REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Mechanical Engineering

OBSTACLES AVOIDANCE FOR PICK-PLACE MOBILE ROBOT IN PRODUCTION LINE BASED ON IMAGE PROCESSING

Zaid Mohammed Abd Ali AL-ZUBAIDI

Master's Thesis

Supervisor

Asst. Prof. Dr. Serdar AY

Co-Supervisor

Asst. Prof. Dr. Mohanned AL-KHAFAJI

Istanbul, 2023

OBSTACLES AVOIDANCE FOR PICK-PLACE MOBILE ROBOT IN PRODUCTION LINE BASED ON IMAGE PROCESSING

Zaid Mohammed Abd Ali AL-ZUBAIDI

Mechanical Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2023

The thesis titled OBSTACLES AVOIDANCE FOR PICK-PLACE MOBILE ROBOT IN PRODUCTION LINE BASED ON IMAGE PROCESSING prepared by ZAID AL-ZUBAIDI and submitted on 14/04/2023 has been **accepted unanimously** for the degree of Master of Science in Mechanical Engineering.

Asst. Prof. Dr. Mohanned Al-KHAFAJI	Asst. Prof. Dr. Serdar AY
Co-Supervisor	Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Serdar AY	Department of Mechanical Engineering, Altınbaş University	_____
Asst. Prof. Dr. İbrahim KOÇ	Department of Mechanical Engineering, Altınbaş University	_____
Asst. Prof. Dr. Haydar UYANIK	School of Applied Science Pilotage, KTO Karatay University	_____

I hereby declare that this thesis meets all format and submission requirements for a Master's Thesis.

Submission date of the thesis to Institute of Graduate Studies: ___/___/___

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Zaid AL-ZUBAIDI

Signature

DEDICATION

This research is dedicated to the supervisor who has believed in me since the first time I asked him for supervision. Whose patience and strict following up and notes have empowered and reinforced this study, as well as to the co-supervisor who spent a great deal of time and effort guiding the outlines and the detailed works, and to My uncle Prof. Dr. Medhat AL-ZUBAIDI and my boss Mr. Ali HADDAD for the valuable financial and logistical support for this research. Certainly, to my wife, who is my entire life, and to my lovely children for their daily support, and to the special man of my life, Mohammed AL-ZUBAIDI, who is my father, and to my everything who is my mother, Fituh AL-TUMAA, and to my encouraging sisters Dr. Noor AL-ZUBAIDI and MSc Biochemist Sara AL-ZUBAIDI, all my family who instilled in me the love of learning, effort, and ambition, and who were by my side every step of the way on my academic path.

ACKNOWLEDGEMENTS

I would like firstly to thank God for giving me the strength and patience to carry out my MSc. Mechanical Engineering Program and giving me the ability and knowledge to develop and successfully submit this thesis. Then I would like to express my sincere gratitude and thanks to my thesis Supervisor, Asst. Prof. Dr. Serdar AY and the co-supervisor Dr. Mohanned M.H. Al-Khafaji For the patience, encouragement, and support for me throughout the writing period, I hope they would be able to reflect upon this thesis with pride and great satisfaction. I would also like to thank all the members of the jury responsible for the evaluation of my thesis. I also extend my thanks to Assoc. Prof. Dr. Süleyman BAŞTÜRK for his valuable guidance throughout my studies. Finally, I would like to extend my thanks to all the academic and non-academic members of Altinbas University for their continuous efforts.

ABSTRACT

OBSTACLES AVOIDANCE FOR PICK-PLACE MOBILE ROBOT IN PRODUCTION LINE BASED ON IMAGE PROCESSING

AL-ZUBAIDI, Zaid Mohammed Abd Ali

M.Sc., Mechanical Engineering, Altınbaş University

Supervisor: Asst. Prof. Dr. Serdar AY

Co-Supervisor: Asst. Prof. Dr. Mohanned AL-KHAFAJI

Date: 04 / 2023

Pages: 124

Industrial robots have revolutionized manufacturing by enabling faster and more diverse production methods. This study uses image processing to find obstacle-free paths in production lines. A*, Hybrid A*, RRT, and PRM were considered in this study for analyzing and contrasting the advantages and disadvantages of each algorithm by simulating the most common production lines and concentrating on the effective parameters that may be modified in each algorithm to fulfill the requirements. This study showed that each algorithm could be adjusted. Still, the Hybrid A* algorithm gives the shortest path but needs too much computational time. In contrast, A* gives a short path with a short time, contrary to RRT, as it does not give a short path compared to the others; its results can be enhanced to find a shorter path as the computational time increases otherwise, it gives a path with very short time and the PRM algorithm gives an unpredictable new path for each iteration, generally the length of the path is inversely proportional to the number of explored nodes at the expense of the computational time. Instead of using a complex algorithm to reduce the amount of time spent processing data and make for easier operation conditions such as camera locating and lighting and other factors, as well as for easier modification to meet the requirements, the use of a conditional command algorithm is performed as a means of overcoming the challenges posed by pick-and-place operations by 6-DOF robotic arms and since this study focuses on path planning; the gripper has been neglected, as the grippier would not affect the path planning. This thesis showed also that there is no perfect path planning algorithm for all cases and

requirements, as the algorithms can be adjusted to be suitable for specific cases according to specific requirements.

Keywords: Image Processing, Path Planning, Mobile Robotic Arm, Pick and Place Applications, Obstacles Avoidance.

TABLE OF CONTENTS

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

A*	:	A-Star Algorithm
RRT	:	Rapidly exploring Random Tree Algorithm
PRM	:	Probabilistic Road Map Algorithm
DH	:	Denavit Hartenberg
RGB	:	Red, Green, and Blue
JPG\JPEG	:	Joint Photographic Expert Group
TIF	:	Tagged Image File
GIF	:	Graphics Interchange
PNG	:	Portable Network Graphics
ANN	:	Artificial Neural Network
GMRs	:	Ground Mobile Robots
GMRMP	:	Ground Mobile Robot Motion Planning
GM	:	Global Method
LM	:	Local Method
SQA	:	Single Query Algorithms
MQA	:	Multi Query Algorithms
LSTM-BIRRT	:	Long Short-term Memory Bi-directional RRT
XABSL	:	Extensible Agent Behavior Specification Language
AI	:	Artificial Intelligence

1 INTRODUCTION

Manufacturing operations are becoming more complicated as a result of the ever-changing requirements of customers due to the dynamics of manufacturing activities and the introduction of industrial revolutions. As a result, robotic solutions play a significant part in the effort to automate manufacturing, achieve operational efficiency, achieve high productivity, and ensure worker safety [1]. In recent years, the global use of robots has expanded due to the necessity to automate and modernize production processes, medicine, security, and society [2]. Nowadays, it's fundamental for every industrial operation to require both high levels of repetition and precision. The introduction of autonomous robotic applications allows for the completion of routine, uncomplicated activities while yet meeting the necessary standards of precision and velocity. Every business in today's rapidly expanding industrial age needs lightning-fast production to meet the demands of their customers. Project's overarching goal is to solve this issue by creating a cheap and flexible robotic arm that can be put to use in any sector [3].

The pressure on the industry to get a more significant market share is increased by the current market evolution and rising levels of competition [4]. And in order to fulfill the various client requests and reduce the expenses associated with holding extensive inventories, the European sector is moving toward being more customer centric. Numerous industrial sectors, like the automobile industry, which has depended on the serial production line paradigm for decades, are exhibiting the paradigm shift. The requirement to adjust production machinery that lacks the cognitive capacity to handle many operations in a dynamic environment seriously jeopardizes the resilience and effectiveness of the serial production model. On the other hand, due to the difficulty of integrating conventional robots in the available workspaces, many EU industries, like aeronautics and aerospace, mainly rely on manual assembly [5]. Figure 1.1 shows the use of multiple robot manipulators to work together will improve efficiency in assembly and transport. Assembly activities sometimes need a robot to complete a portion of the task while another arm holds a work piece or prepares another element. Observing human arms/hands at work reveals this sort of collaboration between two arms/hands in action on a daily basis [6] [7].

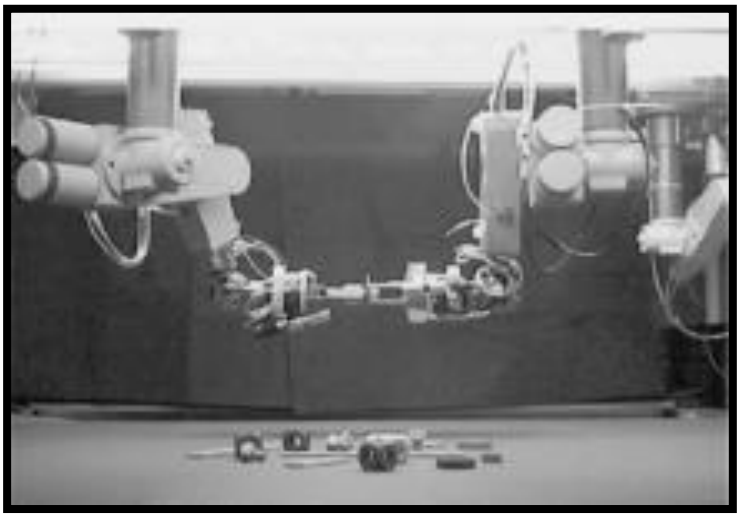


Figure 1.1: Two-Arm Robot System in an Assembly Scenario. [7]

The operational flexibility of robots to handle and assemble various parts has been the main factor in their acceptance in assembly systems, where they have found significant use. Nevertheless, the system that was produced. However, the resultant system level flexibility is constrained because of the below three reasons but however, robotics is a massive possibility since, according to the most recent statistics, at least 85% of manufacturing tasks in essential industries such as electronics, equipment, etc. Can be automated [5].

a. Repeated tasks require an ability that enables the robots to be positioned in a fixed or tightly limited manner on the shop floor to assure a sufficient supply of consumables as well as positioning and reference accuracy.
b. The requirement for part-specific mechanical material handling devices, such as fixtures, grippers, feeders, etc.
c. The complexity of integrating controls and programming additional resources and processes. In order to integrate new equipment, the present control designs employ hierarchical programmable logical controller-based control, which takes time and knowledge.
d. The high expense of maintaining the cell and purchasing auxiliary systems for various parts render automation unaffordable and unsustainable.

Assembly, which comes after manufacturing and is influenced by earlier manufacturing processes, is a critical phase in the manufacturing process. Quality and components prior inaccuracies, defects, and inconsistencies can all have an effect on how things work together. The changes in the delivery of materials and information may result in a completion or service

delay. Throughout the assembly process, the following goals should be met.

a. Reducing assembly and production costs while maintaining product quality.
b. Reducing manufacturing time.
c. Increasing automation.
d. Being able to adapt promptly to changing market demands.
e. Accelerating the introduction of new products.

In small and medium-sized enterprises, a lack of management and evaluation of production costs is a significant challenge. There is generally little skill in estimating real expenditures. As a result, the introduction of assembly automation lacks an important aspect, namely economic justification. The incorporation of economic criteria early in the product development process and subsequently when manufacturing is implemented is one of the essential tasks of computer-aided assembly planning [8]. In the industry, an assembly line is one of several critical places that provide significant possibilities for continual improvement in either process or flow techniques. It contributes to the sector's steady growth. The production of sequences is an important part of assembly procedures. Whenever the assembly process modeling is used in any assembly line that is engaged in acquiring completed items via an efficient procedure flow [9]. Figure 1.2 shows systemic automated assembly line [10].

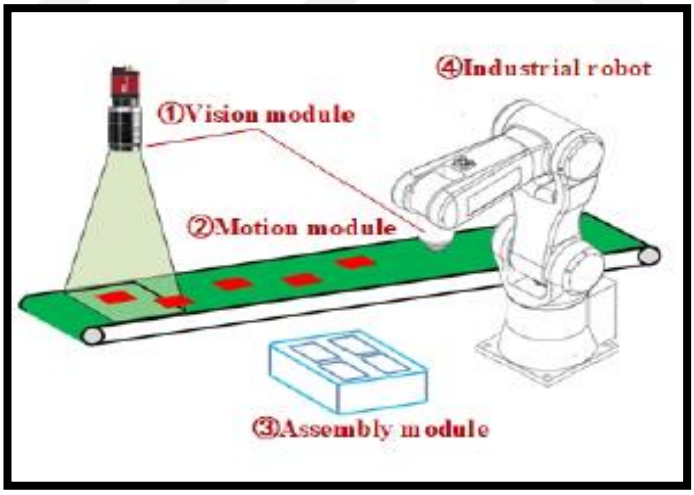


Figure 1.2: System Composition in Assemble Line[10]

The manufacturing workshop is the principal sector/ department on which the corporation relies for producing its production. Monitoring and managing the discrete manufacturing workshop have always been a challenge for the firm since it engages in multi-variety and small-scale production, the production process is intricate, the production schedule is tough, and the discrete manufacturing workshop is engaged. Businesses rely on their capacity to adapt

rapidly to changing needs, which necessitates altering the manufacturing process, shortening the production cycle, and assuring product quality. Simultaneously, as consumer requests become more customized, the unpredictability of survival orders grows. Furthermore, the primary issue must be resolved. Nonetheless, opportunities and challenges coexist. The advancement of networking and information technology has profoundly transformed and influenced traditional business conceptions and management approaches. It has also pointed the way forward for the growth of industrial enterprises [11]. Generally, any production line can be defined as a cluster of workstations linked by handling systems that convey things from one station to the next. From one station to another typically, stochastic flow lines are vulnerable to disruptions induced by variances in processing rates and workstation failure. These glitches disrupt the production stream. To stay idle, resulting in lesser throughput. Buffers are employed to reduce the consequences of these disruptions in order to cut down. Using the machinery throughput is increased with buffer storage. The production line is stopped, and the workstations' starving time enhances the flow line's efficacy. Increasing the buffer distance between machines has the intended effect of increasing the throughput of a stochastic manufacturing line, but only to a point. The quantity of space available for buffers on manufacturing lines is restricted, and stocking costs are also incurred. Furthermore, material handling is required to keep the operational process running; hence, the buffer allocation problem is an optimization stochastic system problem with numerous variables. Buffer zones in a production line divide the line into two stages for a single storage buffer and three stages for two storage buffers at two distinct sites. There are normally n-1 storage buffers available for an n-stage line [12]. The term intelligent assembly system refers to robotic assembly cells and lines with the flexibility of humans that can assemble various items in small batches and handle fast adjustments regarding new products. These systems also have minimum product-specific tooling, are resilient to production tolerances, and are simple to program and re-program. These systems can identify faults and move past them because they are both skillful and compliant. This collection of criteria poses a challenging dilemma for both production engineers and designers of assembly systems [13]. A vital job for every robot is still to detect objects and particular components in the actual environment. Robots can do tasks accurately and with an accuracy rate of approximately 97%. Real-time object recognition uses machine learning and deep learning techniques to identify things using computer vision and artificial intelligence. In India, an average middle-class person may employ intelligent robots daily. In

reality, the primary goal of creating robots is to lessen the physical burden on humans, automate the execution of jobs, and provide more exact and accurate results [14].

1.1 ROBOT APPLICATIONS

The high and accuracy, large workspace, high loading capacity, and high system stiffness are the main functional criteria of robotic production technologies. Building a robotic system with all of the essential tasks, let alone any new features that may be desired in the future as manufacturing processes grow, as is characteristic of modern manufacturing enterprises, is highly challenging. More flexibility at a reasonable price in a robotic production system is needed, particularly in adapting evolving function demands and restrictions (e.g., different positions and orientations and operation tasks, etc.). The adaptive property of a robotic manufacturing system is acquired by adjusting the hardware structure as well as the software structure; in this regard, the adaptive system may also be referred to as the adaptable manufacturing system [15].

One of the most typical jobs for a robot is picking/ holding and putting/ placing. It can be challenging for robots to autonomously design their pick-and-place motion. The surroundings and the thing being grasped are both highly complex geometrically, which contributes to the difficulty. Premises where a pick-and-place job can be completed. When a robot is tasked with a pick-and-place operation in a setting with a large number of seemingly unrelated daily things, it may accidentally place the item to be placed next to or on top of another item. The robot may also occasionally use a bar to hang something. However, there is currently no established method for accurately pinpointing the location and orientation of a stationary object on the surface of an environment during a pick-and-place operation [16].

1.2 TYPES OF ROBOTIC ARMS

A manipulator’s arm generally contains the grippers, the base, and the other components of the manipulator's kinematic system. The arms' links are stationary, yet their joints allow for movement. As a result, the joints are held in a steady position by the linkages. Motors or hydraulic actuators could be used to move the joints. There are mainly two different kinds of robot joints, each of which allows for a different kind of motion [17].

1.2.1 Based On Type of Movement

The manipulator arm may contain one or more types of movements.

1.2.1.1 Rotating movement

This kind of movement can be created by a joint, known as a revolute joint, enables rotary motion about an axis of rotation. One good example is the elbow joint in humans. This type of joint lets links move relative to each other. Most of the time, position sensors are built into these joints so that the relative positions of the links next to them can be measured. When these changes happen at rotary or revolving joints, they are called joint angles.

1.2.1.2 Linear movement

This type of movement can be created by a form of joint that enables extensions or telescopic motion (slide) which is known as prismatic joints. An example is a telescopic vehicle antenna. Sliding (or prismatic) joints are used in several types of manipulators; the joint offset refers to the relative translation between two links [18].

1.2.2 Based On Type of Kinematic Chain

1.2.2.1 Open chain manipulator kinematics

The mechanics of a manipulator in this type of arm can be represented as a kinematic chain of rigid bodies (links) joined by revolute or prismatic joints. This style of arm is used to perform a variety of tasks. A base is used to confine one end of the chain, and an end effector is mounted to the other end of the chain. A chain serial open-loop robot arm is depicted in the following figure. Within the framework of the open chain robot arm, the final motion is achieved through the composition of the fundamental motions of each link in relation to the motion of the link that came before it. Each of the joints need its own independent control.

1.2.2.2 Close chain manipulator kinematics

The Open Chain Manipulator is far easier to perform than the Closed Chain Manipulator. Even analysis needs to take into consideration statics, the limitations imposed by other linkages, and so on. A closed chain constitutes a parallel robot. The Stewart platform is the most effective illustration of this kind of robot. The Stewart platform can be seen in Figure 1.3 [19] [20].

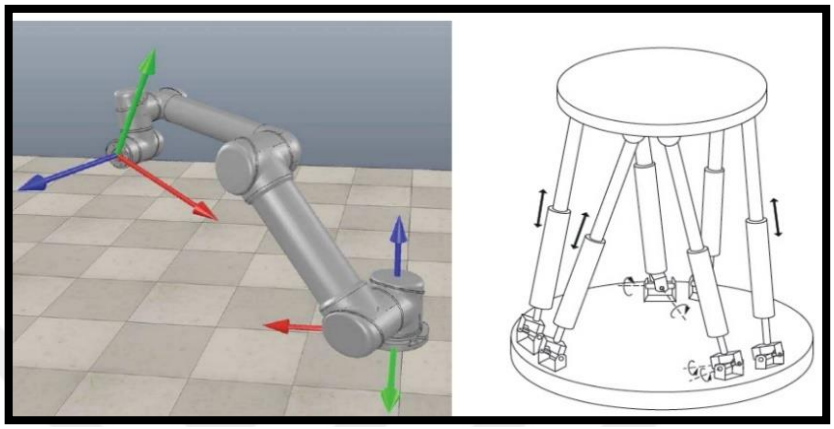


Figure 1.3: Open Chain and Closed Chain. [20]

1.2.3 Examples of Manipulators

1.2.3.1 Gantry robot

This kind of robots, all the motions and working area can be identified by the cartesian coordinates (X, Y, Z axis) as shown in Figure 1.4, For the exact analysis, the cartesian coordinate robot has been chosen with 3 degrees of freedom, working in X, Y and Z axis. This gantry robot can move in the X, Y, and Z axes, making it ideal for pick-and-place applications. It strictly allows for better accuracy. It is easy to program and is a strong dependable mover. It can carry more data and has full support for it [21].

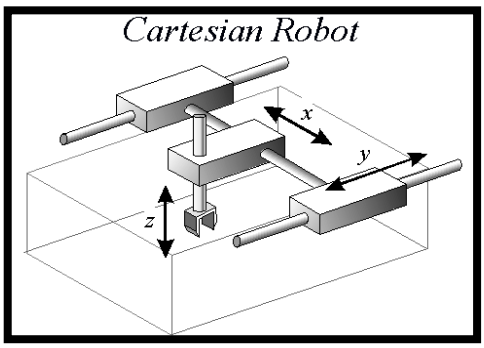


Figure 1.4: Cartesian Robot.[21]

1.2.3.2 Cylindrical robot

A cylindrical robot can be constructed by considering a cartesian robot and replacing one of its prismatic joints with a revolute joint. The movement of a cylindrical robot is determined by a cylindrical coordinate system; Figure 1.5 shows this cylindrical thick shelled as a circular workspace [22].

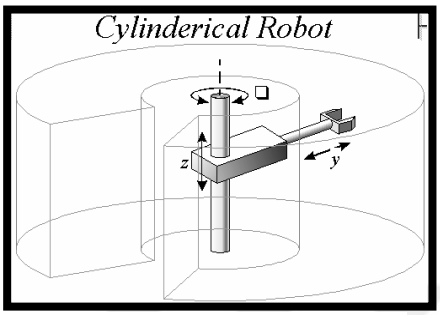


Figure 1.5: Cylindrical Robot.[22]

1.3 SINGLE AND DUAL ARM

Automatic consumer electronics manufacture relies heavily on dual-arm robots, which mimic human motions in the manual production line and can potentially employ human-made equipment. Therefore, replacing human labor may be done with little to no change to existing manufacturing processes. When opposed to single-arm robots, dual-arm robots are more adaptable and can-do jigs [23].

1.3.1 Single Arm

A manipulator known as a single robotic arm consists of a single arm that can move in three dimensions, has only one end effector, and is easier to handle than other types of manipulators. employed for a variety of purposes throughout a variety of industries, including industry, healthcare, and agriculture, among others. Material handling, pick-and-place activities, and assembly line procedures are typical examples of the kind of jobs that call for its application. For example, in the automotive industry, single-arm robots are employed to weld automobile bodywork together in order to speed up the process. Single-arm robots are easier to train and operate compared to dual-arm robots, making them a great choice for smaller-scale operations that demand efficiency and speed [24].

1.3.2 Dual Arm

Dual-arm robots are a type of robotic arm that have two arms with multiple degrees of freedom each, allowing them to work together to complete complex tasks. Dual-arm robots are used in a variety of applications, including manufacturing, assembly, and pick-and-place operations. These robots are designed to mimic human movements, making them ideal for tasks that require a high degree of dexterity and precision. Compared to single-arm robots, dual-arm robots have greater flexibility and can perform tasks that require more complex movements, such as grasping, rotating, and manipulating objects in a 3D space. However, dual-arm robots are typically more expensive and require more complex programming than their single-arm counterparts. Overall, the choice between single-arm and dual-arm robots depends on the specific application requirements and budget constraints [23].

1.4 IMAGE PROCESSING

Generally, the computer cannot recognize any image as the human does, unless dealing with the image as a matrix. As each element of that matrix represents a pixel in the real image, pixel means a single dot on a computer screen. For instance, MATLAB might save an image consisting of 350 rows and 500 columns of distinct colored dots as a 350-by-500 matrix. Colored images (RGB), require a three-dimensional array (that means the computer needs to represent the colored picture into 3 layer matrix (r*c*R), (r*c*G) and (r*c*B), where r stand for rows and c stands for column and the first plane in the third dimension represents the intensities of red (R) pixels, the second plane represents the intensities of green (G) pixels, and the third plane represents the intensities of blue (B) pixels, here all layers should have same dimensions [25]. There are general and important concepts related to image processing.

1.4.1 Image Types

1.4.1.1 Binary images

Each value of the formed matrix for this type of image should be either 0 or 1 as shown in Figure 1.6 (a).

1.4.1.2 Grayscale image

Grayscale means that they are represented by intensities. The brightness of a pixel can range from completely dark (black) to completely bright (white). Typically, the values of displayed pixels are scaled to a range from 0 to 255, as shown in Figure 1.6 (b).

1.4.1.3 Indexed image

Each pixel value in an indexed image corresponds to a color value in a color lookup table rather than an intensity value. Images with an index typically have their data values scaled from 0 to 255 so that they can be properly compared with color databases containing up to 256 colors, as shown in Figure 1.6 (c).

1.4.1.4 RGB image

There are three dimensions to an RGB image, but only two of them are used to describe where exactly a pixel is located inside the image. The pixel colors themselves are defined by the second dimension. In each given image, the color dimension consists of the RGB (red-green-blue) channels as shown in Figure 1.6 (d) [26].

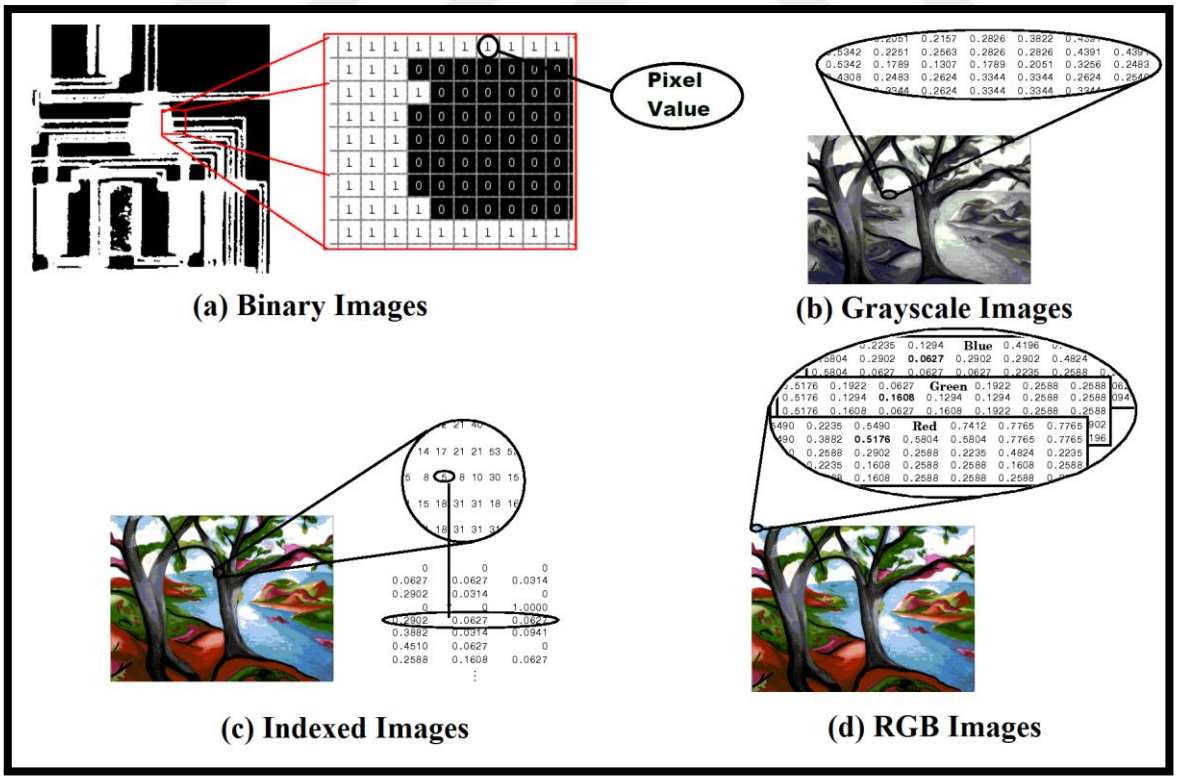


Figure 1.6: Types of Images. [26]

1.4.2 Image Formats

1.4.2.1 JPG (joint photographic expert group)

The most widely used format for reducing file size in digital images is jpeg, sometimes known as jpg. It's commonplace in digital media such as webpages, documents, emails, etc. The reason being the modest size of digital photos compared to other file types. Jpeg photos, however, suffer from extremely poor resolution.

1.4.2.2 TIF (tagged image file format)

The highest quality image for commercial use is a TIFF or TIF file. The files' exceptional quality comes at the expense of their massive size.

1.4.2.3 GIF (graphics interchange format)

GIF is an 8-bit video format used by anyone using a dial-up modem to access the world wide web, which developed by CompuServe which is a time-sharing computer service.

1.4.2.4 PNG (portable network graphics)

PNG files have higher quality than gif and jpg files while being less in size than TIF files. Its transparency has made it a common component of modern web design [27]. The discipline of engineering is observing a meteoric rise in the use of image processing as a tool to carry out non-intrusive precision measurement, autonomous robot navigation, or dependable verification of industrial automation processes. These applications are all being driven by advancements in computer technology. This pattern is anticipated to remain prevalent for the foreseeable future. Image processing methods are becoming increasingly prevalent in the technological landscape, and they constitute a significant topic of study in the fields of engineering and computing. In this method, a picture is processed in order to improve its quality or to extract relevant and valuable data from it. This term refers to a specific category of image processing operations where the picture itself (a photo or video frame, for example) serves as the input signal, with the same properties being returned in the processed output. Mathematical computer operations on a two-dimensional image have been called "image processing" [28] [29]. One of the technologies that are currently expanding at a quick rate, is a process that involves performing a number of operations on the image that is being input in order to obtain informational insights from improved photos. Images that are used as input are

typically captured in the pixel format, where each pixel is correlated with one of the primary colors red, green, and blue RGB, as well as black and white RGB. There are two distinct approaches to this type of processing, which are known respectively as analogue image processing and digital image processing. In the case of physical copies, such as printouts and photographs, an analogue image processing method is utilized. When utilizing the technology of digital image processing, the image must go through three significant stages, which are the processing stage, the stage of enhancement and presentation, and the stage of information extraction [30], [31]. Techniques for acquiring and processing images have been studied extensively in the fields of physics, computer, electrical, and mechanical engineering for a considerable amount of time. The specialized literature in this field is typically devoted to specific aspects such as the geometry of camera projections and computer vision, image acquisition and processing, or either the scientific and industrial applications of image processing and analysis [28]. Recent developments in technology have sparked an increased interest in the study of image processing methods. In general, approaches for image processing are quickly expanding across technologies and comprise a significant study subject in the fields of engineering and computing. This method entails performing several operations on an image with the intention of either improving the overall quality of the image or deriving some information from the image that is both helpful and desirable. It is a sort of operation that can be performed on an image using signal processing in which the picture itself (such as a photo or video frame) is used as an input signal, and the output image retains the same characteristics as the original image. One way to conceptualize image processing is as a mathematical operation carried out by a computer on a two-dimensional photograph [29].

1.5 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is a system's ability to understand external input accurately, learn from it, and adapt to meet certain goals and activities. After multiple ai summers and winters, periods of high and low funding and interest in ai research, we've entered ai fall. This time is marked by harvesting and benefiting from prior research, made possible by growing processing power, statistical improvements, and the infinite availability of big data. These advancements have broadened AI's scope and stimulated interest in AI and robotics within and outside the industry. Financial services, manufacturing, marketing, retailing, and public administration all consider AI for a competitive advantage. [32] The use of artificial

intelligence as a method of controlling robots is now the most modern technology. In terms of performance, it offers numerous benefits, such as precise control and less time spent computing, and it also solves various mathematical issues that arise in the process of motion and path development. Finding a solution for inverse kinematics that is precise and dependable is the primary challenge faced when attempting to control motion in robotic arms. Inverse kinematics must be calculated in order for real-time control to be possible. The solution to inverse kinematics problems is computationally difficult and takes a very long time to compute. [33] based on AI, artificial neural network (ANN) was presented for a nonsurgical robot as an enhanced learning technique to include characteristics without describing an explicit robot system model. Robot manipulators require advanced control. Model uncertainty degrades robot manipulator torque control. Smooth uncertainty is compensated via a neural network in the feedback loop. neural network approximates continuous functions with arbitrary precision. The adaptive weight tuning approximation must be closed-loop stable. Decline neural network's weight. Learning can be tracked, but not performance. Changing tasks and dynamics require a continual flow of information, which online neural network control cannot give. Online neural network robot control uses difficult-to-select basis functions or nonstandard weight tuning. Layers complicate convergence [34].

1.6 PATH PLANNING

Using autonomous ground mobile robots (GMRS) for automated tasks is rapidly expanding [35]. The increasing popularity of GMR can be attributed to the fact that they are easy to deploy and can move quickly through a variety of environments. As a result, autonomous navigation techniques of ground mobile robot motion planning (GMRMP) have emerged as a crucial area of study in robotics [36]. Since path planning allows GMR to navigate autonomously by determining the best route and communicating it to the mobile robot in the form of a track or sequence of points, the navigation path planning problem fits the criteria for an optimization problem. Therefore, the navigational technique selection is a crucial stage in the motion planning of mobile robots at work in both simple and complex environments [37]. There are several permitted paths for GMR to reach the target location in every given task, but the recognition of the optimal path is contingent on the adaptation of optimization criteria such as path length, duration, and energy consumed. Consequently, numerous optimization strategies have been applied to the problem of path planning [35].

Depending on the availability of the surrounding environment, the path planning approaches can be categorized as global method (GM) and local method (LM) path planning. The GM is ideal for scenarios in which only static obstacles exist and all environmental data is available beforehand. The shortest path from origin to destination can be determined in advance. While the LM is applied to the dynamic environment, where there are moving objects and essential information is only available within a limited area, the LM is not applicable to static environments. It does not require any prior knowledge of the surroundings and can find a high-resolution route even when moving items are present. However, it can only be carried out in a real-time context and typically necessitates a large number of sophisticated sensors and massive computations. Due of the aforementioned factors, global and local path planning are typically combined in a hybrid fashion [38]. When it comes to path design, the primary concern that needs to be addressed is ensuring that there will be no collisions along the route. This may be accomplished by taking the following precautions.

a. Keeping obstacles and robots from colliding with one another.
b. Optimizing the path according to the restrictions that have been set, such as the length of the path or its smoothness. In light of the fact that gas operate in the cartesian space, which is often referred to as the SE(3) space, that is also used for the sampling-based planners in order to ensure good comparability [39].

There are two main types of algorithms can be used for finding the suitable path planning:

a. Single query algorithms (SQA): single query algorithms are distinguished primarily by the fact that they seek a path from a start configuration to a goal configuration without making use of any prior information. [40]
b. Multi query algorithms (MQA): in the preprocessing stage of these kinds of algorithms, the configuration space is thoroughly combed over to look for collision states. On the basis of that information, a data structure is recorded, and it will later be utilized during the phase where planning will take place. In the beginning, these sorts of planners are slower than single query methods; nevertheless, the stored data can be used for fresh calculations on the same image.

Generally, SQA is faster than multi-query algorithms due to the fact that no preprocessing phase is required, and this type of algorithm must recalculate even if the start and target configurations were only slightly modified. Rapidly exploring Random Trees (RRT) is the most well-known example, whereas MQA is excellent for static situations. Representatives of renown are probabilistic road map (PRM) [39] [20] In this four path planning algorithms are considered.

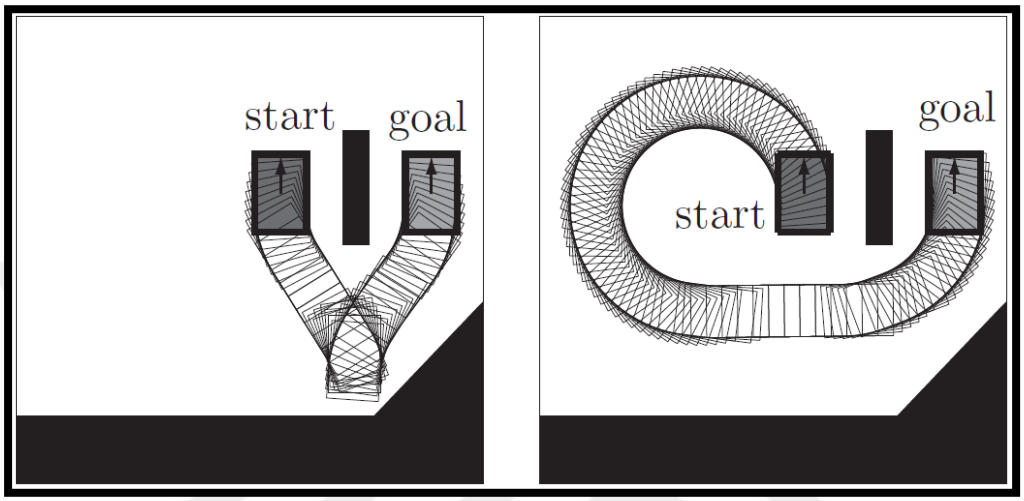


Figure 1.7: Different Algorithms for Path Planning with Obstacles Avoidance.[20]

1.6.1 A* Algorithm

In 1968, Peter Hart, Nils Nilsson, and Bertram Raphael were the ones who came up with this algorithm. A* is one of the "graph search algorithms." grids cover the whole place. Now, there are two dimensions to the search area. Each rectangle in the array can or cannot be walked on. To find the path, squares/rectangles must be figured out to cross to get from the starting point to the end point. After finding the squares/rectangles, the rover must move from the centre of one squares/rectangles called a node, to the next until reaching the target. The path would be found by starting at the starting point, searching for the squares/rectangles nearby, and searching outward until finding the target. Ignore walls, water, and other illegal terrain and count all squares around the starting point that can be reached or walked on which are basically obstacles [41].

1.6.2 Hybrid A* Algorithm

Dolgov et al. [42] gave more information about the algorithm in the years that followed 2007 when the hybrid A* algorithm worked well in the DARPA urban challenge, which was a

competition for robots put on by the U.S. government. The hybrid A* algorithm works in the same way as the A* algorithm. The main difference is that changes from one state to another happen in continuous space instead of discrete space. One of the biggest problems with the old ways of planning paths for non-holonomic robots is that the paths they make are discrete and often can't be used because the changes in direction aren't smooth. Even though the hybrid A* search builds the graph on a discretized grid, any point on the grid can be reached by a vertex. As a continuous search space would not be finite, grid cells are used to break up the space into smaller parts, which limits how big the graph can get. Since there is no set way for transitions from one vertex to the next, it is easy to include the fact that they are not holonomic in the transition between states. Most of the time, the search space is three-dimensional, so the state space x is made up of x, y, and. This makes a discretized cuboid with the base being the x, y position and the height being the heading theta of a vertex.

1.6.3 Rapidly exploring Random Tree (RRT)

A rapidly exploring random tree (RRT) is a data structure and algorithm that makes a space-filling tree to effectively search non-convex high-dimensional spaces. An RRT can also be used to search for the largest Voronoi regions of a graph in configuration space. RRT described as a unidirectional search tree with a branch that leads to the destination node by randomly choosing obstacle-free states and connecting them.

This is a step in the RRT method. In incremental phases, an RRT is built with its roots at the start node and the goal node. The tree explores the space around it, and a simple greedy heuristic is used to move the tree toward its goal [43].

1.6.4 Probabilistic Road Map (PRM)

The general idea of the probabilistic roadmap technique belongs to research team studies since 1990s. Some researchers extended their investigation into it. The following outlines the fundamental concept behind the probabilistic roadmap technique. An undirected roadmap network called $r = (n, e)$ is constructed based on a random sample. In this graph, n represents the nodes of attained milestones by random sampling, and e represents the edge linking these nodes. The probabilistic roadmap approach searches for two nodes, s' and f', that satisfy the conditions that s and s' are directly connected, and f and f' are directly connected. This is done after the starting-points and the finishing-point f have been determined. The planning of a path

is accomplished by doing a search through the edges of the undirected roadmap graph for a sequence of edges that are directly connected to *s′* and *f′*. [44]

1.7 ROBOTS' SELECTION

Even though robots can be automatically controlled, and reprogrammed, are versatile manipulators, and can be programmed in three or more axes. And despite the fact that their capabilities and specifications are heavily utilized in businesses, the determination of which robot is the most appropriate has proven to be a challenging decision, particularly for first-time robot purchasers. This is because there is a vast selection of models available for any application, the necessity of taking into consideration multiple conflicting qualitative and quantitative criteria, and the absence of standards and benchmarks for the evaluation of robot performance would make the selection decision very difficult and mistakable [45]. There are many criteria that can be considered for robot selection which are usually based on the following concepts [46].

a. Object weight handling abilities: this criterion focused on the heaviest object that can be handled by the robot's end-effector (including the end-gripper effector's weight) at one time. Moreover, the weight and the shape of the robot itself are issues that should be considered to achieve compatibility with available environments\ workspaces.
b. Reachability: this criterion focused on the distance from the robot's center to the furthest extending point that the robot end effector can reach according to the boundary conditions of the robot design.
c. Repetitive applications capability: this criterion focused on the precision with which the end effector can be positioned at a predetermined point in the program for a specified number of iterations with keep achieving the required accuracy.
d. Power consumption: this criterion focused on the minimum and the maximum power that the robot needs to do the required tasks, considering the severest conditions.
e. Easiness: this criterion focused on how easy to deal with the robot to full functional achievement with all dexterity**.**

2 LITERATURE REVIEW

Many studies have been conducted to explore the effective image processing algorithm for the required results based on the available circumstances. The other studies tried to find the suitable path planning algorithms for mobile robots based on the required parameters, depending on the required results and available circumstances. Some studies focused on the shortest path while the others focused on the number of nodes that the path composed by, while the others focused on different issues like the time of processing. Reducing the sharpness of the path steering and abrupt changes is also the main factor for the others. All of those factors considered the effective issued for studying the trajectory and the kinematics studying the forward and inverse kinematics, which are very useful for finding the path planning for the articulated robotic arm to perform pack and place applications for repetitive tasks and\or interactive tasks which need intelligence like intelligent assembly applications.

2.1 ROBOTIC ARM

In 2008, Donald R. Dentler [22] showed in his master's thesis that researchers can observe the workings of a robotic system by employing a three-link articulated robotic arm. There are extensive descriptions of the various actuators, controls, and drivers. The arm will be built using physical principles so that it can perform well in both static and dynamic situations. In addition to the obvious structural requirements, it is important to plan for the eventual integration of the controls as part of the design process. It's crucial that the components you use for the design are optimized for both performance and physical properties. The program for simulating the robot arm's movements also allows for coordinate transformations in the opposite direction. The foregoing ideas were utilized to guide the development of a three-link revolute-jointed robot arm, as the thesis concluded. With the help of its CNC driver board and software, it could go wherever it needed to inside its workspace. Its open-loop control mechanism and simple design made it an ideal example for the concepts he explored in his research.

In 2018, Madiha Farman et al. [47] outlined the process of creating a three-axis robotic arm that can pick up and position things of varied weights using a color sorting algorithm. A gripper, two mirror-image connections, a rotary table, and a square base are its primary components. A servomotor drives the rotational motion of each joint. The angular location of

each servomotor shaft is determined by a MATLAB software that gets input from an Arduino microcontroller. The linked MATLAB code provides the appropriate inverse kinematics equations for calculating the required joint angles at a given Cartesian position of the end-effector. Static and dynamic calculations, mechanical property calculations, and prototype testing were all a part of the robotic arm's design process to ensure a sturdy and reliable final product. These efforts revealed that links with a cross-section of two rectangular areas provided the best structural strength to weight ratio, beating out both solid and hollow circular beams. Forward and inverse kinematic analyses were undertaken, and a mathematical equation was devised to depict and display the arm's working space. After a battery of reliability and precision checks, the camera and end effector error values were calculated.

In 2020, Ashwin Misra et al. [48] looked for a less expensive SCARA Robotic arm. Finite element methods and experimental validation are used to assess the initial CAD's practicality. To ensure optimal process conditions and the potential for force feedback control in the future, a comprehensive kinematic and dynamic robotic study is performed. The method described in the article can be used to create a working prototype. Metals and non-metals may be combined in a variety of ways, each with their own strengths, design challenges, and associated costs. Manufacture methods include both additive and more conventional approaches. The Arduino microcontroller is used for all the mechanisms and processes that are under control. The touchscreen interface on the robot facilitates both batch and individual production control. The proposed technique is useful for enhancing the efficiency of assisting Micro to small manufacturing enterprises. Their research led them to the conclusion that combining additive and conventional manufacturing cuts down on both costs and stress tests. In most cases, you can cut costs by 95% if you can predict them with excellent precision. The firmware is based on CNC code, which can be easily modified by factories around the world. When compared to CNC machines, the only real limitation is precision. This issue can be resolved by employing more precise production techniques and higher-resolution motors. Drilling parameter forecasts are made using a neural network. While this article concentrates on the robot's use for drilling, it is equally applicable for milling. The substance of the workpiece dictates the tool and firmware used.

Hasan Salman et al. [49] studied the kinematic model of a three-axis-movable robot arm equipped with a hand gripper in order to select and place an application. For the purpose of identifying the end effector's DH-conforming positions, a forward kinematic model has been

presented. An algebraic strategy based on trigonometric formulas and the geometric method was selected to compute the inverse kinematics of a three-DOF modular manipulator with a shoulder offset. they were able to model and apply the robot arm's emotional aspects in MATLAB by creating a 3D visual software package within the context of GUI design utilizing the Robotic Toolbox simulation (RTB 10.3). At long last, a GUI application-controlled electronic interface circuit based on the Arduino microcontroller was developed. Further investigation into the reality interface design approach of various robot manipulators can yield a more optimal solution in terms of their kinetic properties, and the work presented here can serve as a steppingstone toward that end. A minor disagreement was discovered between the theoretical stance and the actual pose; however, this was fixed by included the value in the new configuration calculation, demonstrating that the theoretical analysis supported the actual implementation for the actuators.

2.2 IMAGE PROCESSING

In 2020, P.S. Minz, [50] tried to develop image processing algorithm for food colorimetry. High-definition photos capture details and improve measuring accuracy. Complex image processing is required for HD pictures' dense image matrix. Two-step picture cropping speeds HD image analysis and reduces computational power. Changing RGB to CIE Lab yields colorimetric parameters. Preprocessing algorithm was applied to reduce picture matrix size. Processing HD photos for color extraction measured CPU, RAM, and disk consumption. Image processing at 1100 * 1100 was finished with consistent CPU use. Skim milk powder CIE Lab values were determined using a food color vision algorithm. HD photos improve food colorimetry accuracy. High-resolution photos are huge and take longer to process. The paper's algorithm can be used to process dense matrix color images and measure skim milk powder color. Food spectrophotometer results validated. The approach generates a sub-image of a high-definition image to reduce image size for faster image processing. Sub pictures automatically remove image backgrounds and unwanted parts. Although tested in “Scilab”, the approach works in MATLAB and LabVIEW. This approach is used when the ROI is preset, or a small picture area is enough to determine image attributes. This study's method helps determine a system's maximum image resolution. It shows if a faster processor or more RAM is needed to process HD images.

In 2021, Gerald K. Ijemaru et al. [29] tried to evaluate MATLAB image processing techniques. MATLAB has advantages over traditional image processing approaches. MATLAB-based technique offers easy debugging, data analysis and visualization, easy implementation, and algorithmic testing without recompilation. MATLAB's computational algorithms can be used to simulate still and video pictures. MATLAB codes are shorter than C++, making browsing and troubleshooting easier. MATLAB handles mistakes before execution by recommending faster code. The suggested technique allows image cropping/resizing, denoising, blur reduction, and sharpening. To provide the latest MATLAB-based image processing tools. They also presented an empirical technique utilizing the 2D-DCT coefficients. Using MATLAB toolbox's latest methods, simulating their proposed technique's performance. The results show MATLAB's usefulness for image processing. They could develop picture color detecting techniques and modified MATLAB code for image processing. Using MATLAB toolbox, they simulated many photos to assess the proposed approach. The results showed that the proposed strategy is state-of-the-art for image processing. Their study intended to give readers MATLAB image processing applications and researchers understanding about MATLAB-based image processing methodologies. MATLAB can document and duplicate image processing steps. MATLAB's image processing algorithms are more advanced than others.

2.3 PATH PLANNING

In 2013, Hazim Nasir and Ali Alhilli [51] used in their master thesis study classical A* algorithm to get the best path with avoidance of any obstacles by considering the starting and ending points and the obstacles as input data and the generated path as points will represent the output, they considered the obstacles as static areas with predefined coordinates and the gotten points will be the optimal path to let the robot start from the starting point to the target without any collision with any obstacles. They took the articulated educational Lab-Volt 5150 robot which is a five degree of freedom robot in comparison with simulations to find out 0.01%-0.14% of accuracy in X-axis and Y-axis.

In 2017, Ahmed T. Sadiq et al. [52] presented a free cartesian space map of the 2-DOF arm to get the joints variable at each position without collision. Moreover, to find out how far away from the destination, they used the D-star technique and the Euclidean distance. An improved version of the Particle Swarm Optimization algorithm that they presented for determining the

best route using local search, D-star, and Euclidean distances. This smooth trajectory is achieved by solving the quintic polynomial problem. They improved PSO algorithm is able to efficiently perform in order to locate the optimal path even in challenging circumstances.

In 2017, Lars Larsen et al. [39] stated a system to find out the path planning for robots in order to expedite the required procedures for this purpose with automatic and intelligent planning system. Thus, they advocate well-known sampling-based methods like PRM or RRT and compare them to CI-based methods like genetic algorithms.

In 2019, Stelian-Emilian Olteana [53] suggested a low-cost mobile robot platform based on a four-wheeled chassis, with plaudits going to the Raspberry Pi and Arduino Uno interfaces. The mobile robot can navigate in 2D environments as a line follower robot thanks to its mapping, navigation, and obstacle avoidance features. A single-degree-of-freedom (DOF) robotic arm is installed on the platform for picking up and relocating the obstructions. To do so, they need to develop a mobile platform that meets the fundamental design requirements of this juncture, such as being a low-cost solution, highly reliable, and extendable, and serving multiple purposes, including those of education (as a didactic stand for microcontrollers, electronics, automation, robotics) and research (including the study of various algorithms of mapping, localization, navigation, obstacle detection, and transportation).

In 2022, Tahseen Fadil Abbas and Alaa Hassan Shabeeb [54] used grey wolf optimization (GWO) to calculate the shortest and safest path for a robot with obstacle avoidance. Their investigation aimed to locate the optimal approach while considering the number of barriers and design characteristics. They utilized MATLAB to simulate the algorithm's performance. Their simulations indicated that the suggested path planning method found the shortest, collision-free path in varied situations. Comparisons with various path-planning algorithms in static contexts confirmed the proposed algorithm's superiority.

In 2022, Aisha Muhammad et al. [55] Compared A*, PRM, RRT, and GLS path planning simulations. Utilizing the same difficult environment. They concluded that the A* algorithm efficiently searches the network for a path from A to B. If the terrain is fully mapped and will remain so during the robot mission, the A* algorithm can be used. RRT's non-optimal solution doesn't ensure correctness. The RRT performs well in clusters. In congested environments, the Generalized Laser Simulator generalized approach and a laser simulator algorithm enhancement work faster. The PRM and GLS path planning algorithms generated a feasible

path shorter in the distance and search time than A*. The Generalized Laser Simulator and A* algorithms can find piecewise linear paths, unlike the RRT algorithm.

In 2022, K. Pribilová and Z. Mäsiarova [56] described and showed examples from RRT, PRM, and the Genetic Algorithm. They began the pathfinding process by determining the beginning and ending places. The facts regarding the three different strategies show that it is essential to select the appropriate route planning algorithm tailored to a particular application and environment.

2.4 TRAJECTORY AND KINEMATICS

In 2016, Alia N. Barakat et al. [57] demonstrated at a conference how they developed parameter designs and studied forward and inverse kinematics to make studying robot arms easier. The DH convention method is then used to create the resulting kinematic equation. An interactive 3D visual simulation of the robot arm, modeled with Robotics equations in MATLAB, displays the results of the trajectory planning algorithms. After identifying the movement of each joint of the robot arm and testing the settings, the simulation achieved the intended outcome of sketching a sine wave on a writing board, allowing for the mathematical development of a model of the robot arm, including complete Kinematics investigations. Forward kinematics equations were written using the DH notation. The developed program makes use of and improves the Jacobian "Velocity Kinematic" and trajectory planning methods. The MATLAB software suite was used to conduct simulation experiments. The researchers built a 3D framework in MATLAB that allowed them to test out the robot's settings.

In 2019, Shuji Yang et al. [58] showed how to use trajectory planning with a dual-armed space robot that is floating freely in space. The coupling effects of the mission arm's maneuvers are essential to capturing the target and establishing control of the base. The base features attitude control, attitude tracking, and position monitoring. To guarantee that the mission arm's joint states capture and regulate the base's motion, an optimization method is employed to solve for the arms' joint velocity under all conditions. The inequality restriction is to prevent the base, the mission arm, and the manipulator arm from colliding. To reduce the impact on the base's attitude caused by the mission arm's joint velocity, a new variable has been introduced. To verify the control strategy during a capture and base maneuver, numerical simulations are performed on planar and spatial dual-arm robots. Because of the arms' dynamic link to the

base, the latter can move around while the former captures its targets. To make it simpler to control the baseline attitude, they swap out the joint velocity of the mission arm for a new variable that lies within the response null space. Here look at three different situations including attitude tracking are investigated, attitude regulation, and position tracking. The optimization of the arm's velocity is transformed into a trajectory planning challenge. Constraints in the code include things like obstacle avoidance, limited movement at the base, and limits on the acceleration and direction of movement at the joints. Accurate capture of targets and smooth base maneuvering were seen in numerical simulations, lending credence to the proposed control strategy. Acquiring signals from a target with pinpoint accuracy and avoiding a collision with a moving barrier are all possible with some base maneuvering. Maneuvering the base using a manipulator saves fuel. Numerical simulations of both 2D and 3D robots have been used to test the effectiveness of the proposed control method. Its core idea can be used in a variety of applications involving base motion management.

In 2021, Kuo-Ching Ying et al. [59] developed a deep learning-based optimization method to plan safe paths for two-armed assembly robots to follow in complex work environments. Due to the high dimensionality of the robotic motion patterns, a Bi-directional Rapidly exploring Random Tree (BIRRT) method merged with the Long Short-term Memory (LSTM-BIRRT) was created to boost the effectiveness and efficiency of the planning process. Extensive numerical analysis demonstrated that the LSTM-BIRRT approach outperforms previous-generation algorithms developed for motion planning of dual-arm robots in both two and three dimensions. The new technology significantly reduces the time and distance required for robotic operations. The LSTM-BIRRT algorithm provides an excellent starting point for further research into the topic of process autonomy within intelligent supply chains.

2.5 PICK AND PLACE TASKS

In 2003, Hsein-I Lin et al. [60] put out a structure for robots to learn pick-and-place jobs from watching humans perform them. This was achieved by using a graphical representation based on Gilbreth's therbligs. One of the fundamental building blocks of their work is robot motion basic. The motion primitives and their order were additionally specified using the Extensible Agent Behavior Specification language (XABSL).

In 2009, Jun Li et al. [61] looked at how different-sized obstacles affected pick-and-place movement limitations caused by obstacles. The results of their experiments showed that the

kinematics of the pick-and-place movement were affected by both the obstacles and the direction in which the movement had to be done. Temporal movement parameters, like the time it took for the movement to happen and its average speed, also changed. These results showed that information about the size of obstacles and the direction of movement was used to plan and carry out the pick-and-place movement. The results and what they mean for controlling robots are also talked about.

In 2014, Vishnu R. Kale and V. A. Kulkarni [62] provided a solution for a mechatronics color sorting system that makes use of image processing, and also presented a smart method for the inspection and selection of objects in a continuous flow. Using a webcam's live feed as input, an image processing procedure detects the objects in the image and extracts color and information to be used for classifying the colored objects coming down the conveyor by color, size, and placing them in the appropriate predetermined location.

In 2015, Pedro Tavares et al. [63] demonstrated an approach that can be used to various settings to detect items in a scene and provide an appropriate response to the requirement of dealing with them. In addition, a case study will be provided to illustrate how the proposed technique was put into practice: the Amazon Picking Challenge, which sought to find ways to fully automate a distribution center. With the help of the ROS framework, they developed a methodology that is structured in three levels: recognition, movement/actuation, and control. They chose ROS as their framework because it simplifies common robotics issues by encouraging the creation of modular, user-friendly programs that meet modern industry standards.

2.6 INTELLIGENT ASSEMBLY

In 1999, Dragica Noe [13] discussed the most promising future direction in this subject is the development and deployment of intelligent assembly systems to automate batch manufacturing, as was addressed in the previous section on the state of the art in sensor technologies and plans for future development of the assembly process. The scientific community and production engineers working for the governments of the world's developed nations have made great efforts toward achieving the concept of intelligent behavior in automated manufacturing. To build smart assembly systems, you need basic sensing components like those found in robot vision systems, touch sensors, and force-torque components. The future of external sensors is inextricably tied to the development of

intelligent assembly systems, flexible assembly systems for batch production, and their application on the shop floor with high availability, as stated by Dragica Noe. As one can see in this survey, there is a significant chasm between the theoretical and practical uses of sensors in assembly systems. Production engineers are more concerned with inexpensive, easy-to-implement solutions, task-oriented systems, and reliable sensors than are scientists who are working to create sensors and systems that are more like the human senses. He had highly developed sensors, but how they are used is up to each individual robot manufacturer.

In 2007, P. Kopacek [8] discussed some concepts and recent research results concerning "intelligent" automation of assembly and disassembly. The automation of assembly processes is a traditional area of robotics use. Assembling processes will increasingly incorporate robots. Because of their adaptability, these cells can be used to automate the assembly and disassembly of medium and small batches. Disassembly families, a method for improving the effectiveness of disassembly cells, are addressed. Initial thoughts on incorporating mobile robots into the disassembling process are discussed, and he concluded his research that assembly automation is common. Disassembly automation is new yet essential in the near future due to End-of-Life Management (EOL) and the rising volume of electronic scrap. Today's market has stupid assembly cells, and researchers are working to make them smarter. Only pilot projects of completely automated single-purpose systems are completed in disassembly automation. The problem is collecting and transporting enough of these things to one spot. An automated or semiautomated cell must dismantle many products. Products with identical disassembly activities are collected in "Disassembly families" - modular cells designed by computer-aided planning tools. A modular concept for developing disassembly cells for different families is also provided. The future hinges on "new technologies" like cheap embedded sensors and RFID. The robot's HMI will become more human-friendly. There are several security issues, such as "software walls or fences". His main goals were modular, intelligent, flexible assembly and disassembly cells with automated part recognition (kind, surface, dimensions, orientation) and universal clamping mechanisms. Mobile robots and Multi-Agent-Systems (MAS) were used to deconstruct huge, heavy objects (e.g., autos). Disassembly-Multi-Agent-Systems (DMAS) avoid building enormous disassembly cells for large items. Today, little steps have been taken. A host computer directs mobile robots. Real DMAS would be practical in the next decade.

In 2014, Sotiris Makris et al. [64] attempted to provide a method for the organic programming of robots with two arms. In order to accomplish this, it was described in detail how to use the proposed dual-arm robotics library with the human language in a task-oriented programming process. The robotics library allows for dual control, which is essential for realistic human simulation. Using a service-oriented approach, the creators of this ROS-based framework made it easy for users to get started. An easy-to-use graphical user interface, depth sensors, noise-cancelling microphones, and a robot platform with two arms are all you need to take command. A dual-arm robot platform has been modified with techniques integrated for use in vehicle assembly. They concluded that the benefits of this proposed framework include:

a. A simplification of the coding process.
b. A robot that a layperson can control through familiar gestures, voice instructions, and user interfaces.
c. Frames and location data from the robot can be imported into a database with little effort.

In 2014, Panagiota Tsarouchia et al. [4] investigated a case study that uses information gathered from a vehicle manufacturer's assembly line. Two-armed robots are discussed, along with their usefulness and necessity. A comprehensive analysis of robotics programs, components, and station layouts is performed. The benefits of utilizing dual-arm robots to undertake tasks that were previously done by people are emphasized, but the advantages of using single-arm robots are also explored. As an example of how this idea is put to use in the automotive sector, they looked at dashboard assembly. Overall, the twin arm robot is a fantastic option for boosting the degree of automation in typically hand-operated assembly lines due to its versatility and affordable price. The flexibility of the dual-arm robot to do tasks either separately or in unison increases its capacity for multitasking. The dual-arm robot can aid in cost reduction by substituting for traditional robots and pricey fittings and grabbing devices. They concluded their study with the following:

a. Increase assembly station automation.
b. Choose a twin arm robot over a single arm robot, which requires more complicated and expensive gripping mechanisms.
c. Increase the robot's cell workspace for storage or additional station.
d. Dual-arm robots have bi-manual control functions, whereas single-arm robots must be synced.

In 2018, Niki Kousi et al. [5] investigated the changes to manual manufacturing lines when workers are given mobile, two-arm assistants. These robot employees can do a wide variety of tasks while moving freely and safely around the factory floor (handling, screwing etc.). They are able to do so in addition to working at a pace of production that is on par with or above the market average. A case study from the automotive industry is used to evaluate the effectiveness of the proposed solution. The results demonstrate that far higher rates of resource usage are feasible. Using an example from automobile suspension assembly, this hybrid paradigm needs to be adapted in three key areas:

a. Production system layout.
b. Process and environment sensing.
c. Integration and control.

Preliminary research suggests that redistributing tasks between robots and humans can significantly improve operator working conditions while keeping output at or above present levels.

2.7 RESEARCH GOALS OF THIS STUDY

Path planning is a very important issue for all robotic fields, especially in the working area that contains obstacles to avoid any collision that may hurt the robot itself and/or the valuable obstacles; this study focuses on path planning algorithms and compares the results of A*, Hybrid A*, Rapidly exploring Random Tree, and Probabilistic Road Map beside comparing proposed robotic arms which are ABB IRB 120 and RoboSEED S6H4 based on the manufacturer drawings to be simulated in the MATLAB code for pick and place tasks with axial planner fixed obstacles and compare the results based on the links lengths and joints rotation range in order to investigate the better robotic arm and path planning algorithm for the proposed tasks and working areas.

3 METHODOLOGY

In this study MATLAB 2022b Academic version is used for the following processes and analyses to achieve the suitable path for a mobile 6DOF robotic arm.

a. In the image processing, the start and target points are identified according to the predefined properties considered as a reference for the proposed system based on the captured image by two cameras one of them in the top of the working area and the second one located in front of the side view of the working area.
b. In path planning process, the proposed system detects the obstacles that the robot should avoid during the task, in order to start the next step which compares the available algorithms (A*, Hybrid A*, RRT, PRM) to find the suitable path planning algorithm to reach the target point after segmenting the workspace and filtering the region.
c. In kinematics analysis, the base area of the mobile robotic arm and the angular values of each joint are calculated to achieve the required configuration to perform the pick and place operations with\ without avoiding the obstacles Next the inverse kinematic calculations are carried out and all these analyses are performed by using MATLAB.

3.1 IMAGE PROCESSING

MATLAB Image Processing toolbox offers a complete collection of algorithms and workflow applications for image processing, analysis, visualization, and algorithm development. Using both deep learning and conventional image processing techniques, picture segmentation, image enhancement, noise reduction, geometric changes, and image registration is possible. It facilitates the processing of 2D, 3D's and arbitrary-sized pictures and the automation of typical image processing processes. The picture data is segmented interactively and evaluated for image registration algorithms and handled for big data sets in bulk. Visualization tools and applications enable the exploration of photos, 3D volumes, and films; contrast adjustment; histogram creation; and region manipulation (rois). The algorithms are accelerated by utilizing multicore CPUs and graphics processing units. Numerous toolbox functions assist the development of C/C++ code for desktop prototyping and the deployment of embedded vision systems [65]. As shown in Figure 3.1, the first step considered in this study is reading the image from the camera or the stored image then convert it to gray scale image to change the dimension of the image to be a row * column matrix instead of row * column*layer matrix

(RGB images), in order to be converted to binary image to enable detecting the obstacles and the start and target points.

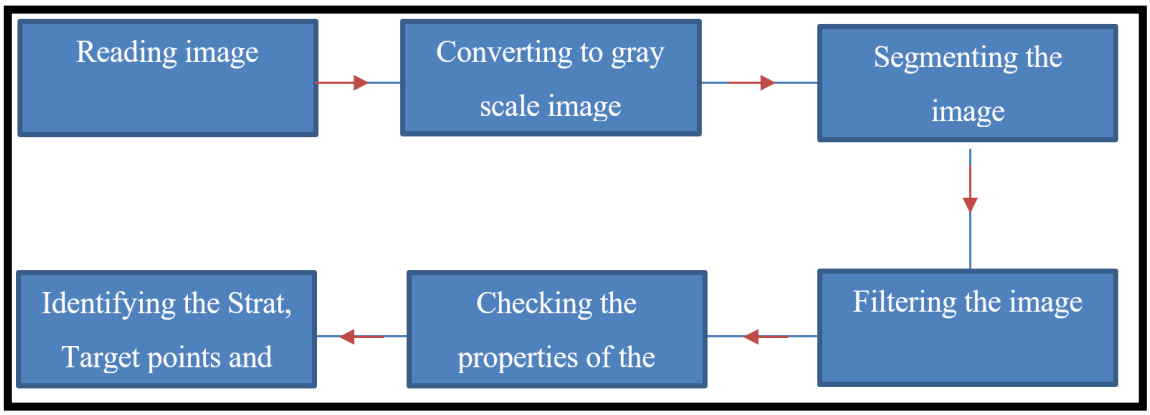


Figure 3.1: Image Processing Procedures Flow.

Set of MATLAB functions are used to segment the image starting from adjusting the image by intensity values or colormap through identifying the thresholds of binarizing the grayscale image to complement image to be easy to identify the objects. Then filtering the image by extracting objects from binary image using properties to identify the objects according to the requirements.

The below functions are used for this purposed image processing algorithm:

a. **J = imadjust (I,[low_in high_in],[low_out high_out],gamma)** maps intensity values in I to new values in J, where gamma specifies the shape of the curve describing the relationship between the values in I and J.
b. **BW = imbinarize (I,'adaptive',Name,Value)** creates a binary image from image I using name-value pairs to control aspects of adaptive thresholding.
c. **J = imcomplement (I)** computes the complement of the image I and returns the result in J.
d. **SE = strel ("disk",r,n)** creates a disk-shaped structuring element, where r specifies the radius and n specifies the number of line structuring elements used to approximate the disk shape. Morphological operations run much faster when the structuring element uses approximations.
e. **J = imdilate(I,SE)** dilates the grayscale, binary, or packed binary image I using the structuring element SE.
f. **BW2 = imfill (BW,'holes') fills** holes in the input binary image BW. In this syntax, a hole is a set of background pixels that cannot be reached by filling in the background from the edge of the image.

g. **BW2 = Bwpropfilt (BW,prop,range)** extracts all connected components (objects) from a binary image BW whose value of property prop is in the specified range. bwpropfilt returns a binary image BW2 containing only those objects that meet the criteria.

h. **Stats = regionprops(output,___)** returns measurements for a set of properties, where output specifies the format of the returned measurements as an array of structures or a table.

3.1.1 Thresholding Technique

Binary images are much easier to examine compared to grayscale photos; nevertheless, raw images frequently cannot be transformed to binary without first being subjected to some form of pre-processing. Because of this, it is frequently necessary to threshold a grayscale image to create a binarized version of the image so that it may be divided into regions that represent the foreground and the background. There is a specific use of intensity quantization known as "binarization," in which an image is created using only the two possible levels of black and white. [66]. In other words, thresholding is the procedure of transforming a picture into a binary image with intensity values of 0 or 1 respectively black and white. In this procedure, as depicted in Figure 3.2, binary number1 is assigned to intensity values that are larger than the threshold value (*K*), and binary number 0 is assigned to values that are less than the threshold value. [67] as shown in the Equation 3.1 [68].

$$output\ (x,y) \quad \begin{matrix} 0, input(x,y) < K \\ 1, input(x,y) > K \end{matrix} \tag{3.1}$$

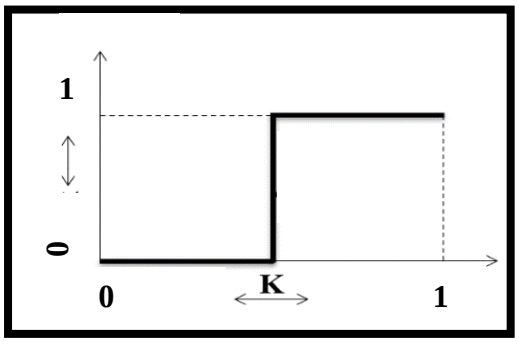


Figure 3.2: Thresholding Function.

3.1.2 Edge Detection

The intensity changes of an image suddenly cause the appearance of its edges. Edge detection is a technique for pinpointing the locations of these sudden shifts in brightness. The threshold image is used to identify the outer contours of the image, and the smoothed image is used to identify the inside contours of the image, making edge detection a two-step procedure. The detection of the snapshot's most crucial contours is ensured by combining the two photos [69]. As one of the most well-known edge detection techniques, canny edge detection is used in most image processing applications. Th pillars of the canny edge detection are low error rate, good edge point localization, and a single answer per edge [70].

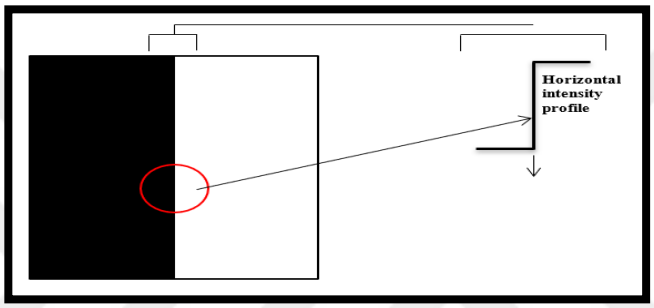


Figure 3.3: Detecting an Edge in the Image.

3.1.3 Object Centroid

In this study, getting the center of the detected object is performed to identify the starting and target points, as the center of the small green circle will be considered bigger red circle as the target for path planning, and the center of the small green square is the picking point, and the center of the bigger red square is the placing point for pick and place tasks. For getting the center of a circle, the "*centroid*" property of MATLAB function named ***regionprops*** is used to get the center of the detected objects as shown in Figure 3.4 using Equation 3.2.

$$
\begin{aligned}
&(\mathrm{Cx}, \mathrm{Cy}) = \big((Ax - Bx), (Ay - Dy)\big) \\
&(\mathrm{Cx}, \mathrm{Cy}) = \big((6-2), (5-1)\big) = (4,4) \\
&\textbf{or} \\
&(\mathrm{Cx}, \mathrm{Cy}) = \big((Dx - Cx), (By - Cy)\big) \\
&(\mathrm{Cx},\mathrm{Cy}) = ((6\text{-}2),(5\text{-}1)) = (4,4)
\end{aligned}
\tag{3.2}
$$

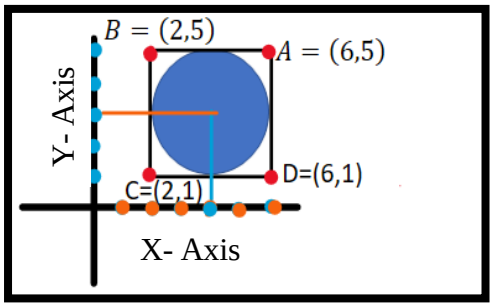


Figure 3.4: Centroid Calculations.

3.1.3 Proposed Image Processing Algorithm

This study used a special algorithm defined according to the proposed cases, two user-defined function is created, the first one is for image segmentation by adjusting the captured image using MATLAB function ***imadjust*** then to binarize it using ***imbinarize*** MATLAB function to be able to get the negative image using ***imcomplement*** MATLAB function to be easy and accurate for segmentation, then the segmented image will be considered for the second user-defined function which is for region filtration which is for recognition the detected objects (using MATLAB *bwpropfilt* function) then getting properties of each detected object through *regionprops* MATLAB function. Figure (3.5) shows the flow chart proposed image processing algorithm.

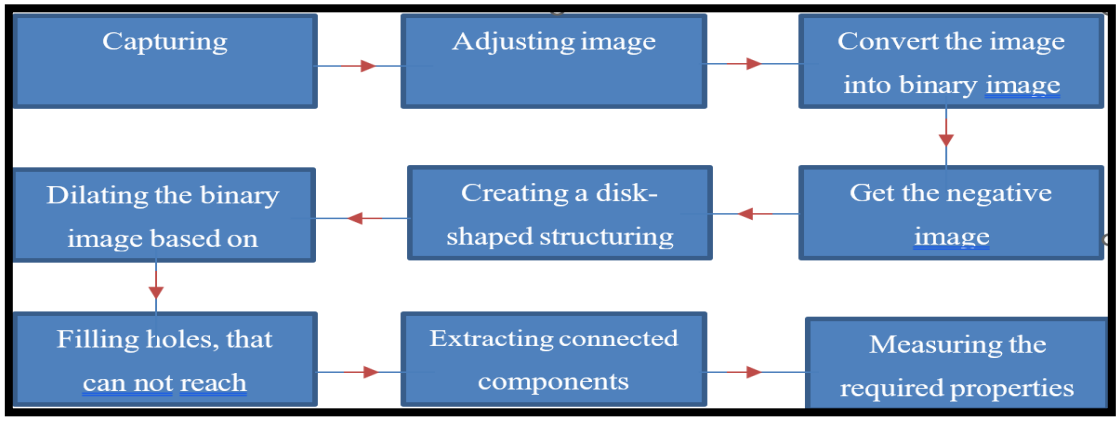


Figure 3.5: Proposed Image Processing Algorithm.

3.2 PATH PLANNING

In this study two main path planning categories are considered:

a. Planner path planning with obstacles avoiding the robot body motion.
b. Path planning for the robotic arm for pick and place tasks with avoiding a single obstacle and the base position of the robot.

Task execution begins with robot body motion from the start point to the target point in order to perform the pick and place operations, so first planner path planning methods are needed. Usually, positions of the start point, and the target point must be identified in cartesian coordinate space also, the initial points which are safety points must be identified in the work envelope of the robot arm. So, the path of all points for the robot to follow. Since it is assumed that there are some obstacles in the cases of this thesis, the proposed method based on image processing lets the robot to move without any collisions by discovering the obstacles in the operation area. The following path planning methods offer more accurate and safer paths in a specific workspace and the robot will decide which nodes could be considered walkable.

3.2.1 A* Algorithm

It is a search algorithm found by Peter Hart, Nils Nilsson, and Bertram Raphael in 1968 searching the route that takes the least amount of time to get from the beginning state to the end state based on"graph search algorithms". It has been utilized in a variety of applications in mapping [71]. The steps of this method are setting up the environment, making sure the path is safe, and then optimizing it, so, it starts with covering the area with grids. To make the search area a two-dimensional area. In this area the captured image of the area contains squares/rectangles in the arrays, each one of them can be walkable or not. Finding the path involves determining which squares/rectangles must be crossed to reach the destination from the beginning one. After identifying the squares/rectangles, the robot must go from the center of one squares/rectangles called "node" to the next until it reaches the target. Starting from the initial point, inspecting nearby squares/rectangles, and searching outward discovering the path until reaching the destination. Ignore walls and foreign objects which are basically obstacles and count all reachable or walkable squares around the starting location. To choose one of the adjacent squares/rectangles from the open list (essentially a list of squares to be checked out) and continue till reaching the destination. The question is which nearby squares/rectangles should be chosen by the algorithm to acquire a low-cost path and reduce the f score.

$$f=g+h \tag{3.3}$$

Where g is the movement cost to travel from the beginning point to a given square on the grid, following the path generated to get there, and h is the anticipated movement cost to move from those squares/rectangles to the end destination, point b. The heuristic can be perplexing. Its name comes from being a guess. Because of obstacles such as walls and machines etc., the

distance is not known when finding the path. The heuristic be formulated in numerous ways [41] [72].

In this work, Manhattan heuristic approach is applied, which calculates the total number of squares/rectangles to move the robot horizontally and vertically from the present square/rectangle to the target square/rectangle, ignoring diagonal movement and impediments. This continues until the destination square is added to the closed list representing a path or to

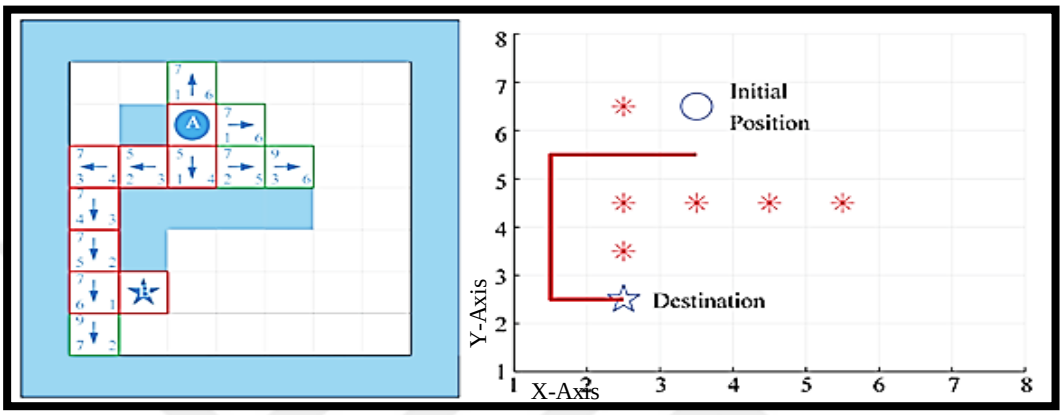


Figure 3.6: A* Path planning for Manhattan Approach. [73]

the open list representing no available path. Now, using the parent pointers for each square/rectangle, work back from the destination square/rectangle to the initial square to find the path, as shown in Figure 3.6 [41].

Table 3.1: A* Algorithm.

```
A* (Start, Goal)
Closed-set = the empty set
Open-set = includes start node
G[Start] = 0, H[Start] = H_calc [Start, Goal], F [Start] = H[Start]
While Open-set ≠ 0 Do
CurNode ←EXTRACT-MIN- F (Open-set)
      If (CurNode == Goal), then return Best Path
  For each Neighbor Node N of CurNode
   If (N is in Closed-set), Then Nothing
   Else If (N is in Open-set),
     calculate N's G, H, F
   If (G [N on the Open-set] > calculated G[N])
     RELAX (N, Neighbor in Open-set, w)
     N's parent=CurNode & add N to Open-set
```

3.2.2 Hybrid A* Algorithm

It is an algorithm proposed based on the traditional A* algorithm. Because of its widespread application in situations where a path planning algorithm for non-holonomic robots, most commonly automobiles, is required, it is typically seen in autonomous vehicles. The heuristic functions are what differentiates hybrid A* from A* as the primary distinction. When determining the cost of the path, the hybrid A* algorithm considers a number of different elements. These include the size of the robot, its kinematic model, which includes the steering angle, and the speed of the robot. Therefore, hybrid A* is able to produce routes that are not only condensed but also smooth and capable of being driven by a vehicle.

This algorithm has two heuristic functions. The first heuristic function, non-holonomic-without-barriers, ignore environmental obstacles and calculates based on the car's non-holomictic. The second heuristic function estimates the shortest distance to the objective while ignoring the car's non-holonomic nature, making it beneficial for recognizing all obstacles and dead ends.

3.2.3 Rapidly Exploring Random Tree (RRT)

It is a searching algorithm which based on sampling-based planning algorithms, RRT is a popular choice that can generate a graph and locate a path, albeit it may not be the best solution for designing routes for autonomous vehicles. The RRT algorithm first plants a tree with its roots at the starting node; the tree is then built-in stages by randomly selecting samples from the search space; finally, the tree expands toward the unknown area due to its natural inclination. For the purposes of developing various planning algorithms, RRT can be thought of as a monte CARLO method for producing open-loop trajectories for autonomous vehicles with state constraints. Below is the algorithm of general function [73].

function BuildRRT(qinit, K, ∆q)
T.init(qinit)
for k = 1 to K
qrand = Sample() -- chooses a random confi guration
qnearest = Nearest(T, qrand) -- selects the node in the RRT tree that is closest to qrand
if Distance(qnearest, qgoal) < Threshold then
return true
qnew = Extend(qnearest, qrand, ∆q) -- moving from qnearest an incremental distance in the direction of qrand
if qnew ≠ NULL then
T.AddNode(qnew)
return false
function Sample() -- Alternatively,one could replace Sample with SampleFree(by using a collision detection algorithm to reject samples in C_obstacle
p = Random(0, 1.0)
if 0 < p < Prob then
return qgoal

elseif Prob < p < 1.0 then
return RandomNode()

3.2.4 Probabilistic Road Map (PRM)

PRM is a search algorithm introduced in the 1990s by Lydia E. Kavraki and Jean-Claude Latombe. Its main benefit is that it requires testing only a small number of points to determine whether or not a set of points and the lines connecting them are free of obstacles. One of PRM's key flaws is that it treats the robot as though it were a single point with 360-degree vision. [73]

3.3 GENERAL KINEMATICS ANALYSIS

The study of how bodies move without considering forces or moments is called kinematics. Robot kinematics is the scientific study of the motion of a robot's manipulator. Developing suitable kinematics models for robot mechanisms is essential for analyzing the performance of industrial manipulators. Modelling the kinematics of manipulators typically involves two different spaces, notably Cartesian space, and Quaternion space. Two Cartesian coordinate systems can be transformed via rotation and translation. Euler angles, Gibbs vectors, Cayley-Klein parameters, Pauli spin matrices, axis and angle, orthonormal matrices, and Hamilton's quaternions are only some of the many ways rotation can be expressed. Most robotics applications use representations based on orthonormal matrices, which are 4x4 real matrices that undergo homogeneous transformations. According to Denavit and Hartenberg's methodology, a generic transformation involving two joints requires the cooperation of four independent variables. The Denavit-Hartenberg (DH) parameters are now universally accepted as the gold standard when describing robot kinematics. While quaternions are a beautiful way to express rotation, homogeneous transformations have seen more use in robotics. The dual quaternion is a compact representation of a rotational and translational transformation. Homogeneous transformations use nine elements instead of only four quaternions to indicate a body's orientation. Regarding the kinematics of robot chains, it provides substantial advantages regarding computational robustness and storage economy [74], [75].

3.3.1 Denavit—Hartenberg (DH) Parameters

Industrial robots with orthogonal workspaces are typically represented as an open series chain with joints linking connections. Similarly, the kinematic model is commonly used to describe

the relationship between the robot's working space and the joint space. The robot will need to develop its own kinematic models. The kinematic relations were defined using the Denavit—Hartenberg (DH) model, one of the most used approaches for robot modelling, to facilitate the application of the results to real-world industrial robots. By specifying the values of four parameters for each link, the kinematics of a robot can be expressed using the Denavit-Hartenberg notation. A link's length and twist angle characterize the link itself, while the offset and joint angle describe its relationship to neighboring connections. The joint variable (q) and the other three values (a, b, and c) are called link parameters (which are always the same) in the case of a standard revolute joint. For prismatic joints, d is the joint variable, whereas the other three integers are link parameters that cannot be changed [76].

3.3.2 Homogeneous Transformation Matrices

The spatial location and orientation of the coordinate frames is described by the term homogeneous transformation matrices (HTM). Every matrix may execute either of the following operations, however a homogeneous transformation matrix of size (4x4) is specifically designed to conduct these transformations between different planes, whether they be 2D or 3D. Any transformation (such rotation or translation) applied to a vector must preserve its integrity as a unitary image. Following is a definition of the homogeneous transformation matrix for the three-dimensional plane [77]. As shown in Figure 3.7 [68].

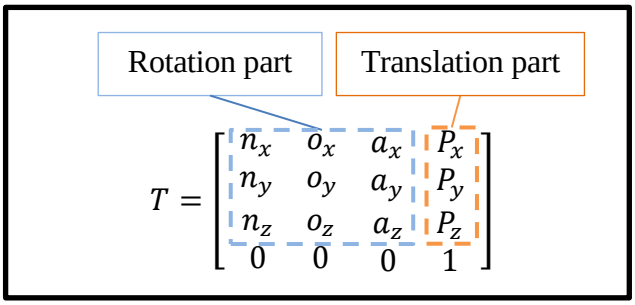


Figure 3.7: Homogeneous Transformation Matrix.

Where:
(***n***, ***o***, and ***a***): Are three-unit vectors as shown in Figure 3.8.

a: Show the angle of the end-tilt effectors in relation to the reference plane.

o: Show the direction of the end-orientation effector's vector.

When (***n*** = ***o*** x ***a***) is multiplied by itself, the result is a vector called a normal vector.

Px, Py, Pz Show where the center point of the end effector is in relation to the reference plane.

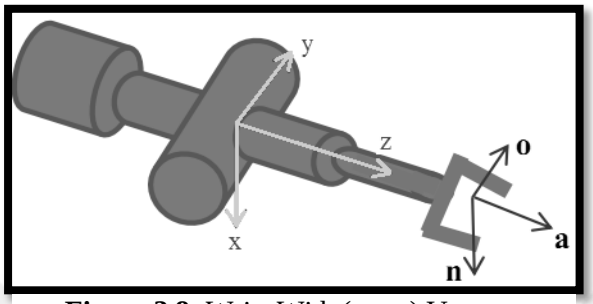


Figure 3.8: Wrist With (***n***,***o***,***a***) Vectors.

3.3.2.1 Translation matrix

The translation of a frame from one plane into another plane can be performed by multiplying it with the homogeneous matrix, the results represent the transformation with no change in orientation, as show in Equation 3.4 [78].

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Where:
t_x: Translation in the x-axis.
t_y: Translation in the y-axis.
t_z: Translation in the z-axis.

3.3.2.2 Rotation matrix

It's 4 by 4 matrix, which represents the axis of rotation as follows in Equations 3.5 [78].

a. Rotation about (x-axis)

$$R(x,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \text{Cos}\,\alpha & -\sin\alpha & 0 \\ 0 & \text{Sin}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5 a}$$

b. Rotation about (y-axis)

$$R(y,\emptyset) = \begin{bmatrix} \text{Cos}\,\emptyset & 0 & \text{Sin}\,\emptyset & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\emptyset & 0 & \text{Cos}\,\emptyset & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5\text{ b})$$

c. Rotation about (Z-axis)

$$R(z,\theta) = \begin{bmatrix} \text{Cos}\,\theta & -\sin\theta & 0 & 0 \\ \text{Sin}\,\theta & \text{Cos}\,\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5\text{ c})$$

3.3.3 Kinematic Analysis

It incorporates the robotic arm's displacement and velocity kinematics, which reflected the interaction between the end-effector and the joint in space. Kinematic analysis is used to determine whether or not the robotic arm crosses correctly. Forward and inverse kinematics are the two categories that may be used to classify the displacement kinematics. Figure (3.9) illustrates the process of kinematic analysis **[79] [68]**.

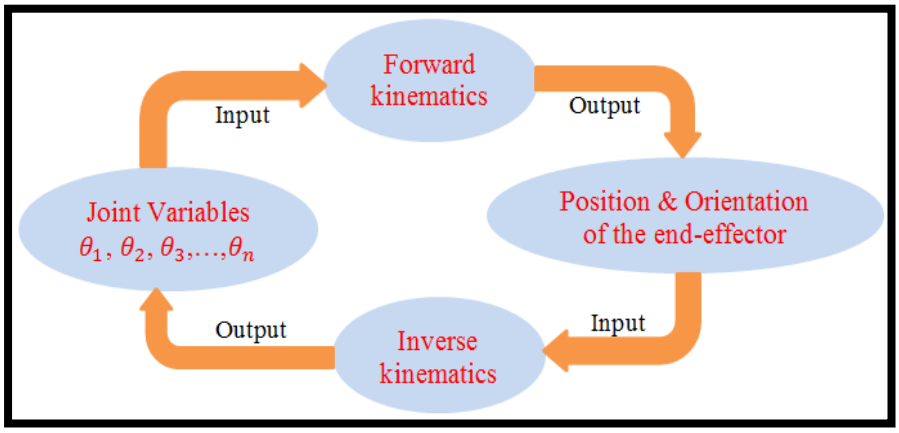


Figure 3.9: Flow Chart of the Process Kinematics Analysis.

3.3.3.1 Forward kinematics analysis

It shows a transformation matrix that extends all the way from the beginning of the robotic arm to the end-effector at the end of the limb. The Denavit-Hartenberg convention, often referred to as the DH convention on occasion, is a convention that is commonly used in robotic applications for the purpose of defining frames of reference [80].

It only requires four parameters link length (***d***), link offset (***a***), theta *i*, and alpha *i* rather of the standard six that are necessary for 3D motion, and which may be expressed in equations [81].

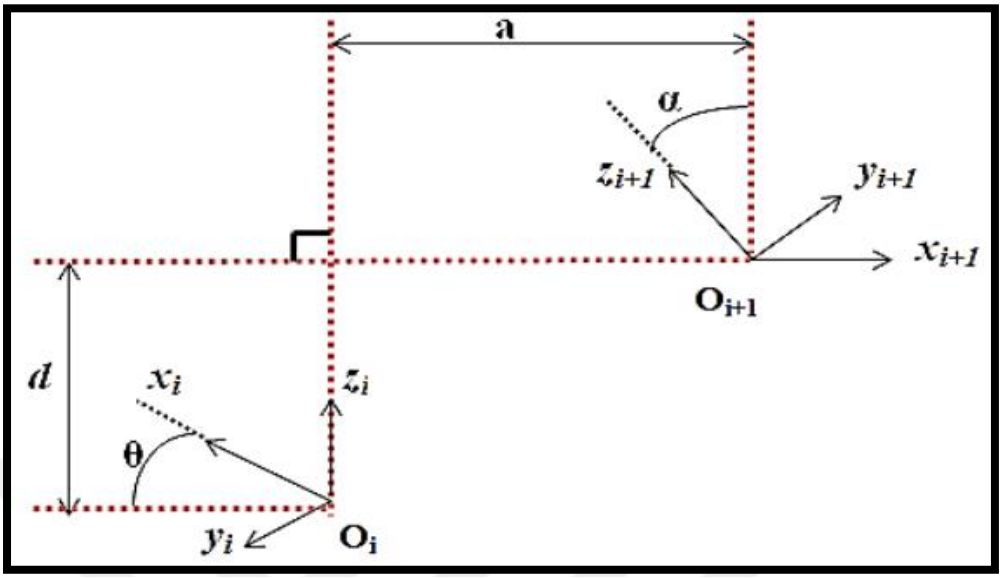


Figure 3.10: DH Frame Assignment.

$$T_i = R_{z,\theta_i}\, T_{z,d_i}\, T_{x,a_i}\, R_{x,\alpha_i}$$

$$T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_i = \begin{bmatrix} c\theta_i & -s\theta_i\, c\alpha_i & s\theta_i\, s\alpha_i & a_i\, c\theta_i \\ s\theta_i & c\theta_i\, c\alpha_i & -c\theta_i\, s\alpha_i & a_i\, s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

Where**:**

$a_i\, c\theta_i$: The end-effector position in x-axis = P_x.
$a_i\, s\theta_i$: The end-effector position in y-axis = P_y.
d_i: The end-effector position in z-axis = P_z.
a_i: Link length.
α_i: Link twist.
d_i: Link offset.
θ_i: Joint angle.

3.3.3.2 Forward kinematic for articulated robot

A robotic arm with six degrees of freedom (also known as 6 DOF) is a type of instructional robot that consists of a base, shoulder, elbow, tool pitch, and tool roll. Each joint in this type of robot is rotating, and it offers five different motions in addition to a grasping action.

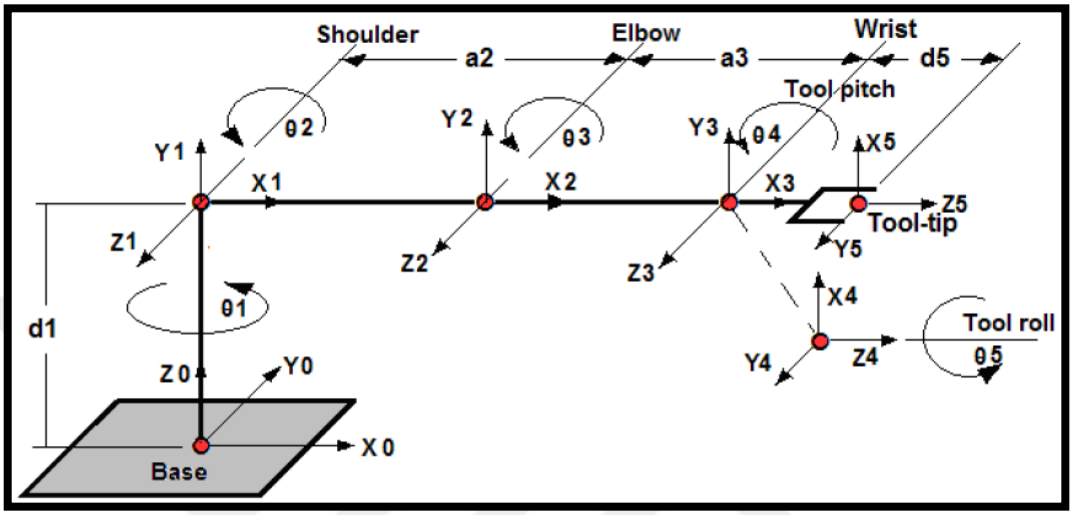


Figure 3.11: Link Coordinate Schematic of the Robotic Arm.

Table 3.2: DH Parameters Table.

(*i*)	a_i (mm)	α_i (degree)	d_i (mm)	θ_i (degree)
1	0	90	255/290	-+160/ -+165
2	275/270	0	0	-90,+110/ -+110
3	93.8/70	0	0	0,+160/-110,+170
4	258.5/302	90	0	-290,+105/ -+160
5	0	0	99/72	-+130/-+120

Where:
i: Link number.

$$T_1^0 = R_{z,\theta_1}\, T_{z,d_1}\, T_{x,a_1}\, R_{x,\alpha_1}$$

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\,(90) & -s(90) & 0 \\ 0 & s(90) & c(90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^0 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

In a similar way will find the T_2^1, T_3^2, T_4^3, and T_5^4.

$$T_2^1 = R_{z,\theta_2}\, T_{z,d_2}\, T_{x,a_2}\, R_{x,\alpha_2}$$

$$T_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = R_{z,\theta_3}\, T_{z,d_3}\, T_{x,a_3}\, R_{x,\alpha_3}$$

$$T_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = R_{z,\theta_4}\, T_{z,d_4}\, T_{x,a_4}\, R_{x,\alpha_4}$$

$$T_4^3 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = R_{z,\theta_5}\, T_{z,d_5}\, T_{x,a_5}\, R_{x,\alpha_5}$$

$$T_5^4 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The position and orientation of the wrist of the robotic arm can be obtained by multiply the matrices (T_1^0 , T_2^1 , and T_3^2):

$$T_3^0 = T_1^0\, T_2^1\, T_3^2 \tag{3.8}$$

$$T_3^0 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^0 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(a_3c_{23} + a_2c_2) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(a_3c_{23} + a_2c_2) \\ s_{23} & c_{23} & 0 & a_3\, s_{23} + a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^3 = T_4^3\, T_5^4 \tag{3.9}$$

$$T_5^3 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^3 = \begin{bmatrix} c_4c_5 & -s_5c_4 & s_4 & d_5s_4 \\ s_4c_5 & -s_4s_5 & -c_4 & -d_5c_4 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^0 = T_3^0\, T_5^3 \tag{3.10}$$

$$T_5^0 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(a_3c_{23} + a_2c_2) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(a_3c_{23} + a_2c_2) \\ s_{23} & c_{23} & 0 & a_3\, s_{23} + a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_4c_5 & -s_5c_4 & s_4 & d_5s_4 \\ s_4c_5 & -s_4s_5 & -c_4 & -d_5c_4 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^0 = \begin{bmatrix} c_{12345} + s_{15} & -s_5c_{1234} + s_1c_5 & c_1s_{234} & c_1(d_5s_{234} + a_3c_{23} + a_2c_2) \\ s_1c_{2345} - c_1s_5 & -s_{15}c_{234} - c_{15} & s_1s_{234} & s_1(d_5s_{234} + a_3c_{23} + a_2c_2) \\ c_5s_{234} & -s_5s_{234} & -c_{234} & -d_5c_{234} + a_3s_{23} + a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.3.3.3 Inverse kinematics for articulated robot

The inverse kinematics of the robotic arm can be obtained by a geometric approach with the link parameters (a_2, a_3, d_1, d_5), as illustrated in Figure 3.12. [82]

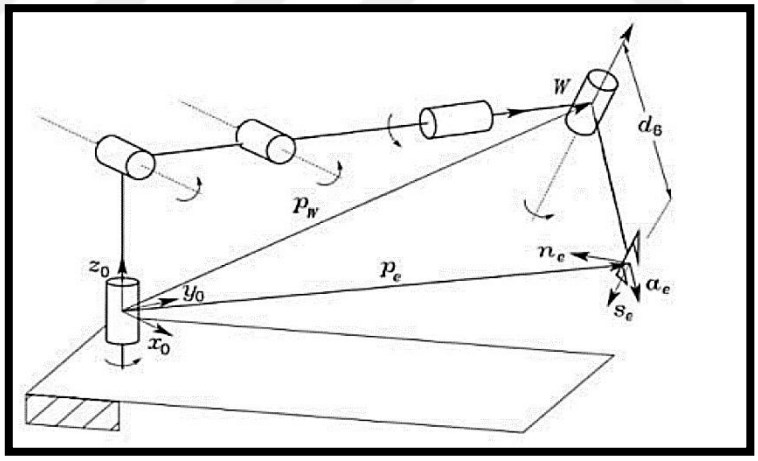


Figure 3.12: Four Link Articulated Robot. [82]

The wrist angle relative to the reference coordinate (θ_{234}) is representing the relation between (θ_2, θ_3, and θ_4): $\theta_{234} = \theta_2 + \theta_3 + \theta_4$

where θ_{234} can be calculated based on pitch wrist orientation angle $\emptyset$

$$90 - \theta_{234} = \pm\emptyset$$

$$R = d_5 * \cos\emptyset$$

$P_{x_w} = P_x - R\cos\theta_1 = X_w$, $P_{y_w} = P_y - R\sin\theta_1 = Y_w$ and $P_{z_w} = P_z + d_5\sin\emptyset = Z_w$

$$R_w = \sqrt{P_x^2 + P_y^2} - R = \sqrt{P_{x_w}^2 + P_{y_w}^2}$$

$$N = \sqrt{\left(P_{z_w} - d_1\right)^2 + R_w^2}$$

a. Solution for $\boldsymbol{\theta_1}$: base angle can be calculated from the figure (20) as follows:

$\theta_1 = \tan^{-1}\left(\frac{P_y}{P_x}\right)$ The other solution will be: $\grave{\theta_1} = \theta_1 + 180$

b. Solution for $\boldsymbol{\theta_2}$: using the law of cosines:

$$a_3^2 = N^2 + a_2^2 - 2a_2\, N\cos(\mu) \Rightarrow \mu = \cos^{-1}\left(\frac{N^2 + a_2^2 - a_3^2}{2a_2\, N}\right)$$

$$\lambda = \tan^{-1}\left(\frac{P_{z_w} - d_1}{R_w}\right)$$

$$\theta_2 = \lambda \mp \mu$$

c. Solution for $\boldsymbol{\theta_3}$:

$$N = \sqrt{(a_2^2 + a_3^2 - 2a_2a_3\cos(\pi - \theta_3))} \quad \text{and} \quad \theta_3 = \pm\cos^{-1}\left(\frac{N^2 - a_2^2 - a_3^2}{2a_2a_3}\right)$$

d. Solution for $\boldsymbol{\theta_4}$:

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3$$

e. Solution for $\boldsymbol{\theta_5}$: To estimate the value of the angles (θ_5), the matrix T_5^3 should be available:

$$T_5^3 = \begin{bmatrix} c_{45} & -s_5c_4 & s_4 & d_5s_4 \\ s_4c_5 & -s_{45} & -c_4 & -d_5c_4 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$n_z = s_5$ and $o_z = c_5$ then $\frac{n_z}{o_z} = \frac{s_5}{c_5} = \tan(\theta_5) \Rightarrow \theta_5 = \tan^{-1}\left(\frac{n_z}{o_z}\right)$

3.4 ROBOTS' DESIGN

Since this study is not focusing on the design, so two robotic arms are chosen to be studied with the main topic of this study which is path planning for obstacles avoidance. Both robots proposed to be located above a mobile platform to make these robotic arms as mobile manipulators to increase the range of tasks that could be covered by these robotic arms.

3.4.1 Mobile Platform

NEW TRACKED MOBILE TANK ROBOT KIT 10022 is a basic mobile robot tank as shown in Figure 3.13 and 3.14, kit that can be programmed. It is a low-cost mobile tracked platform that is utilized for search and rescue operations as well as educational purposes. The tracked mobile tank robot kit has the ability to go over obstacles that are in its path while traversing tough terrain. In addition to that, its climbing ability is rather strong. Because of its superior stability, traction, and reduced ground bearing pressures, the tracked mobile robot tank kit may be used for a larger variety of tasks. This study chooses it to be the mobile platform for the proposed robotic arms, as this platform has been chosen because of the suitability with the proposed robotic arms in the wise of the load and dimensions. Based on the official website of the manufacturer named NEXUS ROBOT [83], Table 3.3 shows the specifications, and the product link also can be found [84].

Table 3.3: Specifications of Mobile Platform.

Dimension:	310 X 300 X 110mm
Body Material:	Aluminium Alloy
Speed:	0.3 m/s
Colour:	Blue
Motor Power:	17 w
Load Capacity:	20 kg
Motor type:	12V DC Motor
Net Weight:	4.3 kg

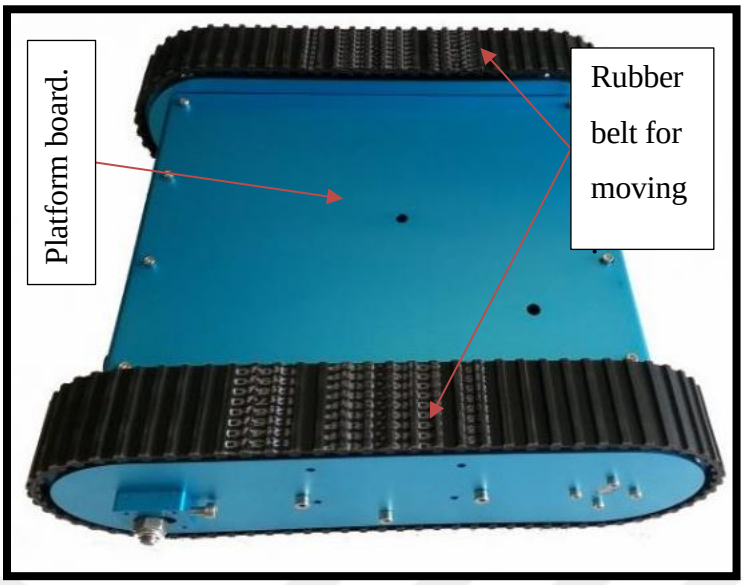


Figure 3.13: Mobile Platform - Top-Side View[84]

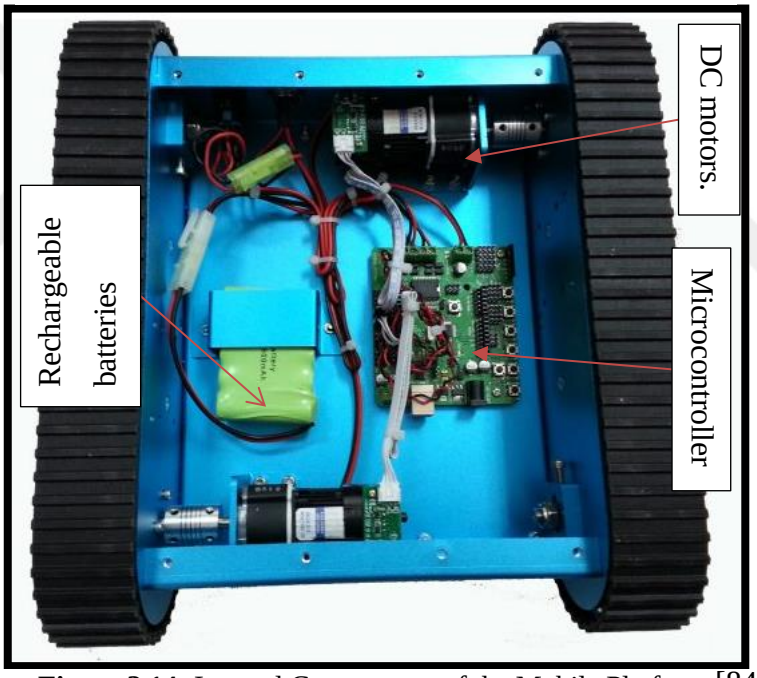


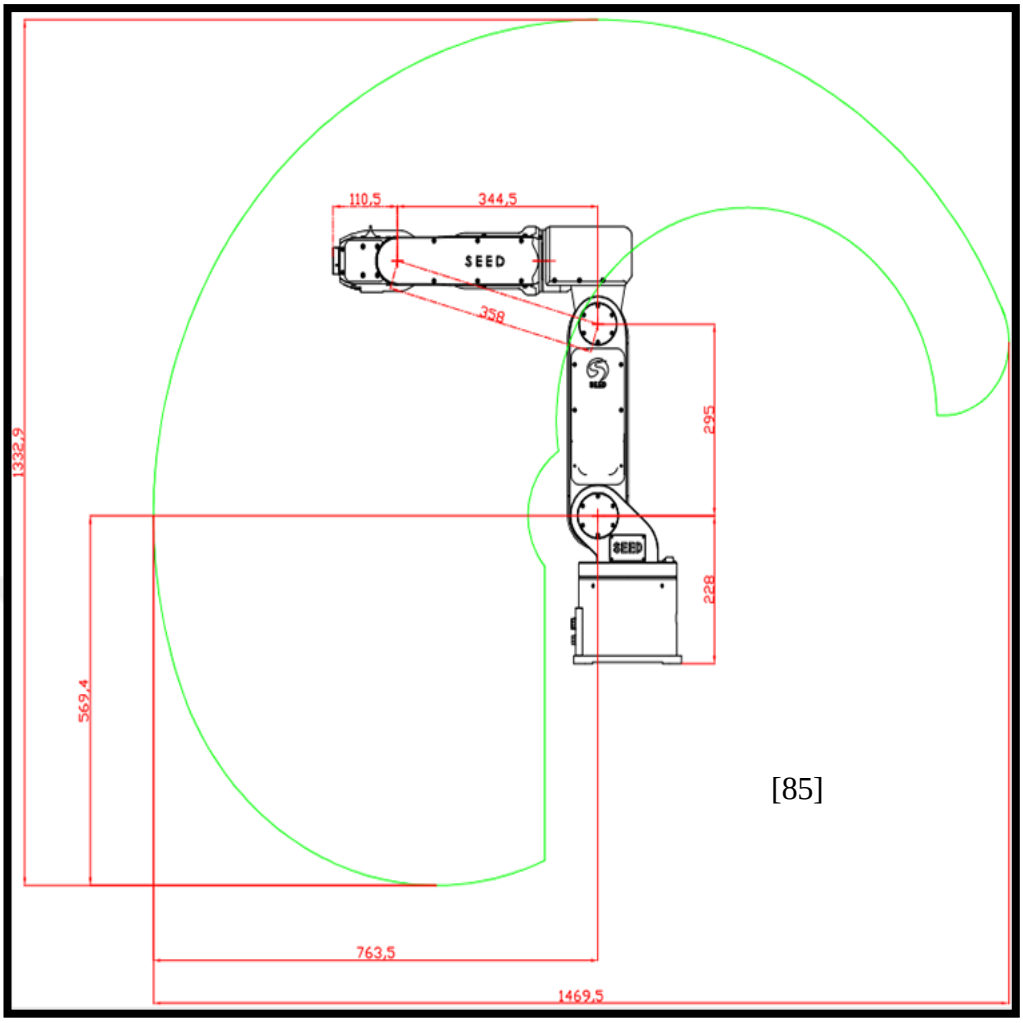
Figure 3.14: Internal Components of the Mobile Platform.[84]

3.4.2 RoboSEED S6H4

Robot Arm six degrees of freedom 6 DOF RoboSEED [85] from ROBOTDIGG is good for comparison because positioning precision of +-0.02 mm, along with strong interpolation, speed, force control, and other performance, the motion control achieves the norm for industrial robots. Extremely adaptable and user-friendly control makes autonomous placement through machine vision a breeze to implement. With the robotic arm connected, the intelligent robot's productivity increases dramatically, and it becomes capable of doing the vast majority of necessary tasks in the house, the office, and other places. Table 3.4 shows the specifications of the robot:

Table 3.4: Specifications of ROBOSEED S6H4.

Payload rated load	1Kg \ the Maximum load is 2 Kg	
Moving Range	JS (Rotary） -160°~ + 160° JL (Upper Arm) -90°~ + 110° JU (Lower Arm) -0°~ + 160°	JR (Arm Rotary) -290°~ + 105° JB (Arm Swing) -130°~ + 130° JT (Arm) -360°~ + 360°
Max Moving Speed	JS (Rotary) 160°/s JL (Upper Arm) 130°/s JU (Lower Arm) 200°/s	JR (Arm Rotary) 300°/s JB (Arm Swing) 400°/s JT (Arm Swing) 500°/s
Repeat location Accuracy	±0.05mm	
Arm Length\Weight	650mm\ 16Kgs without the seat	



[85]

Figure 3.15: RoboSEED S6H4 Dimensions.

This robot is used for input/output control signals, g-code paths, input/output arc/linear interpolation, force control, acceleration/deceleration interpolation, and deceleration/force interpolation demonstration of control method or serial connection to personal computer dimensions outside the body has the ability to manipulate two external axes, making it a true two-dimensional transformer. Figure 3.15 shows the dimensions.

3.4.3 ABB IRB 120

The IRB 120 is a member of the most recent generation of 6-axis industrial robots produced by ABB Robotics as shown in Figure 3.16 [86]. It has a payload of 3 kilograms and was developed primarily for manufacturing industries that make use of flexible robot-based automation. The construction of the robot is open, making it particularly suited for adaptable usage, and it has broad communication capabilities with systems that are external. This robotic arm is used for the suitability specification, and it is a good pattern to be simulated for this study. Its dimensions shown in Figure 3.16 and specifications mentioned in Table 3.5.

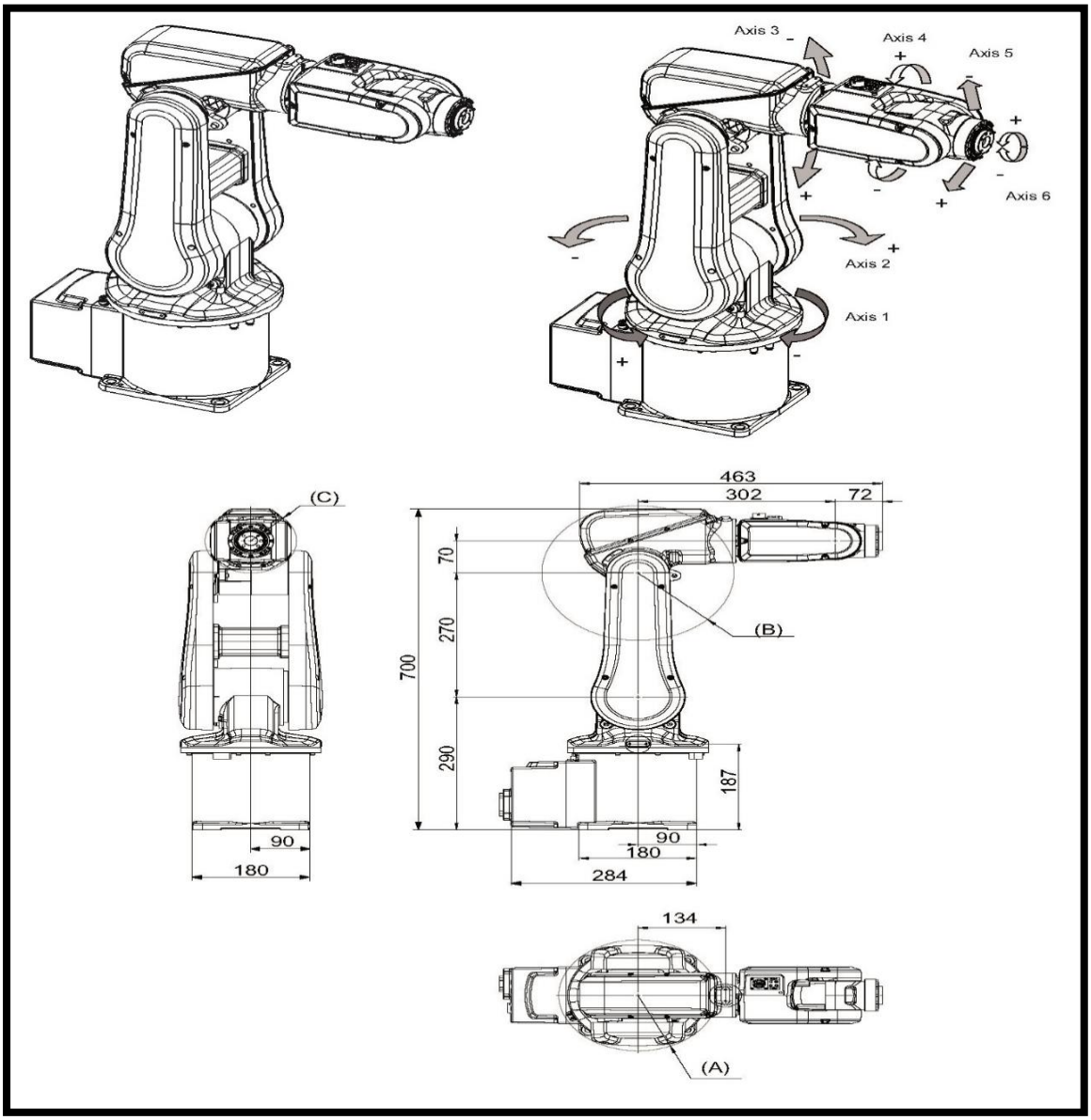


Figure 3.16: ABB IRM 120 Dimensions.[86]

Table 3.5: Motion Range of Each Joint of ABB IRB 120.

Location of motion	Type of motion	Range of movement
Axis 1	Rotation motion	+165° to -165°
Axis 2	Arm motion	+110° to -110°
Axis 3	Arm motion	+70° to -110°
Axis 4	Wrist motion	+160° to -160°
Axis 5	Bend motion	+120° to -120°
Axis 6	Turn motion	+400° to -400° (default) +242 to -242 maximum

3.5 CASES STUDY

Path planning algorithms depend on the working areas. As the algorithm itself deals with the occupied and not occupied area to be considered for finding the suitable path with no collision. The proposed cases are chosen carefully to be studied to check the weaknesses and strengths to make it easy to choose one of them according to the requirements.

3.5.1 Cases For Path Planning Algorithms

For the path planning algorithms, eight cases are studied considering in this thesis as case 1 to case 8 in all cases the black areas represent the occupied area that the robot should avoid while traveling from the starting point, which is represented by the small green circle, to the target, which is represented by a bigger red circle. These eight cases are classified into two types to simulate the common working areas, the first type considered from case 1 to case 4 which represented by a single hall with multi-machines represented by different shapes such as hexagons, diamonds, triangles, and circles with different dimensions. The robot-proposed tasks are to go to the target point without collision and avoid all obstacles to achieve a safe path. The second type is separated rooms in common production lines that are represented in this study by four rooms named respectively WA-1, WA-2, WA-3, and WA-4 in addition to two stores named as Store 1 and Store 2, here the tasks of the robot going to the target to do the required pick-place tasks without collision and avoid all obstacles, so here the occupied areas and obstacles would not be more than lines which represent the walls separated with open doors to pass through. Each case considered purposefully as shown in Figure 3.17 to Figure 3.20.

The following factors are considered for comparison:

a. Real path length in comparative with the different algorithms for the same cases.
b. The explored areas to find the suitable path, which is represented by the nodes that the algorithms discovered to find obstacles avoidance path. The more explored area the better path.
c. Number of path nodes, which are based on the explored area, the minimum number of path nodes the better path, as each node represents a coordinate location as an order to the robot to follow to achieve the path.
d. The consumed time by algorithm to find the path, decreasing the consumed time considered as a positive point for the algorithm.

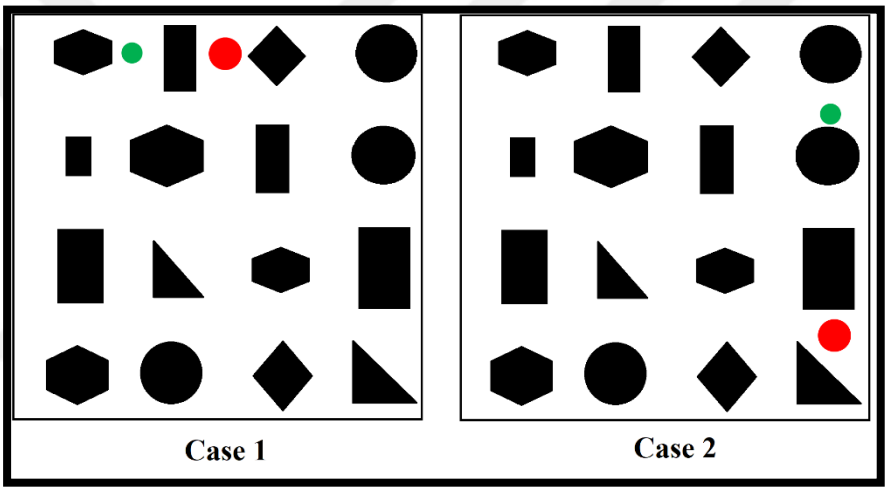


Figure 3.17: Case 1 and 2 for Path Planning Algorithms.

First type of cases for path planning algorithms as shown in Figure 3.17 for case number 1, the starting point, and the target in horizontal line with single obstacle and short distance to check the elongation rate of the resulted path and how the algorithm deals with the near obstacles horizontally, while the second case to check how the algorithms deal with long distances with multi-obstacles and the starting point is very near to the obstacle.

case number 3 is considered to check the algorithms' behavior for the long and inclined paths with multiple obstacles while case number 4 intendedly focuses on the clear area, as the normal path between starting point and the target is supposed to be with no obstacles as the Euclidian distance between them is clear as shown in Figure 3.18.

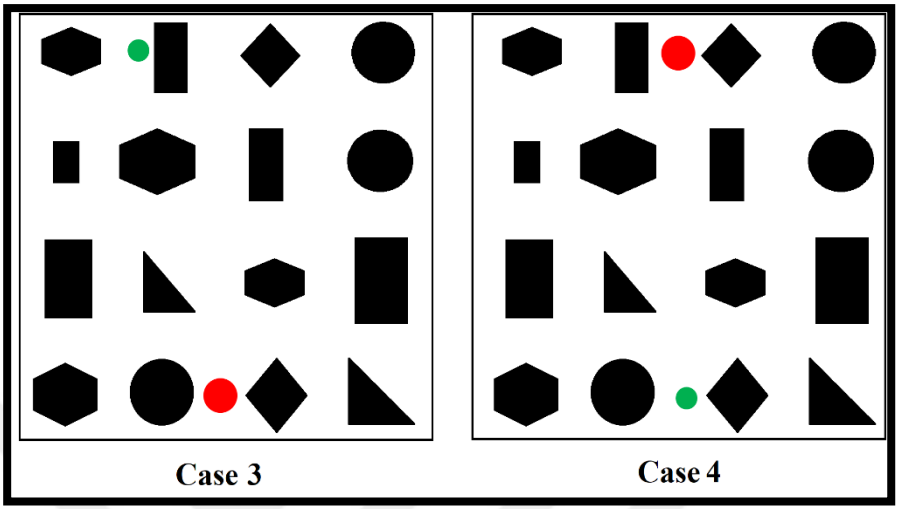


Figure 3.18: Case 3 and 4 for Path Planning Algorithms.

For the second type of cases for path planning algorithms as shown in Figure 3.19, for case number 5 to check when the starting point in the store 2 and the target in WA-4 that means they are near and the only separation is a single line, but the clear path will be through the open doors, while case number 6 to check the algorithms behavior for long and complicated path.

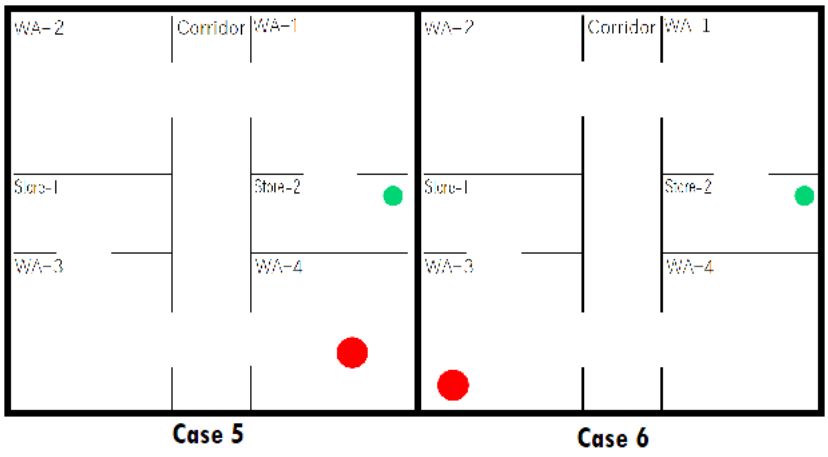


Figure 3.19: Case 5 and 6 for Path Planning Algorithms.

For case number 7 the starting point is very near to the obstacles and the distance is too long while case number 8 to check the algorithms' behavior for the clear starting point and long distance to reach the target which is far and in front of the open door which easy path, so the algorithms supposed to give perfect results as shown in Figure 3.20.

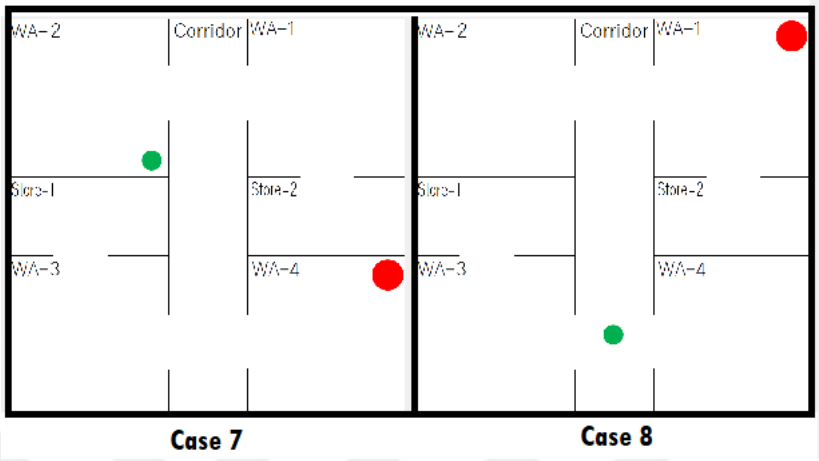


Figure 3.20: Case 7 and 8 for Path Planning Algorithms.

3.5.2 Cases For Pick-Place Tasks

Since this study focused on the path planning, the gripper has been neglected from the simulation and calculations, only the effective links of the robotic arm have been considered for the path planning. 6 DOF robotic arms (regardless the gripper) as the gripper will not affect the path planning as the end of the link no. 5 will be directed attached to the gripper which will do nothing but catching objects.

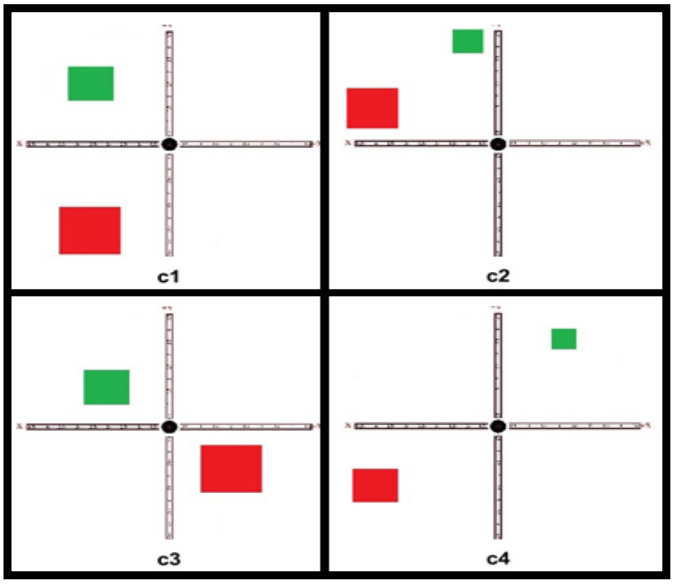


Figure 3.21: Pick-Place Cases.

Figure 3.21 and Figure 3.22 shows the considered cases, as this study tried to use conditional code to avoid the obstacles rather than using algorithms. As the proposed obstacles for the pick-place tasks are not more than a vertical separator to separate the quadrants of the pick-place task area, as each quadrant specialized for doing special task, and the robot here is to pick an object located incorrectly in another quadrant to the corrected place and level. Conditional code is considered, to overcome all possible collisions that may occurs during the pick-place tasks for the proposed working area. As the height of the obstacles, the object to be picked and the target point to the robot are considered and the quadrants and the robot base also considered as the robot cannot move through its base without collision that may destroy the robot.

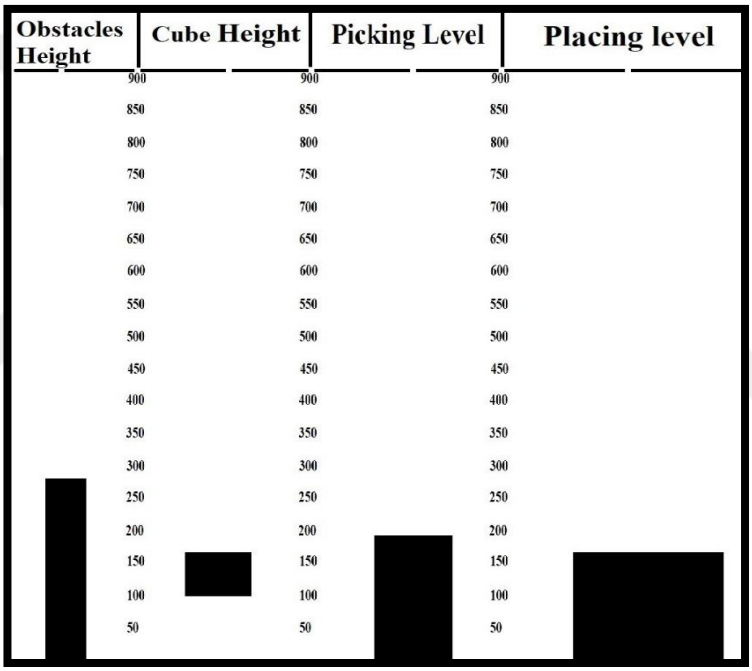


Figure 3.22: Case's Dimensions.

Four different cases are studied to check the suitability of the conditional code using conditional if statements in these cases instead of using complexed algorithms for complex cases that are not proposed working area, as the obstacles fixed and there is no chance to be changed and the complexed algorithms especially for 3-D environment would cost too much time to analyse and needs a complexed procedures for camera calibration moreover the type of camera should be expensive and carefully located in order to cover the 3-D working area.

As shown in Figure 3.21 for all pick-place cases the middle black circle represent the robot base and the graduated vertical and horizontal rulers represent the fixed and unchanged obstacles, the smaller green square represent the picking\ starting point while the bigger red

square represent the target\ placing point, here the green and red squares give the X and Y coordinates for the picking and placing points, while the Z-axis coordinates will be gotten from the image processing of the height image which gives the height of the obstacles and the picking and placing level and the height of the cube, as the cube (picking object) height should be considered in this task in order not to let the object be damaged through the task as shown in Figure 3.22.

One more thing is considered through this study which is the reachability and suitability of the proposed robot, based on the simulation and resulted path and errors of each robot in compare with the required accuracy.

As Figure 3.21 shows that for case number 1 the cube and the box in different quadrant so the vertical obstacle will affect the task according to the applied heights and levels, this case is considered to check the ability of the path planning with overcoming single vertical obstacle while case number 2 is to examine the conditional code in case of no obstacles and case number 3 for the case of the Euclidean distance which represents the shortest distance between the cube and the box is passing through the base of the robot, so here the path planning of the conditional code should overcome this issue to avoid making collision with the robot base with overcoming multi-vertical obstacles while case number 4 the diagonal distance between the cube and box will never pass through the robot base, so this study examine a proposed conditional code prepared for the proposed cases beside the examine the robots complying with the path planning.

For each case three paths will be simulated the first one in red colour to represent the shortest path regardless the obstacles and blue path which represent the resulted path to overcome the obstacle and the green path represent the real path that the robot follows based on the resulted path and the boundaries like motion range of each joint and links' lengths. So, the comparison will be between the resulted path and the real path to check the ability of the proposed robot for the proposed tasks and the shortest path has been simulated to check the effectiveness of conditional code to overcome the obstacles.

4 RESULTS AND DISCUSSION

Two robotic arms are simulated to compare the ability of each one for the proposed cases. The first step of the study is to simulate the motion of the robotic arm from the starting point to the target, to perform the second step which is pick an object and place it in corrected position based on image processing. Eight proposed cases are considered to make comparison between the results and discuss the results according to the algorithms that this study used, these cases are chosen to simulate the common production line working area, there are two groups of cases.

The first group, simulate a hall with machines aligned according to the production requirements, so the robot should move from the starting point to the target without any collision.

The first step of the study is to make MATLAB analyse the captured image to recognize the starting point and the target with two main categories:

The first one is four cases considering from p1 to p4 which representing the working area as a one hall that includes other machines like grinders, millers, 3D printers, laser printers and automatic welding machines aligned according to the production line requirements. Here the mobile manipulator should be moved from the starting point to the target without any collision considering the machines as obstacles using 4 path planning algorithms are used which are (A*, Hybrid A*, RRT and PRM). Safety margins are given to assure that the path will never collide with the obstacles.

The second category is four cases considering from p5 to p8 which representing the working area as four rooms (WA-1, WA-2, WA-3, and WA-4) and two stores. The same algorithms are used to get a suitable obstacle avoidance path. After reaching the robotic arm at the target area, it will perform the pick and place tasks based on image processing for the picking point and the placing point in 4 quadrants board with obstacles boundaries as the separation between quadrants.

4.1 PATH PLANNING RESULTS

Study considered unified properties for each algorithm for all cases as shown in Figure 4.1 to compare the adjusting effects.

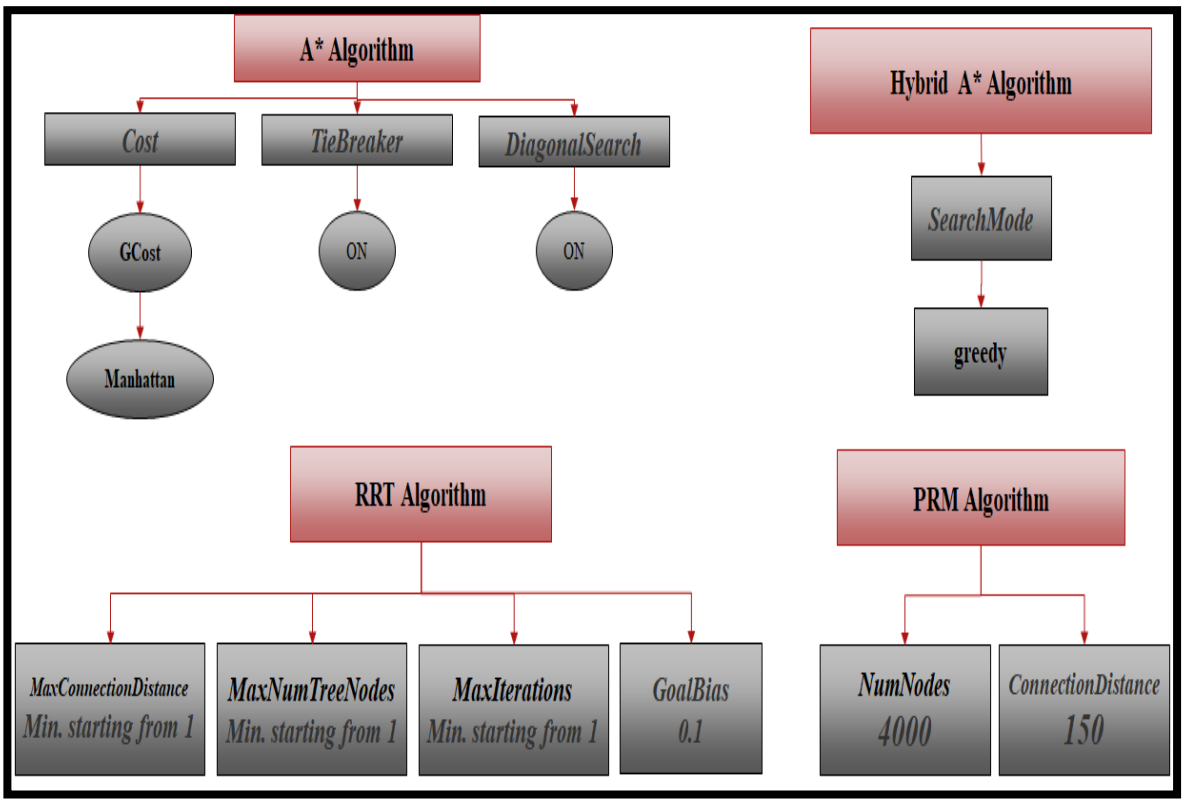


Figure 4.1: Considered Properties for Each Algorithm.

4.1.1 First Category of Path Planning Cases

Table 4.1 shows the result path for the first category for each case, the green highlighted field represents the minimum value while the red highlighted field represents the maximum value. It's easy to notice that the hybrid A* gives the shortest path length while the PRM gives the minimum number of path nodes while the A* consumes minimum time, and the RRT explored minimum nodes which is weak point for this algorithm.

Table 4.1: Algorithms Comparison for Results of First Category Cases.

Case # \ Algorithm / Factors		A*	Hybrid A*	RRT	PRM
p1	Shortest path regardless the obstacles (Per unite of length)	225.4	225.4	225.4	225.4
	Path length (Per unite of length)	331.38	291.381	473.278	303.179
	Total number of explored nodes	30358	292	107	4000
	Number of path nodes	298	292	38	6
	Elapsed time to find the suitable path (Per second)	4.98	70.48	4.08	111.70
p2	Shortest path regardless the obstacles (Per unite of length)	536.367	536.367	536.367	536.367
	Path length (Per unite of length)	612.978	600.928	782.17	647.094
	Total number of explored nodes	40706	601	129	4000
	Number of path nodes	568	601	51	10
	Elapsed time to find the suitable path (Per second)	3.06	356.06	5.94	71.22
p3	Shortest path regardless the obstacles (Per unite of length)	809.281	809.281	809.281	809.281
	Path length (Per unite of length)	922.765	846.772	1011.11	889.727
	Total number of explored nodes	92152	847	145	4000
	Number of path nodes	884	847	59	12
	Elapsed time to find the suitable path (Per second)	3.56	35.15	5.76	80.24
p4	Shortest path regardless the obstacles (Per unite of length)	786.321	786.321	786.321	786.321
	Path length (Per unite of length)	793.87	793.135	989.775	816.317
	Total number of explored nodes	5272	794	143	4000
	Number of path nodes	787	794	58	9
	Elapsed time to find the suitable path (Per second)	3.03	2.19	5.70	80.81

4.1.1.1 Case 1

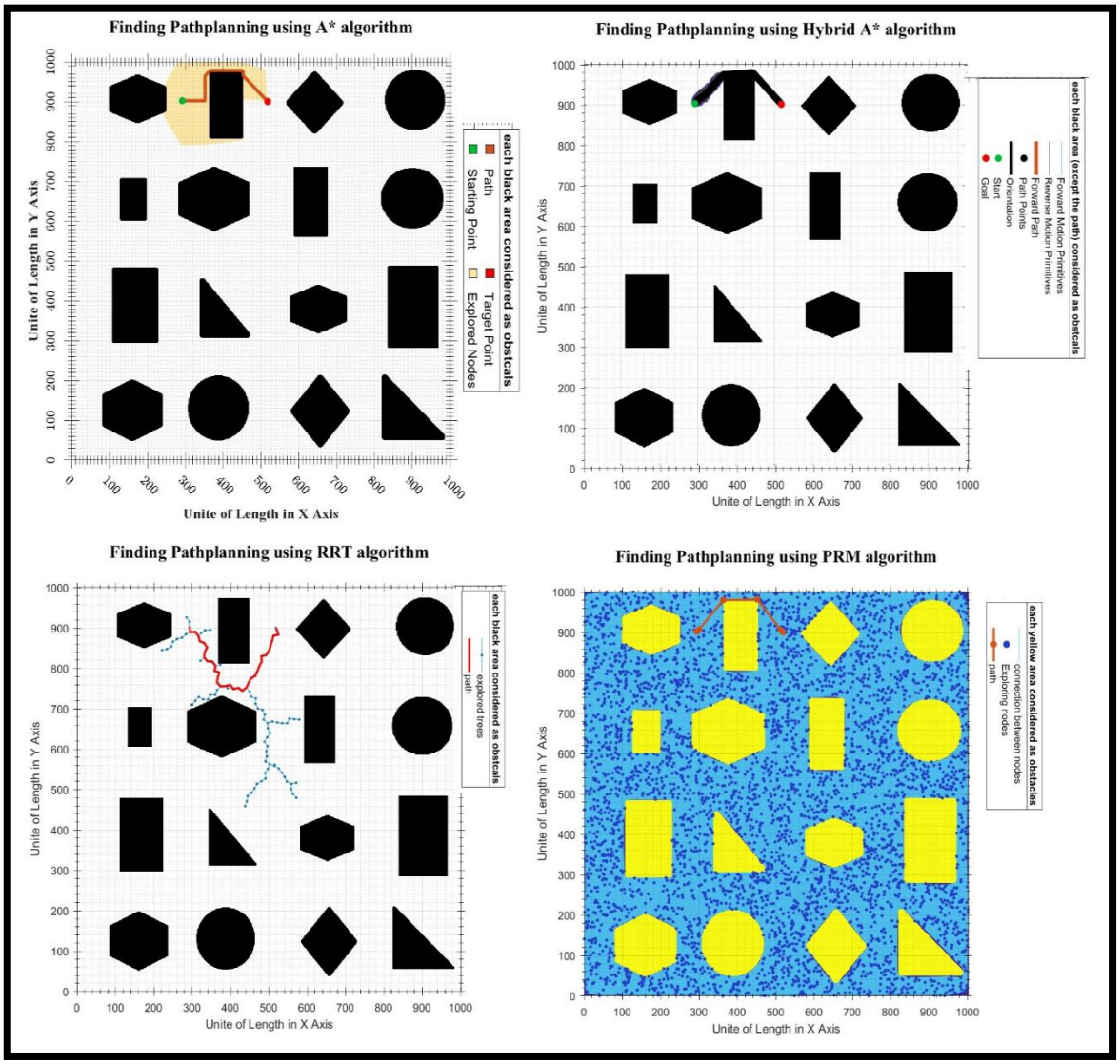


Figure 4.2: Resulted Path of Case Number 1, Using Different Algorithms.

As shown in Figure 4.2, the Hybrid A* algorithm, gives shortest path with minimum sharp turning in the turning points while the analyzing time is not the minimum, while the RRT algorithm gives longest path with many sharping turns but the advantage here is the consumed time.

4.1.1.2 Case 2

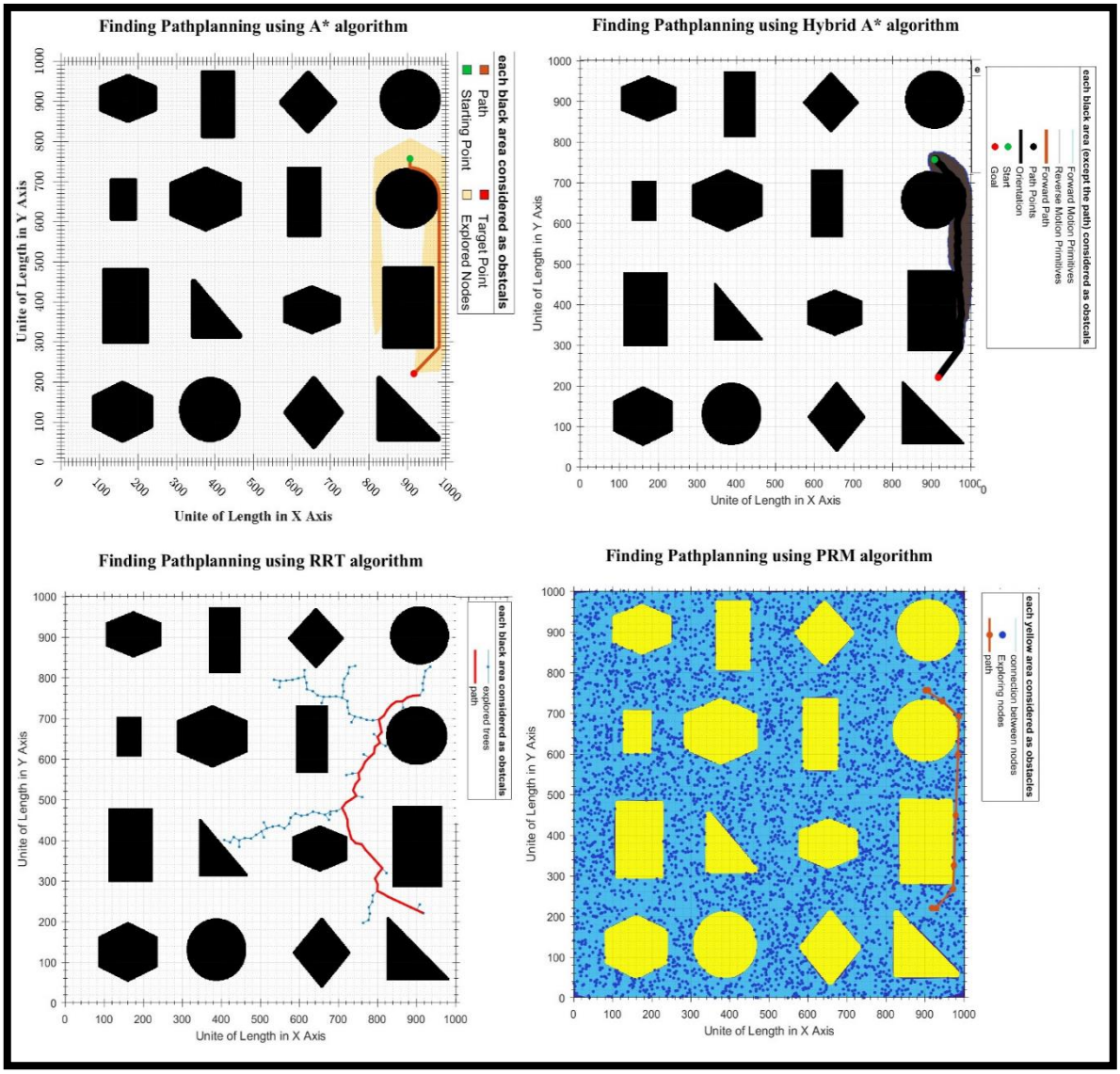


Figure 4.3: Resulted Path of Case Number 2, Using Different Algorithms.

As shown in Figure 4.3, the Hybrid A* algorithm, gives shortest path with minimum sharp turning in the turning points while the analyzing time is the maximum which is the main disadvantage, while the RRT algorithm gives longest path with many sharping turns but the advantage here is the consumed time.

4.1.1.3 Case 3

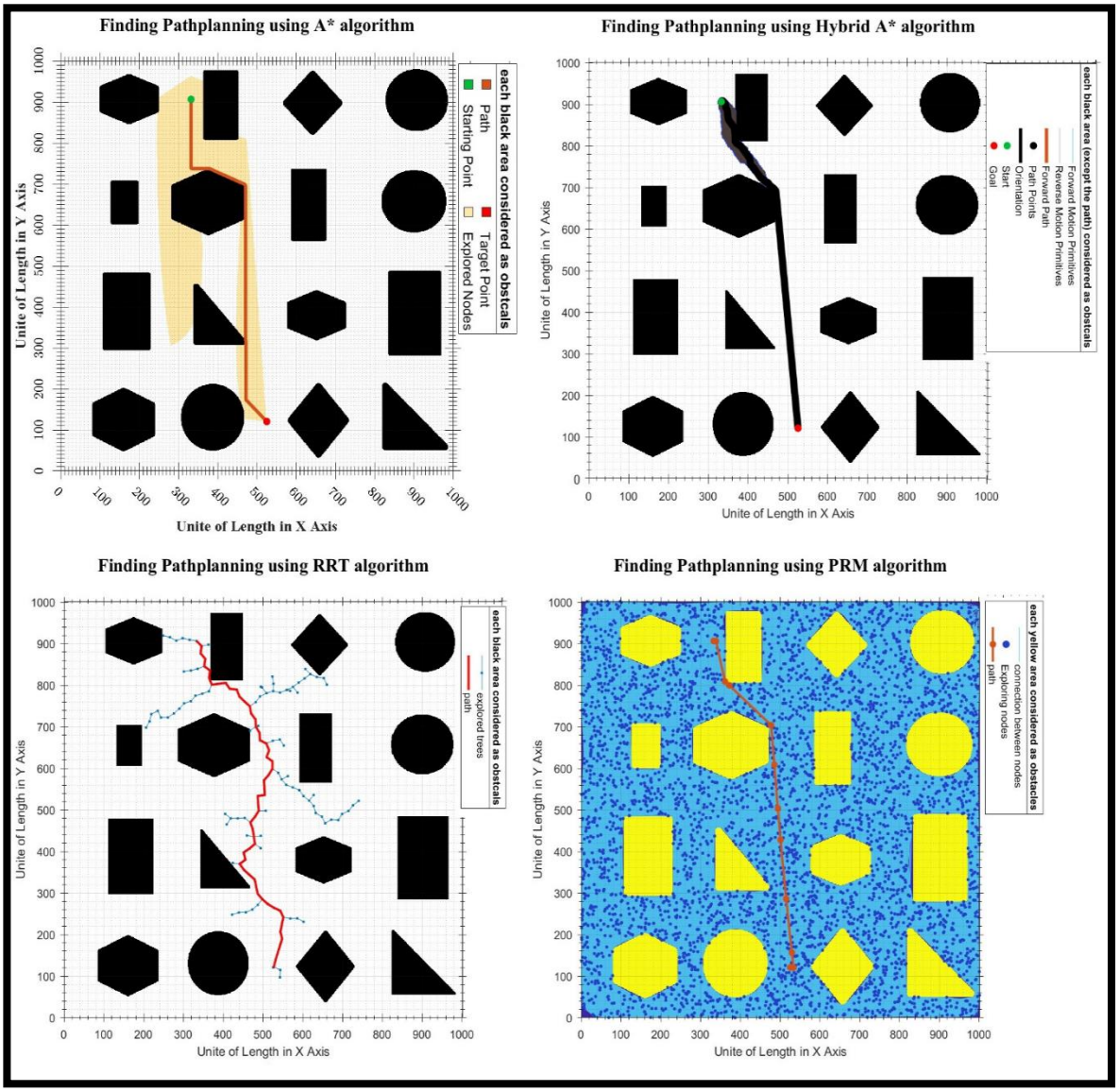


Figure 4.4: Resulted Path of Case Number 3, Using Different Algorithms.

As shown in Figure 4.4, the Hybrid A* algorithm, gives shortest path with minimum sharp turning in the turning points while the analyzing time is not the minimum, while A* algorithm gives a good path in minimum time consuming while RRT gives longest path with many sharping turns.

4.1.1.4 Case 4

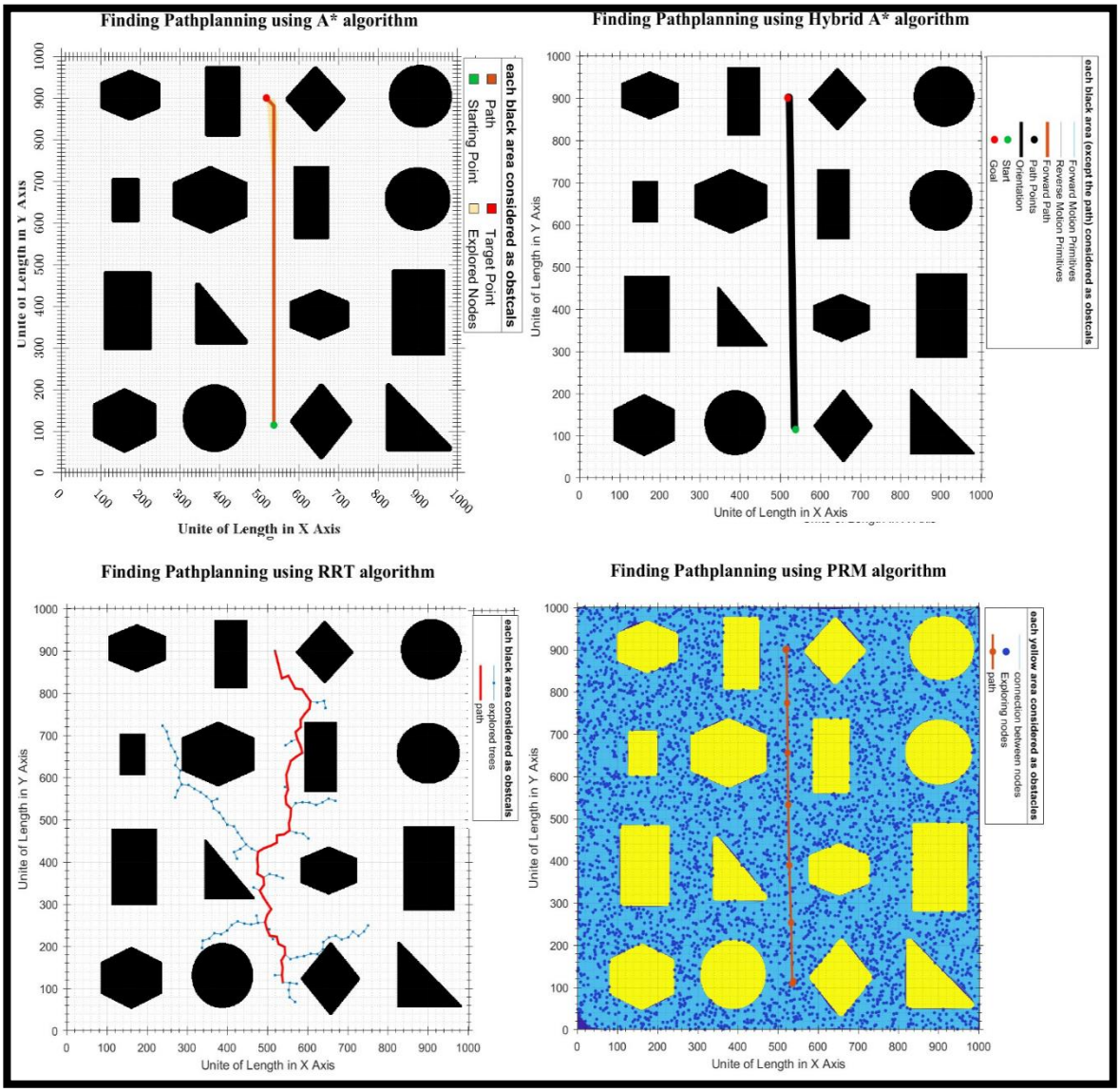


Figure 4.5: Resulted Path of Case Number 4, Using Different Algorithms.

As shown in Figure 4.5, this special case with no obstacles during the Euclidean distance between the starting point and the goal, the Hybrid A* gives the shortest path with minimum consuming time.

4.1.2 Second Category of Path Planning Cases

Table 4.2 shows the result path for the second category for each case, the green highlighted field represents the minimum value while the red highlighted field represents the maximum value.

Table 4.2: Algorithms Comparison for Results of Second Category Cases.

Case #	Factors \ Algorithm	A*	Hybrid A*	RRT	PRM
P5	Shortest path regardless the obstacles (Per unite of length)	411.169	411.169	411.169	411.169
	Path length (Per unite of length)	1262.82	1226.3	1793.27	1292.88
	Total number of explored nodes	277224	1227	192	4000
	Number of path nodes	1135	1227	47	16
	Elapsed time to find the suitable path (Per second)	3.59	1022.98	35.44	87.42
P6	Shortest path regardless the obstacles (Per unite of length)	1009.35	1009.35	1009.35	1009.35
	Path length (Per unite of length)	1442.17	1363.98	1785.91	1418.83
	Total number of explored nodes	465104	1364	195	4000
	Number of path nodes	1205	1364	45	17
	Elapsed time to find the suitable path (Per second)	3.89	24.13	69.78	112.76
P7	Shortest path regardless the obstacles (Per unite of length)	665.487	665.487	665.487	665.487
	Path length (Per unite of length)	1124.93	1060.6	1525.84	1095.76
	Total number of explored nodes	457499	1061	139	4000
	Number of path nodes	988	1061	53	14
	Elapsed time to find the suitable path (Per second)	3.78	70.32	18.95	139.52
P8	Shortest path regardless the obstacles (Per unite of length)	883.681	883.681	883.681	883.681
	Path length (Per unite of length)	1033.34	997.584	1299.76	995.861
	Total number of explored nodes	311855	998	168	4000
	Number of path nodes	908	998	58	13
	Elapsed time to find the suitable path (Per second)	3.46	40.50	15.16	130.28

It's easy to notice that the hybrid A* gives the shortest path length while the PRM gives the minimum number of path nodes while the A* consumes minimum time, and the RRT explored minimum nodes which is weak point for this algorithm.

4.1.2.1 Case 5

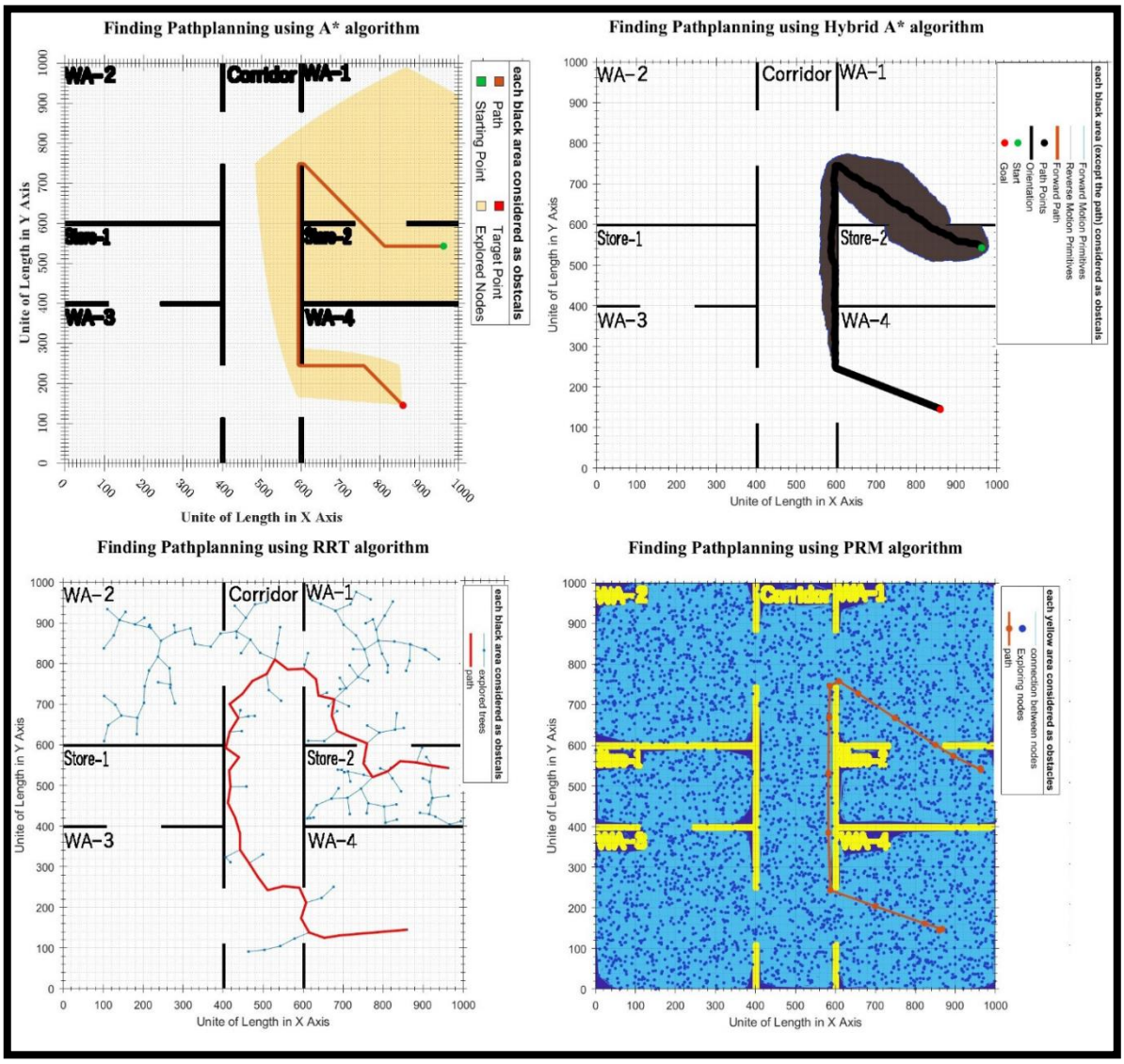


Figure 4.6: Resulted Path of Case Number 5, Using Different Algorithms.

As shown in Figure 4.6, the Hybrid A* algorithm, gives shortest path with minimum sharp turning in the turning points while the analyzing time is not the minimum, while the RRT algorithm gives longest path with many sharping turns but A* gives short path with minimum consumed time.

4.1.2.2 Case 6

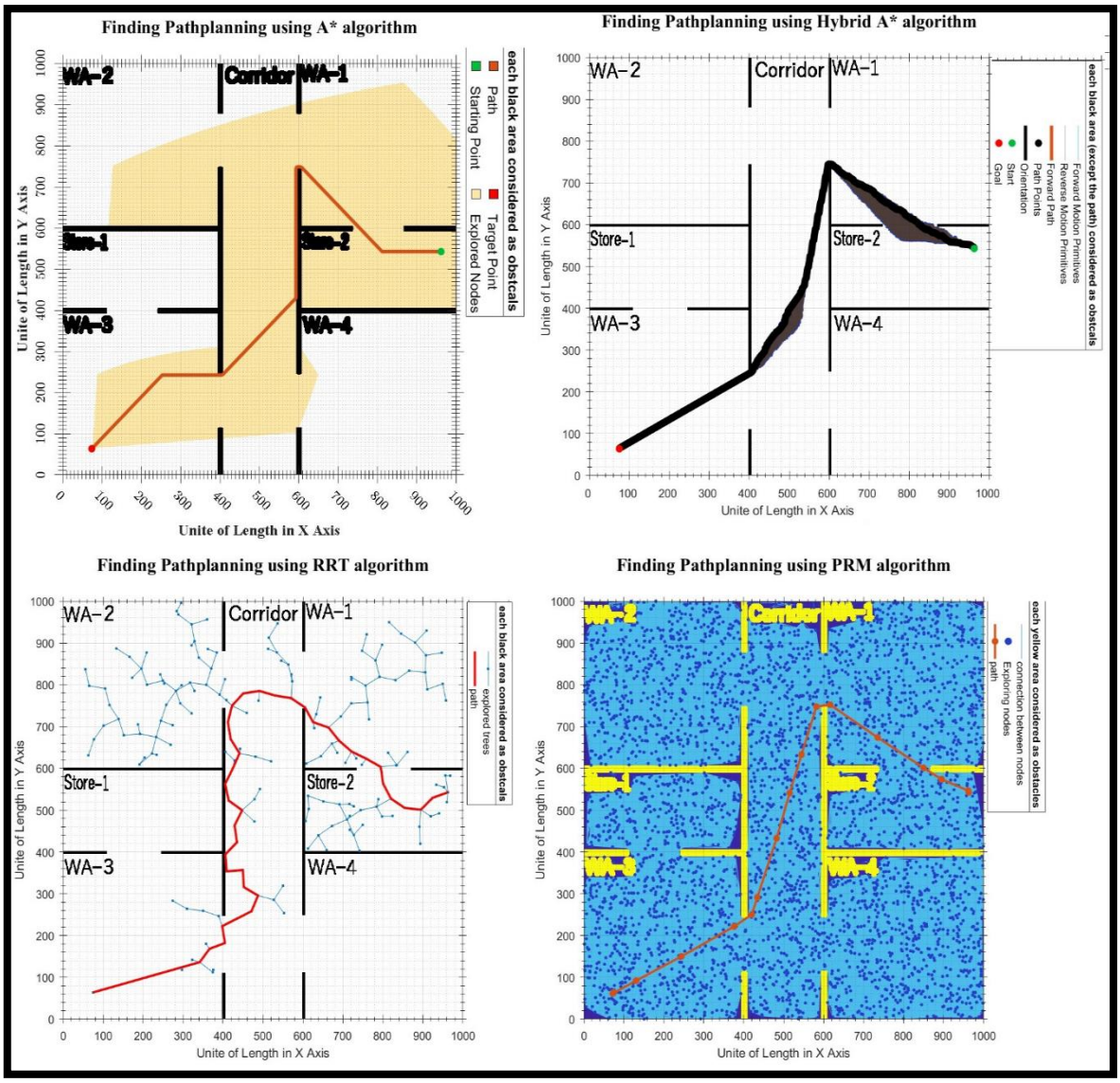


Figure 4.7: Resulted Path of Case Number 6, Using Different Algorithms.

As shown in Figure 4.7, the Hybrid A* algorithm, gives shortest path with minimum sharp turning in the turning points while the analyzing time is not the minimum, but the path contains maximum number of nodes which is a weak point here, while the RRT algorithm gives longest path with many sharping turns but A* gives short path with minimum consumed time.

4.1.2.3 Case 7

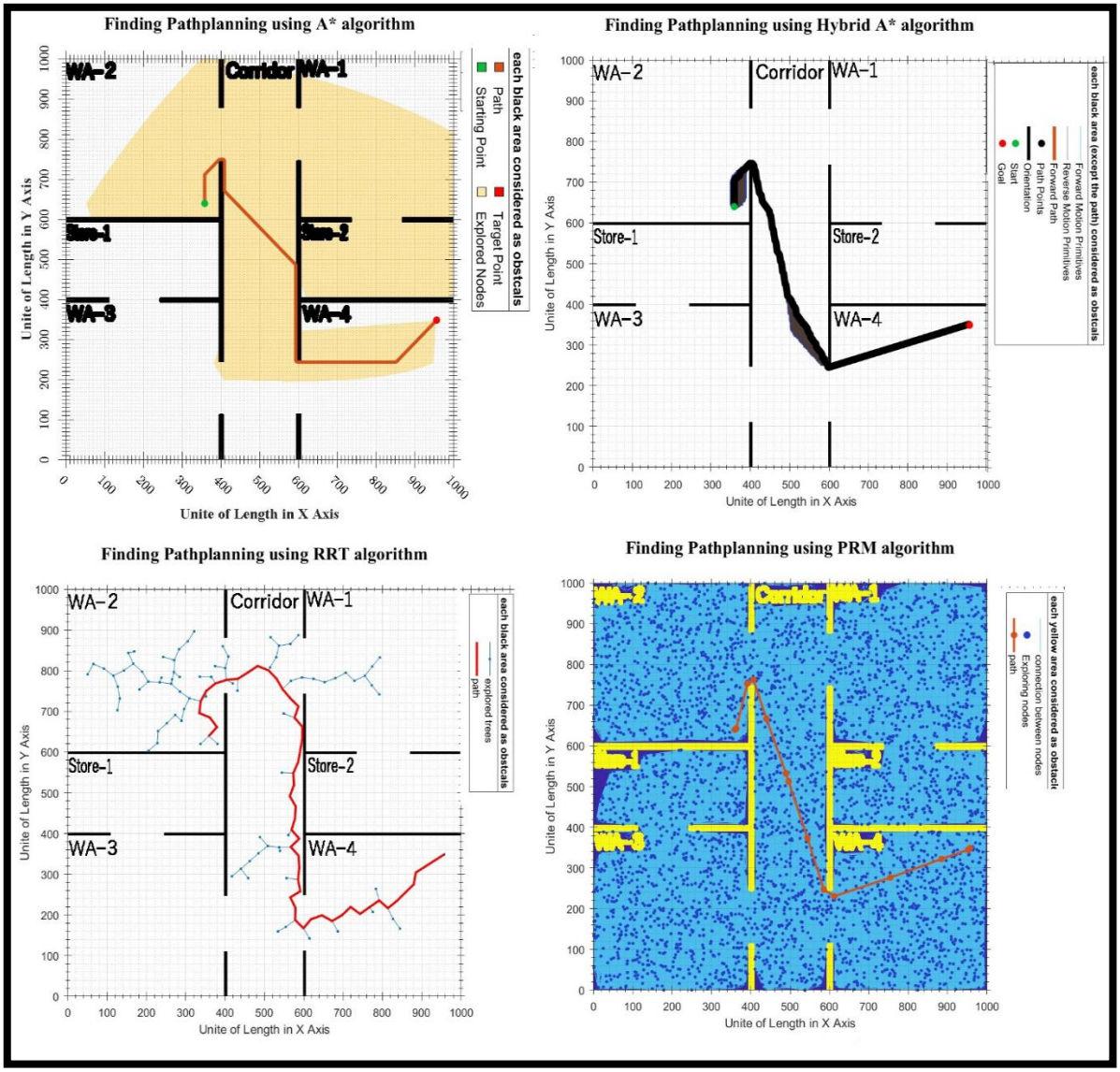


Figure 4.8: Resulted Path of Case Number 7, Using Different Algorithms.

As shown in Figure 4.8, the Hybrid A* algorithm, gives shortest path with minimum sharp turning in the turning points while the analyzing time is not the minimum, but the path contains maximum number of nodes which is a weak point here, while the RRT algorithm gives longest path with many sharping turns but A* gives short path with minimum consumed time.

4.1.2.4 Case 8

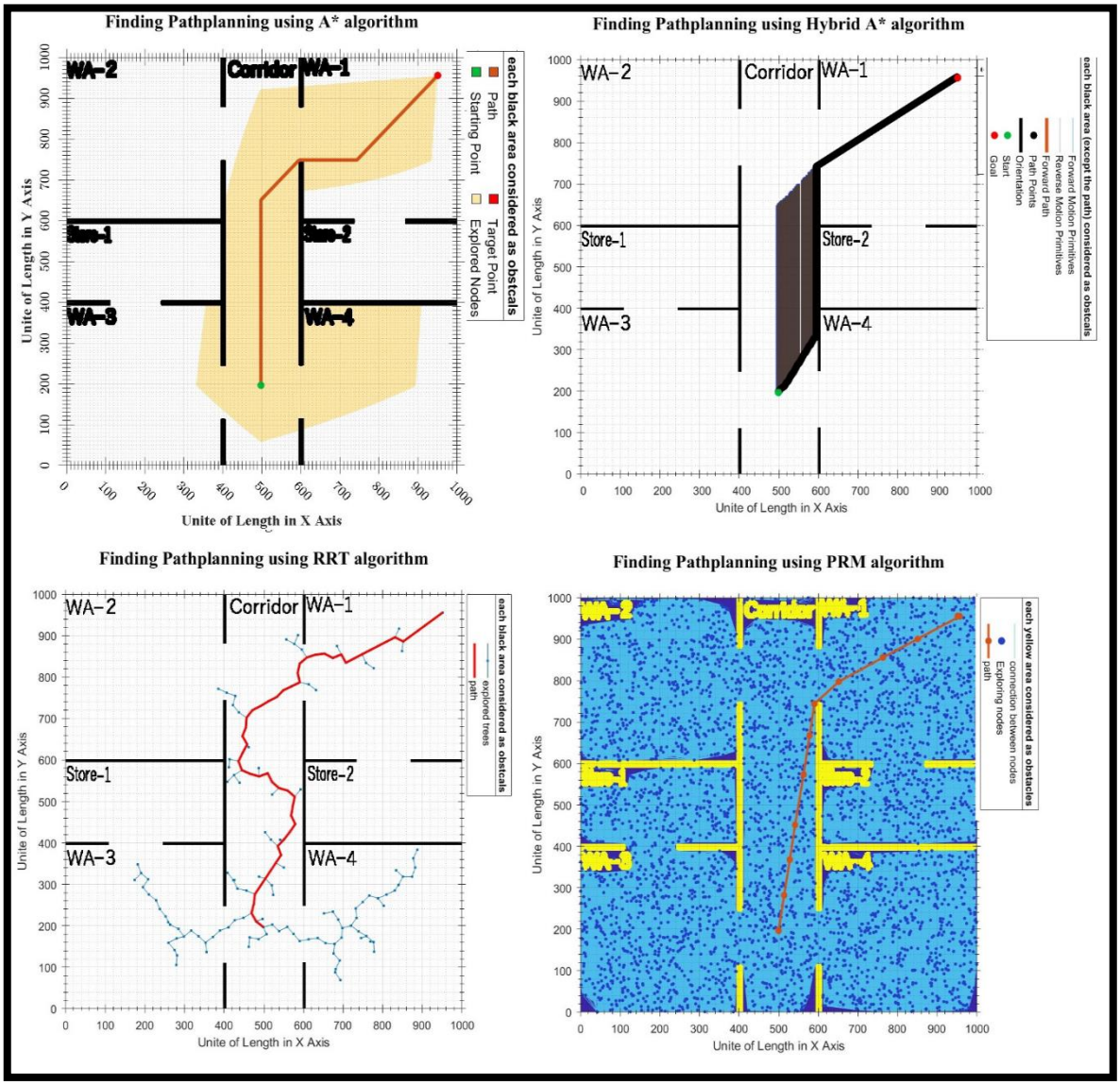


Figure 4.9: Resulted Path of Case Number 8, Using Different Algorithms.

As shown in Figure 4.9, the Hybrid A* algorithm, gives shortest path with minimum sharp turning in the turning points while the analyzing time is not the minimum, but the path contains maximum number of nodes which is a weak point here, while the RRT algorithm gives longest path with many sharping turns but A* gives short path with minimum consumed time.

4.1.3 Effective Modifiable Parameters for Each Algorithm

4.1.3.1 A* Algorithm

a. Cost of moving between two points, the used one in 4.1.1 and 4.1.2 was general cost which explored nodes more than the heuristic cost, which effect negatively for finding the optimal path vice versa in case of decreasing the number of obstacles. Case number 1 and 2 are good example for moving cost method in A* algorithm, in case number 1 there is a single obstacle, so the number of the explored nodes has been decreased to 13286 and the path length has been decreased to 296.818, while in case number 2 the number of obstacles have been increase, the path length increase to 634.884, shown in Figure 4.10.

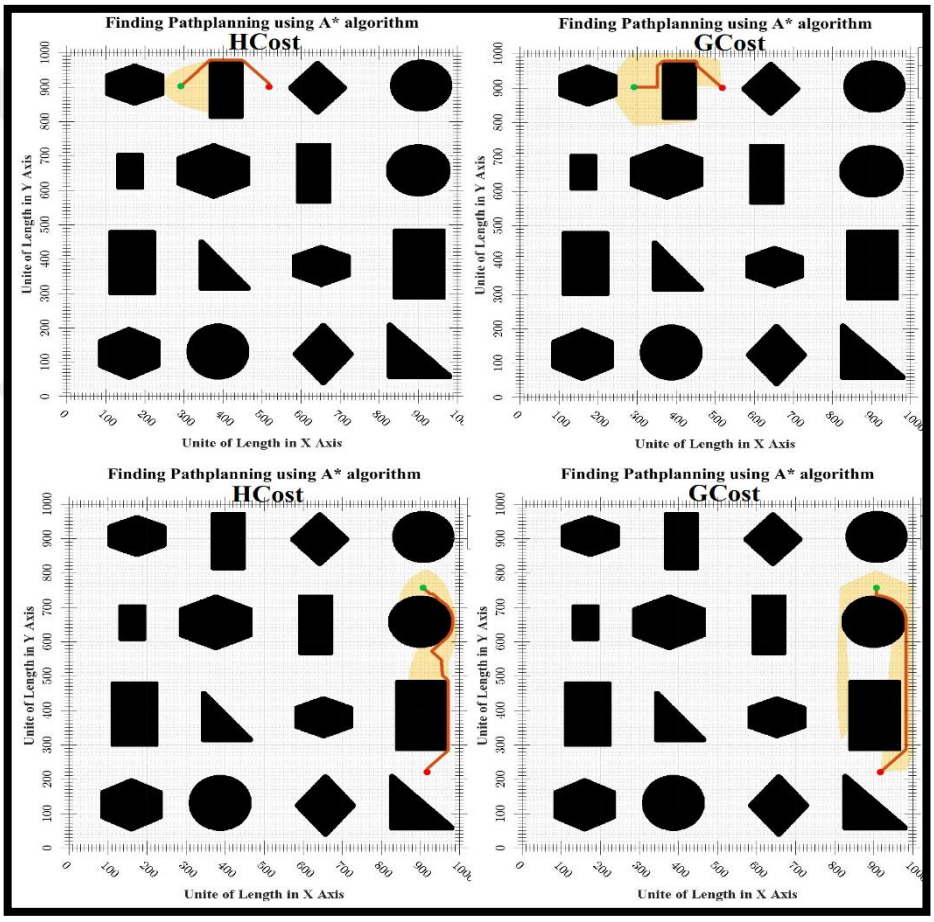


Figure 4.10: Differences Between Using G-Cost and H-Cost.

b. Diagonal search, either to be “on” (as considered in 4.1.1 and 4.1.2) or “off”, when the diagonal search off A* will create a path restrict near the obstacles which make the path longer and less in safe wise for the robot, in case of any error in the motion of the robot as show below for case number 6 as an example as shown in Figure 4.11.

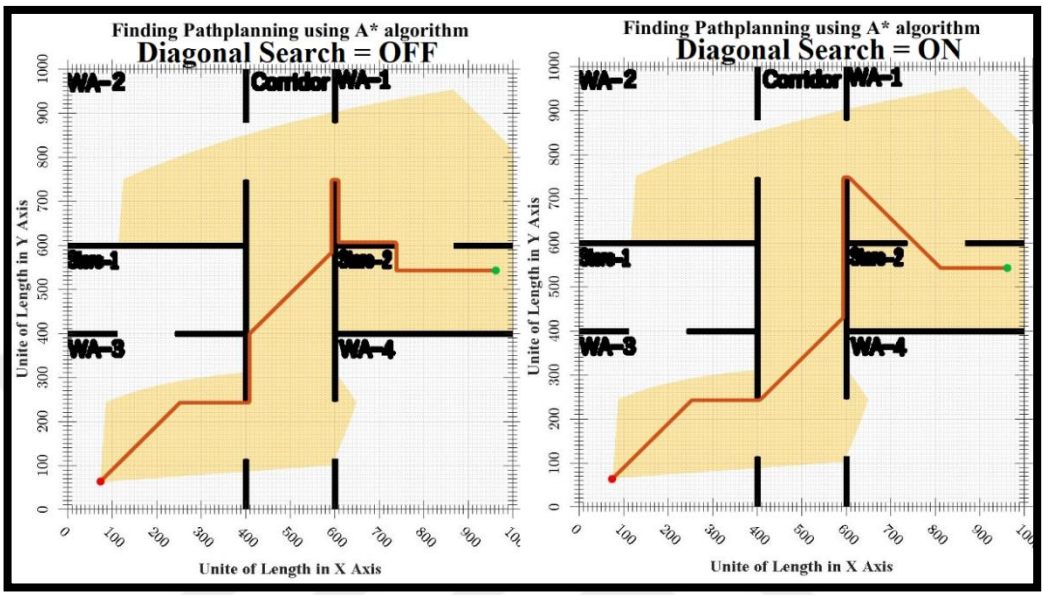


Figure 4.11: Effectiveness of Using Diagonal Search.

c. Tiebreaker either to be “on” (as considered in 4.1.1 and 4.1.2) or “off”, When “TieBreaker” property is enabled, which is a mechanism used to resolve ties that may occur when sorting or ranking data by applying a rule or criterion to determine the order of elements within the sorted list or rank, the A* path planner chooses between multiple paths of the same length by adjusting the heuristic cost value, disabling this property will increase the consuming time with giving same result in case of distances between obstacles are big, for example in case number 6 when disabling the tiebreak property the explored nodes increased to 523704 with keeping the path length same 1442.17 so here the consumed time will be increased. While enabling this property gives shorter path in case of the distances between obstacles are small for example in case number 3 the path length reduced to 865.943. as shown Figure 4.12.

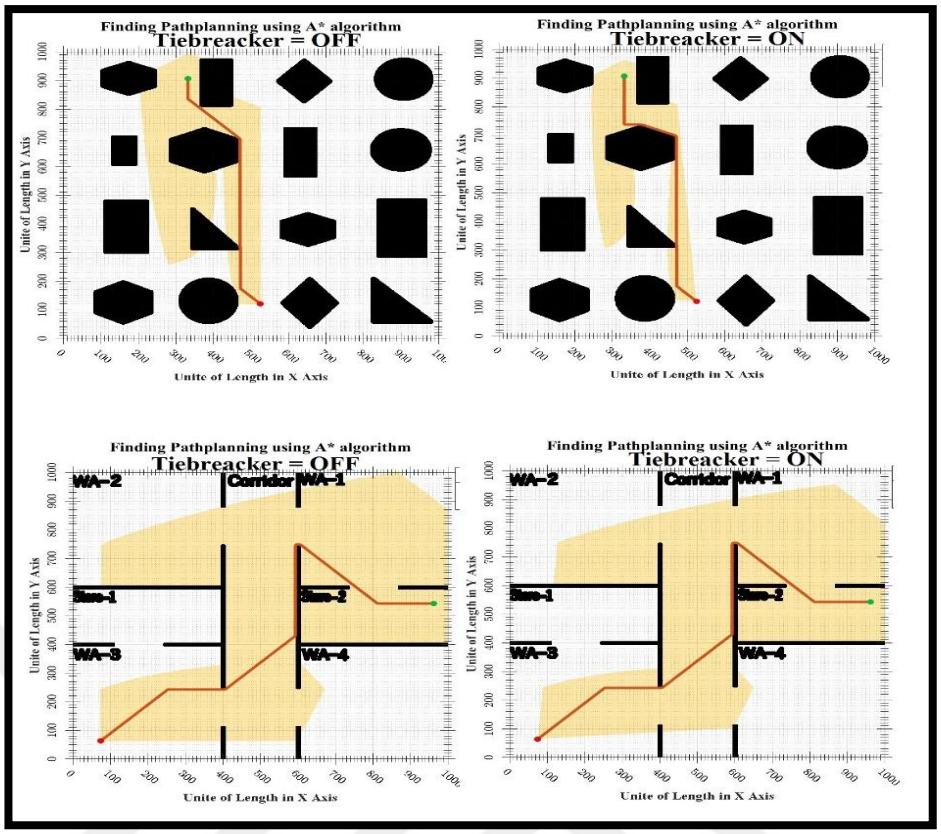


Figure 4.12: Tiebreaker Effect.

d. Method of searching, either to be "Manhattan" (as considered in 4.1.1 and 4.1.2), "Euclidean", "Chebyshev" or "EuclideanSquared", using Euclidean increase the number of turns with keeping the same path length and path nodes in compare with Manhattan in case the distances between the obstacles are big for example in case number 6, while it gives shorter path in case of the distances between the obstacles are small for example case number 3 as the path length reduced to be 867.701. while using Chebyshev instead of Manhattan will increase the path length in case of the distances between the obstacles are big for example case number 6 the path length increases to 1504.3, while it gives shorter path in case of the distances between the obstacles are small for example case number 3 as the path length reduced to be 865.943, as Chebyshev reduce the number of the explored nodes. Otherwise in case of using EuclideanSquared property the path properties will not be affected except the time of analyzing will be increase as shown in Figure 4.13.

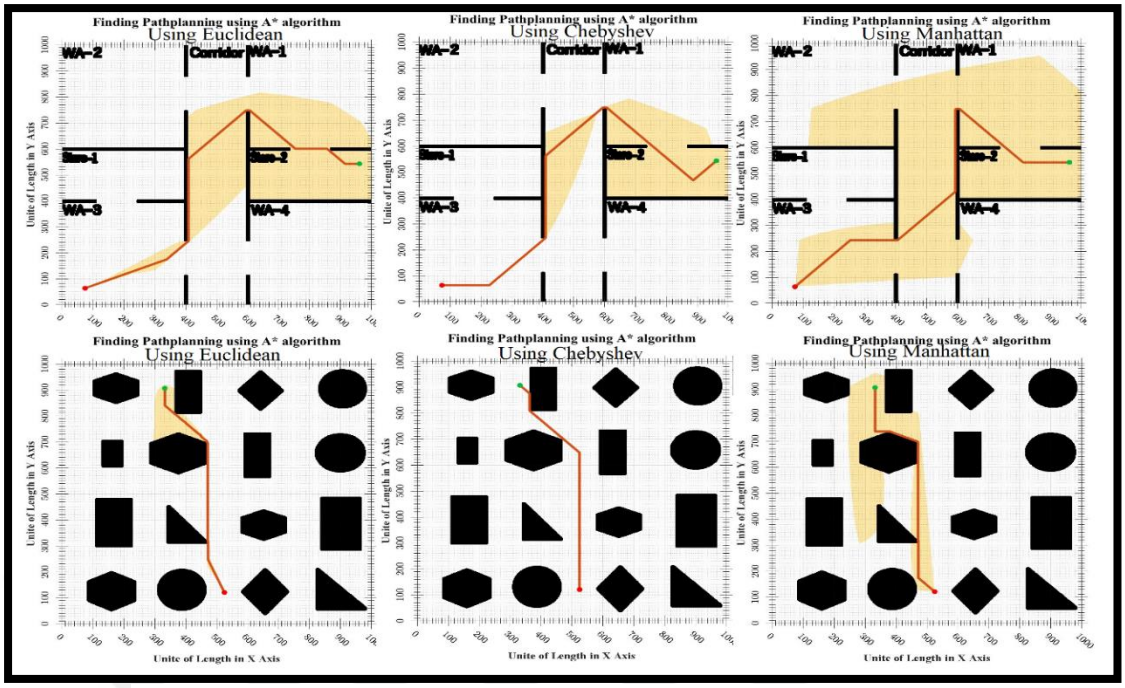


Figure 4.13: Manhattan, Euclidean and Chebyshev Effects.

4.1.3.2 Hybrid A* Algorithm

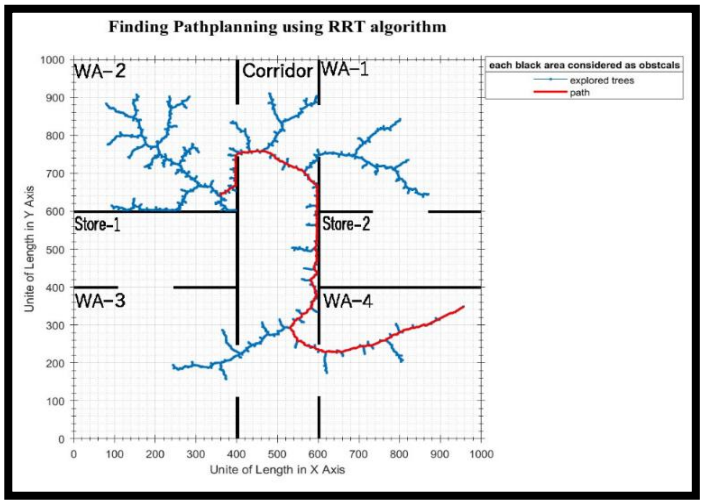
a. The search mode either to be "greedy" (as considered in 4.1.1 and 4.1.2) or "exhaustive" using exhaustive increase the analysing time with very small decreasing the length in compare with greedy for example case number 1 the consumed time increase to 108.58 and for case number 6 the time increase to 106.82 (more than 150%) with (not more than 0.01%) decreasing the length to 291.296 for case number 1 while for case number 6 the path length is increase to be 1367.6.

b. There are other effective properties can be adjusted according to the requirements, like "*MinTurningRadius*" for the minimum turning radius of the path, "*MotionPrimitiveLength*" for length of motion primitives to be generated which cannot be more than one-fourth of the length of the circumference of a circle based on the minimum turning radius, "*NumMotionPrimitives*" for number of motion primitives to be generated, "*ForwardCost*" for considering cost multiplier to travel in forward direction, "*ReverseCost*" for considering cost multiplier to travel in reverse direction, "*DirectionSwitchingCost*" for additive cost for switching direction of motion, "*AnalyticExpansionInterval*" for interval for attempting analytic expansion from the lowest cost node available at that instance, "*InterpolationDistance*" for distance between interpolated poses in output path, adjusting

those parameters costs too much time and requires a highly efficient computer to do the required calculations; this issue was one of the limitations of this study.

4.1.3.3 RRT Algorithm

"MaxConnectionDistance", increasing the value of connection distance between two nodes would increase the path length and reduced the explored nodes and the path nodes and reducing the number of iterations for getting the path, while the analysing time will be increased.
"MaxNumTreeNodes" increasing the value of number of tree nodes would increase the number of iterations and the analysing time. "MaxIterations", increasing the value of number of iterations would increase the chance of finding path. "GoalBias" this property controls the probability of the planner biasing its exploration towards the goal region, as opposed to randomly exploring the entire space. Increasing the value of "GoalBias" increases the likelihood that the planner will explore areas closer to the goal region, and hence can result in faster convergence towards the goal. However, a higher "GoalBias" value can also result in the planner getting stuck in local minima or sub-optimal paths, as it may not explore other regions of the space that could potentially lead to better paths.
In this study especially 4.1.1 and 4.1.2, the minimum number of each above factors are considered and let MATLAB increase the value of MaxNumTreeNodes by 1 and MaxConnectionDistance by 0.1 for each iteration till get the path with minimum values with considering GoalBias as 0.1. For optimal results the number of trees and the number of iterations should be as high as possible and value of connection distance between two nodes as less as possible with putting time consuming in considerations. For example, case number 7, in case of increasing the maximum number of trees to 5018 and the maximum number of iterations to 5018 and the maximum connection distance 2.8, the path length will be reduced to 1356.56 and explored nodes will be increased to 2227 and the path will contain 486 nodes and the analysing time will be increased to 29.06, as shown in Figure 4.14



Connection Distance.

Figure 4.15 shows the effect of changing the value of GoalBias how make the path biasing toward the goal when increasing the value from 0.01 with considering MaxNumTreeNodes = 167 and MaxConnectionDistance =18.5 and MaxIterations = 167 as sown in Figure 4.15 (a) and changing the GoalBias to 0.1 with considering MaxNumTreeNodes = 121 and MaxConnectionDistance =13 and MaxIterations = 121 as sown in Figure 4.15 (b) and changing the GoalBias to 0.7 with considering MaxNumTreeNodes = 123 and MaxConnectionDistance =13.2 and MaxIterations = 123 as sown in Figure 4.15 (c)

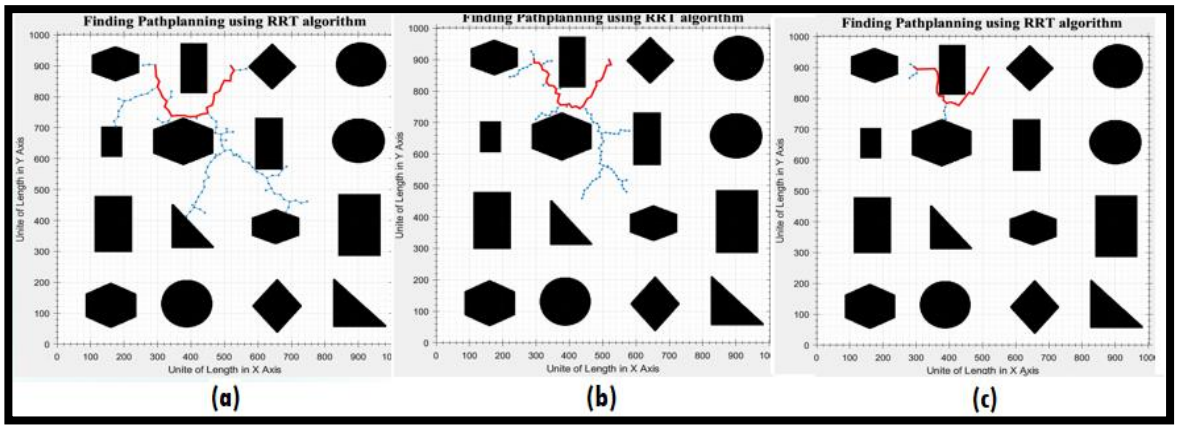


Figure 4.14: GoalBias Effects for RRT.

4.1.3.4 PRM Algorithm

The worst point in this algorithm can be noticed from its name, as it depends on probability that means there is no guarantee to get the same path in each try, but generally the "NumNodes" and "ConnectionDistance" property can be effective on the resulted path, as decreasing the number of nodes that PRM explored will less the chance of finding a path, same for connection distance. In this study especially 4.1.1 and 4.1.2, the number of nodes considered as 4000 with connection distance 150 to assure finding a path, another method for checking this algorithm by considering the minimum number of nodes is considered and let MATLAB increase the value in case of no path is found and increasing the connection distance will increase the probability of finding a path and decreasing the number of path nodes. As show below for case number 3 as an example, with 200 explored nodes and 150 connection distance the path nodes became 11 while with 4000 explored nodes with 50 connection distance the path nodes became 26, as shown in Figure 4.16.

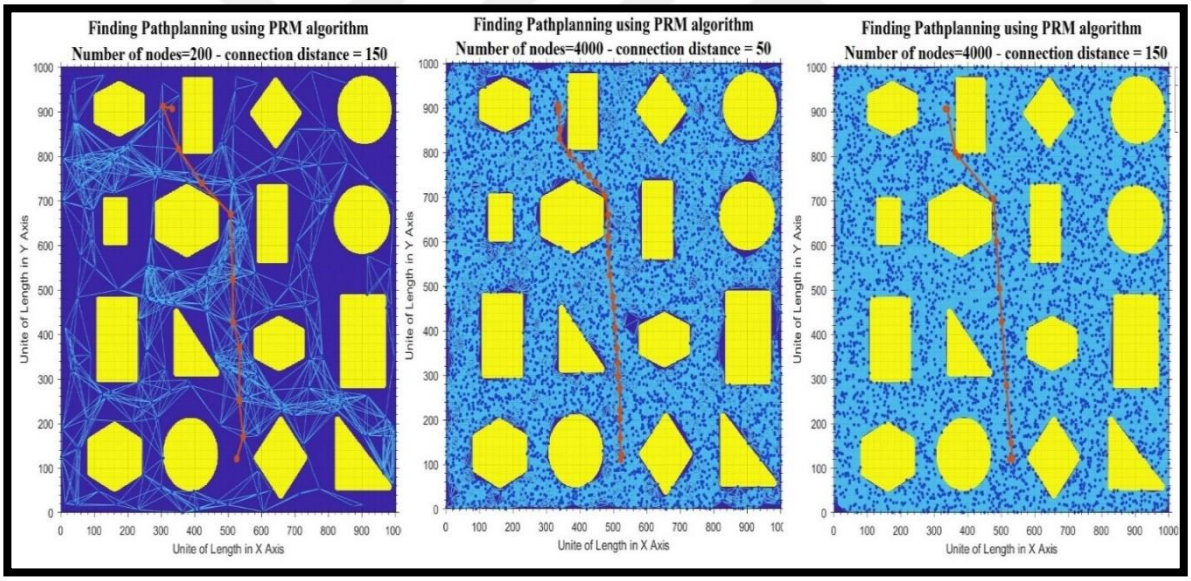


Figure 4.16: Effects of Number of Explored Nodes and Connection Distance.

4.2 PICK AND PLACE CASES

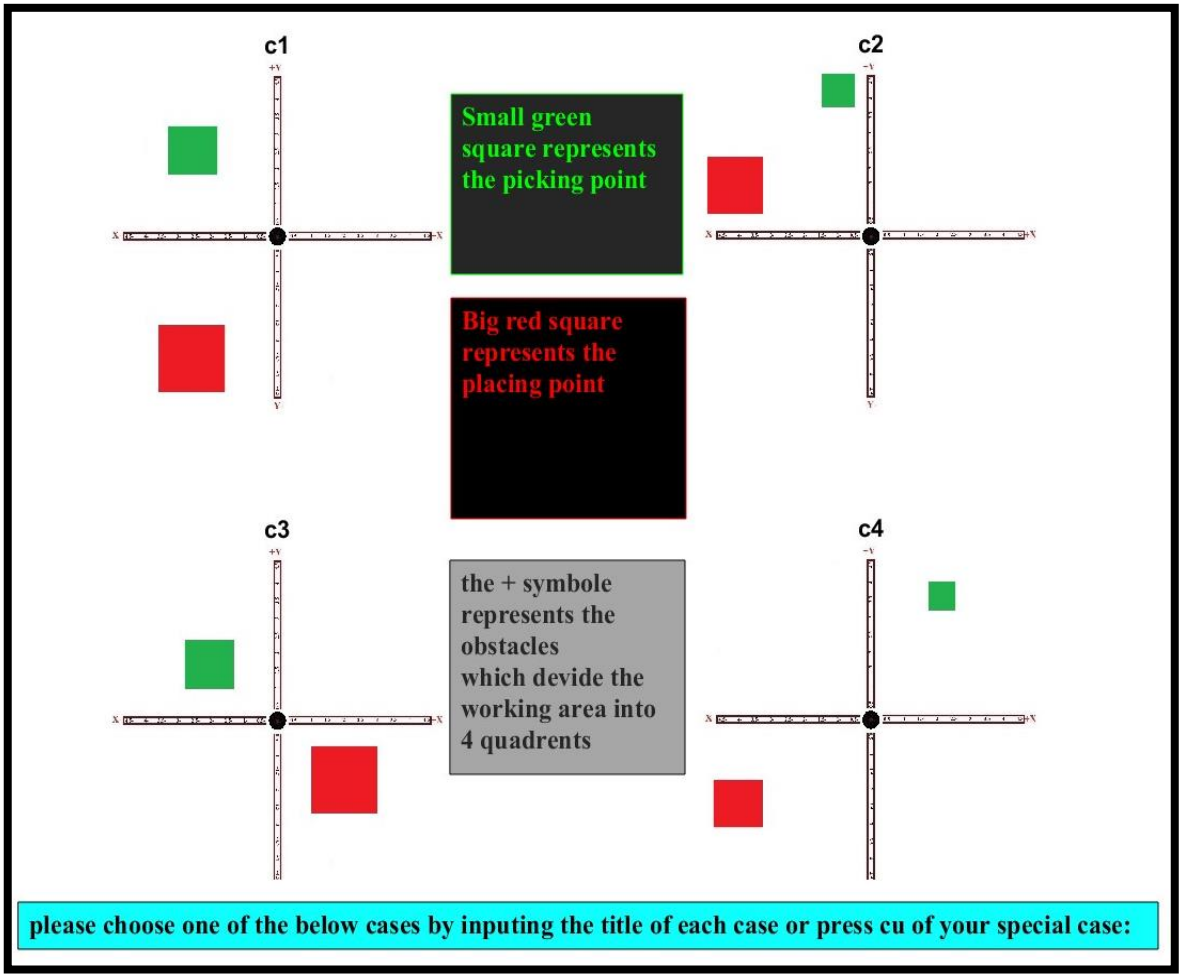


Figure 4.17: Pick and Place Cases.

Figure 4.17 and 4.18 showed the simulated environment that used in this study using MATLAB code, to let the user chose the case and dimensions depending on studied proposals or to amend it manually and the code would act accordingly.

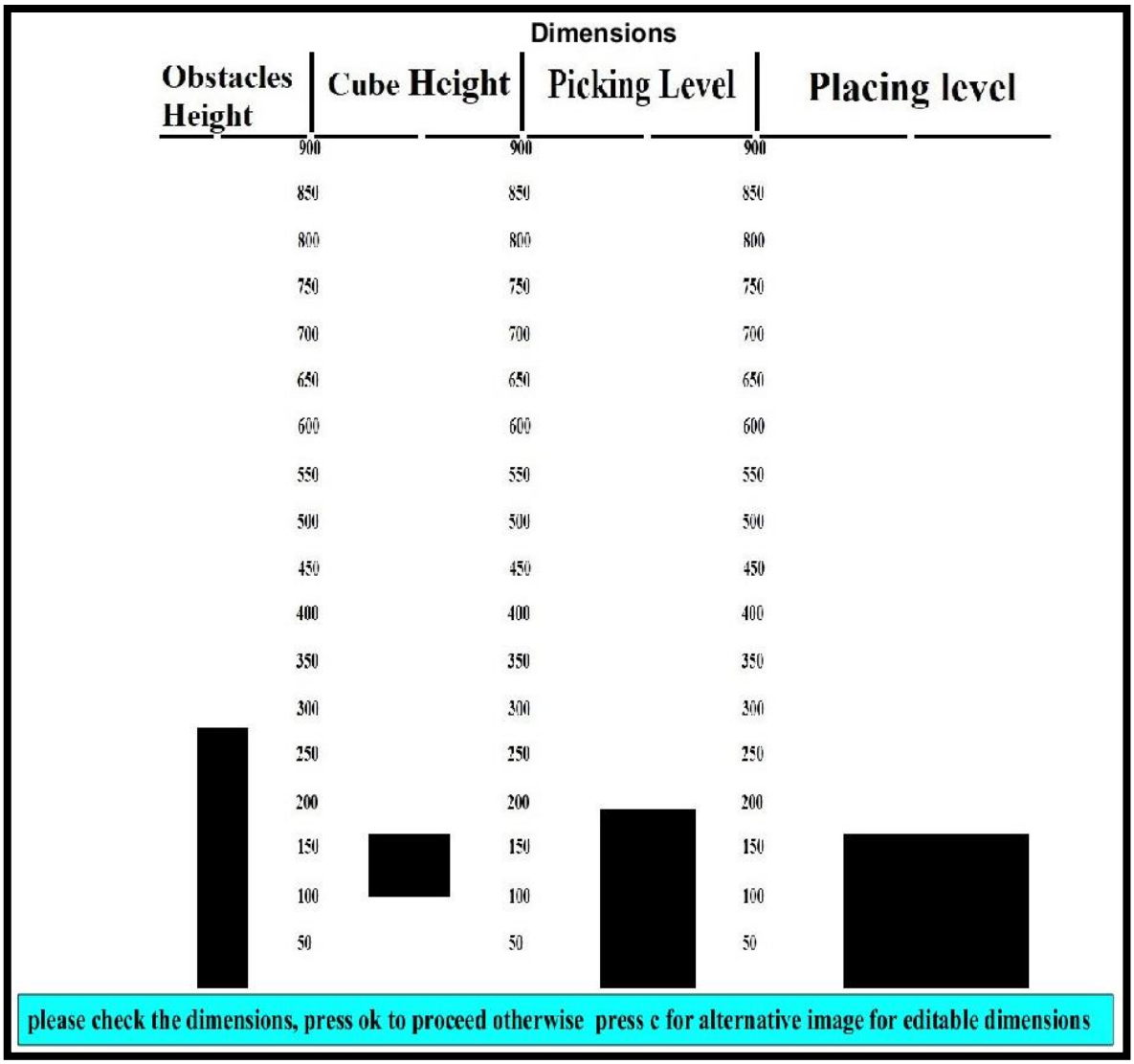


Figure 4.18: Objects and Obstacles Level and Heights.

4.2.1 Case 1

The resulted path of both robots for case number 1 are shown in Figure 4.19, it's easily to be noticed that the conditional code achieved the obstacles avoidance considering the picked object height and the obstacles heights with safety distance, as the red path is regardless the obstacle while the bule one is the resulted path (based on conditional code) that the robot should follow and finally the green path is the real path that the robot follow. The camera simulated the locations and level correctly based on the proposed calibration method by considering the real distance with the digital distance in the processed image.

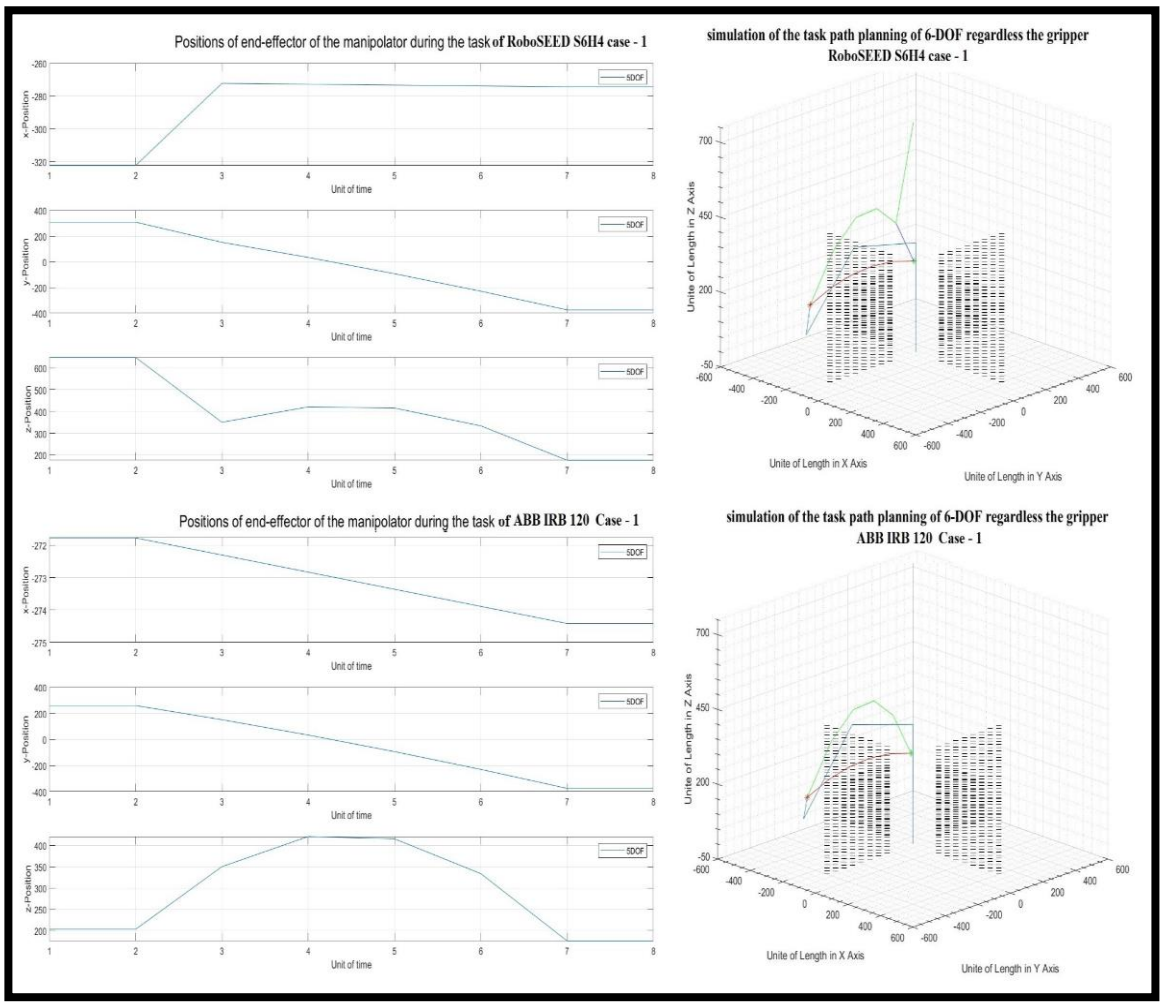


Figure 4.19: Simulations and End-Effector Positions for Case – 1.

In case number 1 RoboSEED S6H4 could not follow the required path in the first picking steps within the second quadrant, because of the limitations of link lengths and joint motion ranges while ABB IRB 120 followed the path with minimum errors. As shown in Figure 4.20 the error of RoboSEED S6H4 in first step and second step is more than 50, 40, 400 units of length in X, Y and Z axis respectively, after that the errors decreased near to zero while ABB IRB 120 achieved the required path with tiny errors not more than $2*10^{-4}$ units of length.

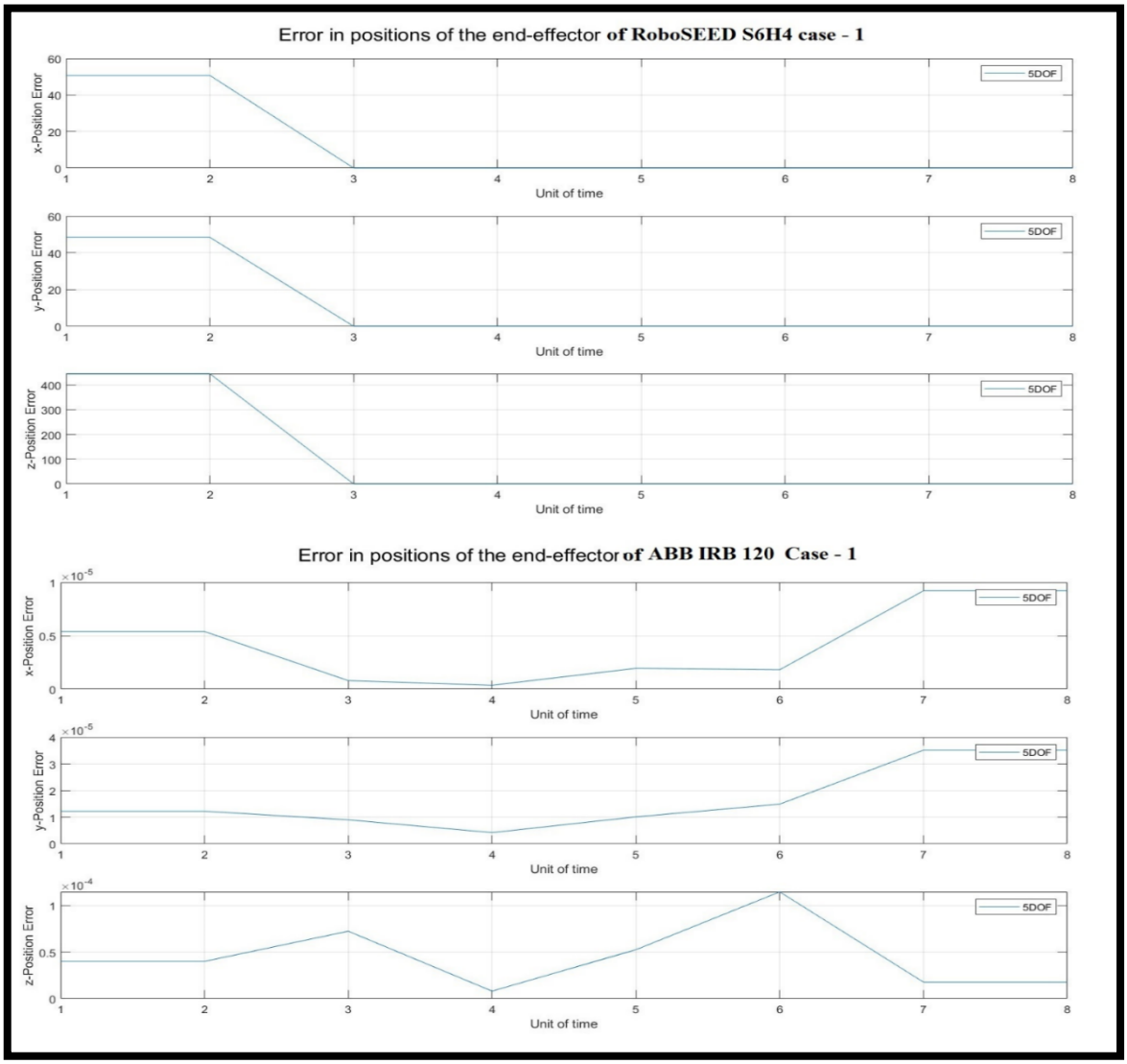


Figure 4.20: Error in Positions of End-Effector for Case – 1.

4.2.2 Case 2

The resulted path of both robots for case number 2 are shown in Figure 4.21, it's easily to be noticed that the conditional code achieved pick and place task considering the picked object height and the obstacles heights with safety distance, all paths in this case will be almost overlayed as there is no obstacles between the cube and box (during the pick-place task) so red path which is regardless the obstacle , the bule one which is the resulted path (based on conditional code) that the robot should follow and the green path is the real path that the robot follow, all of those paths will be almost so in this case all paths will be . The camera simulated the locations and level correctly based on the proposed calibration method by considering the real distance with the digital distance in the processed image.

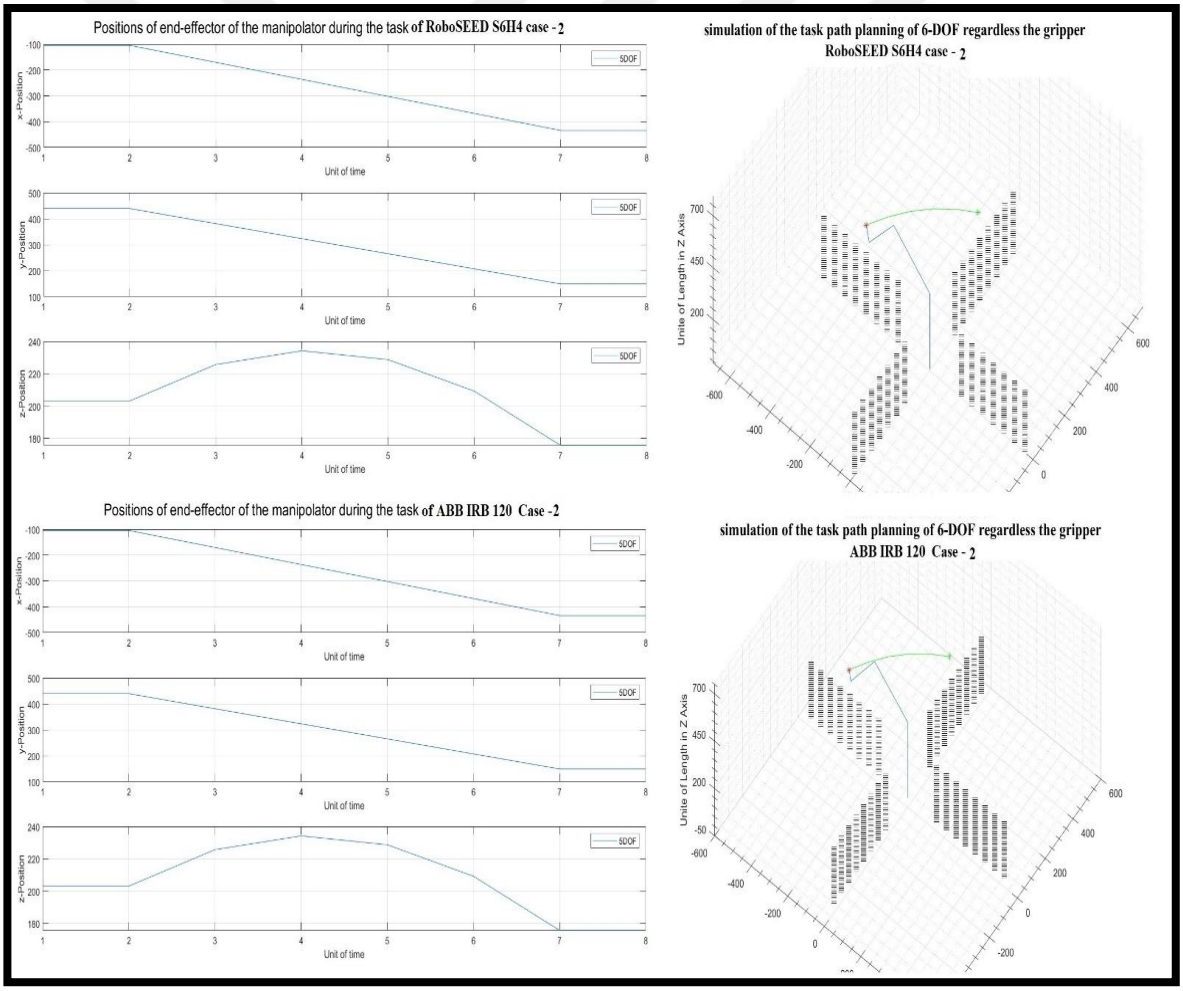


Figure 4.21: Simulations and End-Effector Positions for Case – 2.

In this case the errors for both robots not more than $2*10^{-4}$ units of length as shown in Figure 4.22.

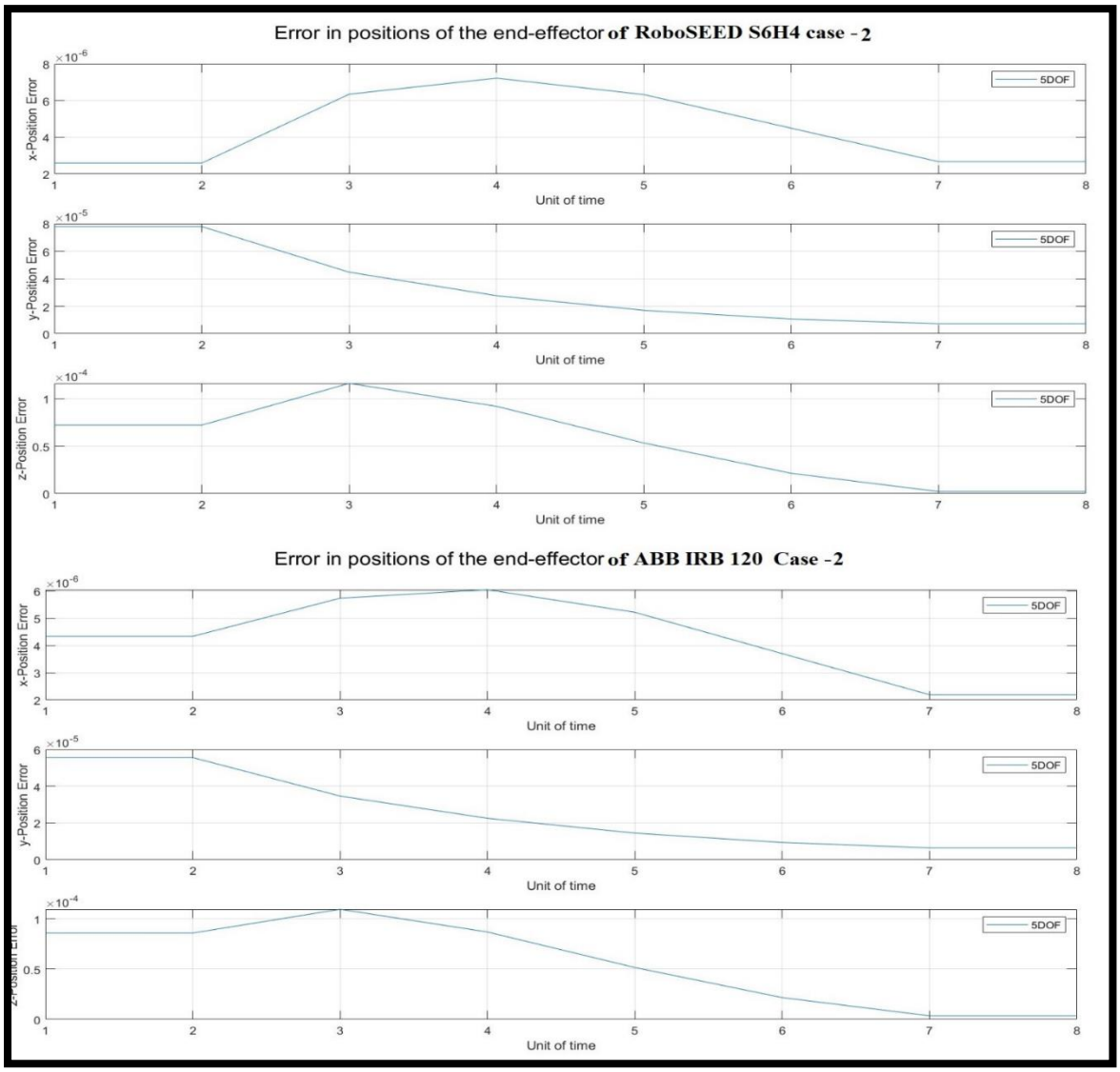


Figure 4.22: Error in Positions of End-Effector for Case – 2.

4.2.3 Case 3

The resulted path of both robots for case number 3 are shown in Figure 4.23, it's easily to be noticed that the conditional code achieved the obstacles avoidance considering the robot base for safety distance, as the red path is regardless the obstacle while the bule one is the resulted path (based on conditional code) that the robot should follow and finally the green path is the real path that the robot follow. The camera simulated the locations and level correctly based on the proposed calibration method by considering the real distance with the digital distance in the processed image.

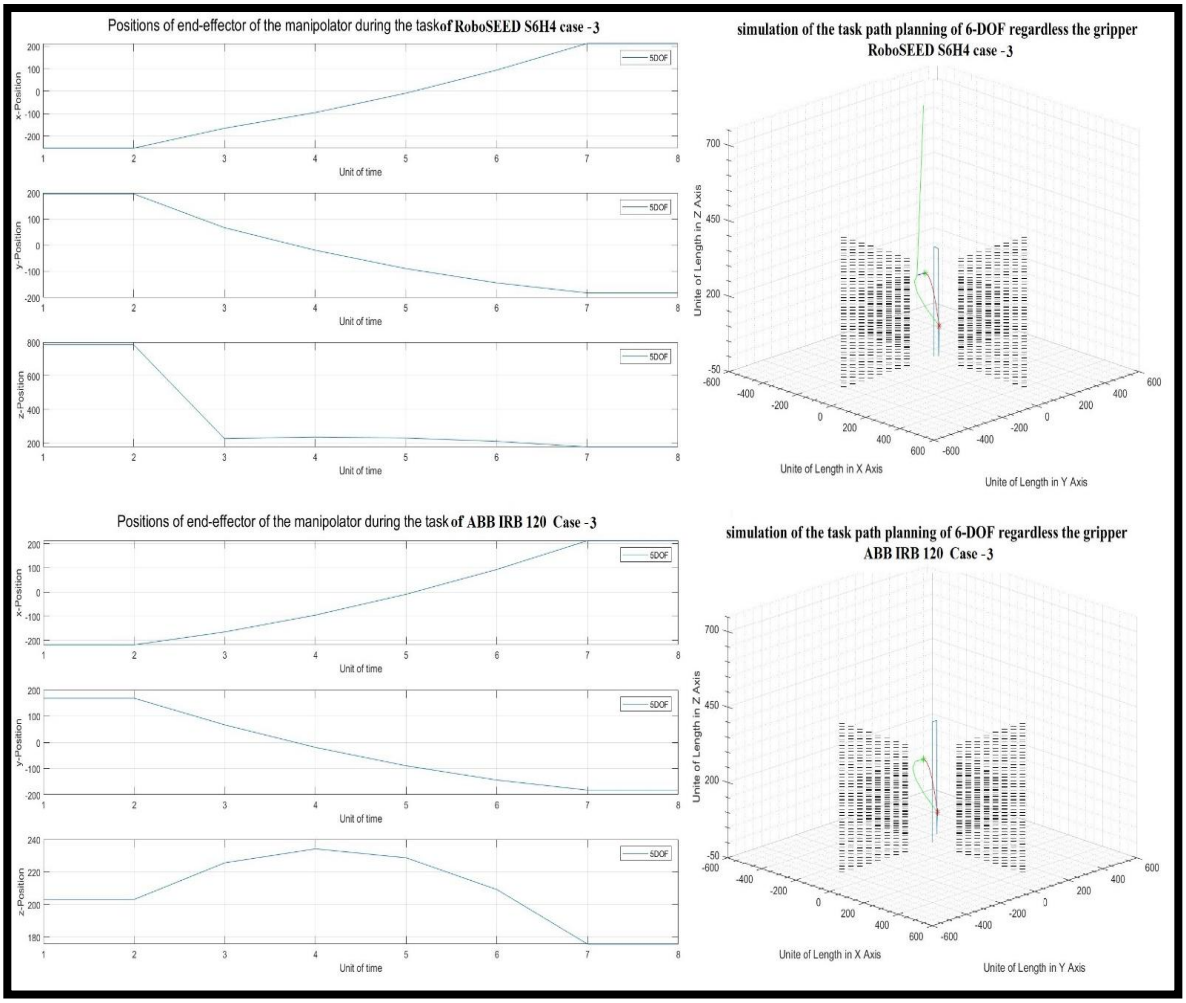


Figure 4.23: Simulations and End-Effector Positions for Case – 3.

In case number 1 RoboSEED S6H4 could not follow the required path in the first picking steps within the second quadrant, because of the limitations of link lengths and joint motion ranges while ABB IRB 120 followed the path with minimum errors. As shown in Figure 4.24 the error of RoboSEED S6H4 in first step and second step is more than 30, 25, 500 units of length in X, Y and Z axis respectively, after that the errors decreased near to zero while ABB IRB 120 achieved the required path with tiny errors not more than $2*10^{-4}$ units of length.

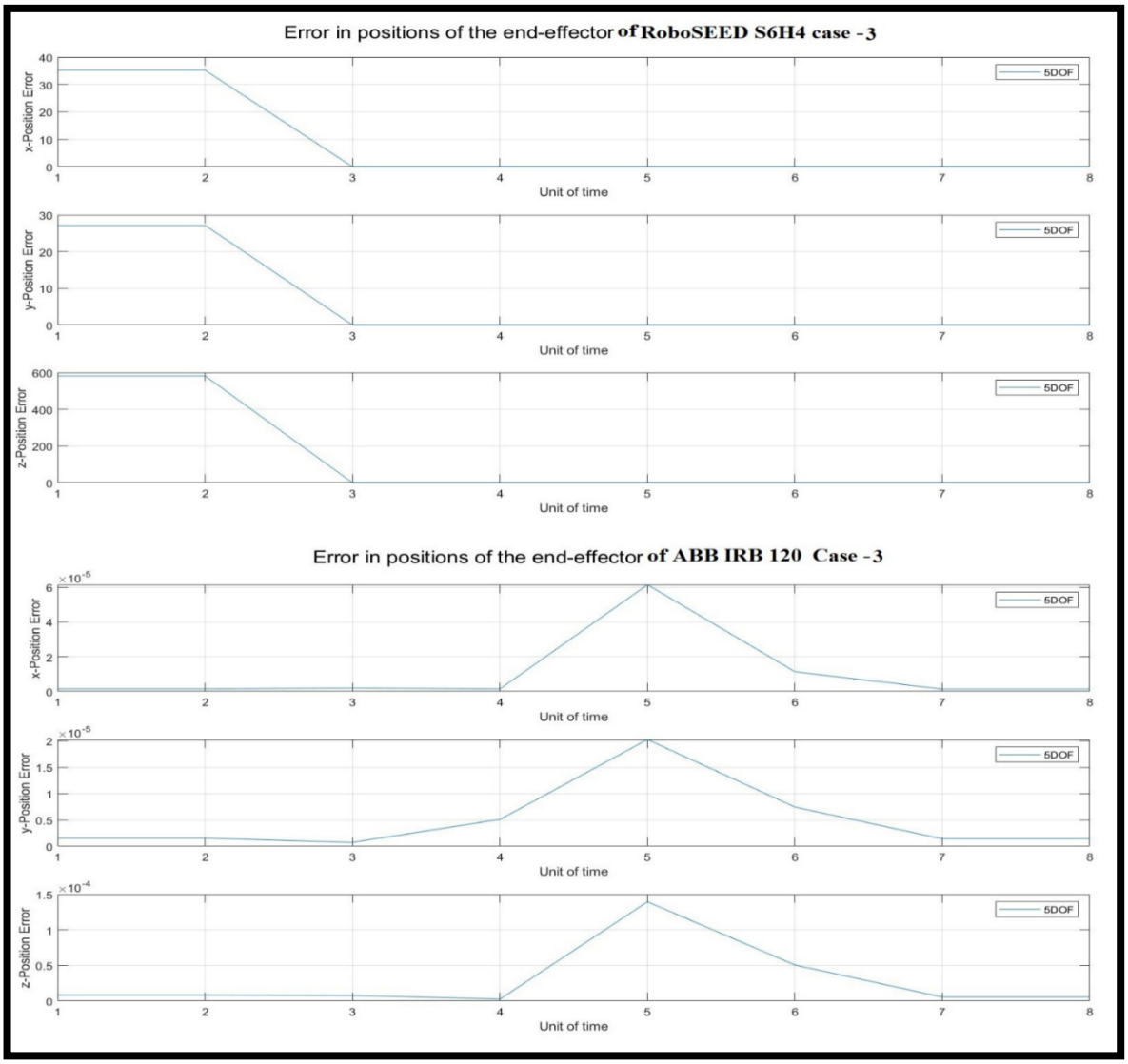


Figure 4.24: Error in Positions of End-Effector for Case – 3.

4.2.4 Case 4

The resulted path of both robots for case number 4 are shown in Figure 4.25, it's easily to be noticed that the conditional code achieved the obstacles avoidance considering the picked object height and the obstacles heights with safety distance, as the red path is regardless the obstacle while the bule one is the resulted path (based on conditional code) that the robot should follow and finally the green path is the real path that the robot follow. The camera simulated the locations and level correctly based on the proposed calibration method by considering the real distance with the digital distance in the processed image.

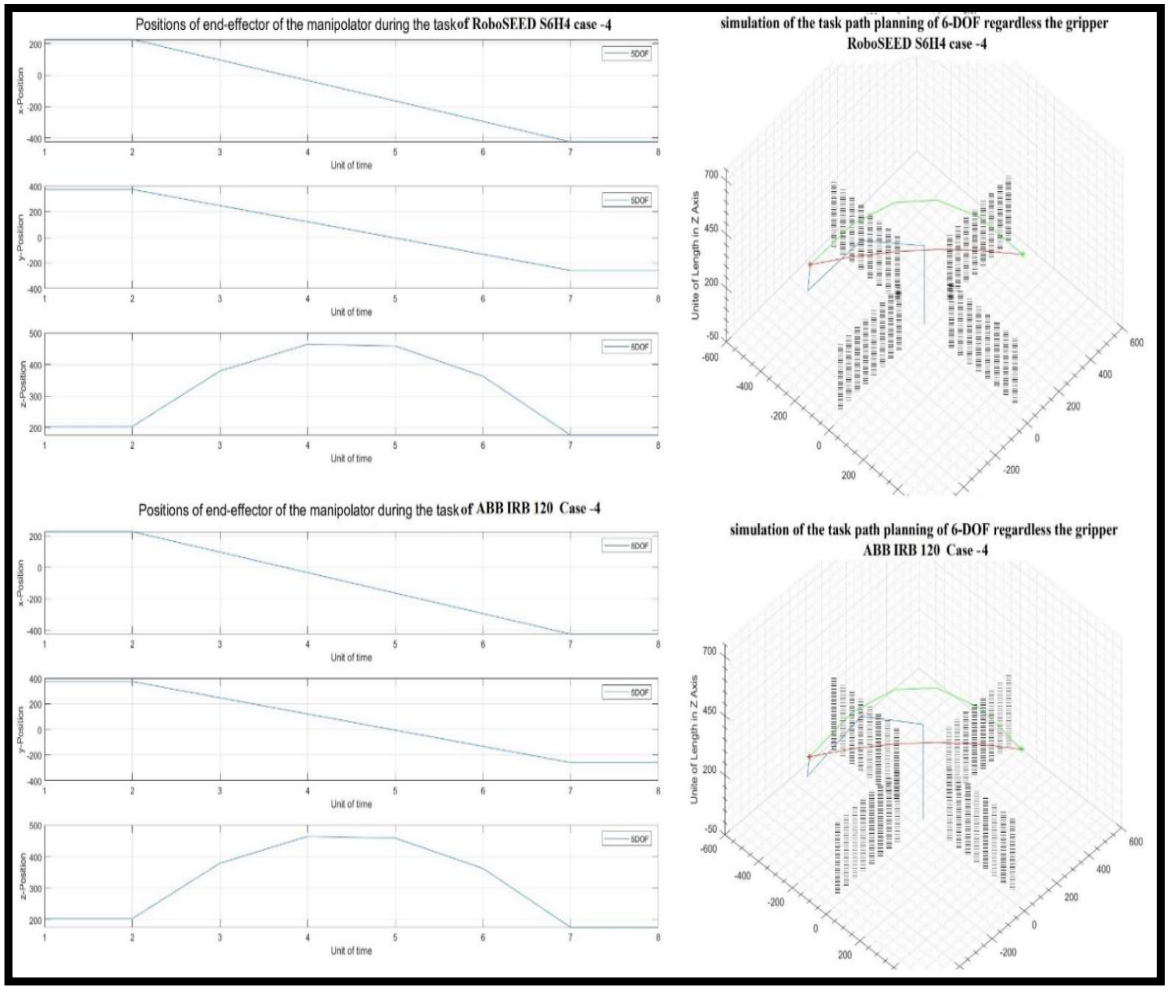


Figure 4.25: Simulations and End-Effector Positions for Case – 4.

In this case the errors for both robots not more than $2*10^{-4}$ units of length as shown in Figure 4.26

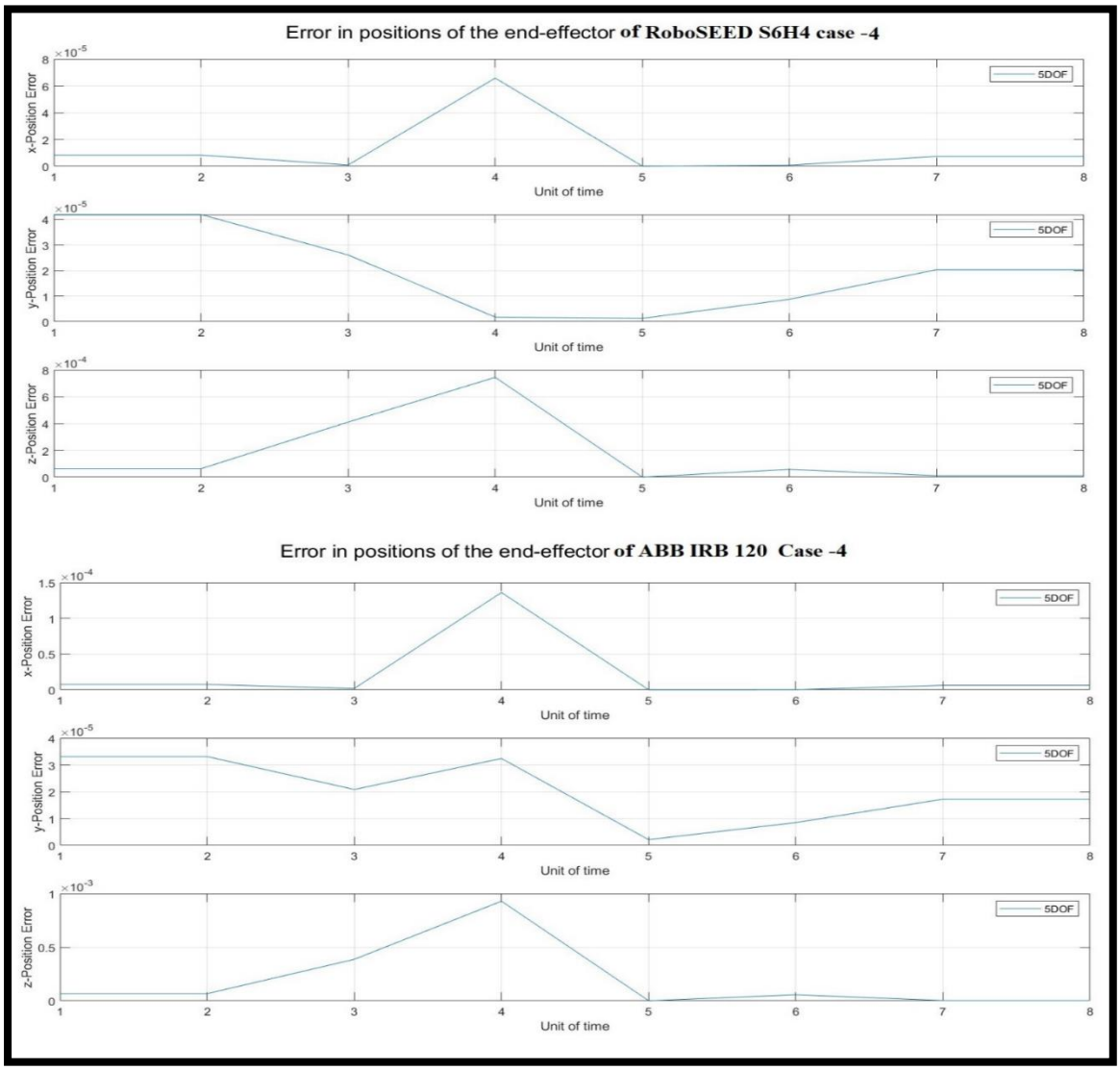


Figure 4.26: Error in Positions of End-Effector for Case – 4.

4.2.5 Special Case

The results of some cases show that the conditional command algorithm could fail to avoid the obstacles. Figure 4.27 (a) shows that the produced path is very near the robot base. And Figure 4.27 (b) shows that the produced path hits the obstacle; such cases are considered failures for the proposed conditional command algorithm, which was prepared to avoid the obstacles and the near area to the robot base, using conditional if statements.

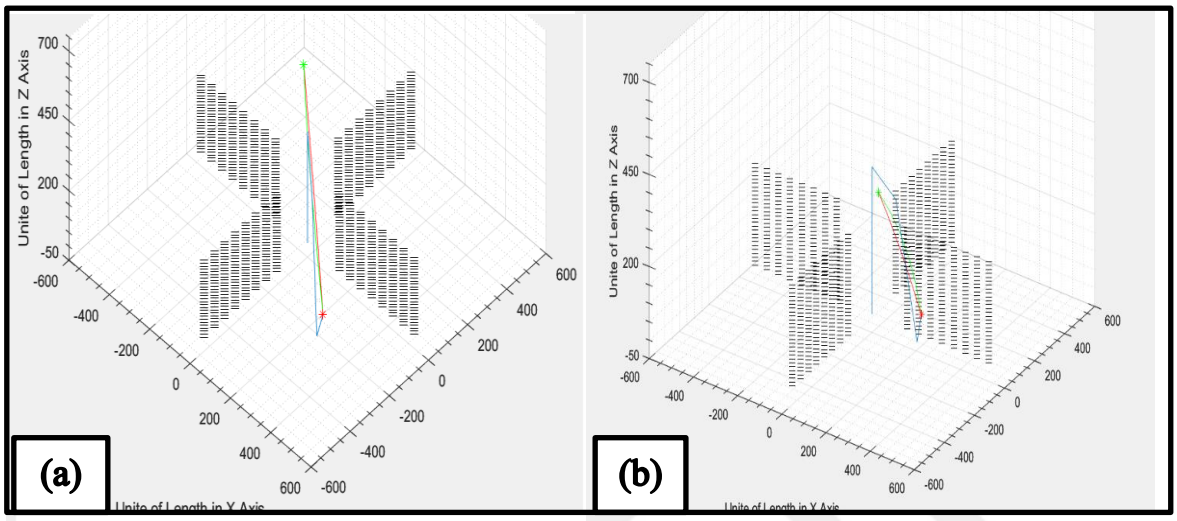


Figure 4.27: Failed Cases of Conditional Command Algorithm.

5 CONCLUSION AND RECOMMENDATIONS

The aim of thesis is to study and compare the path planning algorithms based on total number of explored nodes, real path length, number of path nodes and the consuming time beside simulating the pick and place robotic arms, MATLAB is used for the study using Lenovo ThinkPad X1 Yoga 3rd laptop with processor Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz and 16 GB of memory.

5.1 CONCLUSION

a. A* algorithm is considered very fast compared to the others with a high amount of explored area. It gives an acceptable path length, but on the negative side of A*, the path has many nodes that need many orders to the robot for each node to follow the path, so many intervals decreasing the functionality of each moving order in addition to the discrete path that this algorithm gives sharp turns path, especially with a large number of obstacles which is not acceptable for the high velocity as that increases the possibilities of errors. This algorithm can be adjusted based on the studied properties by using general cost with Euclidean distance equation with enabling tiebreaker and diagonal search let the algorithm gives the shortest path, but the negative side with this algorithm is that the number of nodes of the path cannot be controlled.

b. Hybrid A* algorithm gives a short path length with a continuous flow with fewer sharp turns. Still, the negative side of this algorithm is that it consumes too much time in case of near obstacles to the starting point, and generally, the time here is proportional to the number and sizes of obstacles; moreover, the path here has many points which is considered as a weak point in paths. This algorithm also can be adjusted based on the studied properties either using the greedy or extensive mode, which slightly shortens the path but highly increases the computational time. In other words, Hybrid A* gives the shortest path in comparison with the studied algorithms, but the computational time is very high, almost 1000% in comparison with A*; moreover, this rate the computational time increases as the size and number of obstacles increases and distance between the obstacles and starting and target point decreases, through this study Hybrid A* considered gives shortest path with minimum computational time in case of the obstacles are small and far from the starting and target point.

c. RRT algorithm offers effective properties that can be adjusted to decrease the path nodes, which is inversely proportional to the path length. Moreover, decreasing the consuming time decreases the goal bias but with an inverse proportion of the path length. The main weakness of this algorithm is that it gives unnecessary turns through the path that may negatively affect the path length. In other words, the path length decreases as the maximum number of tree nodes and the maximum connection distance decreases, and the goal bias increases but, in this case, the computational time will highly increase. This algorithm is considered perfect for at least finding a path regardless of the length in a short time when increasing the maximum number of tree nodes and the connection distance between nodes and decreasing the goal bias.

d. PRM algorithm does not give a unified path for each iteration; therefore, it is suitable for dynamic environments, generally the length of the path is inversely proportional to the number of explored nodes, the number of path nodes in this algorithm could be controlled by setting the connection distance, but it will negatively affect to the computational time.

e. Using conditional code algorithm can be useful for simple cases and achieves the requirements in case of static axial and regular obstacles, but it will be not very useful in case of irregular obstacles shapes as the conditional commands will be too much and very hard to cover all cases, in the same time the conditional commands can be easier to amend according to the requirements as the conditions are listed in the code.

f. Robotic arms suitability, the RoboSEED S6H4 has a leak of achieving the required path in the second quadrants while it performs the others, while ABB IRB 120: this robot is more suitable for the proposed cases as the errors are less than RoboSEED S6H4.

5.2 RECOMMENDATIONS

More future studies are needed to cover the following fields:

a. Suggesting easy ways to modify the other algorithms that are not implemented in this study.
b. Appling the methods of this study to a real robot to compare the actual results with the simulations.
c. Expanding the study to cover the path planning algorithms for 3D planes.
d. Going further for studying Hybrid A* algorithm to cover the other effective properties that can be adjusted like the minimum turning radius of the path, the length of motion primitives,

number of motion primitives, switching direction of motion, and distance between interpolated poses in output path.

e. Suggesting modifications to reduce the computational time that Hybrid A* algorithm consumes in complex cases.

6 REFERENCES

[1] Ilesanmi Daniyan, Khumbulani Mpofu, Boitumelo Ramatsetse, Adefemi Adeodu, "Design and simulation of a robotic arm for manufacturing operations in the railcar industry," in *Procedia Manufacturing, 30th International Conference ent Manufacturing (FAIM2020)* , Athens, Greece, 15-18 June 2020.

[2] Dionisis Andronas, Sotiris Xythalis, Panagiotis Karagiannis, George Michalos, Sotiris Makris, "Robot gripper with high speed, in-hand object manipulation capabilities," *Procedia CIRP*, p. 482 – 486, 2020.

[3] Shah R., Pandey A. B., "Concept for Automated Sorting Robotic Arm," in *Procedia Manufacturing\ 2nd International Conference on Materials Manufacturing and Design Engineering*, Gujarat, India, 2018.

[4] Panagiota Tsarouchi, Sotiris Makris, George Michalos, Michael Stefos, Konstantinos Fourtakas, Konstantinos Kaltsoukalas, Dimitris Kontrovrakis, George Chryssolouris, "Robotized assembly process using Dual arm robot," in *Procedia CIRP 23 - Conference on Assembly Technologies and Systems*, patras, greece, 2014.

[5] Niki Kousi , George Michalos , Sotirios Aivaliotis , Sotiris Makris, "An outlook on future assembly systems introducing robotic mobile dual arm workers," in *Procedia CIRP 72 - 51st CIRP Conference on Manufacturing Systems* , Patras, Greece, 2018.

[6] Samad Hayati, Kam Tso and Thomas Lee, "Dual Arm Coordination and Control," *Robotics and Autonomous Systems*, vol. 5, pp. 333-344, 1989.

[7] Jianwei Zhang, Yorck von Collani, Alois Knoll, "Interactive assembly by a two-arm robot agent," *Robotics and Autonomous Systems*, vol. 29, pp. 91-100, 1999.

[8] P. Kopacek, "INTELLIGENT ASSEMBLY AND DISASSEMBLY PAST, PRESENT AND FUTURE," Intelligent Handling and Robotics (IHRT), Vienna University of Technology, Favoritenstraße 9–11, A–1040 Vienna, Austria, 2007.

[9] R. Nithish Kumar, R. Mohan, N. Gobinath, "Improvement in production line efficiency of hemming unit using line balancing techniques," *Materials Today,* vol. Volume 46, no. Part 2, pp. 1459-1463, 2021.

[10] Rui Song, Fengming Li, Tianyu Fu and Jie Zhao, "A Robotic Automatic Assembly System Based on Vision," *Applied Sciences,* vol. 10, no. 3, 8 February 2020.

[11] W. Chen, "Intelligent manufacturing production line data monitoring system for industrial internet of things," *Computer Communications,* vol. Volume 151, p. 31–41, 1 February 2020.

[12] Wadea Ameen, Mohammed AlKahtani, Muneer Khan Mohammed, Osama Abdulhameed, Abdulaziz Mohammed El-Tamimi, "Investigation of the effect of buffer storage capacity and repair rate on production line efficiency," *Journal of King Saud University – Engineering Sciences,* vol. Volume 30, no. Issue 3, p. 243–249, July 2018.

[13] D. Noe, "SENSING AND ACTING IN INTELLIGENT ASSEMBLY SYSTEMS," *Annual Reviews in Control,* vol. Volume 23, pp. 171-176, 1999.

[14] Prasad Ban, Shweta Desale, Revati Barge and Pallavi Chavan, "INTELLIGENT ROBOTIC ARM," in *ITM Web of Conferences 32 \ International Conference on Automation, Computing and Communication 2020 (ICACC-2020)*, Navi Mumbai, MH, India, 2020.

[15] Z.M. Bi, Y.Lin, W.J.Zhang, "The general architecture of adaptive robotic systems for manufacturing applications," *Robotics andComputer-IntegratedManufacturing,* vol. 26, no. 5, p. 461–470, Octobe 2010.

[16] Kensuke Harada, Tokuo Tsuji, Kazuyuki Nagata, Natsuki Yamanobe, Hiromu Onda, "Validating an object placement planner for robotic pick-and-place tasks," *Robotics and Autonomous Systems,* vol. Volume 62, no. Issue 10, pp. 1463-1477, October 2014.

[17] J. J. Craig, "THE MECHANICS AND CONTROL OF MECHANICAL MANIPULATORS," in *Introduction to Robotics: Mechanics and Control\ Third Edition*, Pearson Education International, 2005, pp. 5-7, 61-62.

[18] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo, Robotics: Modelling, Planning and Control/ Advanced Textbooks in Control and Signal Processing, Springer-Verlag London Limited, 2009, p. 4.

[19] K. Singh, "WORKING OF ROBOTIC ARM IN INDUSTRIES," https://www.researchgate.net/publication/353478116, Morai, India, 2021.

[20] Kevin M. Lynch and Frank C. Park, MODERN ROBOTICS: MECHANICS, PLANNING, AND CONTROL, Cambridge University Press, 2017, pp. 1-2.

[21] M. Raj Kumar, T. K. Aravindhan, R. Bhubesh Naveen, N. D. Dhanasekar, S. Gokulrevanth, "Testing and Calibration of Gantry Type Robot," *International Journal of Research in Engineering, Science and Management,* Vols. Volume-2, no. Issue-3, pp. 866 - 869, March 2019.

[22] D. R. D. II, *DESIGN, CONTROL, AND IMPLEMENTATION OF A THREE LINK ARTICULATED ROBOT ARM*, OHIO, USA: The Graduate Faculty of The University of Akron, 2008.

[23] Chien-Pin Chen, Pei-Jui Wang, Wei-Han Wang, Han-Pei Wang, and Che-Chien Chen, "Developing Industrial Dual Arm Robot for Flexible Assembly Through Reachability Map," ResearchGate, Taiwan, 2015.

[24] Bruno Siciliano, Oussama Khatib, Springer Handbook of Robotics, Switzerland: Springer Cham, 2016.

[25] I. -. w. The MathWorks, Image Processing Toolbox - MATLAB User's Guide Version 2022b, Massachusetts, USA: The MathWorks, Inc. - www.mathworks.com, 2022.

[26] "www.irya.unam.mx," IDL 8.2, [Online]. Available: https://www.irya.unam.mx/computo/sites/manuales/IDL/idl_Left.html#CSHID=Gui

deMe%2FImageProcessing%2FBackgroundFastFourierTransform.html|StartTopic=Content%2FGuideMe%2FImageProcessing%2FBackgroundFastFourierTransform.html|SkinName=IDLHelp. [Accessed 24 January 2023].

[27] M. Sinecen, "Digital Image Processing with MATLAB," in *Applications from Engineering with MATLAB Concepts*, London, United Kingdom, Adnan Menderes University, Computer Engineering Department, Aydin, Turkey, 2016.

[28] Paulo Roberto Gardel Kurka, Aldo André Díaz Salazar, "Applications of image processing in robotics and instrumentation," *Mechanical Systems and Signal Processing,* vol. Volume 124, p. 142–169, 1 June 2019.

[29] Gerald K. Ijemaru, Augustine O. Nwajana, Emmanuel U. Oleka, Richard I. Otuka, Isibor K. Ihianle, Solomon H. Ebenuwa, Emenike Raymond Obi, "Image processing system using MATLAB-based analytics," *Bulletin of Electrical Engineering and Informatics,* vol. Vol. 10, no. Issue 5, p. 2566~2577, October 2021.

[30] A.K. Sharadhi, Vybhavi Gururaj, Sahana P. Shankar, M.S. Supriya, Neha Sanjay Chogule, "Face mask recogniser using image processing and computer vision approach," *Global Transitions Proceedings,* vol. 3, no. 1, p. 67–73, June 2022.

[31] Isunuri B Venkateswarlu; Jagadeesh Kakarla; Shree Prakash, "Face mask detection using MobileNet and Global Pooling Block," in *IEEE 4th Conference on Information & Communication Technology (CICT),* 2020.

[32] Michael Haenlein, Andreas Kaplan, "Artificial intelligence and robotics: Shaking up the business world and society at large," *Journal of Business Research,* vol. 124, pp. 405-407, January 2021.

[33] Ahmed R. J. Almusawi, L. Canan Dülger and Sadettin Kapucu, "A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242)," *Computational Intelligence and Neuroscience,* vol. 2016, 2016.

[34] "Adaptive Control of Robotic Manipulators using Deep Neural Networks," *IFAC PapersOnLine*, vol. 55, no. 15, p. 148–153, 2022.

[35] Tahseen Fadhel Abaas, Alaa Hassan Shabeeb, "Path Planning Optimization of a Mobile Robot based on Intelligence Algorithm," in *IOP Conference Series: Materials Science and Engineering*, Baghdad - Iraq, 2020.

[36] B.K. Patle, Ganesh Babu L, Anish Pandey, D.R.K. Parhi, A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, pp. 582-606, 2019.

[37] Farhad Bayat, Sepideh NajaÞ-Nia, Morteza Aliyari, "Mobile Robots Path Planning: Electrostatic Potential Field Approach," *Expert Systems with Applications*, vol. 100, pp. 68-78, 15 June 2018.

[38] Lei Xie, Shuangfei Xue, Jinfen Zhang, Mingyang Zhang, Wuliu Tian, Stein Haugena, "A path planning approach based on multi-direction A* algorithm for ships navigating within wind farm waters," *Ocean Engineering*, vol. 184, pp. 311-322, 2019.

[39] Lars Larsen, Jonghwa Kim, Michael Kupke, Alfons Schuster, "Automatic Path Planning of Industrial Robots Comparing Sampling-Based and Computational Intelligence Methods," in *27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 2017*, Modena, Italy, 2017.

[40] James J. Kuffner, Jr. , Steven M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *In Proc. 2000 IEEE Int'l Conf. on Robotics and Automation (ICRA 2000)*, USA, 2000.

[41] Gopikrishnan. Sa, B. V. S. Shravana, Harshit Golea, Pratik Barvea and Ravikumar. L, "PATH PLANNING ALGORITHMS: A COMPARATIVE STUDY," in *National Conference on Space Transportation Systems*, Thiruvananthapuram, India, 2011.

[42] K. Kurzer, *Path Planning in Unstructured Environments: A Real-time Hybrid A* Implementation for Fast and Deterministic Path Generation for the KTH Research Concept Vehicle*, STOCKHOLM, SWEDEN : www.researchgate.net, 2016.

[43] Bhaavin K. Jogeshwar, K. Lochan, "Algorithms for Path Planning on Mobile Robots," *IFAC PapersOnLine*, vol. 55, no. 1, pp. 94-100, 2022.

[44] Han-ye Zhang, Wei-ming Lin and Ai-xia Chen, "Path Planning for the Mobile Robot: A Review," *symmetry*, vol. 10, no. 10, pp. 1-17, 1 October 2018.

[45] Yusuf Tansel _Ic, Mustafa Yurdakul, Berna Dengiz, "Development of a decision support system for robot selection," *Robotics and Computer-Integrated Manufacturing*, vol. 29, p. 142–157, 2013.

[46] Radu Eugen Breaz,Octavian Bologa, Sever Gabriel Racz, "Selecting industrial robots for milling applications using AHP," *Procedia Computer Scienc - Information Technology and Quantitative Management (ITQM 2017)*, vol. 122, pp. 346-353, 2017.

[47] Madiha Farman, Muneera Al-Shaibah, Zoha Aoraiath, Firas Jarrar, "Design of a Three Degrees of Freedom Robotic Arm," *International Journal of Computer Applications*, vol. 179, no. 37, pp. 12-17, April 2018.

[48] Ashwin Misra, Ayush Sharma, Ghanvir Singh, Ashish Kumar, Vikas Rastogi, "Design and Development ofa Low-Cost CNC Alternative SCARA Robotic Arm," *Procedia Computer Science*, vol. 171, p. 2459–2468, 2020.

[49] Hasan Dawood Salman, Mohsin Noori Hamzah, and Sadeq Hussein Bakhy, "KINEMATICS ANALYSIS AND IMPLEMENTATION OF THREE DEGREES OF FREEDOM ROBOTIC ARM BY USING MATLAB," *The Iraqi Journal For Mechanical And Material Engineering*, vol. 2l, no. 2, pp. 118-129, 19 June 2021.

[50] P.S. Minz, Ish Kumar Sawhney, Charanjiv Singh Saini, "Algorithm for processing high definition images for food colourimetry," *Measurement*, vol. 158, pp. 1-8, 2020.

[51] Hazim Nasir, Ali Hussein mohammed alhilli Alhelli, *Optimum Path Planning And Performance Analysis Of A Robot Manipulator,* Baghdad, Iraq: www.researchgate.net, 2013.

[52] Ahmed T. Sadiq, Firas A. Raheem, Noor Alhuda F. Abbas, "Robot Arm Path Planning Using Modified Particle Swarm Optimization based on D* algorithm," *Al-Khwarizmi Engineering Journal*, vol. 13, no. 3, pp. 27-37, 2017.

[53] Stelian-Emilian Olteana, "Mobile Robot Platform with Arduino Uno and Raspberry Pi for Autonomous Navigation," *Procedia Manufacturing,* vol. 32, p. 572–577, 2019.

[54] Tahseen Fadil Abbas, Alaa Hassan, "Path Planning and Obstacle Avoidance of a Mobile Robot based on GWO Algorithm," *Al-Khwarizmi Engineering Journal,,* vol. 18, no. 2, pp. 13-28, June 2022.

[55] Aisha Muhammad, Nor Rul Hasma Abdullah, Mohammed A.H Ali, Ibrahim Haruna Shanono, Rosdiyana Samad, "Simulation Performance Comparison of A*, GLS, RRT and PRM Path Planning Algorithms," *IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE),* pp. 258-263, 2022.

[56] K. Pribilová, Z. Mäsiarova, "Application of path planning algorithms in the MATLAB environment using Lego Mindstorms," *2022 20th International Conference on Emerging eLearning Technologies and Applications (ICETA),* pp. 543-548, 2022.

[57] Alia N. Barakat, Khaled A.Gouda, Kenz A. Bozed, "Kinematics Analysis and Simulation of a Robotic Arm using MATLAB," in *Proceedings 0f 2016 4th International Connce on Control Engineering & Information Technology*, Hammamet, Tunisia, 2016.

[58] Shuji Yang, Hao Wen, Dongping Jin, "Trajectory planning of dual-arm space robots for target capturing and base manoeuvring," *Acta Astronautica,* vol. 164, p. 142–151, November 2019.

[59] Kuo-Ching Ying, Pourya Pourhejazy, Chen-Yang Cheng, Zong-Ying Cai, "Deep learning-based optimization for motion planning of dual-arm assembly robots," *Computers & Industrial Engineering*, vol. 160, pp. 107603 (1-14), October 2021.

[60] H. -I. Lin, C. -H. Cheng and W. -K. Chen, "Learning a pick-and-place robot task from human demonstration," in *International Automatic Control Conference (CACS)*, Nantou, Taiwan, 2013.

[61] Jun Li, Alexis Maldonado, Michael Beetz and Anna Schuboe, "Obstacle avoidance in a pick-and-place task," in *International Conference on Robotics and Biomimetics*, Guilin, China, 2009.

[62] Vishnu R. Kale, V. A. Kulkarni, "Automation Of Object Sorting System Using Pick & Place Robotic Arm & Image Processing," in *Proceedings of 3rd IRAJ International Conference,*, Mumbai, India, 2014.

[63] Pedro Tavares, José Lima, Pedro Costa and António Paulo Moreira, "ADAPTIVE PICK AND PLACE APPROACH USING ROS FRAMEWORK," in *International Conferences Intelligent Systems and Agents 2015 and Theory and Practice in Modern Computing*, Porto, Portugal, 2015.

[64] Sotiris Makris, Panagiota Tsarouchi, Dragoljub Surdilovic, Jorg Kruger, "Intuitive dual arm robot programming for assembly operations," *CIRP Annals - Manufacturing Technology*, vol. 63, no. 1, pp. 13-16, 2014.

[65] I. The MathWorks, "MathWorks, Inc.," MathWorks, Inc., [Online]. Available: https://www.mathworks.com/products/image.html?s_tid=srchtitle_image%20processing_1. [Accessed 1 September 2022].

[66] B. Padmapriya, T. Kesavamurthi, H. Wassim Ferose c, "Edge Based Image Segmentation Technique for Detection and Estimation of the Bladder Wall Thickness," in *International Conference on Communication Technology and System Design*, India, 2012.

[67] Dr. Ali Abbar Khleif, Athraa M.Salih Ahmed, "Image Based 3D Object Reconstruction System," *Engineering and Technology Journal,* vol. 34, no. 2, pp. 393-405, 2016.

[68] Mohanad Qusay Abbood, Dr. Tahseen Fadhil Abbas, Dr. Ali Abbar Khleif, *Design and Implementation of an Automated System for Object Sorting Based on Color and Geometry,* Baghdad: University of Technology of Baghdad, 2020.

[69] Batuhan Hangün, Önder Eyecioğlu, "Performance Comparison Between OpenCV Built in CPU and GPU Functions on Image Processing Operations," *INTERNATIONAL JOURNAL of ENGINEERING SCIENCE AND APPLICATION,* vol. 1, no. 2, pp. 34-41, 2017.

[70] Lucia Romani, Milvia Rossini, Daniela Schenone, "Edge detection methods based on RBF interpolation," *Journal of Computational and Applied Mathematics,* vol. 349, pp. 532-547, 2018.

[71] "https://www.educative.io/," [Online]. Available: https://www.educative.io/answers/what-is-the-a-star-algorithm. [Accessed 13 January 2023].

[72] T. Bakshi, Algorithms- Notes for Professionals.

[73] Jie Ji, Hong Wang, Yue Ren, Path Planning and Tracking for Vehicle Collision Avoidance in Lateral and Longitudinal Motion Directions, China: MORGAN & CLAYPOOL, 2021.

[74] Kucuk, Serdar & Bingul, Z, "Robot Kinematics: Forward and Inverse Kinematics," in *Industrial Robotics: Theory, Modelling and Control*, Research Gate \ InTech, December 2006, pp. 117 - 148.

[75] JANEZ FUNDA, RUSSELL H TAYLOR, SENIOR MFVRFR AND RICHARD P PAUL FELLOW., "On homogeneous transforms, quaternions, and computational

efficiency," *TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 6, no. 3, pp. 382-388, June 1990.

[76] Ziting Hou, Shumei Ma , Qingfei Zeng , Aiping Li, "Kinematics analysis and self - collision detection of Truss type multi-robot cooperative welding platform," in *52nd CIRP Conference on Manufacturing Systems* , Shanghai 201800 China, 2019.

[77] Carlos Lopez-Franco, Jesus Hernandez-Barragan, Alma Y. Alanis, Nancy Arana-Daniel, "A soft computing approach for inverse kinematics of robot manipulators," *Engineering Applications of Artificial Intelligence*, vol. 74, p. 104–120, 2018.

[78] TaeW.Lim, "Point cloud modeling using the homogeneous transformation for non-cooperative pose estimation," *ActaAstronautica*, vol. 111, pp. 61-76, June–July 2015.

[79] Hee Sung An, Jie Hyeung Lee, Chan Lee, TaeWon Seo, Jeh Won Lee, "Geometrical Kinematic Solution of Serial Spatial Manipulators Using Screw Theory," *Mechanism and Machine Theory*, vol. 116, p. 404–418, 2017.

[80] Hind Hadi Abdulridha, Dr. Tahseen Fadhel Abaas, "DIFFERENTIAL MOTION ANALYSIS OF LAB-VOLT R5150 ROBOT SYSTEM," *Global Journal of Engineering Science and Research Management*, vol. 4, no. 10, pp. 152-160, October 2017.

[81] Serdar Kucuk, Zafer Bingul, "Inverse kinematics Solutions for Industrial Robot Manipulators with Offset Wrists," *Applied Mathematical Modelling*, vol. 38, p. 1983–1999, 2014.

[82] Dr.Tahseen Fadhel Abaas, Hind Hadi Abdulridha, "Inverse Kinematic Analysis of Lab-Volt R5150 Robot system," *IJR*, vol. 4, no. 13, pp. 81-88, October 2017.

[83] N. ROBOT, "NEXUS ROBOT," NEXUS ROBOT, [Online]. Available: https://www.nexusrobot.com/. [Accessed 15 February 2023].

[84] N. Robot, "NEXUS ROBOT," NEXUS ROBOT, [Online]. Available: https://www.nexusrobot.com/product/new-tracked-mobile-tank-robot-kit-10022.html. [Accessed 15 February 2023].

[85] R. E. Makers, "ROBOTDIGG," ROBOTDIGG Equip Makers, [Online]. Available: https://www.robotdigg.com/product/1463/Robot-Arm-six-degrees-of-freedom-6-DOF-RoboSEED. [Accessed 15 February 2023].

[86] A. group, "ABB," ABB, [Online]. Available: https://global.abb/group/en. [Accessed 15 February 2023].

[87] Benjamin Cohen, Sachin Chitta and Maxim Likhachev, "Single- and dual-arm motion planning with heuristic search," *The International Journal of Robotics Research*, vol. Vol. 33, no. Issue 2, p. 305–320, 2014.

[88] Hao Ye, Dong Wang, Jun Wu, Yi Yuec, Yulin Zhou, "Forward and inverse kinematics of a 5-DOF hybrid robot for composite material machining," *Robotics and Computer Integrated Manufacturing*, vol. 65, p. 101961, October 2020.

APPENDIX A

MATLAB CODES, USED IN THIS STUDY

A.1 MAIN CODE FOR PICK-PLACE OPERATIONS

```
clc
clear
close all
cr=input('Please select the robot for simulation\n1- RoboSEED S6H4\n2- ABB IRB
120\n1, 2, or 3 to choose your special robot?\n');
if        cr==1%https://www.robotdigg.com/product/1463/Robot-Arm-six-degrees-of-
freedom-6-DOF-RoboSEED
min1J5dof=(-160);max1J5dof=(160);min2J5dof=(-90);max2J5dof=(110);min3J5dof=(-
0);
max3J5dof=(160);min4J5dof=(-290);max4J5dof=(105);min5J5dof=(-
130);max5J5dof=(130);
links=[110,255,275,93.8,258.5,99];
ActDist=1500;%1469.5;%the actual distance to be compared with the calibrated value
elseif cr==2%https://search.abb.com/library/Download.aspx?DocumentID=3HAC035960-
001&LanguageCode=en&DocumentPartId=&Action=Launch
min1J5dof=(-165);max1J5dof=(165);min2J5dof=(-110);max2J5dof=(110);min3J5dof=(-
110);
max3J5dof=(70);min4J5dof=(-160);max4J5dof=(160);min5J5dof=(-
120);max5J5dof=(120);
links=[110,290,270,70,302,72];
ActDist=1500;%580+580;%the actual distance to be compared with the calibrated
value
else
min1J5dof=input('Please enter the minmum limit (in degree) of the first joint=
');max1J5dof=input('Please enter the mximum limit (in degree) of the first joint=
');min2J5dof=input('Please enter the minmum limit (in degree) of the second joint=
');max2J5dof=input('Please enter the mximum limit (in degree) of the second joint=
');min3J5dof=input('Please enter the minmum limit (in degree) of the third joint=
');max3J5dof=input('Please enter the mximum limit (in degree) of the third joint=
');min4J5dof=input('Please enter the minmum limit (in degree) of the forth joint=
');max4J5dof=input('Please enter the mximum limit (in degree) of the forth joint=
');min5J5dof=input('Please enter the minmum limit (in degree) of the fifth joint=
');max5J5dof=input('Please enter the mximum limit (in degree) of the fifth joint=
');a=input('Please input the hieght of the base: ');b=input('Please input the
length of the first link: ');c=input('Please input the length of the second link:
');d=input('Please input the length of the third link: ');e=input('Please input
the length of the forth link: ');f=input('Please input the length of the fifth
link which is connected to the end-effector: ');
links=  [a,b,c,d,e,f];ActDist=input('Please enter the actual distance for
calibration= ');
end
%mobile platform: httpsosoyoo.storeproducts4wd-omni-wheel-robotic-mecanum-wheel-
robot-car-platform-chassis-with-dc-speed-encoder-motor-for-arduino-raspberry-
pivariant=31634957336687
%% for Cases.
x=figure("Name","Please choose one of the above cases by inputing the title of
each       case       or       press       cu       of       your       special
case:",'WindowState','maximized');annotation(x,'textbox',[0.229449859501228
```

```
0.032930845225028 0.555989583333333 0.0461031833150384],'String',"Please choose
one of the below cases by inputing the title of each case or press cu of your
special
case:",'FitBoxToText','on','FontWeight','bold','FontSize',12,'FontName','Times
New                Roman','BackgroundColor',[0.0588235294117647                1
1]);annotation(x,'textbox',[0.00829166666666667    0.774972557628979    0.1130625
0.176728869374314],'Color',[0 1 0],'String',{'Small green square represents','the
picking      point'},'FontWeight','bold','FontSize',12,'FontName','Times      New
Roman','FitBoxToText','off','EdgeColor',[0                                      1
0],'BackgroundColor',[0.149019607843137                          0.149019607843137
0.149019607843137]);annotation(x,'textbox',[0.00829166666666667
0.535675082327113 0.114625 0.216245883644348],'Color',[1 0 0],'String',{'Big red
square                        represents                          the','placing
point'},'FontWeight','bold','FontSize',12,'FontName','Times                    New
Roman','FitBoxToText','off','EdgeColor',[1     0     0],'BackgroundColor',[0     0
0]);annotation(x,'textbox',[0.00777083333333333    0.285400658616904    0.114625
0.209659714599342],'Color',[0.149019607843137                    0.149019607843137
0.149019607843137],'String',{'the + symbole represents the obstacles','which
devide          the          working          area          into          4
quadrents'},'FontWeight','bold','FontSize',12,'FontName','Times              New
Roman','FitBoxToText','off','BackgroundColor',[0.650980392156863
0.650980392156863          0.650980392156863]);c1=('E:\ME599            Master
Thesis\Code\My_Current_Try\FinalCode\P and P cases\1.jpeg');c2=('E:\ME599 Master
Thesis\Code\My_Current_Try\FinalCode\P and P cases\2.jpeg');c3=('E:\ME599 Master
Thesis\Code\My_Current_Try\FinalCode\P and P cases\3.jpeg');c4=('E:\ME599 Master
Thesis\Code\My_Current_Try\FinalCode\P and P cases\4.jpeg');cu=('E:\ME599 Master
Thesis\Code\My_Current_Try\FinalCode\P                  and                   P
cases\User.jpeg');subplot(2,2,1);imshow(c1);title('c1');subplot(2,2,2);imshow(c
2);title('c2');subplot(2,2,3);imshow(c3);title('c3');subplot(2,2,4);imshow(c4);
title('c4');
cp=input('Please enter the required case you want to check the pick and place
path planning\nYou can choose from c1 to c4 as per the appeared figure or press
cu of your special case: ');close all
%% for Dim.
xh=figure("Name","Please check the dimensions, press ok to proceed otherwise
press       c       for       alternative       image       for       editable
dimensions:",'WindowState','maximized');
annotation(xh,'textbox',[0.206533192834561,0.070252469813392,0.623551106427819,
0.0461031833150384],'String',"please check the dimensions, press ok to proceed
otherwise        press    c    for    alternative    image    for    editable
dimensions",'FitBoxToText','on','FontWeight','bold','FontSize',12,'FontName','T
imes New Roman','BackgroundColor',[0.0588235294117647 1 1]);ok=('E:\ME599 Master
Thesis\Code\My_Current_Try\FinalCode\P and P cases\ok.jpg');
c=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\P and P cases\c.jpg');
subplot(1,1,1);imshow(ok);title('Dimensions');cph=input('please    check    the
dimensions, press ok to proceed otherwise  press c for alternative image for
editable dimensions ');close all
%%%%%%%%
%% for reading and processing Dim. image
Imgh=imread(cph);ImgGrayh=im2gray(Imgh);[BW,maskedImageh]                       =
segmentImageh(ImgGrayh);[BW_outh,propertiesh]                                   =
filterRegionsh(BW);hb=propertiesh(4).BoundingBox(4);hc=propertiesh(3).BoundingB
ox(4);lc=propertiesh(2).BoundingBox(4);ho=propertiesh(1).BoundingBox(4);
%% for reading and processing cases image
Img=imread(cp);ImgGray=im2gray(Img);
%% to use segmentImage function
[BW,maskedImage] = segmentImage(ImgGray);
```

```
%% to use filterRegions function
[BW_out,properties] = filterRegions(BW);
[xxx,yyy]=size(BW);
for ic=1:length(properties)
    if properties(ic).roundness>0.95 && properties(ic).roundness<1.1
    Xc(ic)=properties(ic).Centroid(1);Yc(ic)=properties(ic).Centroid(2);
    end
    if properties(ic).Perimeter > 50
        AreaObj(ic)=properties(ic).Area;Diam(ic)=properties(ic).EquivDiameter;
    end
    CoordObj(ic,:)=properties(ic).Centroid;
end

%% for calibration
InPix=[1004,1004;1,1];
Dist=sqrt((InPix(1,1)-InPix(2,1))^2+(InPix(1,2)-
InPix(2,2))^2);[~,Idx]=maxk(AreaObj,2);BoxLoc=CoordObj(Idx(1),:);
CubeLoc=CoordObj(Idx(2),:);
%% function
[Traj5dof,Ang5,pointsAV,pointsT]=tte5nO(CalibrationVal,links,BoxLocW,ro,rc,Cube
LocW,BoxLoc,CubeLoc,xxx,yyy,hc,hb,lc,ho,min1J5dof,min2J5dof,min3J5dof,min4J5dof
,min5J5dof,max1J5dof,max2J5dof,max3J5dof,max4J5dof,max5J5dof);
figure('Name','Positions of end-effector of the manipolator during the
task',WindowState='maximized')
subplot(3,1,1)
plot(Traj5dof(:,1))
grid on
xlabel('Unit of time')
ylabel('x-Position')
legend('5DOF')
subplot(3,1,2)
plot(Traj5dof(:,2))
grid on
xlabel('Unit of time')
ylabel('y-Position')
legend('5DOF')
subplot(3,1,3)
plot(Traj5dof(:,3))
grid on
xlabel('Unit of time')
ylabel('z-Position')
legend('5DOF')
sgtitle('Positions of end-effector of the manipolator during the task')
figure('Name','angles value in each step for each joint',WindowState='maximized')
for i = 1:5
subplot(5,1,i)
plot(Ang5(:,i))
grid on
xlabel('Steps of movement')
ylabel('angles per Degreee')
legend('angles values')
sgtitle('angles value in each step for each joint')
end
figure('Name','Error in positions of the end-effector','WindowState','maximized')
subplot(3,1,1)
plot(abs(Traj5dof(:,1)-pointsAV(:,1)))
grid on
```

```
xlabel('Unit of time')
ylabel('x-Position Error')
legend('5DOF')
subplot(3,1,2)
plot(abs(Traj5dof(:,2)-pointsAV(:,2)))
grid on
xlabel('Unit of time')
ylabel('y-Position Error')
legend('5DOF')
subplot(3,1,3)
plot(abs(Traj5dof(:,3)-pointsAV(:,3)))
grid on
xlabel('Unit of time')
ylabel('z-Position Error')
legend('5DOF')
sgtitle('Error in positions of the end-effector')
```

A.2 USER-DEFINED FUNCTIONS USED IN THIS STUDY

```
function
[Traj5dof,Ang5,pointsAV,pointsT]=tte5nO(CalibrationVal,links,BoxLocW,ro,rc,Cube
LocW,BoxLoc,CubeLoc,xxx,yyy,hc,hb,lc,ho,min1J5dof,min2J5dof,min3J5dof,min4J5dof
,min5J5dof,max1J5dof,max2J5dof,max3J5dof,max4J5dof,max5J5dof)
robot = rigidBodyTree('DataFormat','column','MaxNumBodies',10);
body = rigidBody('link1');
joint = rigidBodyJoint('joint1', 'revolute');
joint.PositionLimits = [min1J5dof,max1J5dof];
body.Joint = joint;
body = rigidBody('link2');
joint = rigidBodyJoint('joint2','revolute');
joint.PositionLimits = [min2J5dof,max2J5dof];
body.Joint = joint;
body = rigidBody('link3');
joint = rigidBodyJoint('joint3','revolute');
joint.PositionLimits = [min3J5dof,max3J5dof];
body.Joint = joint;
body = rigidBody('link4');
joint = rigidBodyJoint('joint4','revolute');
joint.PositionLimits = [min4J5dof,max4J5dof];
body.Joint = joint;
body = rigidBody('link5');
joint = rigidBodyJoint('joint5','revolute');
joint.PositionLimits = [min5J5dof,max5J5dof];
body.Joint = joint;
body = rigidBody('tool');
joint = rigidBodyJoint('fix1','fixed');
body.Joint = joint;
numberOFcontrolingPoints=3;
intervalDistancebetweentowPoints=0.2;
if (((xxx/2 < BoxLoc(1) && xxx/2 > CubeLoc(1)) || (xxx/2>BoxLoc(1) &&
xxx/2<CubeLoc(1))) && ((yyy/2<BoxLoc(2) && yyy/2>CubeLoc(2)) || (yyy/2>BoxLoc(2)
&& yyy/2<CubeLoc(2))))
if ((CubeLocW(1)+BoxLocW(1))/2<75*CalibrationVal&&(CubeLocW(1)+BoxLocW(1))/2>0)
pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
pyT=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
```

```
pz=[hc,hc+lc,hb];pzT=[hc,hc+lc,hb];
elseif ((CubeLocW(1)+BoxLocW(1))/2>75*CalibrationVal)
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];py=[CubeLocW(2),(CubeLoc
W(2)+BoxLocW(2))/2,BoxLocW(2)];pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxL
ocW(1)];pyT=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.7*(ho+(2*lc))),hb];
pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];
pzT=[hc,hc+lc,hb];
end
elseif                                              (((CubeLocW(1)+BoxLocW(1))/2>-
75*CalibrationVal&&(CubeLocW(1)+BoxLocW(1))/2<0))
px=[CubeLocW(1),((CubeLocW(1)+BoxLocW(1))/2)+((-100*CalibrationVal)-
(CubeLocW(1)+BoxLocW(1))/2),BoxLocW(1)];
py=[CubeLocW(2),((CubeLocW(2)+BoxLocW(2))/2)+((100*CalibrationVal)-
(CubeLocW(2)+BoxLocW(2))/2),BoxLocW(2)];
pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
pyT=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
elseif ((CubeLocW(1)+BoxLocW(1))/2<-75*CalibrationVal)
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];py=[CubeLocW(2),(CubeLoc
W(2)+BoxLocW(2))/2,BoxLocW(2)];
pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
pyT=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.7*(ho+(2*lc))),hb];pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];pzT=[hc,hc+lc,hb];
end
elseif                                               ((CubeLocW(1)+BoxLocW(1))/2>-
75*CalibrationVal&&(CubeLocW(1)+BoxLocW(1))/2<0)
px=[CubeLocW(1),((CubeLocW(1)+BoxLocW(1))/2)+((-100*CalibrationVal)-
(CubeLocW(1)+BoxLocW(1))/2),BoxLocW(1)];
py=[CubeLocW(2),((CubeLocW(2)+BoxLocW(2))/2)+((-100*CalibrationVal)-
(CubeLocW(2)+BoxLocW(2))/2),BoxLocW(2)];
pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
pyT=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
elseif ((CubeLocW(1)+BoxLocW(1))/2<-75*CalibrationVal)
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
py=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
pyT=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.7*(ho+(2*lc))),hb];
pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];pzT=[hc,hc+lc,hb];
end
elseif
((CubeLocW(1)+BoxLocW(1))/2<75*CalibrationVal&&(CubeLocW(1)+BoxLocW(1))/2>0)
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2+((100*CalibrationVal)-
(CubeLocW(1)+BoxLocW(1))/2),BoxLocW(1)];
py=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2+((-100*CalibrationVal)-
(CubeLocW(2)+BoxLocW(2))/2),BoxLocW(2)];
pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];pzT=[hc,hc+lc,hb];
elseif ((CubeLocW(1)+BoxLocW(1))/2>75*CalibrationVal)
```

```
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
py=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
pyT=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.7*(ho+(2*lc))),hb];pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];
pzT=[hc,hc+lc,hb];
end
else
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
py=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];
pyT=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.7*(ho+(2*lc))),hb];
pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];
pzT=[hc,hc+lc,hb];
end
end
elseif ((CubeLocW(1)>0&&BoxLocW(1)>0) && (CubeLocW(2)>0&&BoxLocW(2)>0))
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];py=[CubeLocW(2),(CubeLoc
W(2)+BoxLocW(2))/2,BoxLocW(2)];pxT=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxL
ocW(1)]; pzT=[hc,hc+lc,hb];
elseif ((CubeLocW(1)+BoxLocW(1))/2<150*CalibrationVal
px=[CubeLocW(1),((CubeLocW(1)+BoxLocW(1))/2)+((140*CalibrationVal)-
((CubeLocW(1)+BoxLocW(1))/2)),BoxLocW(1)];py=[CubeLocW(2),(CubeLocW(2)+BoxLocW(
2))/2,BoxLocW(2)];pz=[hc,hc+lc,hb];pzT=[hc,hc+lc,hb];
elseif         (CubeLocW(1)+BoxLocW(1))/2>=150*CalibrationVal                  &&
((CubeLocW(2)>0 && BoxLocW(2)<0))
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];py=[CubeLocW(2),0,BoxLoc
W(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.5*(ho+(2*lc))),hb];
pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];pzT=[hc,hc+lc,hb];
end
elseif             ((CubeLocW(1)+BoxLocW(1))/2>-150*CalibrationVal              &&
(CubeLocW(1)+BoxLocW(1))/2<0))
px=[CubeLocW(1),((CubeLocW(1)+BoxLocW(1))/2)-
((140*CalibrationVal)+((CubeLocW(1)+BoxLocW(1))/2)),BoxLocW(1)];py=[CubeLocW(2)
,(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
elseif        ((CubeLocW(1)+BoxLocW(1))/2<=-150*CalibrationVal)               &&
((CubeLocW(2)>0 && BoxLocW(2)<0)
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];py=[CubeLocW(2),0,BoxLoc
W(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.5*(ho+(2*lc))),hb];pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];pzT=[hc,hc+lc,hb];
end
elseif             ((CubeLocW(2)+BoxLocW(2))/2<150*CalibrationVal              &&
(CubeLocW(2)+BoxLocW(2))/2>0)
```

```
py=[CubeLocW(2),((CubeLocW(2)+BoxLocW(2))/2)+((140*CalibrationVal)-
((CubeLocW(2)+BoxLocW(2))/2)),BoxLocW(2)];px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(
1))/2,BoxLocW(1)];pz=[hc,hc+lc,hb]; pzT=[hc,hc+lc,hb];
elseif         (CubeLocW(2)+BoxLocW(2))/2>=150*CalibrationVal                  &&
((CubeLocW(1)>0 && BoxLocW(1)<0)
px=[CubeLocW(1),0,BoxLocW(1)];
py=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLocW(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.5*(ho+(2*lc))),hb]; pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];pzT=[hc,hc+lc,hb];
end
elseif           (((CubeLocW(2)+BoxLocW(2))/2>-150*CalibrationVal)            &&
(CubeLocW(2)+BoxLocW(2))/2<0 )
py=[CubeLocW(2),((CubeLocW(2)+BoxLocW(2))/2)-
((140*CalibrationVal)+((CubeLocW(2)+BoxLocW(2))/2)),BoxLocW(2)];
px=[CubeLocW(1),(CubeLocW(1)+BoxLocW(1))/2,BoxLocW(1)];pz=[hc,hc+lc,hb];
pzT=[hc,hc+lc,hb];
elseif           ((CubeLocW(2)+BoxLocW(2))/2<=-150*CalibrationVal             &&
(CubeLocW(2)+BoxLocW(2))/2<0 )
px=[CubeLocW(1),0,BoxLocW(1)];py=[CubeLocW(2),(CubeLocW(2)+BoxLocW(2))/2,BoxLoc
W(2)];
if (ho>hc) || (ho> hb)
pz=[hc,(1.5*(ho+(2*lc))),hb];
pzT=[hc,hc+lc,hb];
else
pz=[hc,hc+lc,hb];
pzT=[hc,hc+lc,hb];
end
end
n=numberOFcontrolingPoints-
1;nn=0:n;coff=factorial(n)./(factorial(nn).*factorial(n-nn));t=0;
for i=1:numberOFcontrolingPoints
    b(i,:)=coff(i)*t.^nn(i).*(1-t).^(n-nn(i));
end
bx=zeros(1,numel(t));by=zeros(1,numel(t));bz=zeros(1,numel(t));bxT=zeros(1,nume
l(t));byT=zeros(1,numel(t));bzT=zeros(1,numel(t));
for i=1:numberOFcontrolingPoints
bx=b(i,:)*px(i)+bx;bxT=b(i,:)*pxT(i)+bxT;by=b(i,:)*py(i)+by;byT=b(i,:)*pyT(i)+b
yT;bz=b(i,:)*pz(i)+bz;bzT=b(i,:)*pzT(i)+bzT;
end
pointsT=    [CubeLocW(1),CubeLocW(2),hc;bxT',byT',bzT';BoxLocW(1),    BoxLocW(2),
hb;];
pointsAV=  [CubeLocW(1),  CubeLocW(2),  hc;bx',by',bz';BoxLocW(1),  BoxLocW(2),
hb;];q0 = homeConfiguration(robot);ndof = length(q0);qs = zeros(height(pointsAV),
ndof);
inverseKinematics('RigidBodyTree',
robot,'SolverAlgorithm','LevenbergMarquardt');
weights = [0.025, 0.025, 0.025, 1, 1, 0.1];qInitial = q0;
for i = 1:height(pointsAV)
point                    =                  pointsAV(i,:);qSol                    =
ik(endEffector,trvec2tform(point),weights,qInitial);qs(i,:) = qSol;Traj5dof(i,:)
= tform2trvec(getTransform(robot,qSol,'tool','base'));qInitial = qSol;
end
r=sum(links)-links(1);[X,Y,Z]                                                    =
sphere(100);x1=r*X;y1=r*Y;z1=links(1)+r*Z;Idx=find(z1(:,1)>0);
```

```
figure('Name','simulation of the task path planning of 6-DOF regardless the
gripper','WindowState','maximized')
show(robot,qs(1,:);
hold on
title ('simulation of the task path planning of 6-DOF regardless the gripper','
','Color','k','FontWeight','bold','FontName','Times New Roman','FontSize',16)
plot3(pointsT(1:end,1),pointsT(1:end,2),pointsT(1:end,3),'r')
plot3(pointsAV(1:end,1),pointsAV(1:end,2),pointsAV(1:end,3),'b')
plot3(Traj5dof(1:end,1),Traj5dof(1:end,2),Traj5dof(1:end,3),'g')
plot3(CubeLocW(1),CubeLocW(2),hc,'*','Color','g')
y222=0;
for x222=-500*CalibrationVal:50*CalibrationVal:-150*CalibrationVal
for z222=0:10*CalibrationVal:ho
plot3(x222,y222,z222,'_','Color','k')
hold on
end
end
hold on
for x222=150*CalibrationVal:50*CalibrationVal:500*CalibrationVal
for z222=0:10*CalibrationVal:ho
plot3(x222,y222,z222,'_','Color','k')
hold on
end
end
framesPerSecond = 15;
r = rateControl(framesPerSecond);
for i = 1:height(pointsAV)
show(robot,qs(i,:);
drawnow
waitfor(r);pause(0.5)
end
Ang5=qs;
end
```

A.3 MATLAB CODES FOR PATH PLANNING ALGORITHMS

```
clc
clear
close all
x=figure("Name","Please choose one of the below cases by inputing the title of
each case","WindowState","maximized");
annotation(x,'textbox',[0.103645833333334  0.950618384373906  0.830208333333333
0.0461031833150384],'String','Please enter the required case you want to check
the path planning which is starting from case1 to case8 as per the appeared
figures or u for editable
case','FontWeight','bold','FontSize',12,'FontName','Times New
Roman','BackgroundColor',[0.0588235294117647 1 1]);
annotation(x,'textbox',[0.00881250000000006 0.805708013172338 0.104208333333333
0.110867178924261],'Color',[0 1 0],'String',{'Small green point represents','the
starting point'},'FontWeight','bold','FontSize',12,'FontName','Times New
Roman','FitBoxToText','off','EdgeColor',[0 1
0],'BackgroundColor',[0.149019607843137 0.149019607843137 0.149019607843137]);
annotation(x,'textbox',[0.010375 0.658616904500549 0.0990000000000001
0.0998902305159178],'Color',[1 0 0],'String',{'Big red point represents
the','target'},'FontWeight','bold','FontSize',12,'FontName','Times New
Roman','FitBoxToText','off','EdgeColor',[1 0 0],'BackgroundColor',[0 0 0]);
annotation(x,'textbox',[0.0106770833333334 0.482985729967069 0.0940104166666666
0.110964873765097],'Color',[0.149019607843137 0.149019607843137
0.149019607843137],'String',{'All black areas
represent','obstacles'},'FontWeight','bold','FontSize',12,'FontName','Times New
Roman','FitBoxToText','off','BackgroundColor',[0.650980392156863
0.650980392156863 0.650980392156863]);
case1=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\1.png');case2=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\2.png');case3=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\3.png');
case4=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\4.png');case5=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\5.png');case6=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\6.png');
case7=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\7.png');case8=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\8.png');u=('E:\ME599 Master Thesis\Code\My_Current_Try\FinalCode\PP
cases\User.png');
subplot(2,4,1);imshow(case1);title('case1');subplot(2,4,2);imshow(case2);title(
'case2');subplot(2,4,3);imshow(case3);title('case3');subplot(2,4,4);imshow(case
4);title('case4');subplot(2,4,5);imshow(case5);title('case5');subplot(2,4,6);im
show(case6);title('case6')
subplot(2,4,7);imshow(case7);title('case7');subplot(2,4,8);imshow(case8);title(
'case8');
cp=input('Please enter the required case you want to check the path
planning\nwhich is starting from case1 to case8 as per the appeared figure or u
for your editable case: ');
close all
picOBS=imread(cp);
picG_OBS=im2gray(picOBS);
[BW,maskedImage] = segmentImageP(picG_OBS);
[BW_out,properties] = filterRegionsP(BW);
for ic=1:length(properties)
```

```
    if   (properties(ic).roundness>0.99   &&   properties(ic).roundness<1.1)   &&
((properties(ic).Area   <=   2250   &&   properties(ic).Area   >=   2100)   ||
(properties(ic).Area   <=   5250   &&   properties(ic).Area   >=   5100))   &&
((properties(ic).Perimeter  <=  165  &&  properties(ic).Perimeter  >=  160)  ||
(properties(ic).Perimeter <= 255 && properties(ic).Perimeter >= 250))
        AreaObj(ic)=properties(ic).Area;
        st(ic)=properties(ic).EquivDiameter;
    end
end
[~,o]=maxk(AreaObj,2);
Gc=properties(o(1)).Centroid;
Sc=properties(o(2)).Centroid;
plT=(properties(o(1)).PixelList);
xT=plT(:,2);
yT=plT(:,1);
plS=(properties(o(2)).PixelList);
xS=plS(:,2);
yS=plS(:,1);
[~,kk]=maxk(st,2);
sT=(properties(kk(1)).EquivDiameter/2);
sS=(properties(kk(2)).EquivDiameter/2);
pic=imcomplement(picOBS);
picG=im2gray(pic);
picBW=imbinarize(picG);
picBW=double(picBW);
```

A.3.1 A* Algorithm

```
tic
map=occupancyMap(picBW);
setOccupancy(map, [xS,yS])
setOccupancy(map, [xT,yT])
PP=plannerAStarGrid(map,C,A,'TieBreaker',TB,'DiagonalSearch',DS);
Sc=[Sc(1),1000-Sc(2)];
Gc=[Gc(1),1000-Gc(2)];
[path,details]=plan(PP,round(Sc),round(Gc),"world");
path(:,3)=0;
figure('Name','result',"WindowState","maximized")
show(PP,"ExploredNodes","on")
ax = gca;
ax.YAxis.Exponent = 0;
ax.XAxis.Limits = [0 1000];
ax.YAxis.Limits = [0 1000];
ax.XAxis.TickValuesMode='manual';
ax.YAxis.TickValuesMode='manual';
ax.XAxis.TickValues = 0:100:1000;
ax.YAxis.TickValues = 0:100:1000;
ax.XAxis.Label.String = 'Unite of Length in X Axis';
ax.YAxis.Label.String = 'Unite of Length in Y Axis';
ax.XAxis.TickLabelColor='k';
ax.YAxis.TickLabelColor='k';
ax.XAxis.TickLabelInterpreter='latex';
ax.YAxis.TickLabelInterpreter='latex';
ax.XAxis.TickLabelRotation = -45;
ax.YAxis.TickLabelRotation = 90;
ax.XAxis.TickLabelsMode='auto';
ax.XAxis.LabelHorizontalAlignment = 'center';
ax.YAxis.LabelHorizontalAlignment = 'center';
ax.YAxis.TickDirectionMode='auto';
ax.XAxis.TickDirection='both';
ax.YAxis.TickDirection='both';
ax.XAxis.TickLength = [0.02 0.035];
ax.YAxis.TickLength = [0.02 0.035];
ax.XAxis.MinorTick = 'on';
ax.YAxis.MinorTick = 'on';
ax.XAxis.MinorTickValues = 0:10:1000;
ax.YAxis.MinorTickValues = 0:10:1000;
ax.XAxis.FontName = 'Times New Roman';
ax.YAxis.FontName = 'Times New Roman';
ax.XAxis.FontSize = 12;
ax.YAxis.FontSize = 12;
ax.XAxis.FontWeight = 'bold';
ax.YAxis.FontWeight = 'bold';
ax.XAxis.FontAngle='normal';
ax.XAxis.FontSmoothing = 'on';
ax.XAxis.FontSmoothing = 'off';
ax.XGrid=('on');
ax.YGrid='on';
ax.XMinorGrid="on";
ax.YMinorGrid="on";
```

```
title(legend({'Path','Starting          Point','Target          Point','Explored
Nodes'},'Position',[0.709895833333335    0.799259055982437    0.244270833333333
0.116081229418221],'NumColumns',2),'each black area considered as obstcals')
dubinsSpace       =       stateSpaceDubins([0,1000;0,1000;0,0]);pathobj       =
navPath(dubinsSpace);
append(pathobj,path);
for i = 20:length (path)
len(i)=sqrt(((x(i)-x(i-1))^2)+(((y(i)-y(i-1))^2)));
end
len=sum(len);
fprintf('- The shortest path regardless of the obstaclses= %g Unit of length\n-
The real path length = %g Unit of length\n- The number of explored nodes = %g\n-
The   number   of   the   selected   nodes   to   be   the   path   nodes   =   %g
only\n',eD,len,details.NumNodesExplored,length(path))
toc
```

A.3.2 Hybrid A* Algorithm

```
tic
map=occupancyMap(picBW);
setOccupancy(map, [xS,yS], zeros(length(plS),1),"grid")
setOccupancy(map, [xT,yT], zeros(length(plT),1),"grid")
inflate(map,1);
ss = stateSpaceSE2;
ss.StateBounds = [map.XWorldLimits;map.YWorldLimits;[-pi,pi]];
sv = validatorOccupancyMap(ss);
sv.Map = map;
planner = plannerHybridAStar(sv);
startPose = [Sc(1),1000-Sc(2),pi/2];
goalPose = [Gc(1),1000-Gc(2),-pi/2];
path = plan(planner,startPose,goalPose,SearchMode='greedy');
figure('Name','result',"WindowState","maximized")
show(planner)
legend('show');
title(legend('Position',[0.70546875 0.681942919868277 0.260416666666667
0.220087815587267],'NumColumns',1),'each black area (except the path)
considered as obstcals')
ax = gca;ax.XAxis.TickValues = 0:100:1000;ax.YAxis.TickValues = 0:100:1000;
ax.XAxis.Label.String = 'Unite of Length in X Axis';ax.YAxis.Label.String =
'Unite of Length in Y Axis';
ax.XAxis.TickDirection='both';ax.YAxis.TickDirection='both';
ax.XAxis.MinorTick = 'on';ax.YAxis.MinorTick = 'on';
ax.XGrid=('on');ax.YGrid='on';
ax.XMinorGrid="on";ax.YMinorGrid="on";
title ('Finding Pathplanning using Hybrid A* algorithm','
','Color','k','FontWeight','bold','FontName','Times New Roman','FontSize',16)
len = pathLength(path);
fprintf('the shortest path regardless the obstaclses= %g\n while the real path
length is = %g\nthe number of path nodes is %g\n',eD,len,path.NumStates)
toc
```

A.3.3 Probabilistic Road Map Algorithm

```
tic
map=occupancyMap(picBW);
setOccupancy(map, [xS,yS], zeros(length(plS),1),"grid")
setOccupancy(map, [xT,yT], zeros(length(plT),1),"grid")
inflate(map,6);
Probabilistic_Road_path= mobileRobotPRM;
Probabilistic_Road_path.NumNodes=4000;
Probabilistic_Road_path.ConnectionDistance=150;
Probabilistic_Road_path.Map=map;
Sc=[Sc(1),1000-Sc(2)];
Gc=[Gc(1),1000-Gc(2)];
path=findpath(Probabilistic_Road_path,Sc,Gc);
figure('Name','result',"WindowState","maximized")
show(Probabilistic_Road_path)
ax = gca;ax.XAxis.TickValues = 0:100:1000;ax.YAxis.TickValues = 0:100:1000;
ax.XAxis.Label.String = 'Unite of Length in X Axis';ax.YAxis.Label.String =
'Unite of Length in Y Axis';
ax.XAxis.TickDirection='both';ax.YAxis.TickDirection='both';
ax.XAxis.MinorTick = 'on';ax.YAxis.MinorTick = 'on';
ax.XGrid=('on');ax.YGrid='on';
ax.XMinorGrid="on";ax.YMinorGrid="on";
title ('Finding Pathplanning using PRM algorithm','
','Color','k','FontWeight','bold','FontName','Times New Roman','FontSize',16)
    pause(0.5);
title(legend({'connection between nodes','Exploring
nodes','path'},'Position',[0.711718750000001 0.784028540065863
0.192708333333333 0.115806805708013],'NumColumns',1),'each yellow area
considered as obstacles')
ax = gca;ax.XAxis.TickValues = 0:100:1000;ax.YAxis.TickValues = 0:100:1000;
ax.XAxis.Label.String = 'Unite of Length in X Axis';ax.YAxis.Label.String =
'Unite of Length in Y
Axis';ax.XAxis.TickDirection='both';ax.YAxis.TickDirection='both';
ax.XAxis.MinorTick = 'on';ax.YAxis.MinorTick = 'on';
ax.XGrid=('on');ax.YGrid='on';ax.XMinorGrid="on";ax.YMinorGrid="on";
title ('Finding Pathplanning using PRM algorithm','
','Color','k','FontWeight','bold','FontName','Times New Roman','FontSize',16)
flip(path);path(:,3)=1;dubinsSpace = stateSpaceDubins([0,1000;0,1000;-pi,pi]);
pathobj = navPath(dubinsSpace);
append(pathobj,path);
eD=sqrt(((Gc(1)-Sc(1))^2)+(((Gc(2)-Sc(2))^2)));len = pathLength(pathobj);
fprintf('the shortest path regardless the obstaclses= %g\n while the real path
length is = %g\nThe number of nodes = %g\nThe number of nodes that selected to
be path nodes =
%g\n',eD,len,Probabilistic_Road_path.NumNodes,pathobj.NumStates)
toc
```

A.3.4 Rapidly exploring Random Tree Algorithm

```
tic
map = occupancyMap(picBW,1);
setOccupancy(map, [xS,yS])
setOccupancy(map, [xT,yT])
sv.Map = map;
sv.ValidationDistance = 0.01;
ss.StateBounds = [map.XWorldLimits;map.YWorldLimits;[-pi pi]];
i=1;
j=1;
k=1;
planner =
plannerRRT(ss,sv,GoalBias=0.1,MaxNumTreeNodes=j,MaxConnectionDistance=i,MaxIter
ations=k);
start =[Sc(1),1000-Sc(2),0];
goal = [Gc(1),1000-Gc(2),0];
rng(10,"threefry");
[pthObj,solnInfo] = plan(planner,start,goal);
figure('Name','result',"WindowState","maximized")
show(map)
ax = gca;ax.XAxis.TickValues = 0:100:1000;ax.YAxis.TickValues = 0:100:1000;
ax.XAxis.Label.String = 'Unite of Length in X Axis';ax.YAxis.Label.String =
'Unite of Length in Y Axis';
ax.XAxis.TickDirection='both';ax.YAxis.TickDirection='both';
ax.XAxis.MinorTick = 'on';ax.YAxis.MinorTick = 'on';
ax.XGrid=('on');ax.YGrid='on';
ax.XMinorGrid="on";ax.YMinorGrid="on";
title ('Finding Pathplanning using RRT algorithm','
','Color','k','FontWeight','bold','FontName','Times New Roman','FontSize',16)
hold on
plot(solnInfo.TreeData(:,1),solnInfo.TreeData(:,2),'.-')
plot(pthObj.States(:,1),pthObj.States(:,2),'r-','LineWidth',2)
title(legend({'explored trees','path'},'Position',[0.709375000000001
0.817919868276619 0.1828125 0.0897365532381997],'NumColumns',1),'each black
area considered as obstcals')
len = pathLength(pthObj);
fprintf('the shortest path regardless the obstaclses= %g\nThe real path length
= %g\nThe number of explored nodes = %g\nThe number of nodes that selected to
be path nodes = %g\nThe minimum number of iteration =
%g\n',eD,len,solnInfo.NumNodes+1,pthObj.NumStates,planner.MaxIterations)
toc
```