HIGH SPEED 3D PRINTING STRATEGY USING 6 DOF NON-MOBILE ROBOT MANIPULATOR BASED ON SHORTEST DISTANCE ALGORITHM

BEE SHENG CHONG

A report submitted in partial fulfilment of the requirements for the degree of Bachelor of Mechatronics Engineering

Faculty of Electrical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2018

"I hereby declare that I have read through this report entitled "HIGH SPEED 3D PRINTING STRATEGY USING 6 DOF NON-MOBILE ROBOT MANIPULATOR BASED ON SHORTEST DISTANCE ALGORITHM" and found that it complies the partial fulfillment for awarding the degree of Bachelor of Electrical Engineering"

Signature	: ....
Supervisor's Name	: PM DR MUHAMMAD FAHMI BIN MISKON
Date	: ....

I declare that this report entitled "HIGH SPEED 3D PRINTING STRATEGY USING 6 DOF NON-MOBILE ROBOT MANIPULATOR BASED ON SHORTEST DISTANCE ALGORITHM" is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently in candidature of any other degree.

Signature	: ....
Name	: BEE SHENG CHONG
Date	: ....

To my beloved mother and father

ACKNOWLEDGEMENT

Firstly, I would like to express my sincere appreciation to my FYP supervisor PROFESOR MADYA DR. MUHAMMAD FAHMI BIN MISKON for his patience and kindness in guiding me on this project. His passion and sense of responsibility had help me to keep the progress of the project at a satisfactory level. Besides that, I would also very grateful to his professional comment and critics which greatly promote my understanding on the project.

Meanwhile, I sincerely thank you to both of my FYP panels, DR. MOHD SHAHRIEEL BIN MOHD ARAS and Pn NURDIANA BINTI NORDIN for the generous enlightenment they gave to me on the shortcomings and weakness of my works after the presentation. The suggestions they gave strengthen my report and make it more complete.

Lastly, I am very thankful for all the supports and helps which came from my course mates and friends. Their attitudes of always willing to help and fast respond resolved most of my confused moment during the work for this project. I would also like to thank my family for their caring and tolerant to me during this tough time.

ABSTRACT

3D printing is an increasing demanded technology nowadays and the 3D printers sold on the mass market today are limited on its workspace. In order to solve the problem, a strategy that combining non-mobile articulated robot and 3D printing is introduced in this project. The main focus in this project is to study, design and evaluate the shortest distance algorithm in order to achieve high speed 3D printing that using 6 DOF non-mobile robot manipulator. The Dijkstra's algorithm is referred in this project for determine the shortest distance travelled from coordinate to coordinate. The performance of the shortest distance algorithm is examined and analysed by using Scilab. Meanwhile, the performance of the manipulator is simulated in V-REP and corresponding analysis is carried out to measure the speed of the 3D printing. The result shows that the designed algorithm able to shorten the distance travelled by percentage ratio of 72.57%. On the other hand, time complexity of the designed algorithm is $O(n^2)$ while 3D printing analysis result indicates that the algorithm can increase the efficiency of the 3D printing but the starting coordinate may affect the efficiency of the 3D printing caused by different trajectory path.

ABSTRAK

Pencetakan 3D adalah teknologi yang semakin menuntut pada masa kini dan pencetak 3D yang dijual di pasaran massa hari ini terhad pada ruang kerjanya. Untuk menyelesaikan masalah ini, satu strategi yang menggabungkan artikulasi robot bukan mudah alih dengan percetakan 3D diperkenalkan dalam projek ini. Tumpuan utama dalam projek ini adalah untuk mengkaji, merekabentuk dan menilai algoritma jarak terpendek untuk mencapai pencetakan 3D berkelajuan tinggi yang menggunakan 6 DOF robot bukan mudah alih. Algoritma Dijkstra dirujuk dalam projek ini untuk menentukan jarak terpendek dari koordinat ke koordinat. Prestasi algoritma jarak terpendek diperiksa dan dianalisis dengan menggunakan Scilab. Sementara itu, prestasi manipulator disimulasikan dalam V-REP dan analisis yang sama dijalankan untuk mengukur kelajuan percetakan 3D. Hasilnya menunjukkan bahawa algoritma tersebut dapat memendekkan jarak perjalanan dengan nisbah peratusan sebanyak 72.57. Di samping itu, kerumitan masa algoritma yang direka adalah $O(n^2)$ manakala hasil analisis percetakan 3D menunjukkan bahawa algoritma dapat meningkatkan kecekapan pencetakan 3D tetapi koordinat permulaan mungkin mempengaruhi kecekapan pencetakan 3D disebabkan oleh laluan trajektori yang berbeza.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

SP	Shortest Path
RP	Rapid Prototyping
DOF	Degree Of Freedom
CAD	Computer-Aided Drawing
3D	Three-Dimensional
dist	Distance
STL	STereoLithography
AM	Additive Manufacturing

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Nowadays, 3D printing becomes a trending technology due to the impact it had brought to rapid prototyping [1]. Many people believe that it will be widespread and become more common in future because of those benefits it brought to us. For example, a prototype can be created rapidly in just a few hours by 3D printing technique compared with traditional manufacturing methods which taken days or even weeks to receive a prototype[2]. Due to the advantage of 3D printing on saving cost and time, mitigate risks, able to create complex products and many more, this technology is increasing demanded on various field, such as, architecture and construction, healthcare and medical, mechanic, aeronautics and space, and so on [1].

However, 3D printer on the market nowadays cannot create a large enough prototype because of its limited workspace and size. For example, the 3D printer produced by CreatBot named as 'D600' can only print the largest build volume of 600*600*600*mm and it's already 30 times larger than it of other ordinary 3D printer in today mass market. Obviously, this is one of the limitation of 3D printer nowadays. Hence, if we can implant 3D printing technology on 6 DOF non-mobile robot manipulator system we may overcome this limitation.

Besides that, 3D printing using 6 DOF non-mobile robot manipulator system possess a great advantage compared with the traditional additive manufacturing method. Firstly, conventional method is restricted by both the gravity and printing environment[3]. Meanwhile, the uses of 6 DOF robot makes the 3D printing on irregular, or non-horizontal surfaces become possible. The robot can change the orientation of extrusion for the best strength and appearance of a part during 3D printing.

Lastly, the robot is using shortest distance algorithm to analyze and decide the shortest path for the 3D printing process. This energy optimizing technique reduces the time and energy consumption by the robot to the lowest as possible. Consequently, it makes the construction of light-weighted, high mobility and longer energy lasting 3D printing robot become possible.

1.2 Problem statement

The first problem of high speed 3D printing strategy using 6 DOF non-mobile robot manipulator system by using shortest distance algorithm is to design a shortest distance algorithm that can determine the sequence of the coordinates to achieve high speed 3D printing.

The designed algorithm should be able to minimize the travelled path in order to increase the speed of 3D printing. Next, the problem is to code the algorithm to the computer and conduct analysis on its performance.

The next problem is the complexity of trajectory generation for the robot. The complexity of trajectory generation of the robot is due to its multiple degree of freedom [4]. The robot has to find the way to reach a specific point and it got many ways to get a same point due to its multiple degree of freedom [5].

The performance of the robot is demonstrated and observed firstly by simulation using V-rep. This is also one of the problems we concerned about as how to simulate these trajectories and represent them in the computer.

Based on the problems described, the project need to address a method on how to achieve high speed 3D printing with different shape and dimension.

1.3 Objective

The objectives of this project are:

1. To analyse the shortest path problem of 3D printing that using a 6 DOF manipulator (IRB 4600 industrial robot).
2. To design a shortest path algorithm which can generate corresponding trajectory based on the shape and dimension given.
3. To evaluate the efficiency and speed of the 3D printing that using IRB 4600 industrial robot.

1.4 Scope

The scope of this project concentrated on:

1. Study on the shortest path algorithm to achieve a high-speed 3D printing with 6 DOF non-mobile robot manipulator system.
2. Use Scilab simulator to demonstrate the performance of the algorithm.
3. Use V-rep simulator to simulate the 3D printing process.
4. 3D printed a U-shape box but not exceed the workspace limit of the robot.
5. Use IRB 4600-40-255 industrial robot to conduct the 3D printing.
6. Represented the 3D printed object by octree and the dimensions of the object is acted as references.
7. To find the efficiency of the algorithm by using method of analysis.
8. The trajectory path that generated by the algorithm.
9. Assuming no obstacles and external factors in the simulation environment.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter is intended to review, synthesises, analyse and present the information of previous research works from literature related to 3D printing using robot. Nevertheless, the shortest path algorithm which is used to optimize the performance of the robot and resulted in high speed 3D printing is prioritized in this chapter. In line with this, a few shortest path algorithms such as Dijkstra’s algorithm, Bellman Ford's algorithm and Floyd–Warshall's algorithm will be introduced and compared in the coming section. A most suitable SP algorithm is selected as reference with the appropriate reason provided. Then, time complexity is introduced for the analysis of algorithm. Subsequently, the chapter is focus on 3D printing and technology related to it. The 3D printing and its process are enlightened in the following section. Related subjects such as STL files, slicing software and G-code are presented and explained along with 3D printing process. The information of existing 3D printing robot is gathered and shown in the later section.

2.2 Shortest path algorithm based on distance comparison

Shortest path algorithm is widely implement in various field especially in internet addressing computing, intelligent transportation systems, urban geographic information systems, and military geographical information systems. In this project, shortest distance algorithm is the main focus which is also the biggest problem urged to be solved. At present, there are several types of numerical algorithm available for solving optimization problem. Here are their names: Dijkstra algorithm, Bellman Ford's algorithm and Floyd–Warshall's algorithm.

2.2.1 Dijkstra’s algorithm

Dijkstra’s algorithm is an algorithm used to find the shortest distance between nodes in a graph. It was devised by computer scientist Edsger W. Dijkstra in 1956 and published 3 years later [6].

Dijkstra’s algorithm applies only if [7]:

i. The link values (edges cost) must be positive values (the algorithm will be broken if the value is negative) but the links can be directional
ii. All vertices in the graph is connected (the algorithm does not work if there is unconnected part exists)

The steps to apply Dijkstra’s algorithm are shown below [8]:
The starting node is called as the initial node.

1. An uncertain distance value is assigned to every node: initial node is set as zero and other nodes are set as infinity.
2. Place the initial node as current and set all other nodes as unvisited. Make the unvisited set which is a set of all the unvisited nodes.
3. Consider all the neighbors of current node and compute their tentative distances. The current assigned value will be replaced by newly calculated tentative distance if the value is smaller. For example, if the assigned value is 7 for current node A and the connected edge to neighbor node B has a length of 3, then the distance to B will be 7 + 3 = 10. If B was previously assigned as a value greater than 10 then it will be changed to 10. Otherwise, the value will be remained.
4. When all the neighbors of the current node are done considering, the current node is marked as visited and it will be removed from unvisited set. A visited node will not be visit again.
5. The algorithm will continue until the destination node has been marked visited or the smallest tentative distance among the nodes in the unvisited sets is infinity.

6. If none of the two conditions is achieved, the unvisited node with the smallest tentative distance will be selected and set as new ‘current node’ then repeat from step 3.

Figure 2.1 & 2.2 and Table 2.1 show the demonstration of the algorithm.

‘A’ is the starting node.

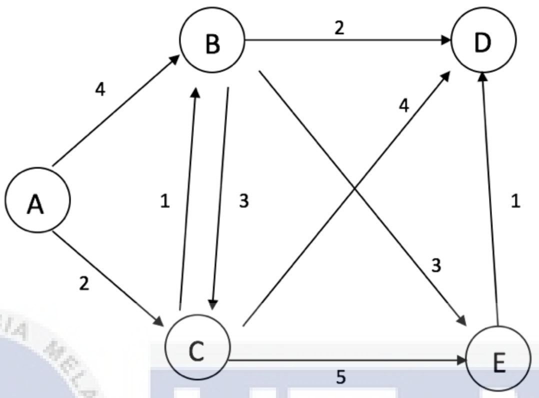


Figure 2.1: Five vertices shortest path problem example.

Table 2.1: Result of Dijkstra’s algorithm on solving Figure 2.1 example.

Step	Current node	Unvisited set	A	B	C	D	E
1	A	{B,C,D,E}	0	4	2	∞	∞
2	C	{B,D,E}	0	3	2	6	7
3	B	{D,E}	0	3	2	5	6
4	D	{E}	0	3	2	5	6
5	E	{}	0	3	2	5	6

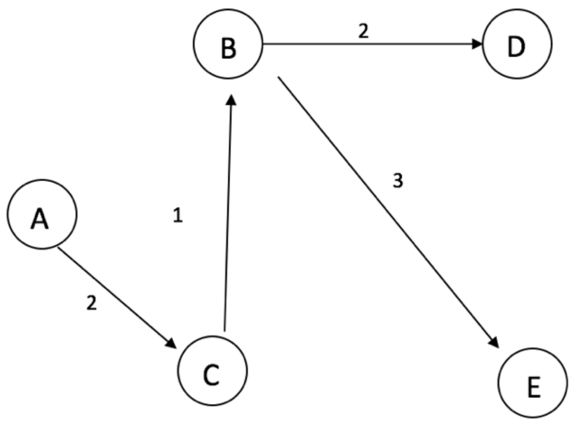


Figure 2.2: The shortest path of the Figure 2.1 example.

2.2.2 Bellman-Ford's algorithm

Bellman-Ford's algorithm is similar to Dijkstra's algorithm but it is more versatile because it can work with graph in which the edges can have negative weights [7]. This algorithm uses the principle of relaxation which is same as Dijkstra's algorithm but the only difference is Bellman-Ford algorithm relaxes all the edges instead of choosing the nearest vertex that has not been checked (also known as greedy strategy). The ordering of its relaxation makes the algorithm can handles negative weights and also detects negative cycles. It is important to note that if negative cycle occurs, there will be no shortest path exists on the graph.

Below is the step on how the algorithm works:

1. Allocate the starting node as zero and the rest of the vertices are set as infinity.
2. Start from the first point (starting node), give all the neighbors of the node a tentative distance cost.
3. Continue with the next node, the node which cannot be defined on the moment is skipped.
4. After all nodes are checked once, the first iteration is done.
5. Repeat from step 2 until 4, update the tentative distance cost when there is a smaller value get in this process.

6. The process will stop when there is no change compared with previous iteration (The algorithm takes at most |V – 1| iteration but it can be ended earlier if no alteration is found on the next iteration).

The algorithm is demonstrated on Figure 2.3 and Table 2.2 as below.
'S' is the starting node

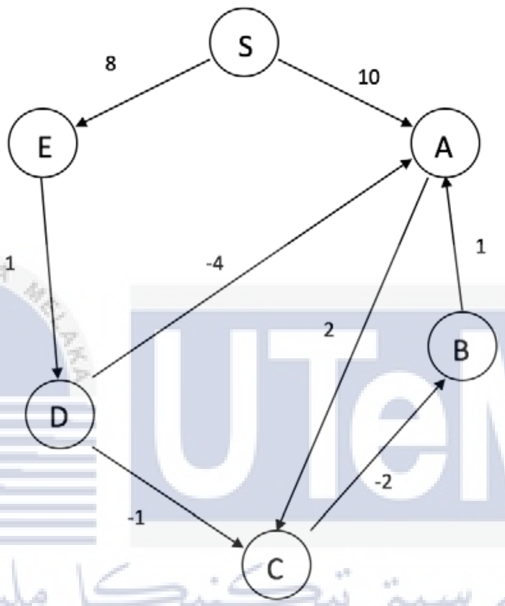


Figure 2.3: Six vertices graph with negative edge weights example.

Table 2.2: Result of Figure 2.2 solved by Bellman-Ford's algorithm.

Iteration	S	A	B	C	D	E
0	0	∞	∞	∞	∞	∞
1	0	10	10	12	9	8
2	0	5	10	8	9	8
3	0	5	5	7	9	8
4	0	5	5	7	9	8

2.2.3 Floyd-Warshall algorithm

Floyd-Warshall algorithm is an algorithm for finding shortest distance in a weighed graph same as Dijikstra's algorithm and Bellman-Ford's algorithm [7]. However, it is an all-pairs shortest path algorithm instead of single-source shortest path algorithm (Dijkstra's algorithm & Bellman-Ford's algorithm). This means that it computes the shortest distance between every pair of vertices in the graph instead of starting from a single initial node. Floyd-Warshall algorithm is an example of dynamic programming, the problem is break down into smaller sub-problems and the answers of those sub-problems are combined to solve the large, primary problem. It can solve the graph contains negative weighed edges but no negative-weighed cycles (same as Bellman-Ford algorithm). It is useful at handling multiple stops on the route because it can compute the shortest distance between all relevant nodes.

The implementation of Floyd-Warshall algorithm is shown below.

1. A distance array table is constructed to track the shortest path between nodes.
2. Fill in the corresponding weighs into the table by looking at the edges between the nodes of the graph.
3. Replace the value on the table by the newly calculated value based on equation (2.1) if the condition is met;

   $$\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j] \tag{2.1}$$

   i , j and k are value from 1 until number of vertices in the graph, eg. if the graph contains 4 vertices then;

   i = 1 2 3 4 ; j = 1 2 3 4 ; k = 1 2 3 4
4. The algorithm is done as all values of i, j and k have been calculated once.

The example of Floyd-Warshall algorithm is shown in Figure 2.4 below.

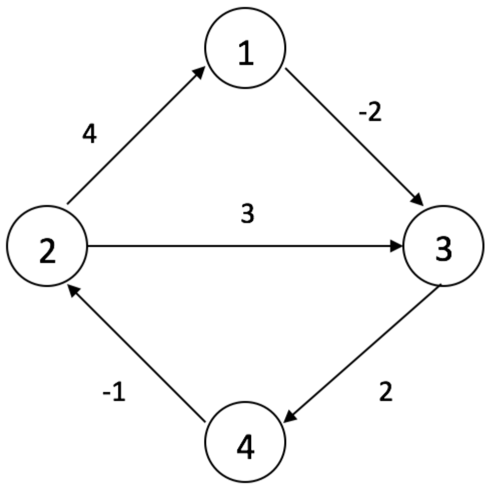


Figure 2.4: Example of four vertices graph with negative edge weights.

Step 1: Constructs a table and fill in the corresponding edges weights. The result is shown in Table 2.3.

Table 2.3: Result of the Step 1 of Floyd-Warshall algorithm.

	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

Step 2: Computes them by using the formula with different value of i, j and k

Let k = 1, i = 1 and j = 2,

dist[i][j] > dist[i][k] + dist[k][j]

dist[1][2] > dist[1][1] + dist[1][2]

$\infty > 0 + \infty$

$\infty > \infty$ (False, nothing happened)

Let k = 1, i = 2 and j = 3,

dist[i][j] > dist[i][k] + dist[k][j]

dist[2][3] > dist[2][1] + dist[1][3]

3 > 4 + (-2)

3 > 2 (True, the value on [2][3] is replaced by 2)

Step 3: The algorithm is done when all values of i, j and k have been calculated once. Table 2.4 shows the final result of the algorithm.

Table 2.4: The final result of Figure 2.4 example by using Floyd-Warshall algorithm.

	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0

2.3 Comparison of the SP algorithm

The comparison of the shortest path algorithm is shown in Table 2.5.

Table 2.5: The comparison of the shortest path algorithm.

Shortest path algorithm	Advantage	Disadvantage
Dijkstra’s algorithm	Fast and simple	Cannot solve graph that contains negative edge weights.
Bellman-Ford’s algorithm	More versatile and can solved graph contains negative edge weights.	Slower than Dijsktra’s algorithm.
Floyd-Warshall algorithm	Can find the shortest path between the vertices and solved graph contains negative edge weights.	Only show shortest path cost between nodes without intermediate nodes.

Dijkstra’s algorithm is the earliest and simplest SP algorithm among these three algorithms [9]. Dijkstra’s algorithm is fast because of the greedy strategy it’s used. Dijkstra’s algorithm greedily selects the nearest vertex that has not been processed yet and performs the relaxation process on all its outgoing edges. However, Dijkstra’s algorithm cannot handle graph with negative edge weights, it will break down if the graph contains negative edge weights.

Compared with Dijkstra’s algorithm, Bellman-Ford’s algorithm is much slower on solving the same problem. However, it is outstanding on its versatility because it able to solve the graph with negative edge weights due to the way of its relaxation.

Floyd-Warshall algorithm does not have a starting node, it shows all the shortest path cost between each node. This is suitable for problem which has few stops in a route, but in the other hand it is difficult to know the sequence and the pattern of the shortest path.

In a nutshell, Dijkstra’s algorithm possesses the fastest and simplest characteristic among these three algorithms. It cannot solve the graph contains negative edge weights however it is no necessary to deal with the graph contains negative edge weights in this project. Hence, based on the above reasons, Dijkstra’s algorithm may be the most suitable SP algorithm for this project.

2.4 Time complexity of algorithm

The performance of algorithm may vary with different algorithm and there are a few parameters that can denote that. For example, time complexity, space complexity, correctness, simplicity and generality. Time complexity indicates how fast the algorithm runs while space complexity deals with the extra space the algorithm requires.

As technology advanced, the computer's speed and memory have improved by many orders of magnitude. Now the amount of extra space required by an algorithm is typically not of as much concern. However, time issue has not diminished quite to the same extent. Many experiment and research show that for most problems, we can get a greater progress in speed than in space.

Figure 2.5 below shows the comparison of the performance of insertion sort and quick sort by implemented quick sort on Intel 486 and insertion sort on an IBM SP2. IBM SP2 is a super computer while Intel 486 is only a personal computer. The result indicates that a fast computer with an inferior algorithm may perform worse than a slow computer with a superior algorithm[10].

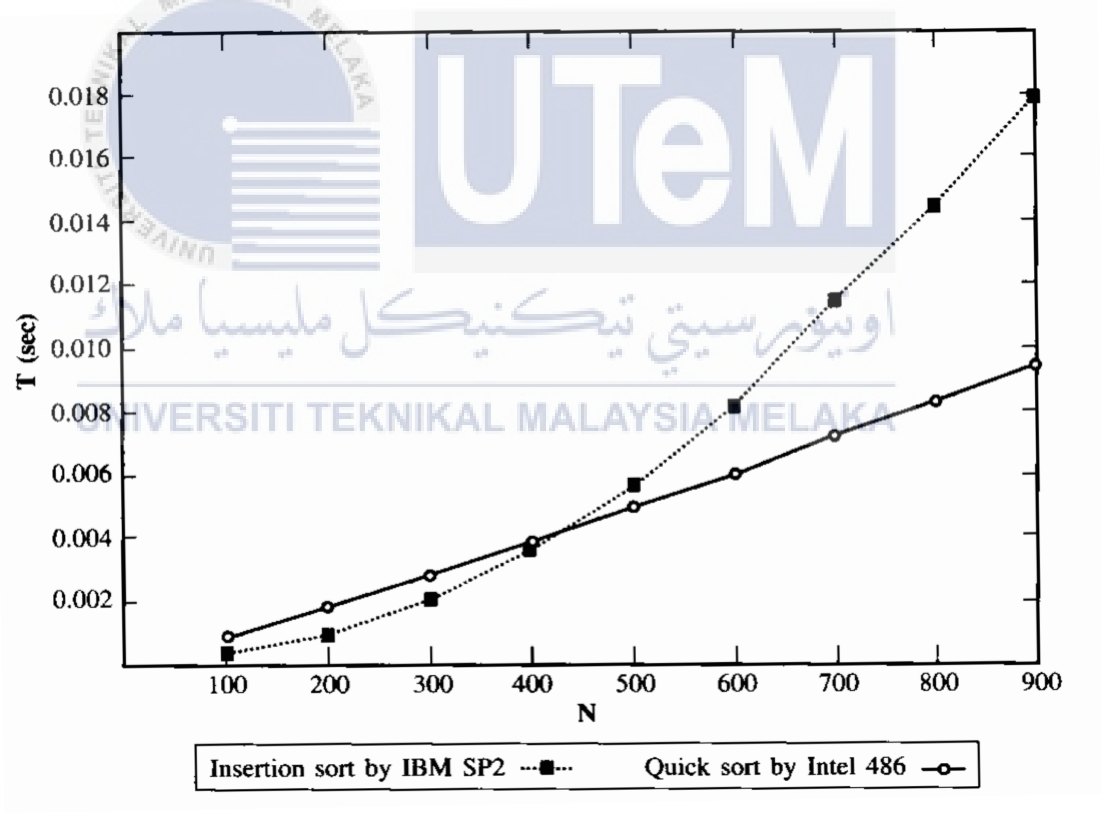


Figure 2.5: The comparison of the performance of insertion sort and quick sort.[10]

Hence, time complexity should be the focus of the analysis of the algorithm. Next, it is important to know that the time efficiency of an algorithm is not a measure of the running time of a program implementing the algorithm in unit of second or millisecond because the time taken is affected by the speed of particular computer, capability of the programmer, the complier used in generating the machine code and the difficulty of clocking the actual running time of the program. Instead, the basic operation which contributing the most to the total running time should be identified and computed for the number of times the basic operation is executed.

It is essential to distinguish the order of growth of an algorithm and Table 2.6 below demonstrates the significance of order of growth on the algorithm. Time complexity with lower order is better than that with higher order. The order of growth affected significantly to the running time as the input size *n* grows larger.

Table 2.6: Time complexity function with respect to problem size

Time complexity function	Problem size: *n*			
	10	10^2	10^3	10^4
$\log_2 n$	3.3	6.6	10	13.3
n	10	10^2	10^3	10^4
$n \log_2 n$	0.33×10^2	0.7×10^3	10^4	1.3×10^5
n^2	10^2	10^4	10^6	10^8
2^n	1024	1.3×10^{30}	$> 10^{100}$	$> 10^{100}$
$n!$	3×10^6	$> 10^{100}$	$> 10^{100}$	$> 10^{100}$

Meanwhile, computer scientists used three notations: O (big oh), Ω (big omega), and Θ (big theta) to compare and rank order of growth. In this research, O-notation (big oh) will be the only notation used to represent the time complexity of the algorithm. O-notation is defined as [11]

A function t(n) is said to be in O(g(n)), denoted t(n) $\in$ *O(g(n)), if t(n) is bounded above by some constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some nonnegative integer* n_0 *such that t(n)* $\leq$ *cg(n) for all n* $\geq n_0$

In addition, there are three situations have to be concerned for any algorithm which are the best case, the average case and the worst case. This is because many algorithms which running time depends not only on an input size but also on the specifics of a particular inputs.

2.4 3D Printing

3D printing or additive manufacturing is a process of creating a 3D product from digital model by adding successive layers of materials until the product is completed [12]. Each of these layers can be seen as a finely sliced horizontal cross-section of the final object. It is the reverse of subtractive manufacturing which the final product is bigger than the initial material [12]. It can produce complex shapes using less material than traditional manufacturing method. 3D printing was initially used in rapid prototyping and it had started to develop into next-generation manufacturing technology that has the potential to allow the local, on-demand production of final products or parts.

2.4.1 The process of 3D Printing

The 3D printing process flowchart is shown in Figure 2.6 below.

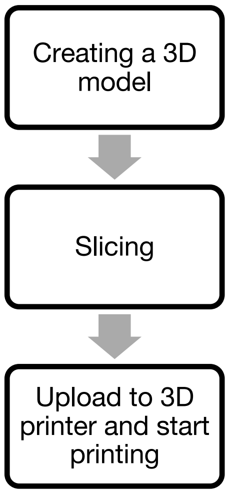


Figure 2.6: The 3D printing process flowchart

3D printing process starts with creating a 3D model by computer. The 3D model can be created either with 3D modelling software or based on data generated from 3D scanner. A STL file is obtained after the 3D model is created. STL (abbreviation of **ST**ereo**L**ithography) file encodes the surface geometry of the 3D model by tessellation.

Tessellation is a process of tiling a surface with one or more geometric shapes such that there are no overlaps or gaps. The outer two-dimensional surface of the 3D model is tessellated using tiny triangles (facets) [13] and the information of the facets is stored in a STL file. STL file format offers two ways to store the information which are in binary format or ASCII format [12]. Both of them represents the same information but they are different in the size of the file. Binary format file is smaller in size than that of ASCII format file, however binary format file is unreadable by human while ASCII format file can visually read and checked by human. Even though there are a lot of formats created for rapid prototyping, STL is the most widely adopted format by various developers of CAD packages [14].

The next stage is slicing which divide the 3D model into numerous thinly layers by using slicing software [1]. A tool path is then created and the information is bundled up into a G-code file which is the term used by 3D printer to produce the prototype [14]. It is important to notice that the quality of the print is influenced by the slicing software and its settings.

The final stage is upload the file to the 3D printer and the product is ready to be fabricated. The file can be uploaded via USB, SD or Wi-Fi depends on the brand and type of 3D Printer or AM machine. The fabrication of the product is done by adding layer by layer of successive materials until all the separated two-dimensional layers are reassembled as a 3D object on the print-bed.

2.4.2 Existing 3D Printing Robot

Nowadays, 3D printing is not long being undergo in a box instead it can be done in an open environment by using articulated robot. There are few tech company started to develop this technology and their details are shown in Table 2.7 below:

Table 2.7: The general information of existing 3D Printing Robot.

Developer	Branch Technology	Joris Laarman Lab	MIT
Robot	KUKA KR 120 R2500 robot	MX3D Metal Robot	5-axis Altec AT40GW mobile hydraulic arm and a 6-axis KUKA robotic arm
Project	Shop Pavillion, Cheekwood Playhouse and etc.	MX3D Bridge, Gradient Screen, Cucuyo, Buttlefly Screen and Dragon Benches	Digital Construction Platform (DCP)

As a partner of KUKA robotics company, Branch Technology combines industrial robot with 3D printing and utilises them in architectural fabrication. They adopted KUKA KR 120 R2500 robot to do the 3D printing with the aids of the algorithm they developed themselves. The geometry and robotics motion are created by the algorithm to build complex geometries in open space.

MX3D Metal Robot is the result of research made by Joris Laarman Lab associated with Acotech and supported by Autodesk. They combine industrial robot with welding machine and develop a software to drive the robot. The robot can 3D printing various 3D metal products with complicated shape because of the flexibility brought by its multiple axis.

A project named as "Digital Construction Platform (DCP)" was initiated by MIT for the purposes of improve the safety, speed and quality of construction; develop the automated construction technology which can be used in disaster, hazardous environment and interplanetary exploration [15]. The DCP consists of a 5-axis Altec AT40GW mobile hydraulic arm with a 6-axis KUKA robotic arm mounted at its endpoint [15]. These two robots imitate the biological model of human shoulder and hand, the large arm is used for gross positioning while the small arm can perform fine positioning, provide oscillation compensation and improve force control bandwidth.

2.5 Conclusion

The SP algorithm is the main focus on this chapter, the comparison between the three different algorithms is shown in a table. After the analysis, Dijkstra's algorithm is the most suitable algorithm for this project because it is fast and there is not necessary to deal with negative edge weights in this project. Time complexity is important for the analysis of algorithm and the running time of an algorithm hugely affected by the order of growth of the algorithm.

3D printing and its process are detailed in the later section, it shows how the 3D printing works and the elements inside it. STL file and slicing software play an important role in 3D printing, however it will not be included in the research scope. Then, the information of existing 3D printing robots is gathered and shown in a table. There are three technology company are developing this technology with different approach. All of them used 3D printing for architectural innovation and their works are worth to be referred as they are the pioneer in this sector.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter discusses about the procedure and the method used in this project. It covers all the elements involved in making this project to achieve its goal and objective. It gives the reader an insight on the process of this project from initial stage to final stage.

3.2 Project Flowchart

This project flowchart lists down the steps that are taken throughout this project. It shows from survey on prior research work to the simulation and experimental test. The project flowchart is shown in Figure 3.1.

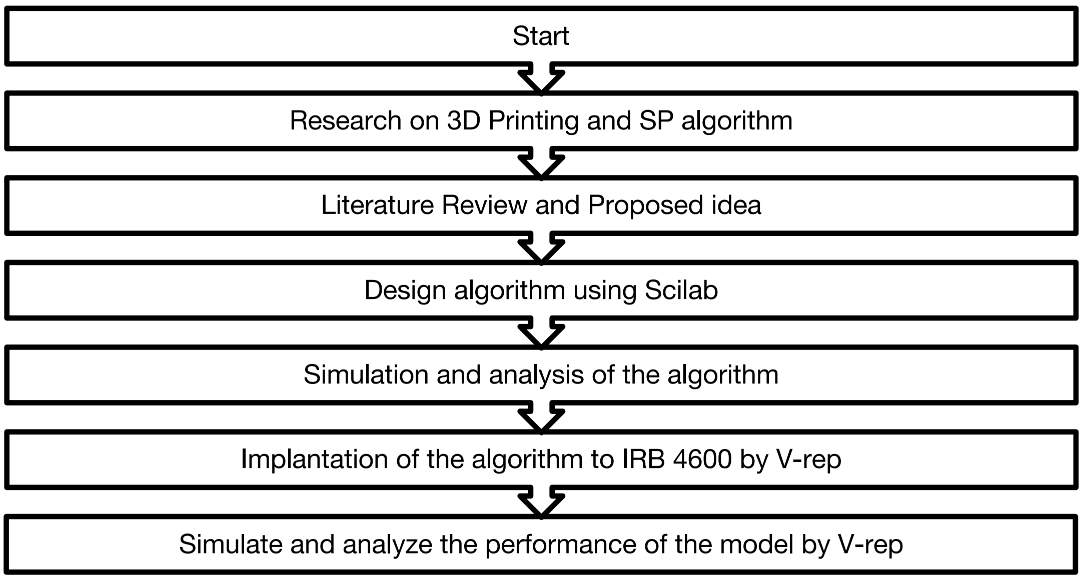

Start
Research on 3D Printing and SP algorithm
Literature Review and Proposed idea
Design algorithm using Scilab
Simulation and analysis of the algorithm
Implantation of the algorithm to IRB 4600 by V-rep
Simulate and analyze the performance of the model by V-rep

Figure 3.1: Project flowchart

3.3 Project Methodology Flowchart

The project methodology flowchart is shown in Figure 3.2.

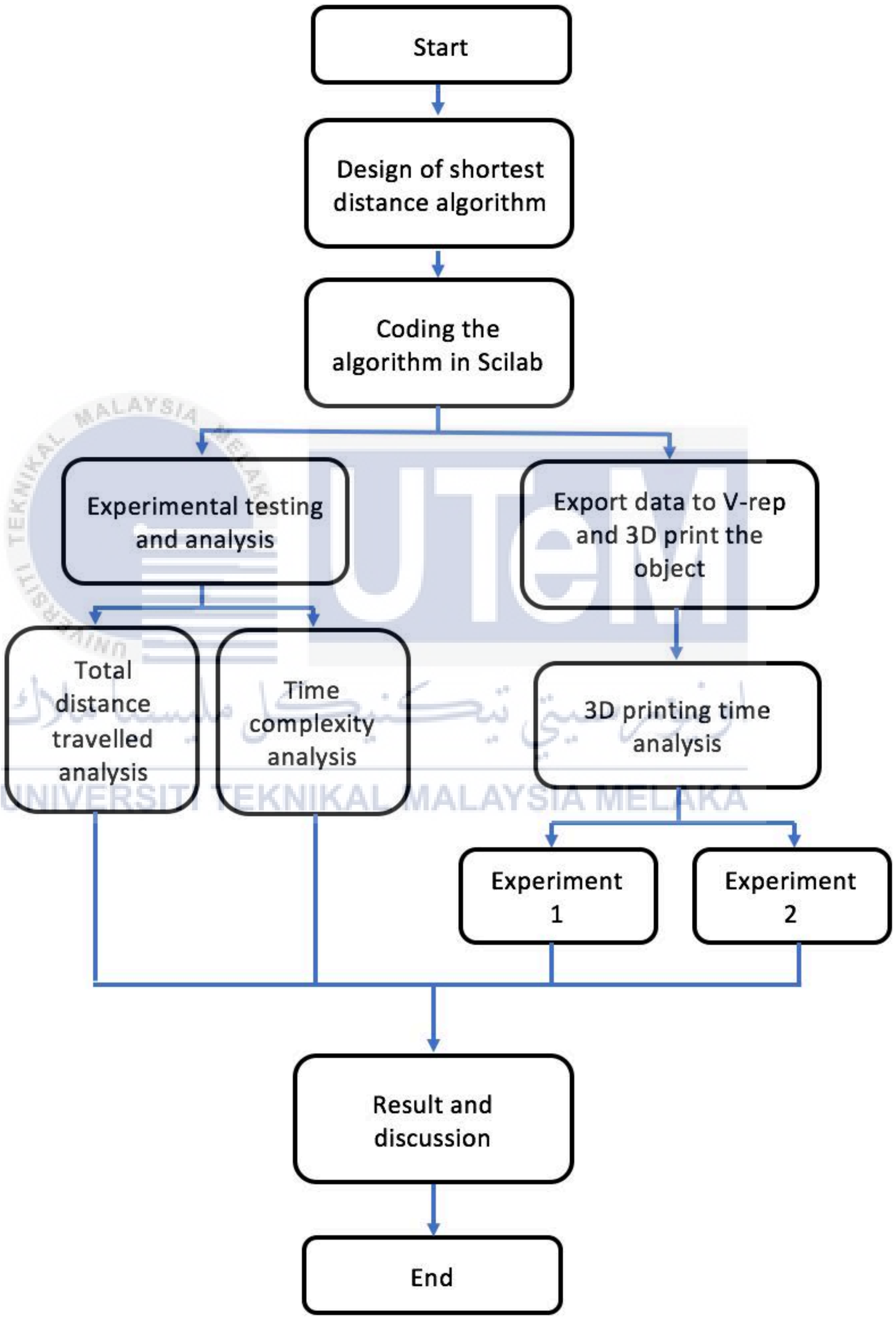

Start
Design of shortest distance algorithm
Coding the algorithm in Scilab
Experimental testing and analysis
Export data to V-rep and 3D print the object
Total distance travelled analysis
Time complexity analysis
3D printing time analysis
Experiment 1
Experiment 2
Result and discussion
End

Figure 3.2: Project methodology flowchart.

3.4 Theoretically description of proposed idea

The main focus of the idea is on finding the most efficient trajectory path to complete the 3D printing by implementation of shortest path algorithm. In this project, Dijkstra’s algorithm is referred as shortest path algorithm to achieve the goal. The overview of elements in the proposed idea is shown in Figure 3.3 below.

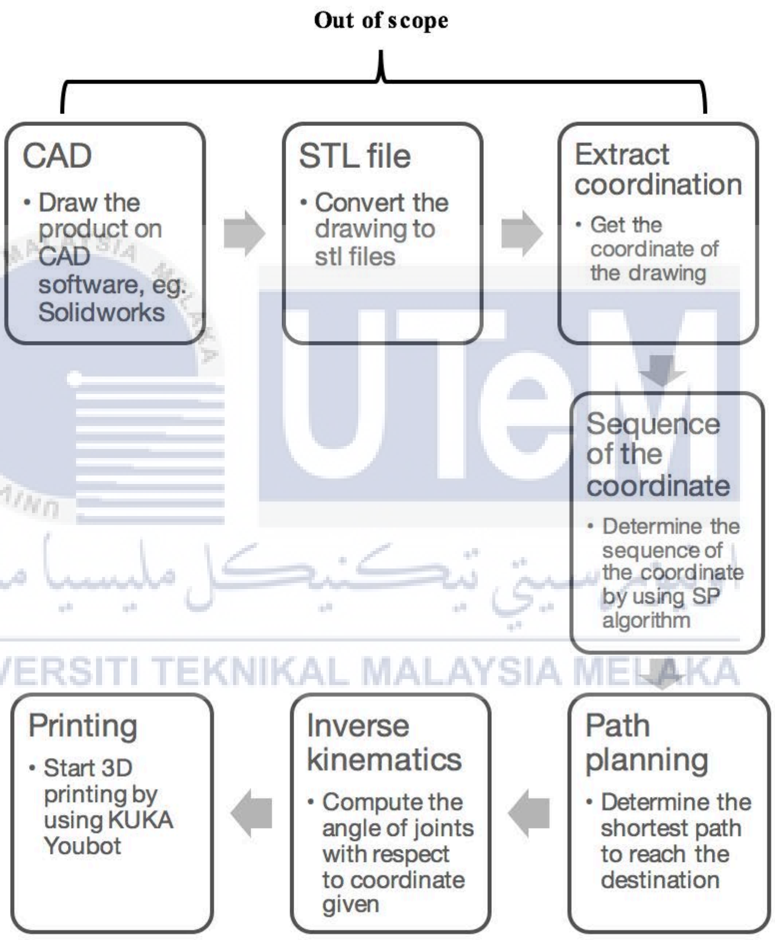

Out of scope
CAD
• Draw the product on CAD software, eg. Solidworks
STL file
• Convert the drawing to stl files
Extract coordination
• Get the coordinate of the drawing
Sequence of the coordinate
• Determine the sequence of the coordinate by using SP algorithm
Path planning
• Determine the shortest path to reach the destination
Inverse kinematics
• Compute the angle of joints with respect to coordinate given
Printing
• Start 3D printing by using KUKA Youbot

Figure 3.3: Overview of elements in the proposed idea.

From the figure, we can see that the stages from CAD drawing until extract coordination are not included in this project. Hence, the research is started from creating a U-shape box model. The algorithm will automatically generate the coordinate of the U-shape box and subsequently determine the shortest way to complete it.

IRB 4600-40-255 industrial robot is programmed using LUA language (a type of programming language) to apply shortest path algorithm in the 3D printing. The joint angles of the manipulator ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_{5,}, \theta_6$) corresponding to the Cartesian coordinate are calculated by using the inverse kinematic module in the V-REP simulator. The V-REP simulator also provides the position, velocity and acceleration profiles of every joints. The performance of the manipulator is tested, simulated and evaluated in the V-REP simulator before it can be tested in the reality. In the working environment, it is assumed that there is no obstacle along the path.

3.5 Shortest path algorithm design

The brief description of SP algorithm used in this project is shown in Figure 3.4 below.

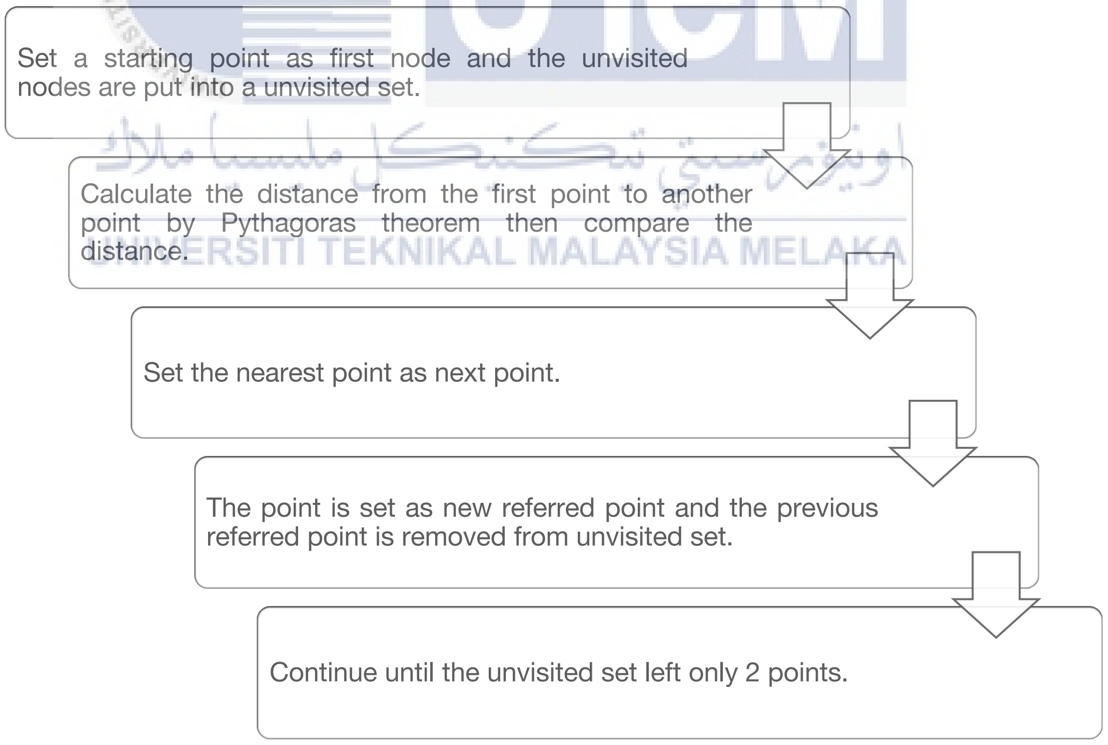

Set a starting point as first node and the unvisited nodes are put into a unvisited set.
Calculate the distance from the first point to another point by Pythagoras theorem then compare the distance.
Set the nearest point as next point.
The point is set as new referred point and the previous referred point is removed from unvisited set.
Continue until the unvisited set left only 2 points.

Figure 3.4: Flowchart of SP algorithm used in this project.

Distance calculation by Pythagoras theorem as (3.1)

$$\text{Distance, } d_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{3.1}$$

where	d	=	distance between the referred node and its neighbor nodes.
	x_i	=	x position of the referred node
	x_{i+1}	=	x position of the neighbor node
	y_i	=	y position of the referred node
	y_{i+1}	=	y position of the neighbor node

The demonstration of the SP algorithm used in this project is displayed in Figure 3.5.

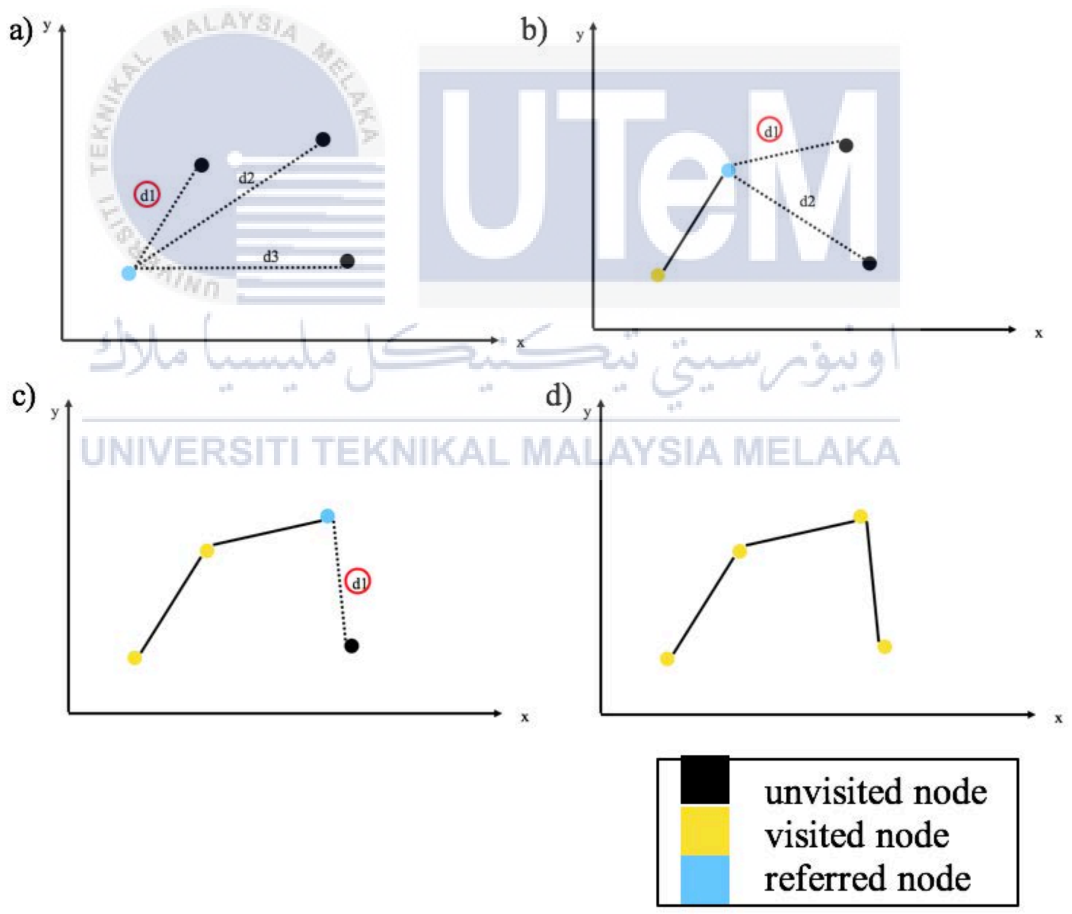


Figure 3.5: Demonstration of SP algorithm in graph.

a) The initial point is set as starting node and the distance between it and other points is calculated by using Pythagoras Theorem. Meanwhile, all unvisited points are put into an unvisited set.
b) The nearest point is selected and set as new referred point. On the same time, the previous referred point is removed from unvisited set.
c) The algorithm continue until there are only 2 points left in the unvisited set.
d) The sequence of the points is determined.

3.6 Find the joint angle by inverse kinematics module

In this project, the mathematical calculation of inverse kinematics is not being emphasized because this research is focusing on the trajectory generation only. Nevertheless, the joint angle of the robot can be found by using the inverse kinematics module (IK module) in V-REP simulator [16].

3.7 Find the velocity and acceleration of the joint

The velocity and acceleration can be obtained by using Jacobian method as shown (3.2) & (3.3) below [17].

$$\dot{\theta} = J(\theta)^{-1}.\dot{X} \tag{3.2}$$

$$\ddot{\theta} = J(\theta)^{-1}.(\ddot{X} - \left(\frac{d}{dt}J(\theta)\right).\dot{\theta}) \tag{3.3}$$

where θ = angle of joint

$\dot{\theta}$ = angular velocity of joint

$\ddot{\theta}$ = angular acceleration of the joint

J = Jacobian matrix

$\dot{X}$ = velocity of the end-effector

$\ddot{X}$ = acceleration of the end-effector

The angular velocity and acceleration of each joints are obtained from the V-REP simulator in this project.

3.8 Material and equipment

3.8.1 Scilab and V-REP simulator

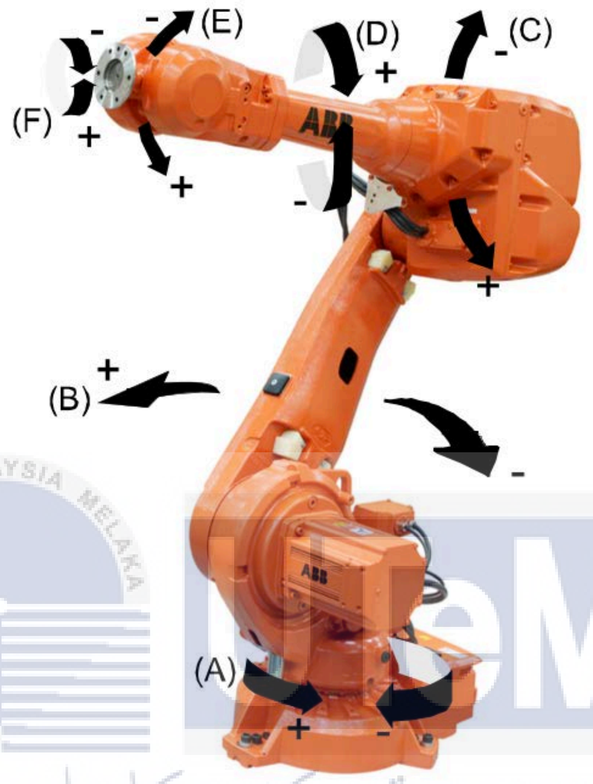
Scilab and V-REP are the simulator to simulate the performance of the algorithm and manipulator. Scilab is an open source, high-level numerical oriented programming language for numerical calculation, representation and programming. In addition, data analysis, numerical computation and creating model are done by using Scilab.

V-REP is a 3D robot simulator which allows the users to model, edit and program various type of manipulator [18]. It was chosen for this work since it currently has a free full license for education purposes, with all the characteristics of the commercial version, and also includes, on its library, the model of the IRB 4600-40-255 industrial robot [19]. The trajectory planning of the manipulator and time analysis are performed and simulated in the V-REP environment.

3.8.2 IRB 4600-40-255 industrial robot

IRB 4600 series is the product of ABB Robotics which comes in 4 versions, and they are IRB 4600-20-250, IRB 4600-40-255, IRB 4600-45-205 and IRB 4600-60-205. They are highly productive general-purpose robot boosted for short cycle times where compact robots can help create high density cells. They have ultra-wide working range with flexible mounting with floor, tilted, semi-shelf and inverted mounting.

The manipulator axes of the IRB 4600-40-255 industrial robot is displayed in the Figure 3.6.



Pos	Description	Pos	Description
A	Axis 1	B	Axis 2
C	Axis 3	D	Axis 4
E	Axis 5	F	Axis 6

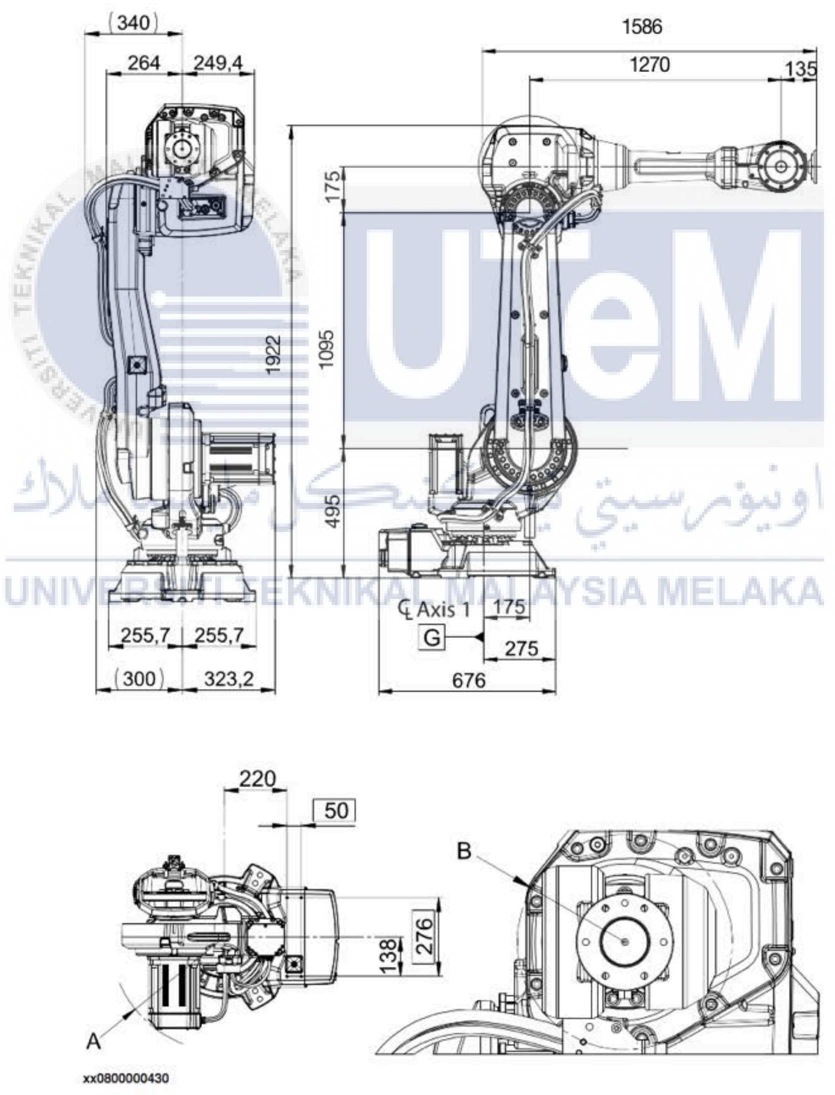
Figure 3.6: IRB 4600-40-255 industrial robot manipulator axes.[20]

General info

- Number of axes: 6
- Mounting: Floor, shelf, inverted or tiled
- Controller: IRC5 Single cabinet, IRC5 Dual cabinet
- Reach: 2.55 m
- Height: 1922 mm
- Weight: 425 kg

- Dimensions robot base: 512 x 676 mm
- Payload: 40 kg
- Armload: 20 kg
- Position repeatability: 0.06 mm
- Path repeatability: 0.28 mm
- Supply voltage: 200-600V, 50-60 Hz

The dimensions of the IRB 4600-40-255 industrial robot is shown in Figure 3.7.



Pos	Description
A	R 400 Minimum turning radius of axis 1
B	R 138 Minimum turning radius of axis 4

Figure 3.7: IRB 4600-40-255 industrial robot dimensions.[20]

The workspace of the IRB 4600-40-255 industrial robot is illustrated as Figure 3.8.

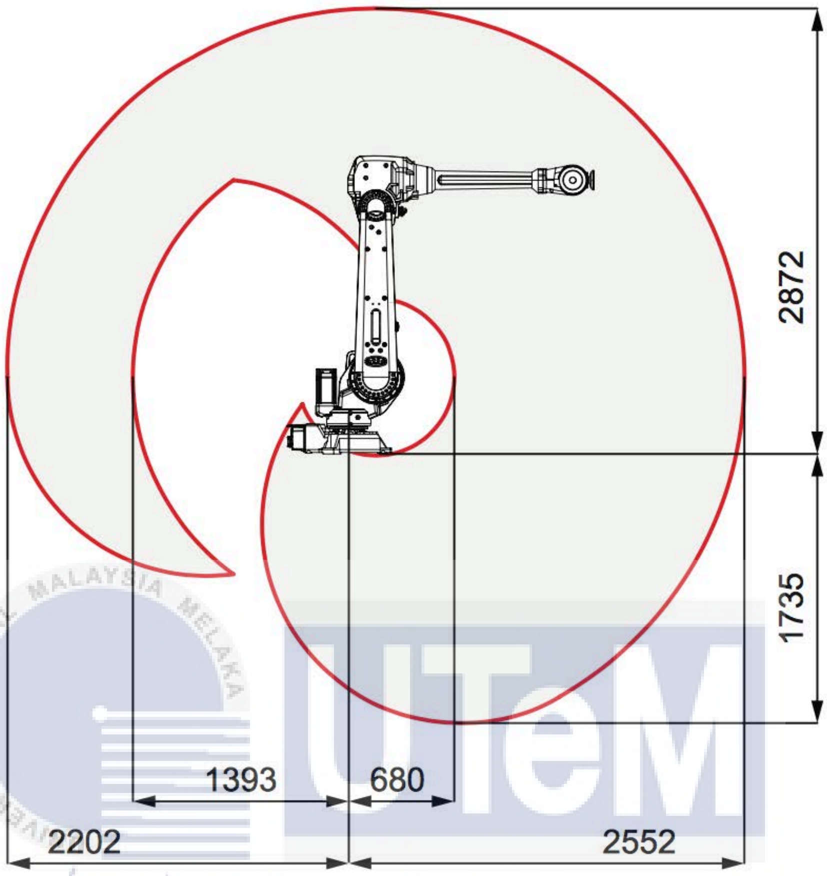


Figure 3.8: Workspace of IRB 4600-40-255 industrial robot.[20]

The working range and axis max speed of the IRB 4600-40-255 industrial robot are displayed in the Table 3.1.

Table 3.1: The working range and axis max speed of IRB 4600-40-255 industrial robot with respect to its joints. [20]

Axis movement	Working range	Axis max speed
Axis 1	$+180°\ to\ –180°$	175°/s
Axis 2	$+150°\ to\ –90°$	175°/s
Axis 3	$+75°\ to\ –180°$	175°/s
Axis 4	$+400°\ to\ –400°$	250°/s
Axis 5	$+120°\ to\ –125°$	250°/s
Axis 6	$+400°\ to\ –400°$	360°/s

3.9 Data collect and method of analysis

The data gathered in this project are total distance travelled from coordinate to coordinate, basic operation count in the algorithm, execution time of the program and time taken for a complete 3D printing.

The collected data will be analyzed by few method, such as total distance travelled analysis, time complexity analysis and 3D printing time analysis.

3.9.1 Total distance analysis

The objective of this analysis is to examine the efficiency of the algorithm on optimizing. The performance of the SP algorithm can be evaluated by consider the total cost of the outcome or in other words the total distance travelled [21].

In detail, two models with 100 random coordinates are firstly generated in Scilab as Figure 3.9 below.

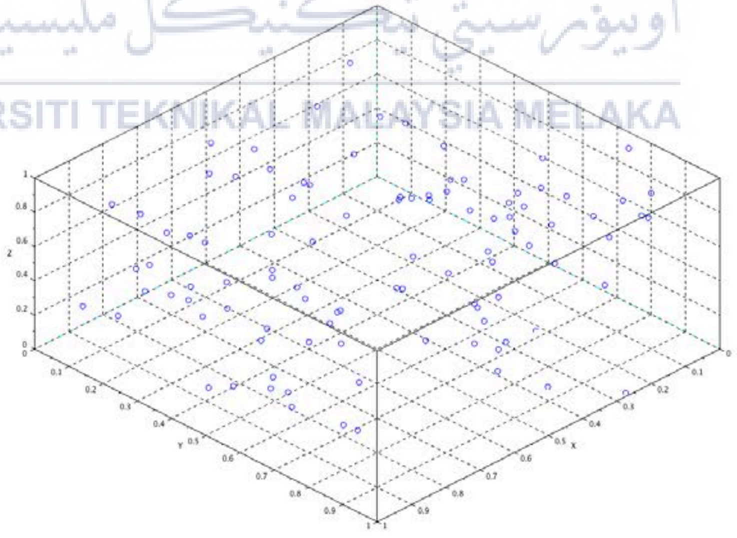


Figure 3.9: Model with 100 random coordinates.

Then, the algorithm is implanted on one of the models while another without the algorithm. The total distance travelled by both models are calculated and recorded.

The total distance travelled from coordinate to coordinate can be calculated mathematically by equation (3.4) & (3.5):

$$T = d_1 + d_2 + d_3 + d_4 + \cdots + d_{i-1} + d_i \tag{3.4}$$

$$T = \Sigma d_i \tag{3.5}$$

where T = Total distance travelled from coordinate to coordinate
d = Distance between the connected coordinates

The data are collected from 5 different set of models and the reduction percentage between two models is calculated. Subsequently, the mean of the reduction percentage is computed to show the average value of the result. Equation (3.6) to (3.8) are used to analyse the recorded data:

$$Reduction\ percentage = \left(1 - \frac{Total\ distance\ travelled\ of\ 1st\ model}{Total\ distance\ travelled\ of\ 2nd\ model}\right) \times 100\% \tag{3.6}$$

$$Mean, \mu = \frac{Sum\ of\ percentage\ ratio}{5} \tag{3.7}$$

After that, in order to examine the robustness of the algorithm, 5 different shapes in two dimensional are constructed in Scilab and their total distance travelled between coordinates are recorded and comparison between model with algorithm and without algorithm is presented in a table.

3.9.2 Time complexity analysis

In this analysis, the objective is to analyse the time efficiency of the algorithm. Basic operation count in the algorithm and execution time of the program are measured and evaluated mathematically after programmed in Scilab.

The method of calculating the time complexity started from decide on a parameter indicating an input size. Next, identify the algorithm's basic operation and check whether the number of times the basic operation is executed depends only on the size of an input. If it also depends on some additional property, the worst-case,

average-case and, if necessary, best-case efficiencies have to be investigated separately. Then, set up a sum expressing the number of times the algorithm's basic operation is executed. Finally, using standard formulas and rules of sum manipulation, either find a closed-form formula for the count or, at the very least, establish its order of growth [11].

Later, execution time of the algorithm on different number of input is measure and recorded. A graph of execution time versus number of input is plotted and compared it with the calculated time complexity in order to examine the result. Theoretical execution time is calculated by using equation (3.9) below:

$$Execution\ time, T(n) = c_{op}C(g(n)) \qquad ; c_{op} = constant \tag{3.9}$$

Where

$n = Number\ of\ input$

$c_{op} = Execution\ time\ of\ an\ algorithm's\ basic\ operation$

$C\big(g(n)\big) = Time\ complexity\ of\ the\ algorithm$

3.9.3 3D printing time analysis

The objective of this analysis is to evaluate the efficiency of the algorithm on 3D printing. It is made up of two experiments, first experiment is to investigate the relationship between the presence of algorithm and the time taken on 3D printing while second experiment is to analyse the efficiency of the algorithm based on starting coordinate.

The first experiment started by constructing two equal size U-shape box model in Scilab. Subsequently, one of them is implanted with algorithm while another doesn’t. The sequence of coordinates is generated in Scilab and exported to V-rep. Then, the model is 3D printed in v-rep for 2, 3, 4, 5 and 6 layers separately and the time taken for each of them is recorded. Lastly, the time taken for both models are compared in a table and a graph of time taken versus number of layers is plotted to observe the differences. On the other hand, in order to investigate the robustness of the design, 4 more models with different shape are being created and 3D printed in v-rep simulation environment separately. The time taken for completing each model are recorded and their comparison are displayed in a bar chart.

For second experiment, three models of equal size U-shape box are created and implanted with algorithm in Scilab. The starting coordinate are set as (0.1,0), (0.3,0) and (0.6,0) respectively. Next, the sequence of coordinates is generated in Scilab and exported to V-rep. The model is 3D printed in v-rep for 2, 4, 6, 8 and 10 layers separately and the time taken for complete the 3D printing is recorded. After all, the collected data are shown in a table and a graph of time taken versus number of layers is plotted to observe the differences.

CHAPTER 4

RESULT

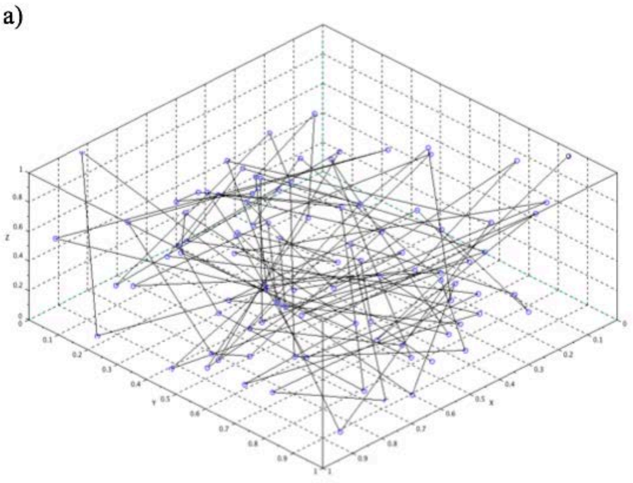
4.1 Introduction

In this chapter, the result will be presented with the aids of figures and tables. The result is obtained from the simulation and experimental analysis by Scilab and V-rep simulator.

4.2 Total distance analysis result

The Dijkstra’s algorithm is modified and applied to determine the sequence of the coordinates. In the initial phase, 100 random generated coordinates are used as a model to illustrate the performance of the algorithm. In addition, both the coordinates and the sequence are generated randomly by using Scilab. To demonstrate the performance of the algorithm, a model with algorithm and a model without algorithm are created and the comparison is made according to the outcome. The graphical result from Scilab is shown in Figure 4.1.

a)



b)

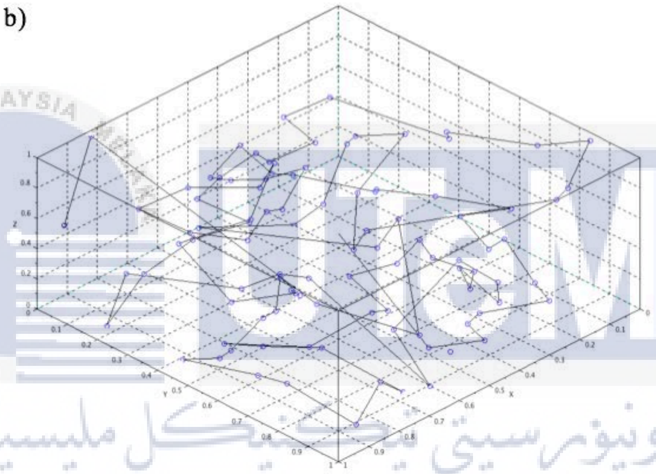


Figure 4.1: Graphical result from Scilab
a) Model without algorithm
b) Model with algorithm

From observation, the model without algorithm is disorganized and the sequence of the coordinates is random. Meanwhile, the model with algorithm is arranged in order.

In order to examine the efficiency of the algorithm, the total distance travelled by both models is measured and the result is shown in Table 4.1 below.

Table 4.1: Result of total distance travelled analysis.

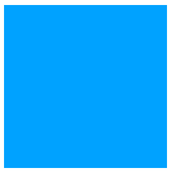
Model	Total distance travelled (m)				
	1st	2nd	3rd	4th	5th
Without algorithm	68.24	64.68	72.77	69.54	67.33
With algorithm	19.18	18.06	18.47	18.84	19.29
Reduction percentage	$\left(1-\frac{19.18}{68.24}\right) \times 100\% = 71.89\%$	$\left(1-\frac{18.06}{64.68}\right) \times 100\% = 72.08\%$	$\left(1-\frac{18.47}{72.77}\right) \times 100\% = 74.62\%$	$\left(1-\frac{18.84}{69.54}\right) \times 100\% = 72.91\%$	$\left(1-\frac{19.29}{67.33}\right) \times 100\% = 71.35\%$

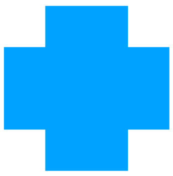
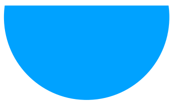
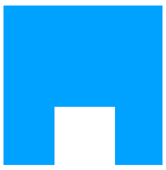
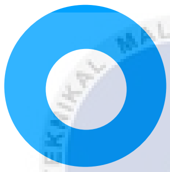
$$\text{Mean of reduction percentage, } \mu = \frac{71.89+72.08+74.62+72.91+71.35}{5} = 72.57\% \qquad (4.1)$$

In a nutshell, the total distance travelled between coordinates is able to be reduced to the mean of 72.57% in the presence of the algorithm.

Besides that, in order to investigate the robustness of the algorithm, few shapes are being created and the comparison of total distance travelled between two models are displayed on the Table 4.2.

Table 4.2: Comparison of total distance travelled among two models in different shapes.

Shape	Total distance travelled (m)		Reduction percentage
	Without algorithm	With algorithm	
	21.05	12.70	$\left(1-\frac{12.70}{21.05}\right) \times 100\% = 39.67\%$

	10.86	6.51	$\left(1-\frac{6.51}{10.86}\right)\times 100\%$ $= 40.06\%$
	27.16	15.88	$\left(1-\frac{15.88}{27.16}\right)\times 100\%$ $= 41.53\%$
	17.47	11.10	$\left(1-\frac{11.10}{17.47}\right)\times 100\%$ $= 36.46\%$
	28.65	17.19	$\left(1-\frac{17.19}{28.65}\right)\times 100\%$ $= 40.00\%$

In brief, the total distance travelled between coordinates on different shapes can be reduce to 36.46% ~ 41.53% in the presence of the algorithm. However, the efficiency of the algorithm on shapes are lower than that compared with the 100 random numbers model. The factor that caused the result is the variability of the sequence of input data.

4.3 Time complexity analysis result

After the algorithm is being programmed in Scilab, the basic operation of the program is defined and shown below:

```
ALGORITHM
//Determine the sequence of the coordinates by choosing nearest coordinate.
//Input: An array of random coordinates $A[c_1, c_2, \ldots, c_{n-1}, c_n]$
//Output: The coordinates are arranged by nearest coordinate strategy.
Line 1          While $n>2$ do
Line 2          for i = 2 to n-1 do
Line 3    $d = \sqrt{c_1^2 - c_i^2}$    //Find the distance by Pythagoras Theorem
Line 4    M = min(d)                    //Find the minimum distance
Line 5    A [$c_1$]= {}                 //Eliminate referenced coordinate from the set
Line 6    $c_M = c_1$                   //Coordinate M become referenced coordinate
Line 7    end
```

In the above algorithm, *n* number of coordinates are firstly included in a set of A. When the size of *n* is more than 2, it will fall into the for loop. In the for loop, the distance between coordinates will be calculated by Pythagoras Theorem from *n*-1 to 2 times in each loop which the distance to last coordinate is not necessary to be calculated. In a meanwhile, the minimum distance or nearest coordinate from referenced coordinate will be selected as new referenced coordinate and the previous referenced coordinate will be eliminated from the set A. Moreover, the algorithm will access *n*-1 coordinates once so that is no need to distinguish among the worst, average and best cases.

Let us denote G(*n*) the number of times this operation is executed. There are 3 operations from line 4 to line 6 that will be executed once per loop hence there is constant number 3 times with *n*-1 for n number of coordinates.

$$G(n) = \sum_{i=3}^{n-1}(i+3) = 3(n-1) + \sum_{i=3}^{n-1} i \qquad (4.1)$$

For distance calculation, the algorithm will repeat the loop from *n*-1 times to finally 2 times. For example, if there are 4 coordinates, the first loop will calculate 3 times, the second loop will calculate 2 times and the loop will stop at there as only left the last coordinate in the set A. Hence, according to Summation Formulas:

$$\sum_{i=3}^{n-1} i = (n-1) + (n-2) + \cdots + 3 + 2 = \frac{n(n+1)}{2} \approx \frac{1}{2}n^2 \qquad (4.2)$$

Hence, the time complexity of the algorithm in O(g(n) is

$$G(n) = 3(n-1) + \frac{1}{2}n^2 = O(n^2) \tag{4.3}$$

The execution time, *t* for different number of input, *n* is recorded and a graph of execution time versus number of input is plotted and shown in Figure 4.2 below.

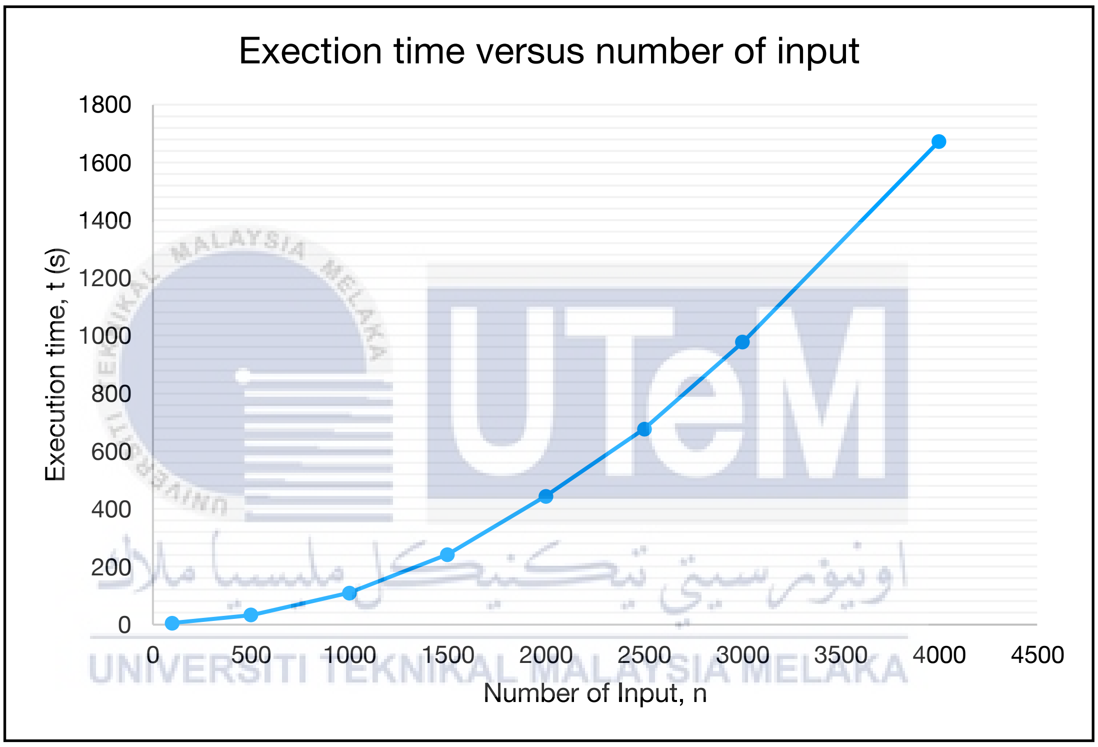


Figure 4.2: Graph of execution time versus number of input.

From the Figure 4.2 above, we can see that the execution time increased quadratically as the input size growth. In order to examine and compare the collected data, a graph is plotted in Scilab according to the calculated equation (4.4) below:

$$Execution\ time, T(n) = c_{op}\left(3(n-1) + \frac{1}{2}n^2\right) \qquad ; c_{op} = constant \tag{4.4}$$

Where

$$n = Number\ of\ input$$

$$c_{op} = Execution\ time\ of\ an\ algorithm's\ basic\ operation$$

The result is shown in Figure 4.3 below,

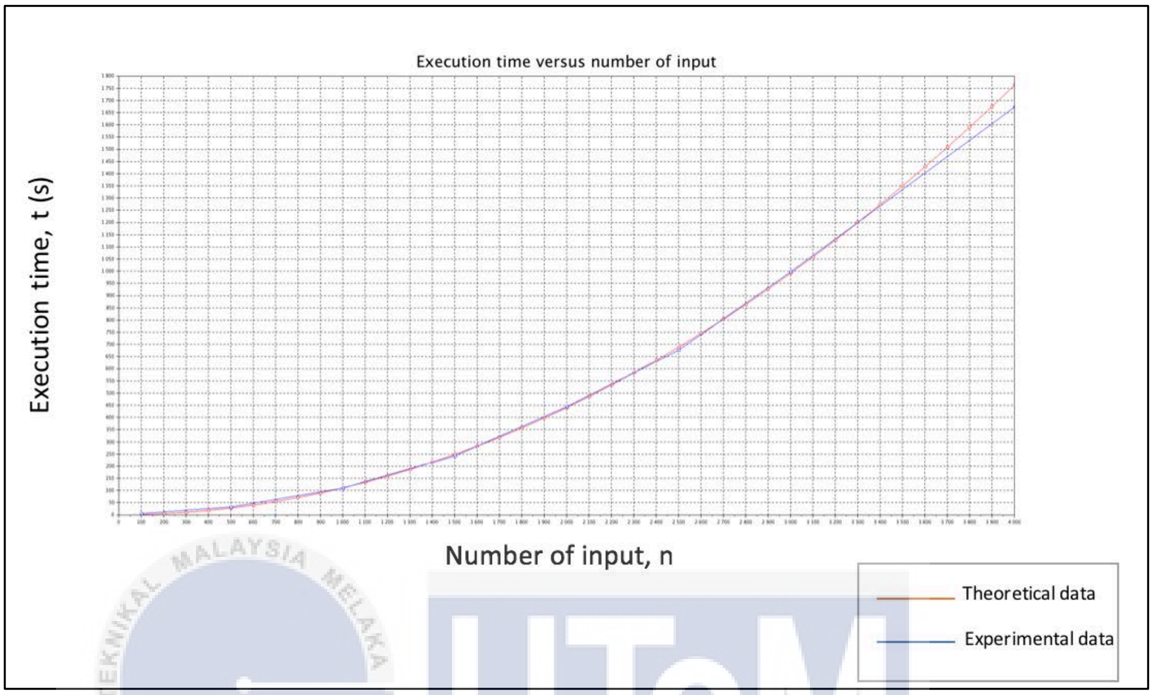


Figure 4.3: Comparison between theoretical and experimental data.

Overall, the experimental data is matched with the theoretical data except for the last segment and this may be caused by the variability of input data. In conclusion, the time complexity of the algorithm is $O(n^2)$ and it is proved by comparing theoretical data with experimental data.

4.4 3D printing time analysis result

The model of U-shape box with dimension 1x1 which created in Scilab is shown in Figure 4.4.

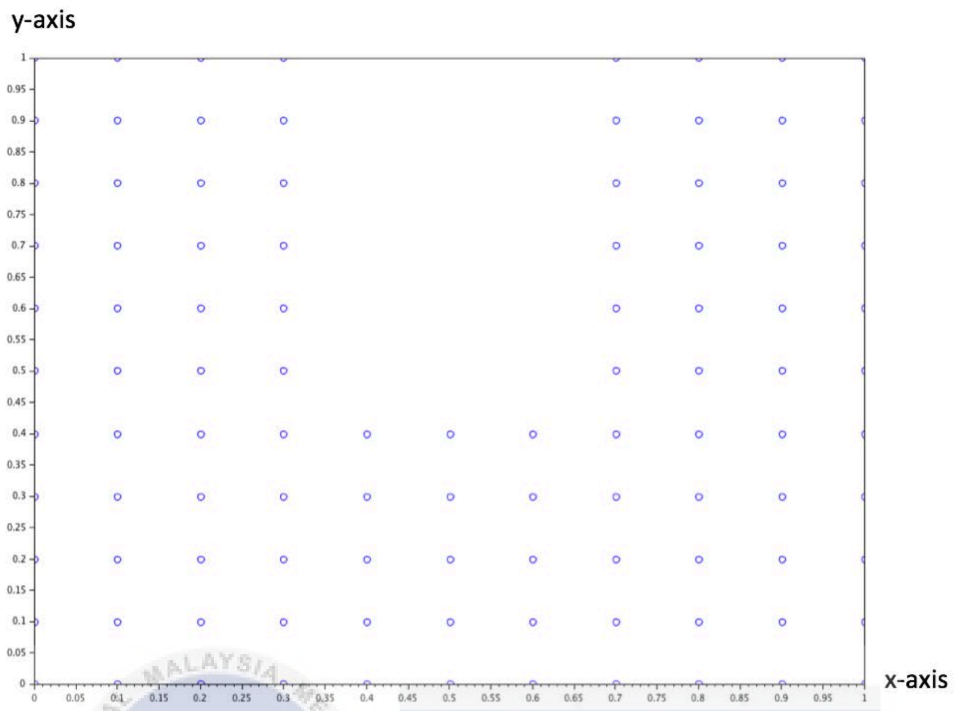


Figure 4.4: Scatter graph of U-shape model in Scilab.

Figure 4.5 below is showing the simulation in v-rep.

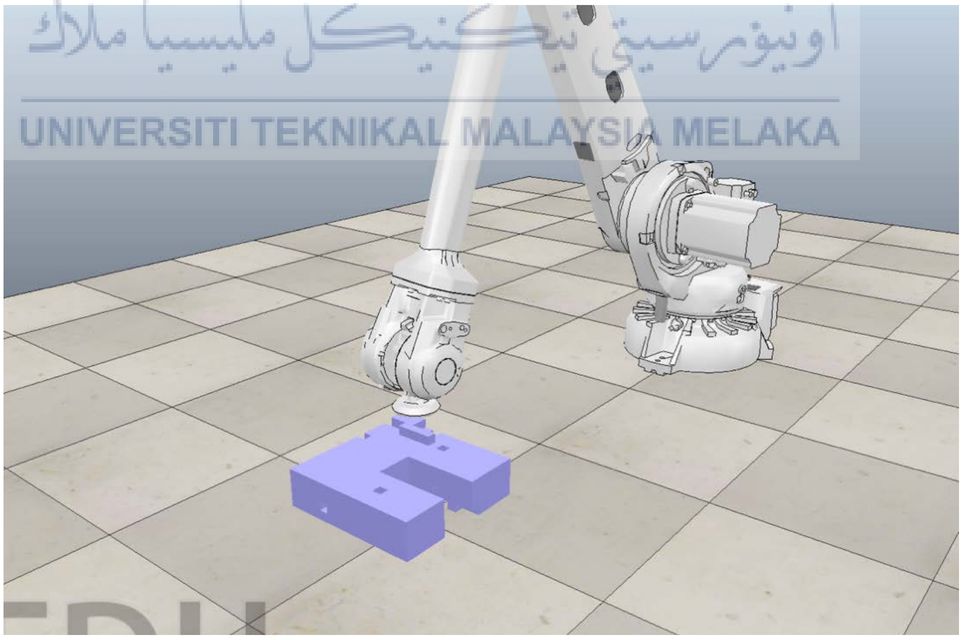


Figure 4.5: Simulation in V-rep

<u>Experiment 1</u>

Figure 4.6 below shows the trajectory path of both models.

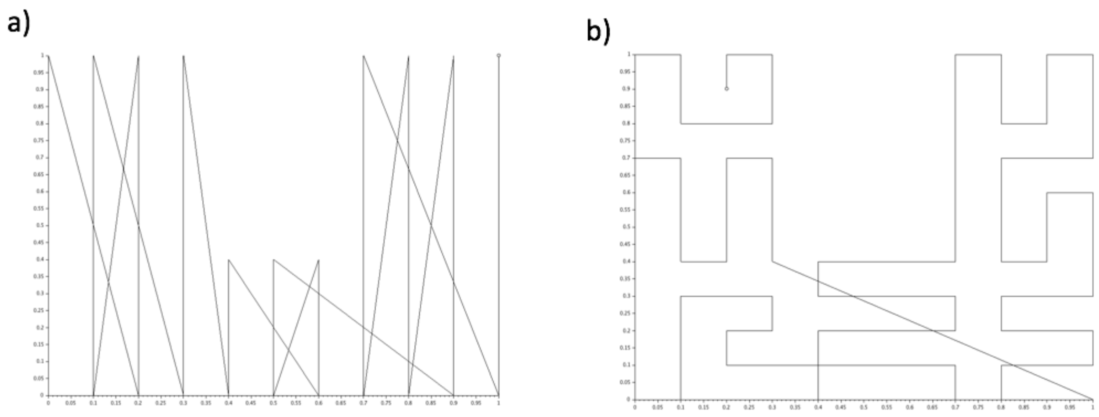


Figure 4.6: Trajectory path of a) Model without algorithm; b) Model with algorithm

The graph in Figure 4.7 below displays the comparison of time taken between two models on different number of layers.

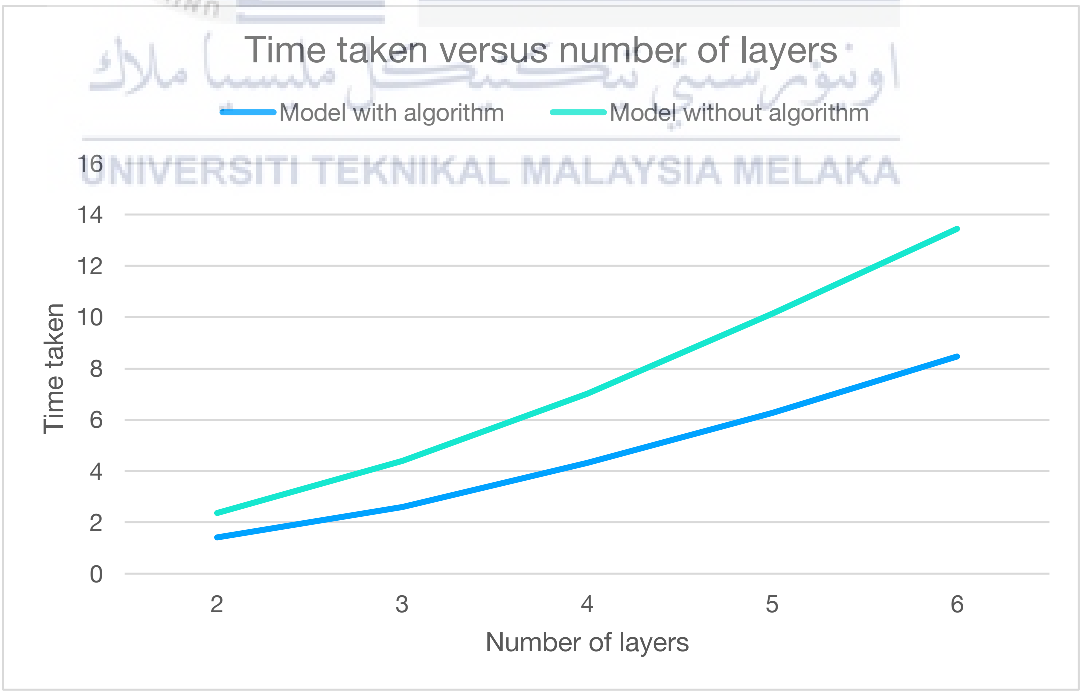
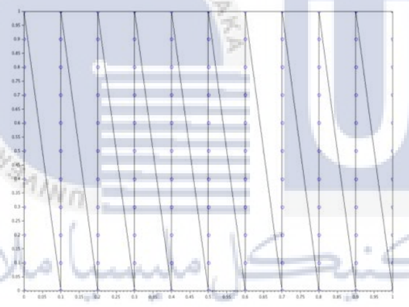
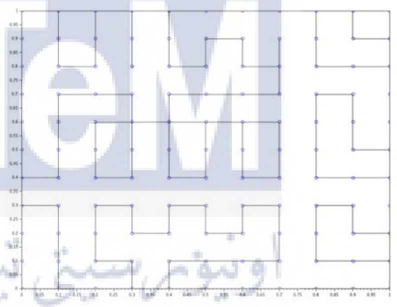
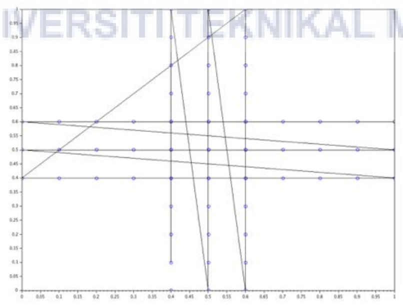
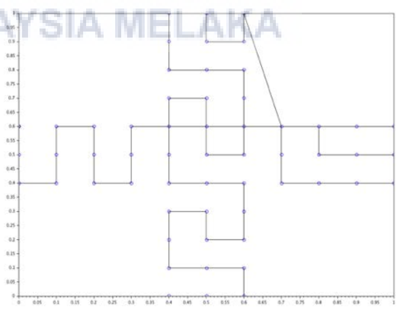
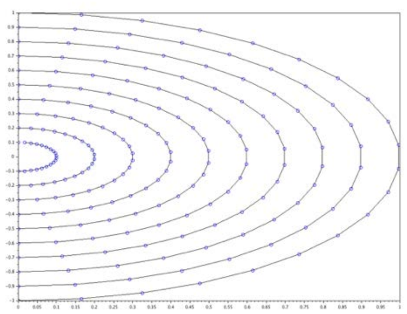
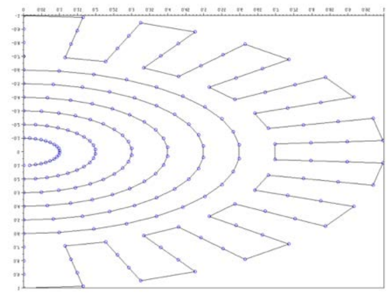


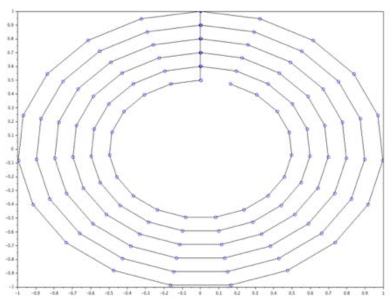
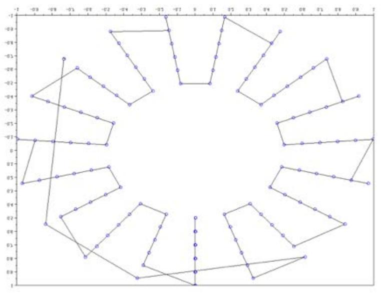
Figure 4.7: Time taken versus number of layers for experiment 1.

From the graph above, we can notice that the time taken for model with algorithm is always less than that of model without algorithm. Besides that, the time gap between them is increasing as the number of layers increased. This indicates that with the trajectory path generated by algorithm able to decrease the time taken to do the 3D printing. As the number of layers increases, the effectiveness of the algorithm is getting larger.

In order to examine the robustness of the algorithm on 3D printing, another 4 models with different shapes are created and 3D printed with 5 layers. The result is shown in Table 4.3 and Figure 4.8 below.

Table 4.3: Comparison of trajectory path between two models in different shapes.

Shapes	Without algorithm	With algorithm
Square		
Plus		
Semi-circle		

Donut		

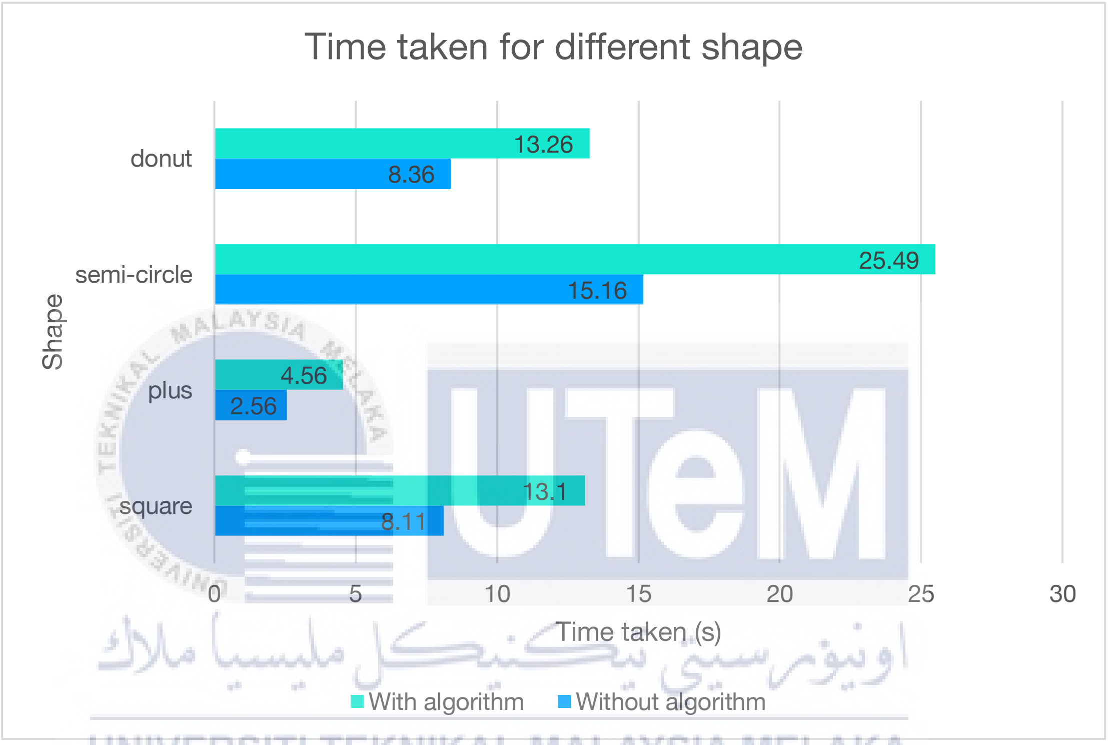


Figure 4.8: Comparison of time taken for 3D printing between two models in different shape.

Overall, the efficiency of the 3D printing on different shapes can be increased with the implementation of the algorithm. However, the magnitude of the optimizing is different according to the complexity and the volume of the object.

Experiment 2

Figure 4.9 below shows the trajectory path of three models.

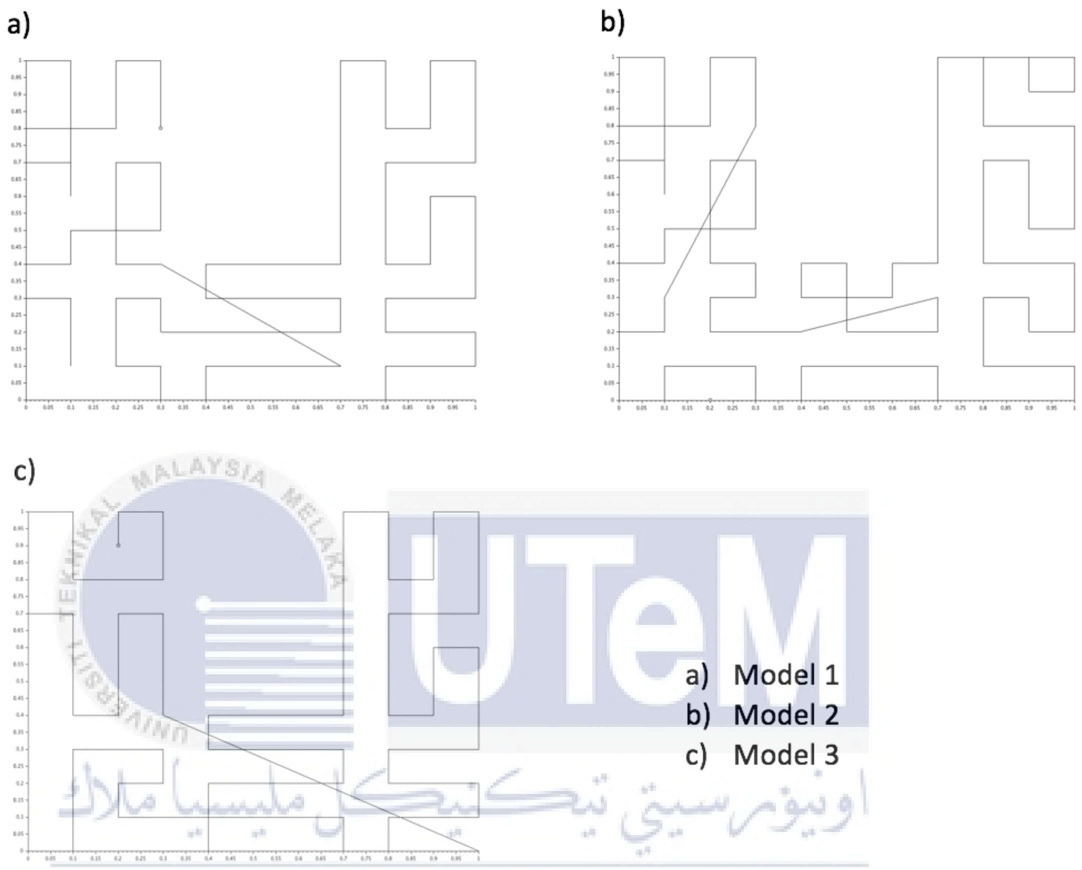


Figure 4.9: Trajectory path of three models.

Figure 4.10 below displays the comparison of time taken between three models on different number of layers.

Model 1 starting coordinate at (0.1,0)

Model 2 starting coordinate at (0.3,0)

Model 3 starting coordinate at (0.6,0)

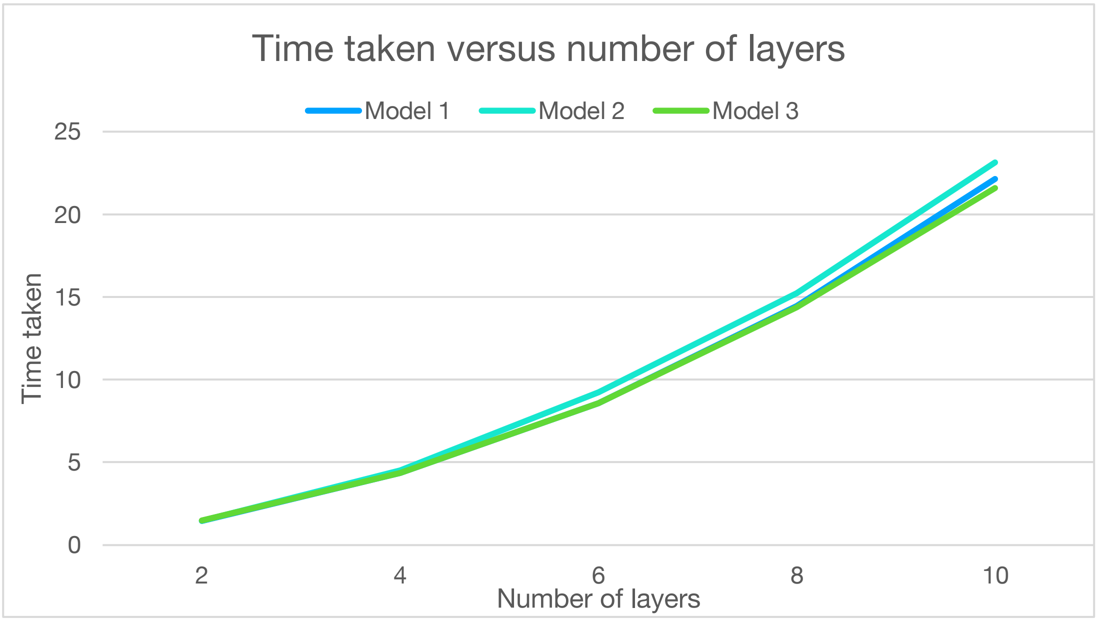


Figure 4.10: Comparison of three models in graph.

From the observation, time taken of model 1 and 3 which started at (0.1,0) and (0.6,0) respectively are almost equal while time taken is longer for model 2 which started at (0.3,0) compare to both model 1 and 3. This shows that the starting coordinate may influence the efficiency of algorithm on 3D printing. The trajectory path is different when the starting coordinate is altered.

In conclusion, trajectory path affects the efficiency of 3D printing and the algorithm can generate a more efficient path. The effectiveness of the algorithm is increasing proportionally with the increasing of volume of the object. Meanwhile, the algorithm may start at any coordinate and it will affect the efficiency. With different starting coordinate, the algorithm may generate a more efficient or less efficient trajectory path.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In conclusion, the shortest distance algorithm is designed based on Dijkstra's algorithm. The result shows positive outcome which indicate that the algorithm is able to increase the efficiency of 3D printing.

Three sets of experimental analysis are conducted to evaluate the performance of the algorithm. The analysis result is presented by equation, calculation or comparison with the aids of figures and tables. The comparison graph is generated from Scilab and Microsoft Excel to illustrate the difference between few models.

The total distance travelled analysis shows that the total distance travelled between coordinates is reduced to a mean of 72.57% with standard deviation of 1.14 in the presence of the algorithm. In addition, robustness testing shows that the algorithm still able to optimize the travelling path to 36.46% ~ 41.53%. Moreover, the time complexity analysis indicates the algorithm is in order of growth of n^2 which representing time complexity of $O(n^2)$. The result is proven by comparing theoretical result which generated by Scilab with experimental result.

Subsequently, 3D printing time analysis illustrates the efficiency of the algorithm on 3D printing. Experiment 1 shows that the 3D printing speed is respectively high when the algorithm is implanted. Besides that, the robustness test indicates that the 3D printing speed can still be improved on different shapes. Experiment 2 denotes the starting coordinate of the 3D printing may affects the speed of the 3D printing as the trajectory path will be different for different starting point.

Lastly, most of the research in this project are focus on the design and analysis of the shortest distance algorithm, such as analysis of time complexity. Besides that, fundamental knowledges about 3D printing, Scilab and V-rep simulation are also learned from this research.

5.2 Future work

The future work of this project is to increase the speed of 3D printing by improve the efficiency of the shortest distance algorithm. Although the result is showing positive outcome, however, the trajectory path that generated by the algorithm still is not the optimum result. Besides that, it is recommended to establish a research on extracting coordinates from CAD drawing for variety of 3D printed objects. It can strengthen this research on its robustness testing by analyse the result from various 3D printed objects. Finally, conducting the experiment in real environment is suggested in order to investigate the effects of external factors on the performance of the 3D printing.

REFERENCES

[1] R. C. Luo and P.-K. Tseng, “Trajectory generation and planning for simultaneous 3D printing of multiple objects,” *2017 IEEE 26th Int. Symp. Ind. Electron.*, pp. 1147–1152, 2017.

[2] H. J. Nyman and P. Sarlin, “From bits to atoms: 3D printing in the context of supply chain strategies,” *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 4190–4199, 2014.

[3] K. H. Hee, P. C. Youn, and L. M. Cheol, “A study on the 3D printing simulator for construction and application of robust control Using SMCSPO,” *2017 IEEE Int. Conf. Multisens. Fusion Integr. Intell. Syst.*, 2017.

[4] A. Pepe, D. Chiaravalli, and C. Melchiorri, “A hybrid teleoperation control scheme for a single-arm mobile manipulator with omnidirectional wheels,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2016–Novem, pp. 1450–1455, 2016.

[5] K. Lee, J. Oh, O. Sim, H. Bae, and J. H. Oh, “Inverse kinematics with strict nonholonomic constraints on mobile manipulator,” *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 2469–2474, 2017.

[6] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numer. Math.*, vol. 1, pp. 269–271, 1959.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithm*, Second Edi. The MIT Press, 2001.

[8] R. H. Möhring, H. Schilling, B. Schütz, D. Wagner, and T. Willhalm, “Partitioning graphs to speedup Dijkstra’s algorithm,” *ACM J. EA*, vol. 11, pp. 1–29, 2006.

[9] X. Yang, D. Liu, L. Cong, and L. Liang, “Shortest path algorithm based on distance comparison,” pp. 3137–3139, 2014.

[10] R. C. T. Lee, S. S. Tseng, R. C. Chang, and Y. T. Tsai, *Introduction to the Design and Analysis of Algorithms*. McGraw-Hill Education (Asia), 2005.

[11] A. (Villanova U. Levitin, *Introduction to The Design & Analysis of Algorithms*, 2nd ed. Pearson Education, 2007.

[12] C. C. K., L. K. F., and L. C. S., *Rapid Prototyping: Principles and Applications*,

Second Edi. World Scientific Publishing Co. Pte. Ltd., 2003.

[13] V. Seheda, “Paralepiped-plane optimizing algorithm for finding the trajectory of cutting instrument in full perpendicular processing of three-dimensional component through the information obtained from the STL-file,” *Mod. Probl. Radio Eng. Telecommun. Comput. Sci. 2008 Proc. Int. Conf.*, pp. 59–62, 2008.

[14] A. C. Brown and D. De Beer, “Development of a stereolithography (STL) slicing and G-code generation algorithm for an entry level 3-D printer,” *IEEE AFRICON Conf.*, 2013.

[15] S. J. Keating, J. C. Leland, and L. C. and N. Oxman, “Toward site-specific and self-sufficient robotic fabrication on architectural scales,” *Sci. Robot.*, vol. Vol. 2, no. Issue 5, 2017.

[16] V. Vladareanu, S. B. Cononovici, R. I. Munteanu, H. Wang, Y. Feng, and L. Vladareanu, “Modelling Inverse Kinematics for Virtual Environment Robot Simulation,” *Proc. - 2017 21st Int. Conf. Control Syst. Comput. CSCS 2017*, pp. 500–505, 2017.

[17] R. K. Mittal and I. J. Nagrath, *Robotics and control*. Tata McGraw-Hill Publishing Company Limited, 2003.

[18] E. Rohmer, S. P. Singh, and M. Freese, “V-rep: A versatile and scalable robot simulation framework,” *Intell. Robot. Syst. 2013 IEEE/RSJ Int. Conf. on. IEEE*, pp. 1321–1326, 2013.

[19] R. F. T. Alen and M. F. Silva, “Development and simulation on V-REP of an algorithm for the RoboCup@Work BNT,” *2014 IEEE Int. Conf. Auton. Robot Syst. Compet. ICARSC 2014*, pp. 315–320, 2014.

[20] A. A. Robotics, “Data sheet IRB 4600 features and specification,” 2018. [Online]. Available: https://library.e.abb.com/public/241c8d349f9140839b6bd453f911ac45/IRB4600_ROB0109EN_J_datasheet.pdf?x-sign=UqzkUZ/cqzAWEWu97+z4/sI1iWhCuLz2RL4VYVIoTFQJJC1r+jwpADlXiAo0zny3.

[21] B. Rahnama, M. C. Ozdemir, Y. Kiran, and A. Elci, “Design and implementation of a novel weighted shortest path algorithm for maze solving robots,” *Proc. - Int. Comput. Softw. Appl. Conf.*, pp. 328–332, 2013.

APPENDIX A

Total distance travelled analysis coding in Scilab

```
x=rand(1,100)
y=rand(1,100)
z=rand(1,100)
[j,s]=size(x)

scatter3(x,y,z)
comet3d(x,y,z)
j=1
Q(1)=0
r=1
while j<s-1
    j=1:s-1
    p=sqrt((x(j)-x(j+1))^2+(y(j)-y(j+1))^2+(z(j)-z(j+1))^2)
end
Q=sum(p)

scf()
scatter3(x,y,z)
E(1)=0
q=1
cx(1)=x(1);cy(1)=x(1);cz(1)=x(1);
while s>2
[j,s]=size(x)
i=1:s-1
d=sqrt((x(1)-x(i+1))^2+(y(1)-y(i+1))^2+(z(1)-z(i+1))^2)

[m,k]=min(d)
c1= x(k+1)
c2= y(k+1)
c3= z(k+1)
```

```
[row,column]=find(x==c1 & y==c2 & z==c3)
x(:,column)=[]
y(:,column)=[]
z(:,column)=[]
x(1)=c1
y(1)=c2
z(1)=c3
cx(q+1)=c1
cy(q+1)=c2
cz(q+1)=c3
E(q)=m
q=q+1
crdx=cx'
crdy=cy'
crdz=cz'

end
N=sum(E)
comet3d(crdx,crdy,crdz)
```