bat algorithm (BA) can be described as shown in Algorithm 1.



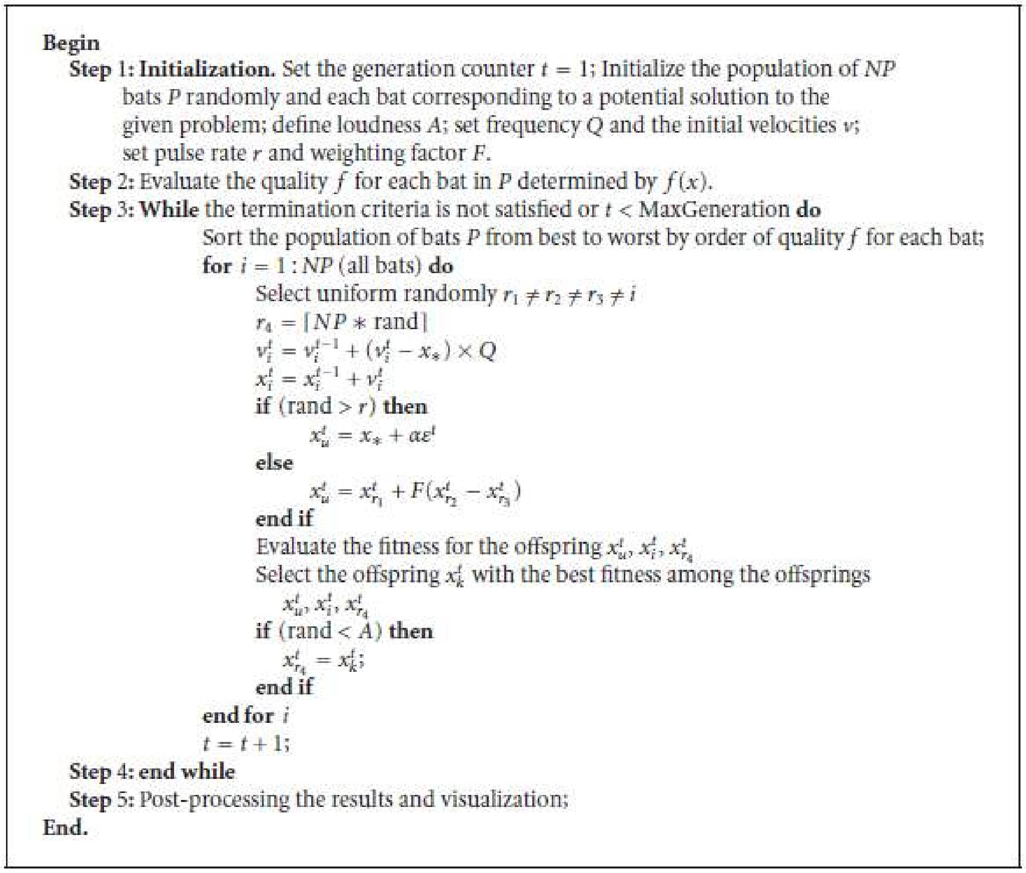
```
Begin
  Step 1: Initialization. Set the generation counter $t = 1$; Initialize the population of $NP$ bats $P$
          randomly and each bat corresponding to a potential solution to the given problem;
          define loudness $A_i$, pulse frequency $Q_i$ and the initial velocities $v_i$ $(i = 1, 2, \ldots, NP)$; set
          pulse rate $r_i$.
  Step 2: While the termination criteria is not satisfied or $t <$ MaxGeneration do
              Generate new solutions by adjusting frequency, and updating velocities
              and locations/solutions [(7)]
              if (rand $> r_i$) then
                  Select a solution among the best solutions;
                  Generate a local solution around the selected best solution
              end if
              Generate a new solution by flying randomly
              Generate a new solution by flying randomly
              if (rand $< A_i$ and $f(x_i) < f(x_*)$) then
                  Accept the new solutions
                  Increase $r_i$ and reduce $A_i$
              end if
              Rank the bats and find the current best $x_*$
              $t = t + 1$;
  Step 3: end while
  Step 4: Post-processing the results and visualization.
End.
```

Figure 2-6 Bat Algorithm

2.4.2 Bat Algorithm with Mutation

Storn and Price invented the Differential Evolution (DE) method, which is a basic Evolutionary Algorithm (EA) that develops new candidate solutions by merging the parent individual with a few additional members of the same population. A candidate can only take the place of a parent if it is physically fitter. This is a fairly opportunistic selection strategy that frequently outperforms typical EAs. DE has several advantages, including ease of implementation, simplicity of structure, speed, and robustness[15].

In most circumstances, DE is capable of searching worldwide and quickly locating the global ideal value. At the exploitation stage, however, it is difficult to develop the solutions further. Standard BA, on the other hand, is not very good in exploring the answer. As a result, the BA[16] had undergone a mutation. The distinction between BAM and BA is that instead of using the random walk in basic BA, the mutation operator is utilised to enhance the original BA, creating new solutions for each bat with a probability 1 –r. In this approach, BAM searches the space both globally and locally, based on the DE and BA mutations, respectively. DE and BA[16] can be completely used in this method.



```
Begin
  Step 1: Initialization. Set the generation counter $t = 1$; Initialize the population of $NP$
      bats $P$ randomly and each bat corresponding to a potential solution to the
      given problem; define loudness $A$; set frequency $Q$ and the initial velocities $v$;
      set pulse rate $r$ and weighting factor $F$.
  Step 2: Evaluate the quality $f$ for each bat in $P$ determined by $f(x)$.
  Step 3: While the termination criteria is not satisfied or $t <$ MaxGeneration do
          Sort the population of bats $P$ from best to worst by order of quality $f$ for each bat;
          for $i = 1 : NP$ (all bats) do
              Select uniform randomly $r_1 \neq r_2 \neq r_3 \neq i$
              $r_4 = \lfloor NP * \text{rand} \rfloor$
              $v_i^t = v_i^{t-1} + (v_i^t - x_*) \times Q$
              $x_i^t = x_i^{t-1} + v_i^t$
              if (rand $> r$) then
                  $x_u^t = x_* + \alpha\varepsilon^t$
              else
                  $x_u^t = x_{r_1}^t + F(x_{r_2}^t - x_{r_3}^t)$
              end if
              Evaluate the fitness for the offspring $x_u^t, x_i^t, x_{r_4}^t$
              Select the offspring $x_k^t$ with the best fitness among the offsprings
                $x_u^t, x_i^t, x_{r_4}^t$
              if (rand $< A$) then
                $x_{r_4}^t = x_k^t$;
              end if
          end for $i$
          $t = t + 1$;
  Step 4: end while
  Step 5: Post-processing the results and visualization;
End.
```

Figure 2-7 Bat Algorithm with Mutation

2.5 Summary

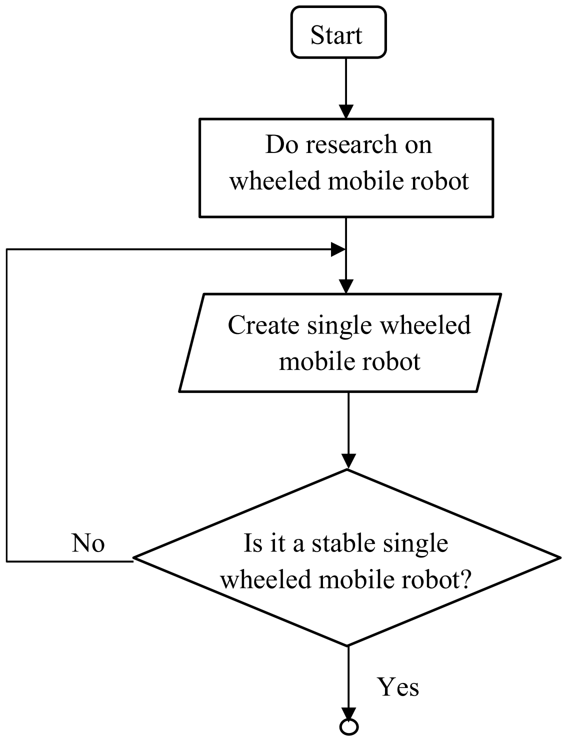
To summarize this chapter, I had reviewed on some of the optimal path planning algorithm used for mobile robots to manoeuvre autonomously in a known environment More details on the methodology to achieve the objectives will be discussed in chapter 3.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter includes a detailed explanation on the methodology used to ensure the project achieved the objective of the project. Figure 3-1 below shows the flowchart of the project.


Start
Do research on wheeled mobile robot
Create single wheeled mobile robot
Is it a stable single wheeled mobile robot?
No
Yes

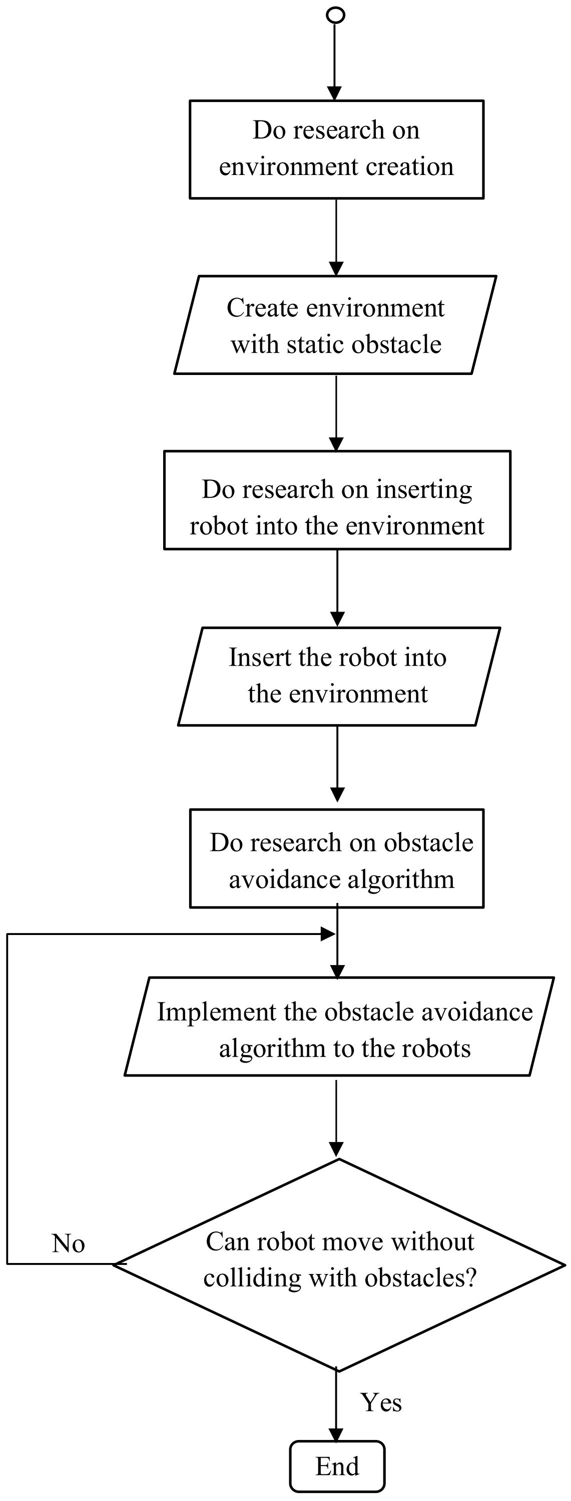

Do research on environment creation
Create environment with static obstacle
Do research on inserting robot into the environment
Insert the robot into the environment
Do research on obstacle avoidance algorithm
Implement the obstacle avoidance algorithm to the robots
Can robot move without colliding with obstacles?
No
Yes
End

Figure 3-1 Flowchart of the project

3.2 Single Wheeled Mobile Robot Model

A single wheeled mobile robot (WMR) model is created in MATLAB Simulink. It is consists of the few blocks such as, input source, controllers, WMR block and output scope for linear velocity and angular velocity.

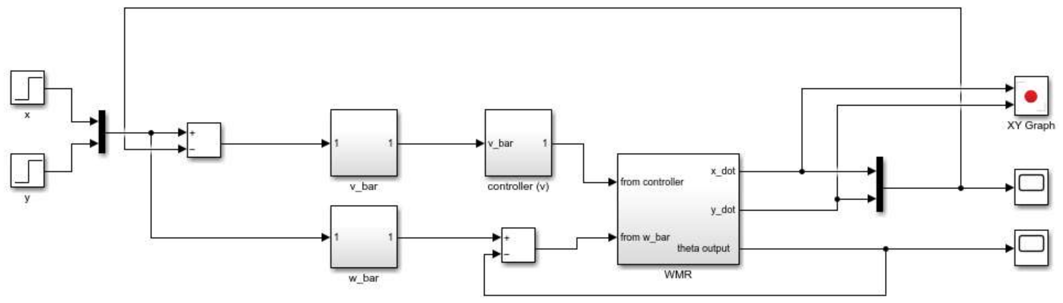


Figure 3-2 Single Wheeled Mobile Robot Block Diagram

I had chosen step input as the input source of the robot. The input source will then added up to enter the v_bar and w_bar block. The v_bar block provides modified error signal for linear velocity controller while w_bar block provides modified error signal for angular speed controller. The details of these two blocks will be discussed in subchapter.

After the signal is processed by the controllers, the signal will enter the kinematic model of the robot to determine the robot motion such as the left (wL) and right (wR) angular velocity. The details of the Kinematic model will be discussed in next subtopic.

As this wheel mobile robot is a closed loop system, both left and right angular velocity and the orientation angle will be feedback to the system.

3.2.1 Kinematic Model

A differential drive wheeled mobile robots is used in the project. This robot consists of 2 wheels. This wheeled mobile robot [17] is configured by three generalized coordinates, as written in Eq. (3.1), which are *x*, *y* and *θ*. *x* and *y* is the position of robot and *θ* is the orientation angle of the robot.

$$Q = (x, y, \theta) \qquad (3.1)$$

The movement of this wheeled mobile robot is determined by the left wheel angular velocity ω_l, right wheel angular velocity ω_r and also the orientation angle of the robot. The mathematical model for differential drive wheeled mobile robot is derived as follow:

$$\dot{x} = \frac{r}{2} (\omega_r + \omega_l) \cos \theta \qquad (3.2)$$

$$\dot{y} = \frac{r}{2} (\omega_r + \omega_l) \sin \theta \qquad (3.3)$$

$$\theta = \frac{r}{D} (\omega_r - \omega_l) \qquad (3.4)$$

which r is the radius of the robot's wheel and D is the distance between the robot's wheel. To implement the mathematical model of the differential drive wheeled mobile robot, a Unicycle model with translational and angular velocity of robot *v* and *ω* is implemented as the equation below:

$$\dot{x} = v \cos \theta \qquad (3.5)$$

$$\dot{y} = v \sin \theta \qquad (3.6)$$

$$\theta = \omega \qquad (3.7)$$

where $\dot{x}$ and $\dot{y}$ are the velocity on x and y direction as shown in figure 3-4 and figure 3-5 and θ is the angular rate of the wheel. By rearranging the inputs from Eq. (3.2) - Eq. (3.7), we can get the value of left and right angular wheel velocity, ω_l and ω_r as follow:

$$\omega_l = \frac{2v - \omega D}{2r} \qquad (3.8)$$

$$\omega_r = \frac{2v + \omega D}{2r} \qquad (3.9)$$

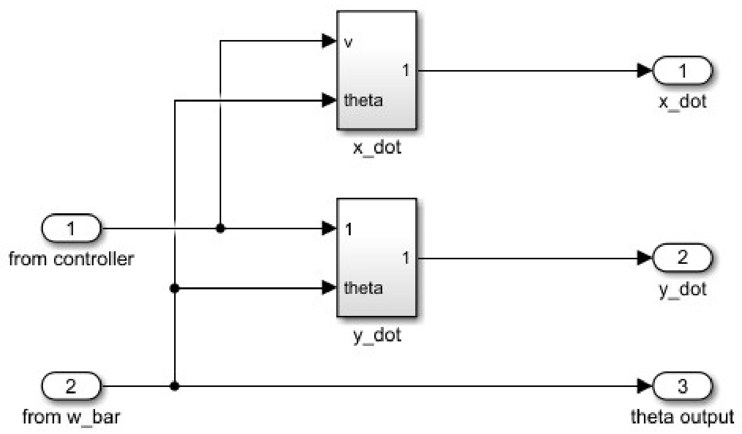


Figure 3-3 Interior structure of the kinematics model in which the outputs are velocity at x and y direction, and the angular rate of the wheel

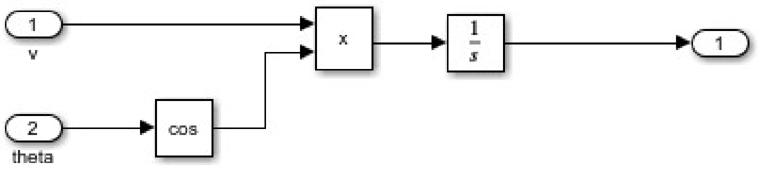


Figure 3-4 Structure to produce velocity at x direction

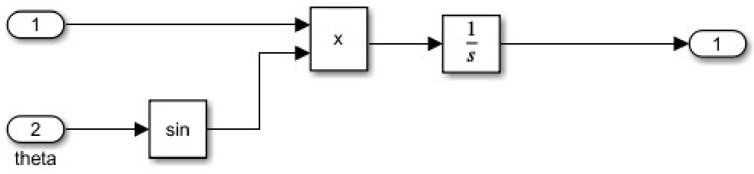


Figure 3-5 Structure to produce velocity at y direction

3.3 Controller of robot using PID Controller

The controller used to control the movement of the robots is PID controller with the inputs of velocity and angular speed. However, only velocity input will be manipulated, while angular speed input will remain the same.

The velocity of WMR [17] contain velocity along x-axis, v_x and y-axis, v_y as written in Eq. (3.10).

$$v = \sqrt{v_x^2 + v_y^2} \tag{3.10}$$

The paths x and y are set to a set of desired input values. To create an error signal, these numbers will be compared to the actual values. However, an error must be added to the velocity and angular speed inputs in order for the system to accept them. As stated in Eqs. (3.11) and (3.12), two distinct functions are devised to translate these error values into the two inputs of the WMR system [17].

$$\tilde{v} = \sqrt{(x_{desire} - x_{actual})^2 + (y_{desire} - y_{actua}\ )^2} \tag{3.11}$$

$$\tilde{w} = \text{atan2}\ ((y_{desire} - y_{actual}),(x_{desire} - x_{actual})) \tag{3.12}$$

where $\tilde{v}$ is modified error signal for velocity controller, $\tilde{w}$ is modified error signal for angular speed controller, *xdesire* is target of x position, *ydesire* is target of y position, *xactual* is actual value of *x* position, *yactual* is actual value of *y* position, and *atan*2 is four-quadrant inverse tangent, where it will return values in the closed interval [−π, π].

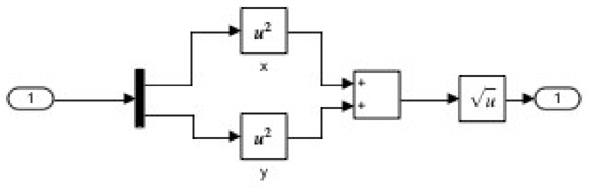


Figure 3-6 Structure of modified error signal for velocity controller

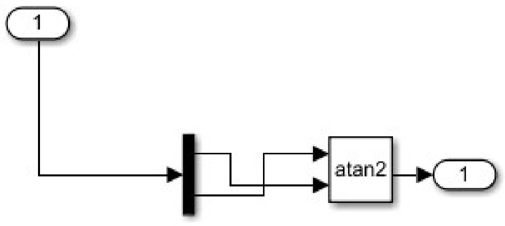


Figure 3-7 Structure of modified error signal for angular speed controller

PID controller with the inputs of velocity and angular speed used in this project is as shown in Eq. (3.13) below.

$$V = K_p\,\tilde{v} + K_i \int \tilde{v}\ \mathrm{dt} + K_d \frac{d\tilde{v}}{dt} \qquad (3.13)$$

where K_p is proportional gain, K_i is integral gain and K_d is differential gain. The system is using Proportional controller (P controller) only and K_i and K_d are equal to zero, which means this PID controller will use all of the gain in the controller.

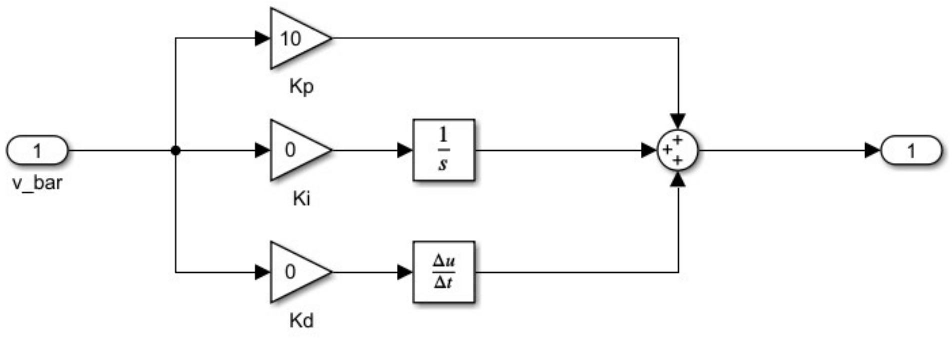


Figure 3-8 P controller only will be used in this system

3.4 Obstacle Avoidance Algorithm for a single mobile robot

Before creating the obstacle avoidance algorithm, an environment with static obstacle is created to insert the mobile robot into it.

3.4.1 Environment Creation

An environment with static obstacle is created by using Simulation Map Generator under the Apps tab in MATLAB.

Steps to create a simulation map:

1. Prepare a map image and load into the Simulation Map Generator by clicking the Load map Image button.

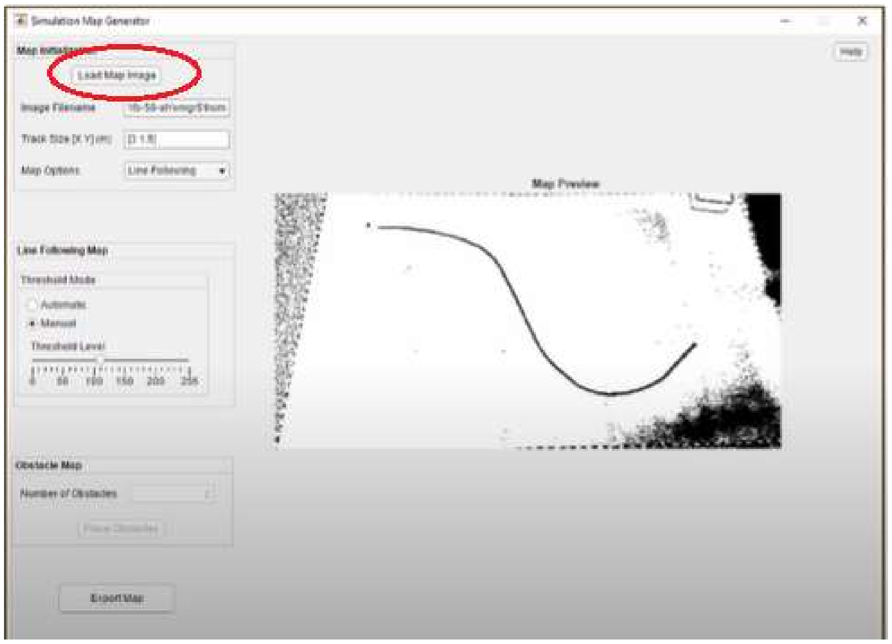


Image Filename: 10-58-...
Track Size [X Y] (m): [3 1.5]
Map Options: Line Following

Figure 3-9 Load map image into the map generator

2. Insert the track size as [3 1.5] and press Enter.

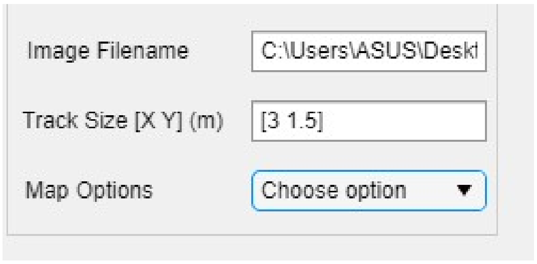


Image Filename	C:\Users\ASUS\Deskt
Track Size [X Y] (m)	[3 1.5]
Map Options	Choose option ▼

Figure 3-10 Track Size [3 1.5]

3. Under the Map Options, select Obstacle.

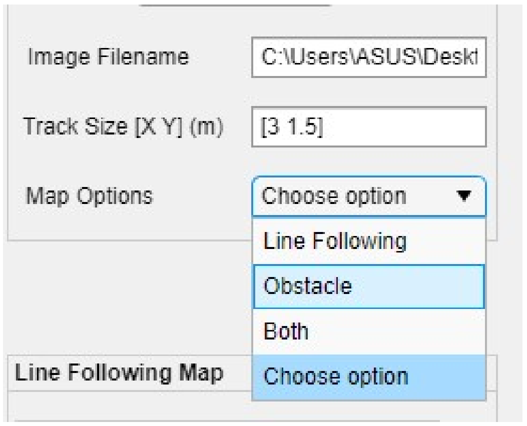


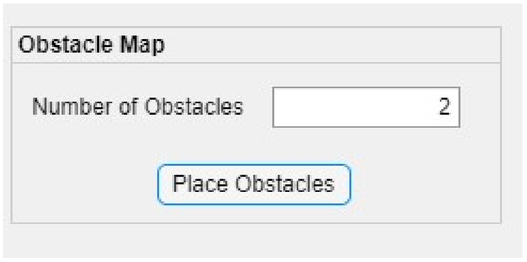
Image Filename: C:\Users\ASUS\Deskt

Track Size [X Y] (m): [3 1.5]

Map Options: Choose option

Figure 3-11 Select Obstacle for Map Options

4. Fill 2 for the number of obstacle and press the Place Obstacle button.

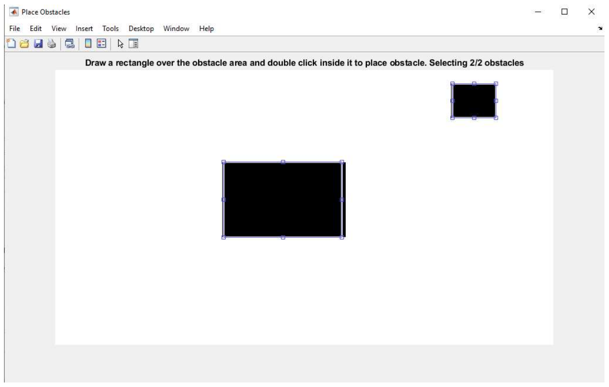


Obstacle Map

Number of Obstacles: 2

Figure 3-12 Number of Obstacle

5. Trace the obstacle position based on the map image and press enter. The obstacle will be shown in the map preview.



Draw a rectangle over the obstacle area and double click inside it to place obstacle. Selecting 2/2 obstacles

Figure 3-13 Obstacle Placing

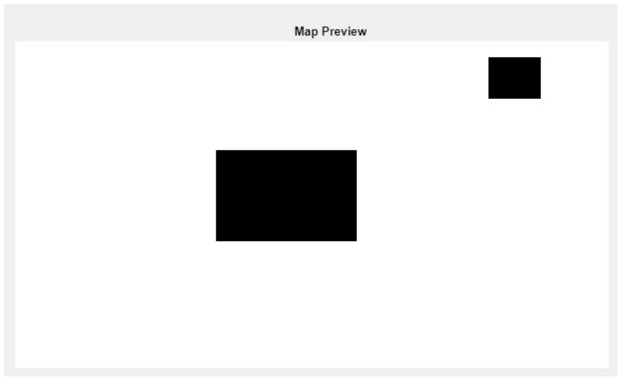


Figure 3-14 Map preview

6. Press the Export Map button and save the map generated.

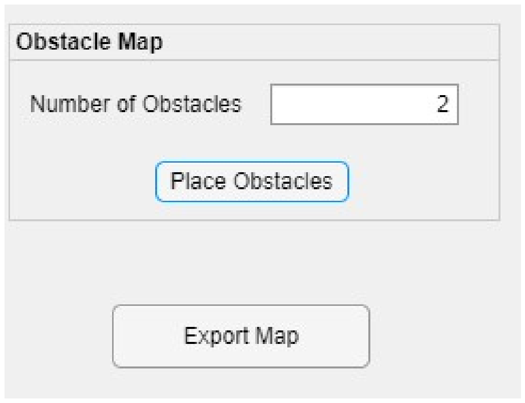


Figure 3-15 Export the generated map

3.4.2 Flow Chart of the Obstacle Avoidance Algorithm

A flow chart of the obstacle avoidance algorithm is created before creating the coding in Simulink model. An optimum path is planned in advanced as the robot is moving in a known environment. Figure below shows the planned path of the robot.

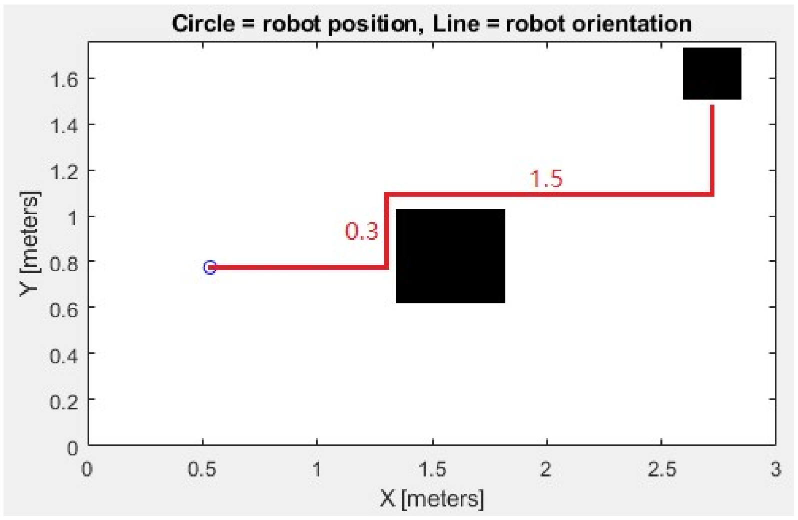
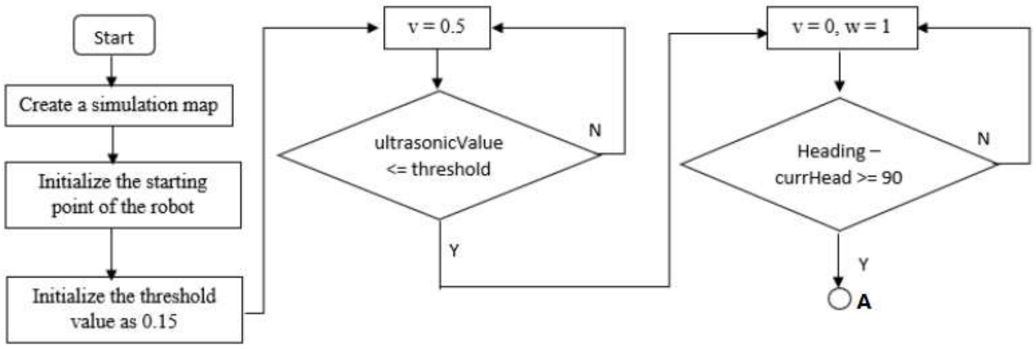


Figure 3-16 Planned path of the robot in a known environment

Based on the flow chart below, a simulation map will be created. Next, the starting point of the robot will be initialized. In this project, the starting point of the robot is (0.3076, 0.77562). Then a fixed threshold value is set to 0.15. This threshold value is the distance reserved to avoid collision between the robot and the obstacle. The further details regarding the algorithm will be discussed in the next subtopic.


Start
Create a simulation map
Initialize the starting point of the robot
Initialize the threshold value as 0.15
v = 0.5
ultrasonicValue <= threshold
N
Y
v = 0, w = 1
Heading – currHead >= 90
N
Y
A

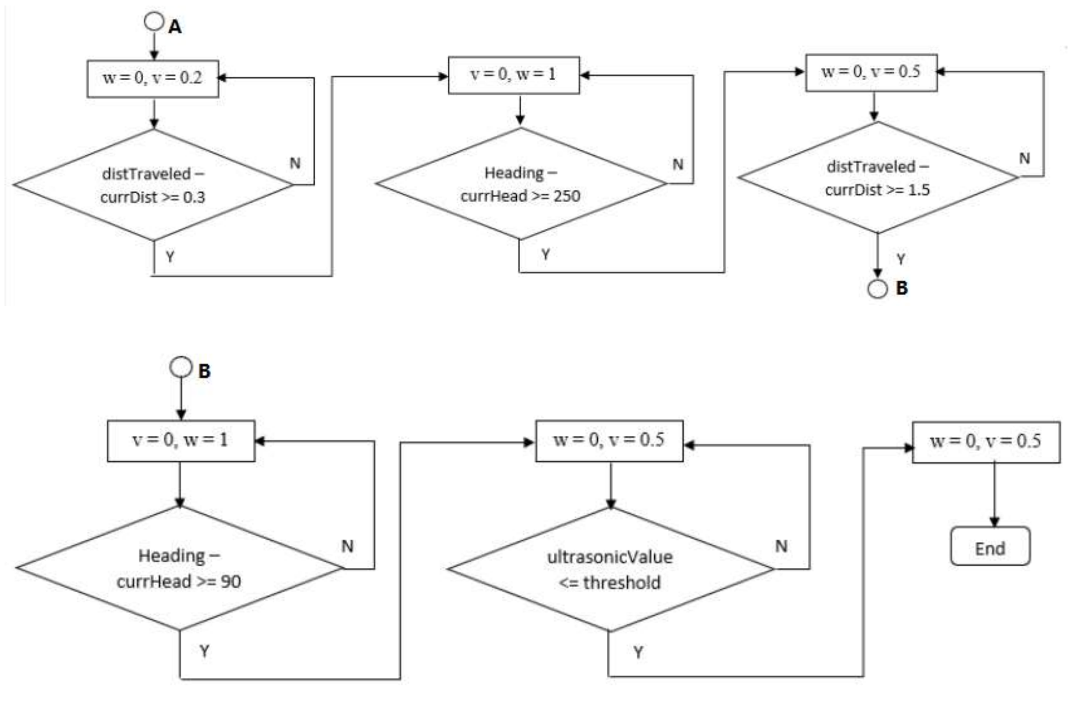


Figure 3-17 Flow Chart of Obstacle Avoidance Algorithm

3.4.3 State flow in Simulink

After the flow chart is done, the algorithm is then transformed into coding which presented in the form of state flow chart in Simulink. After the initialization of the simulation map, starting position of the robot and the threshold value, the robot will start moving in the velocity of 0.5. Then it will stop when the distance between the robot and obstacle (ultrasonicValue) is less than or equal to the threshold value. The robot will then rotate its heading angle to 90 degree and continue to move in velocity of 0.2. After travelling for a distance of 0.3, the robot will rotate its heading angle again for 250 degree.

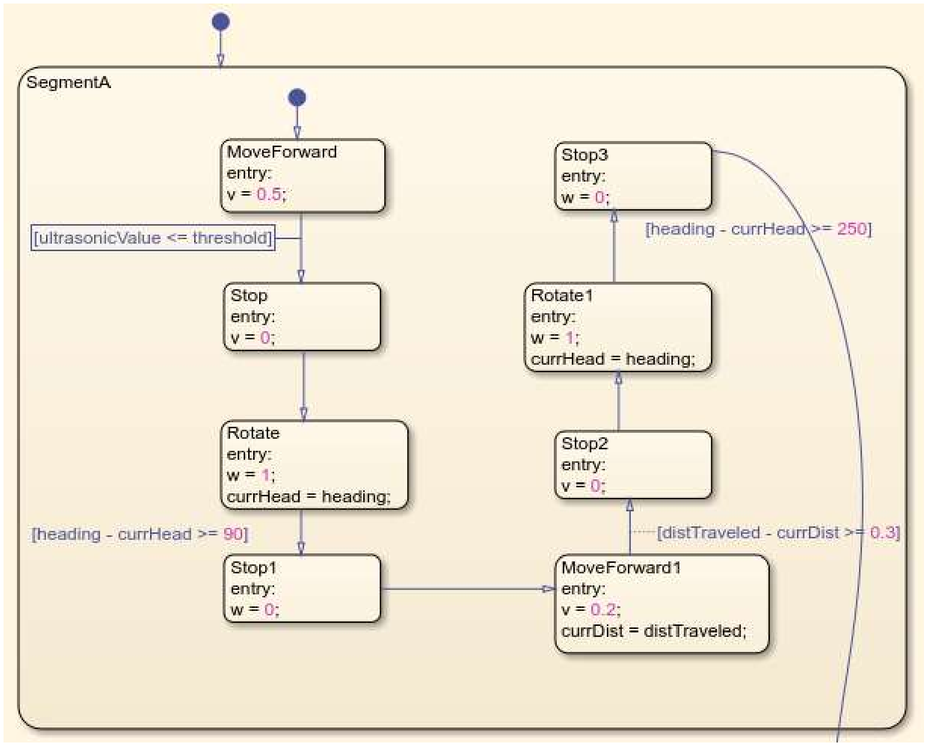


Figure 3-18 First part of the coding

Furthermore, the robot will move with a velocity of 0.5 and stops after travelling for a distance of 1.5. The robot will then rotate its heading angle again for 90 degree and continue to move with a velocity of 0.5. The robot finally stop when its distance with the obstacle is less than or equal to the threshold value. At this moment the obstacle is assumed as the ending point of the robot.

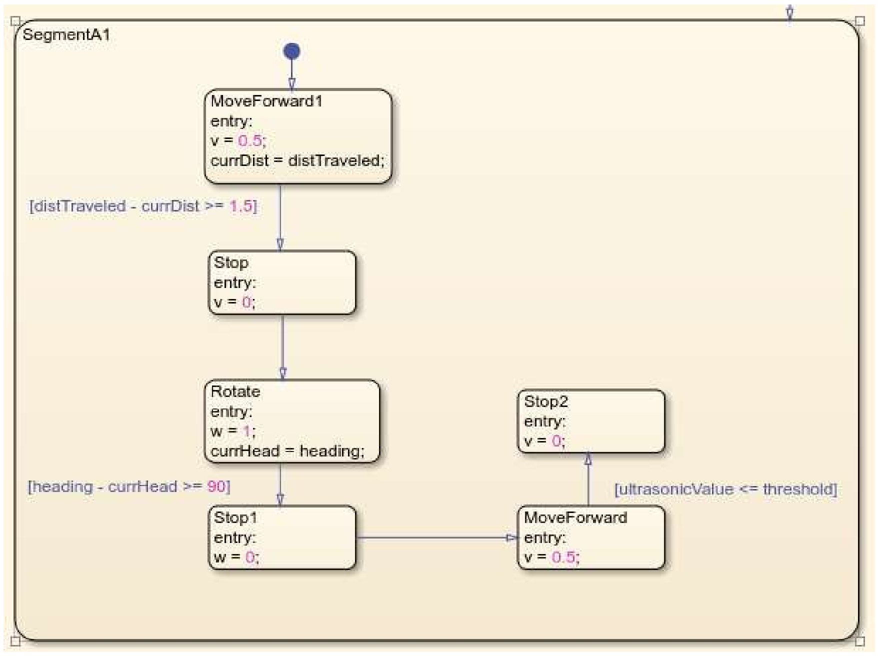


Figure 3-19 Second part of the coding

3.5 BAM Algorithm for swarm robots

Below is the pseudocode of the BAM Algorithm implement on the swarm robots.

Begin

Step 1: **Initialization**. Set the generation counter $t = 1$; Initialize the population of NP bats P randomly and each bat correcponding to a potential solution to the given problem; define pulse frequency Q; set loudness A_i, the initial velocities v_i and pulse rate r_i ($i = 1, 2, \ldots, NP$); set weighting factor F.

Step 2: **Generating the coordinates of the path**. Calculate the path to be taken by the robot from starting point to ending point by considering the obstacles present in the environment.

Step 3: Evaluate the threat cost J for each bat in P.

```
Step 4: while The halting criteria is not satisfied or $t < MaxGeneration$ do
    Sort the population of bats P from best to worst by order of threat cost J for each bat;
    for $i = 1 : NP$ (all bats) do
            Select uniform randomly $r_1 \neq r_2 \neq r_3 \neq i$
            $r_4 = [NP * rand]$
            $v_i^t = v_i^{t-1} + ( v_i^t - x_* ) x Q$
            $x_i^t = x_i^{t-1} + v_i^t$
            if (rand > r) then
                $x_u^t = x_* + \alpha\varepsilon^t$
            else
                $x_u^t = x_{r1}^t + F( x_{r2}^t - x_{r3}^t )$
            end if
            Evaluate the fitness for the offsprings $x_u^t, x_i^t, x_{r4}^i$
        Select the offspring $x_k^t$ with the best fitness among
            the offsprings
            $x_u^t, x_i^t, x_{r4}^i$
        if (rand < A) then
            $x_{r4}^i = x_{r4}^{i;}$
        end if
    end for $i$
```

```
Evaluate the threat cost for each bat in P.
Sort the population of bats P from best to worst by order of threat cost $J$ for each bat;
t = t + 1;
Step 5: end while
Step 6: Inversely transform the coordinates in final optimal path into the original
        coordinate, and output
End.
```

3.6 Complete System

Figure below shows the complete system created in Simulink. The blocks used in creating this system are shown in the table below

Type of Block	Function
Threshold constant value	A fixed value set as the safety distance between the robot and the obstacle.
Ultrasonic Sensor	To compute the distance between the robot and obstacle based on the robot position and sensor offsets.
Encoder	To measure the rotation of left and right wheels in ticks for both side of motor.
Calculator	To calculate the travelled distance of the robot and the heading angle of the robot.

WLWR block	To convert the robot linear velocity (v) and robot angular velocity (w) into left (wl) and right (wr) wheel angular velocities using robot parameters.
1-D lookup table	To represent the speed characteristics of the robot motor.
Motor model	Uses LTI systems to model and simulate the robot motors.
Robot simulator	To visualize a robot based on the given angular velocities.

Table 1 Function of every block used in the system

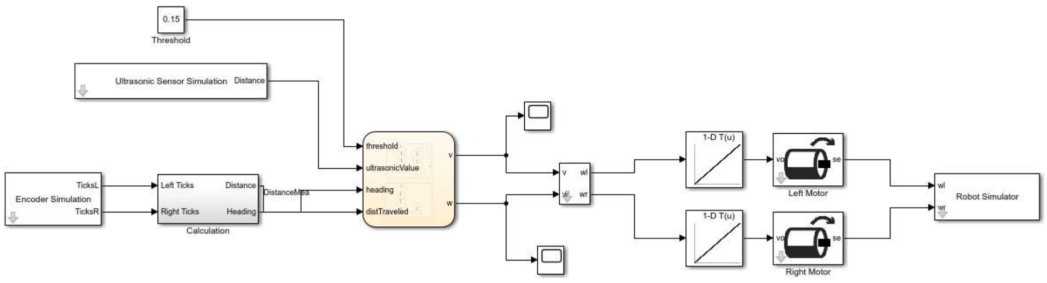


Figure 3-20 Complete system created in Simulink

3.6.1 Calculator

In this calculator, the distance travelled by the robot and the heading angle of the robot is calculated. First, the left and right ticks are both multiplied with the gain to obtain the left and right wheel travel distance based on the equation (3.13) below.

$$Distance = \frac{Total\ Encoder\ Ticks}{tick\ count\ per\ rotation}\ x\ 2\pi R \qquad (3.13)$$

For most of the robot wheel, tick count per rotation is 9 and 2πR is the circumference of the wheel.Since the controller used in this robot is only P controller, as mentioned in 3.3.2.2 PID controller, the left and right wheel travel is then multiplied with 0.5, in which also means that divided by 2, to obtained the real distance travelled by the robot.

The sum of left and right wheel travel distance is also multiplied with the gain of $\frac{1}{axlelengt}$ to get a radian value, and then convert the radian value to degree by using the R2D block to obtain the heading angle, which is the direction of the robot moving.

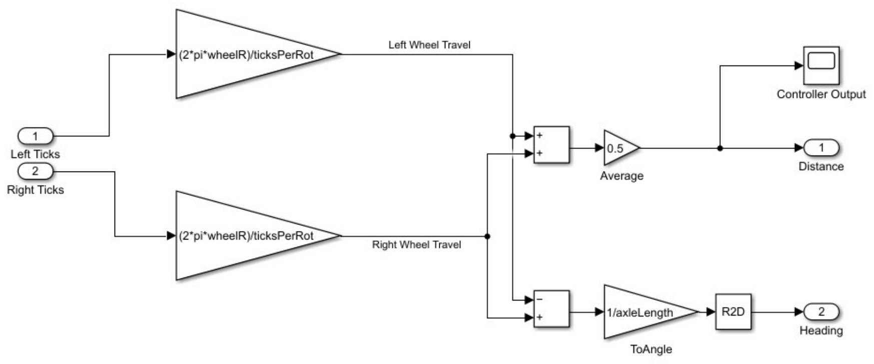


Figure 3-21 Details of the Calculator block

3.6.2 WLWR block

This WLWR block is used to convert the robot linear velocity (v) and robot angular velocity (w) into left (wL) and right (wR) wheel angular velocities. The left and right wheel angular velocities can be calculated by using the equation (3.14) and equation (3.15) as below.

$$wL = \frac{(v - w) * (\frac{axleLengt}{2})}{wheelRadius} \tag{3.14}$$

$$wR = \frac{(v + w) * (\frac{axleLengt}{2})}{wheelRadius} \tag{3.15}$$

The left and right wheel angular velocities are also can be calculated by using matrix notation of wLwR as equation (3.16) below.

$$[wL; wR] = \left(\frac{1}{wheelRadius\ x\ u}\right) * (\mathrm{K} * \mathrm{u}) * [v; w] \tag{3.16}$$

$$(K * u) = \left[1 - \frac{axleLength}{2}; \frac{1\ axleLength}{2}\right] \tag{3.17}$$

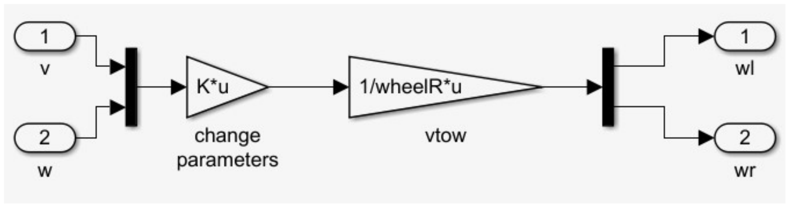


Figure 3-22 Details of WLWR block

3.6.3 Motor model

After receiving the output from 1-D lookup table, the angular velocities will enter the motor plant. The details of motor plant will be discussed in the next subtopic. The angular velocities will then choose to pass through either first input or third input. If control port 2, which is the sensorType is greater or equal to threshold value set, or equal to 0, then the angular velcities will enter the first input, where it will pass through the discrete-time integrator. The function of the discrete-time integrator is to create a purely discrete value. The value will then enter the R2D block to then convert the radian value to degree value. If control port 2 do not fulfill the requirement, the angular velocities will enter the third input.

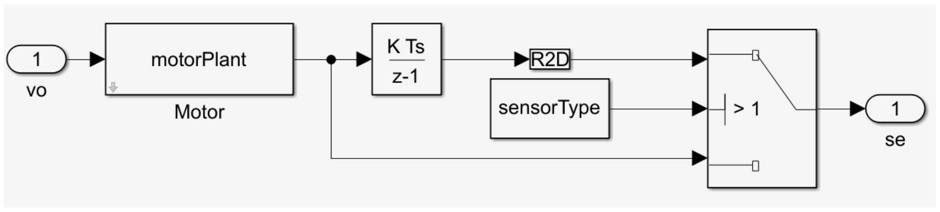


Figure 3-23 Details of motor model

3.6.3.1 Motor Plant

A state space model is used in this motor plant. It is a Multiple Input and Single Output (MISO) models. The inputs of the model are left (wL) and right (wR) angular velocity, while the outputs of the models could be x-position, y-position or θ as attitude angle. In this project, state space model is used to obtain the value of u, where $u = [u_1\ u_2] = [wL\ wR]$.

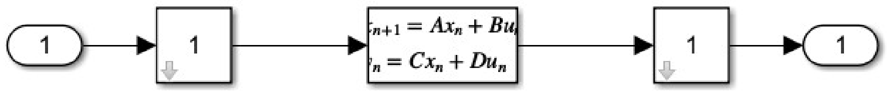


Figure 3-24 State Space Model in Motor Plant

3.7 Summary

In this chapter, I had introduce the methodology of the project. I had discussed about the construction of a single wheeled mobile robot creation, creation of the virtual environment with static obstacle, the obstacle avoidance algorithm applied to the robot and the details of the system. The results of each subchapter in chapter 3 will be discussed in the chapter 4.

CHAPTER 4

RESULTS

4.1 PID Controller of one robot

From the block diagram based on Fig. 3-1, a single wheeled mobile robot is created in Simulink. The controller used for this robot is PID controller with Proportional gain, $K_p = 10$. The time response graph obtained is as shown in Fig. 4-1.

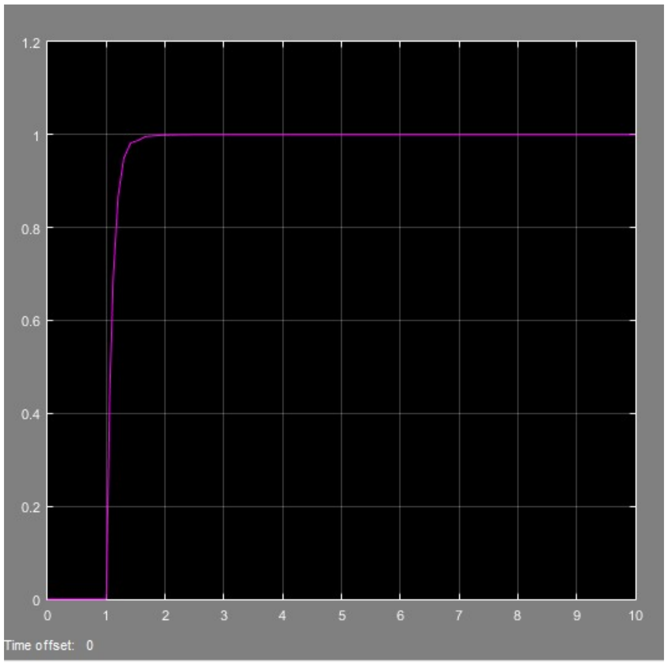


Figure 4-1 Time response when Kp = 10

Robot is set to have the same angular velocity for both left and right side of wheel, therefore it moves in a straight line.

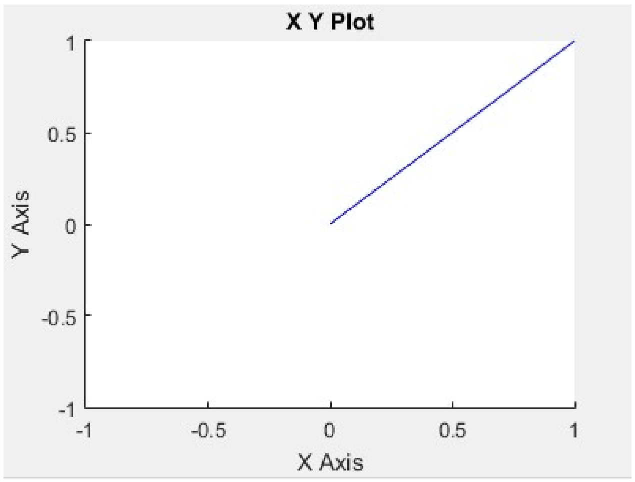


Figure 4-2 Same angular velocity for both side of wheel

4.2 Obstacle Avoidance Algorithm

Based on the Stateflow chart, the path travelled by the robot is shown in the figure below.

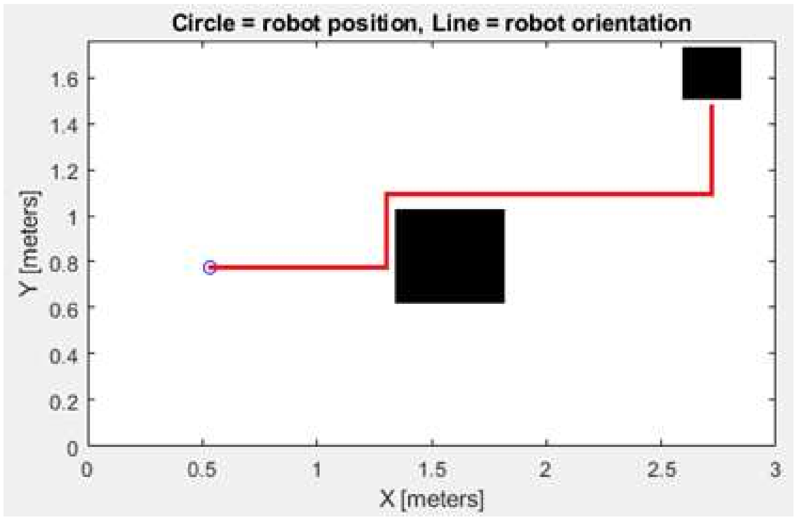


Figure 4-3 The robot's path

4.2.1 Virtual Environment

Below is the virtual environment with static obstacle created by using Simulation map generator.

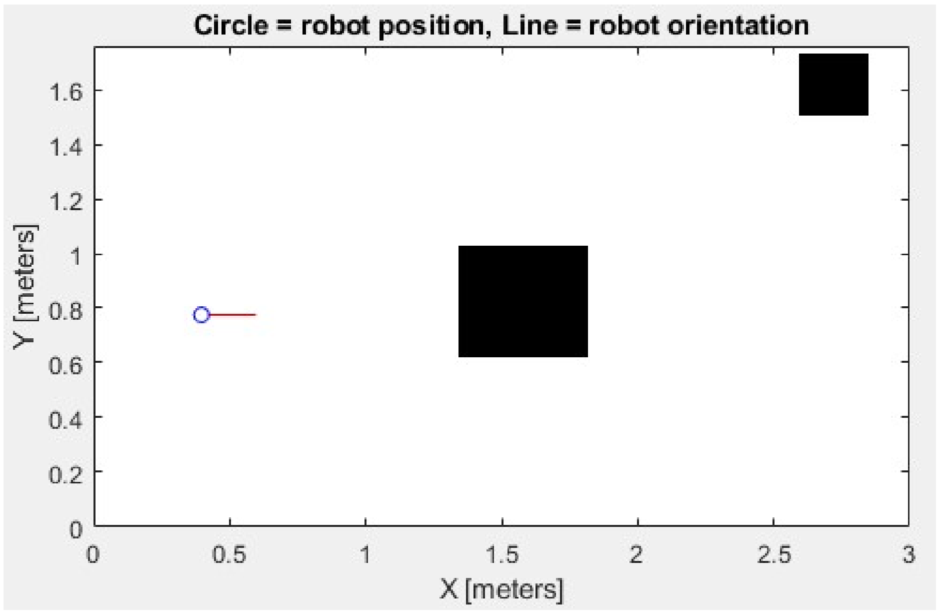


Figure 4-4 Virtual environment with static obstacle

4.2.2 Robot Linear velocity (v)

From the scope result, the velocity is 0.5 at point A, E, and G, 0.2 at point C. While the velocity is 0 at point B, D, F and H. It has shown that the linear velocity of the robot is synchronize with the coding created in stateflow chart.

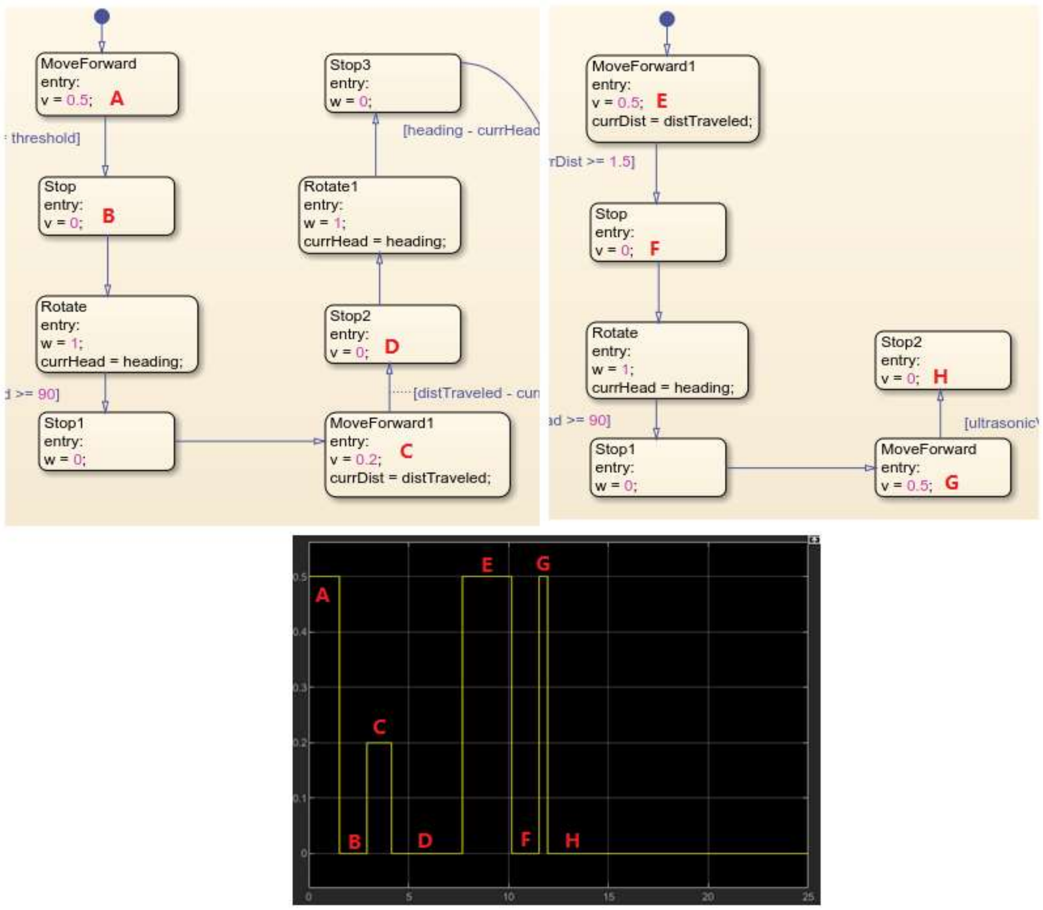


Figure 4-5 Comparison between stateflow chart and the linear velocity obtained from the scope.

4.2.3 Robot Angular velocity (w)

The robot rotates its heading angle when the angular velocity is equal to 1. From the scope, the angular velocity is equal to 1 at point A, C and E while angular velocity is equal to 0 at point B, D and F. It has shown that the angular velocity of the robot is synchronize with the coding created in stateflow chart.

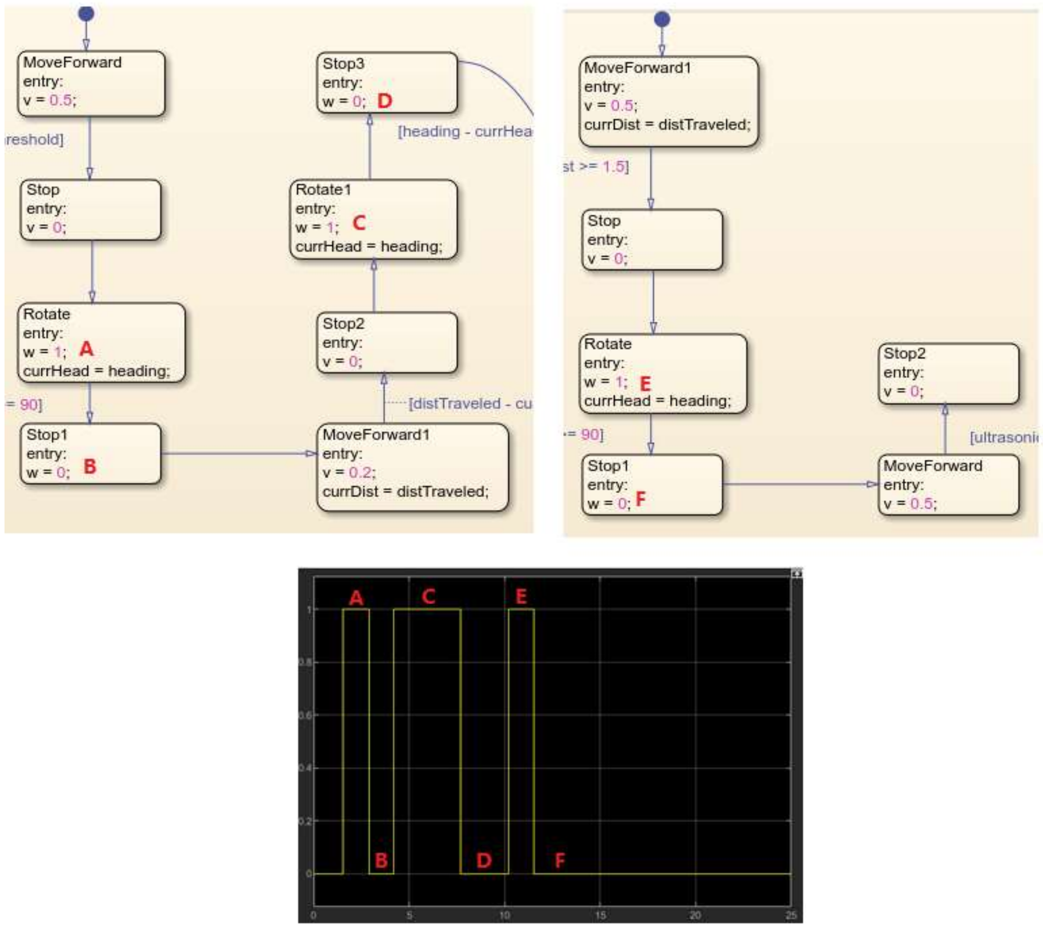


Figure 4-6 Comparison between stateflow chart and the angular velocity obtained from the scope.

CHAPTER 5

CONCLUSION

5.1 Conclusion

In conclusion, I had created a robot that is able to manoeuvre in a known situation and equipped with the ability of obstacle avoidance. Referring to the results in subtopic 4.2 and 4.3, the value of linear velocity (v) and angular velocity (w) of the robot obtained from the scope is synchronize with the coding created in the state flow. This shows that the robot is able to implement the algorithm without having error.

This robot is able to be implemented in the situation where static obstacle is present in an environment and it is able to take the best path to reach its destination. For instance, a delivery robot that travels in the same path to deliver items. The improvement that can be done towards this project will be discussed in the next subtopic.

5.2 Future Works

There are some improvement that can be done towards this project, such as implementing it into swarm robots where the robots are equipped with obstacle avoidance algorithm and applying the swarm robots in an unknown environment in which the robots will need to plan for a new path depending on its current environment.

REFERENCES

[1] Meerza, S. I., Islam, M., & Uzzal, M. M. (2018). Optimal path planning algorithm for swarm of robots using particle swarm optimization technique. *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)*. doi:10.1109/icitisee.2018.8720988

[2] Pebrianti, D., Bayuaji, L., Arumgam, Y., Riyanto, I., Syafrullah, M., & Ayop, N. Q. (2019). PID Controller Design for Mobile Robot Using Bat Algorithm with Mutation (BAM). *2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*.

[3] Korkmaz, M., & Durdu, A. (2018). Comparison of optimal path planning algorithms. *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*. https://doi.org/10.1109/tcset.2018.8336197

[4] Rashid, R., Perumal, N., Elamvazuthi, I., Tageldeen, M. K., Ahamed Khan, M. K., & Parasuraman, S. (2016). Mobile robot path planning using Ant Colony Optimization. *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*. https://doi.org/10.1109/roma.2016.7847836

[5] Samadi, M., & Othman, M. F. (2013). Global Path Planning for Autonomous Mobile Robot Using Genetic Algorithm. *2013 International Conference on Signal-Image Technology & Internet-Based Systems*. https://doi.org/10.1109/sitis.2013.118

[6] Guo , Z., & Ge , X. (2010). A particle swarm optimization for the motion planning of wheeled mobile robot. *2010 8th World Congress on Intelligent Control and Automation*. https://doi.org/10.1109/wcica.2010.5554396

[7] Brand, M., Masuda, M., Wehner, N., & Yu, X.-H. (2010). Ant Colony Optimization algorithm for robot path planning. *2010 International Conference On Computer Design and Applications*. https://doi.org/10.1109/iccda.2010.5541300

[8] Kulatunga, A., Liu, D., Dissanayake, G., & Siyambalapitiya, S. B. (2006). Ant Colony Optimization based Simultaneous Task Allocation and Path Planning of Autonomous Vehicles. *2006 IEEE Conference on Cybernetics and Intelligent Systems*. https://doi.org/10.1109/iccis.2006.252349

[9] Behring, C., Bracho, M., Castro, M., & Moreno, J. A. (2001). An Algorithm for Robot Path Planning with Cellular Automata. *Theory and Practical Issues on Cellular Automata*, 11–19. https://doi.org/10.1007/978-1-4471-0709-5_2

[10] Patil, D. A., Upadhye, M. Y., Kazi, F. S., & Singh, N. M. (2015). Multi Robot Communication and Target Tracking system with controller design and implementation of SWARM robot using Arduino. *2015 International Conference on Industrial Instrumentation and Control (ICIC)*. https://doi.org/10.1109/iic.2015.7150777

[11] Şahin, E. (2005). Swarm Robotics: From Sources of Inspiration to Domains of Application. *Swarm Robotics*, 10–20. https://doi.org/10.1007/978-3-540-30552-1_2

[12] Ducatelle, F., Di Caro, G. A., Pinciroli, C., Mondada, F., & Gambardella, L. M. (2011). Communication assisted navigation in robotic swarms: Self-organization and cooperation. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. https://doi.org/10.1109/iros.2011.6094454

[13] Saxena, A., Satsangi, C. S., & Saxena, A. (2014). Collective collaboration for optimal path formation and goal hunting through swarm robot. *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*. https://doi.org/10.1109/confluence.2014.6949364

[14] Nair, S. C., Coronado, E. M., Frye, M. T., & Qin, Y. (2015). Swarm intelligence for the control of a group of robots. *2015 10th System of Systems Engineering Conference (SoSE)*. https://doi.org/10.1109/sysose.2015.7151906

[15] Wang, G., Guo, L., Duan, H., Liu, L., & Wang, H. (2012). A Bat Algorithm with Mutation for UCAV Path Planning. *The Scientific World Journal*, *2012*, 1–15. https://doi.org/10.1100/2012/418946

[16] Wang, G.-G., Chu, H. C. E., & Mirjalili, S. (2016). Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerospace Science and Technology*, *49*, 231–238. https://doi.org/10.1016/j.ast.2015.11.040

[17] Pebrianti, D., Hao, Y. H., Suarin, N. A., Bayuaji, L., Musa, Z., Syafrullah, M., & Riyanto, I. (2018). Motion Tracker Based Wheeled Mobile Robot System Identification and Controller Design. *Lecture Notes in Mechanical Engineering*, 241–258. https://doi.org/10.1007/978-981-10-8788-2_23

[18] Ghanem, W. A., & Jantan, A. (2017). An enhanced Bat algorithm with mutation operator for numerical optimization problems. *Neural Computing and Applications*, *31*(S1), 617–651. https://doi.org/10.1007/s00521-017-3021-9

[19] Xie, J., & Wei, D. (2018). Multi-sensor Detection Network based on Improved Bat Algorithm. *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. https://doi.org/10.1109/iaeac.2018.8577249

[20] Shadbahr, E., Aminnejad, B., & Lork, A. (2021). Determining post-fire residual compressive strength of reinforced concrete shear walls using the BAT algorithm. *Structures*, *32*, 651–661. https://doi.org/10.1016/j.istruc.2021.03.002

APPENDICES

Appendix A: Estimated Detailed Progress Gantt Chart

Estimated Detailed Progress Gantt Chart

No	Description	Weeks (2020/2021 II)														Weeks (2021/2022 I)													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Understanding of Project Title	🟥	🟥	🟥																									
2	Research reading on single wheeled mobile robot			🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																
3	Single wheeled mobile robot creation				🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																
4	Checking the stability of mobile robot											🟥	🟥	🟥	🟥														
5	Research reading on environment creation															🟨	🟨	🟨											
6	Environment creation																🟨	🟨											
7	Research reading on point-to-point navigation																		🟨	🟨	🟨								
8	Implementing point-to-point navigation to the robot																			🟨	🟨	🟨							
9	Inserting robot into the environment																					🟩	🟩						
10	Research reading on Obstacle Avoidance Algorithm																							🟩	🟩	🟩			
11	Implementation of obstacle avoidance algorithm to the robots																									🟩	🟩		
12	Debugging																										🟩	🟩	
13	Checking if objective is achieved																												🟩