Model Free Control of Geomagic OMNI Manipulator

By

Xu Zhang

Submitted in partial fulfillment of the requirements for

the degree of Master of Applied Science

at

Dalhousie University

Halifax, Nova Scotia

August, 2024

Table of Contents

List of Tables

List of Figures

Abstract

In this thesis, we explore the trajectory tracking control of the OMNI manipulator using various enhanced beetle bee algorithms. A metaheuristic algorithm mimics the beetle's ability to locate food in an unknown environment using its two antennas. Beetles decide to go left or right based on the strength of the scent until they reach their target. However, the conventional Beetle Antennae Search (BAS) takes a long to converge, especially when dealing with higher-dimensional systems. To address this issue, we propose improved beetle bee methods that combine other techniques with the original algorithm. The proposed approaches enhance the convergence speed of the algorithm and make it more efficient for higher dimensional systems.

The three proposed hybrid algorithms will be introduced in this thesis. The first one is adding the incremental PID control to the original algorithm. The second one combines the BSO (Beetle Swam Optimization) algorithm. The last one is counting the angular velocity to the objective. The simulation results from the proposed and cutting-edge metaheuristic algorithms will be compared. Moreover, the experimental results based on the Omni manipulator system with an improved Beetle Bee algorithm are also obtained. Experimental results demonstrate significant improvements in trajectory optimization compared to the original algorithm.

List of Abbreviations Used

DOF	Degree of Freedom
MIMO	Multi-Input and Multi-Output
PD	Proportional-Derivative
PID	Proportional-Integral-Derivative
PSO	Particle Swarm Optimization
CS	Cuckoo Search
GWO	Grey Wolf Optimizer
WOA	Whale Optimization Algorithm
BSO	Beetle Swarm Optimization
BAS	Beetle Antenna Search
HBO	Honey Bee Optimization
MS	Monkey Search
DPO	Dolphin Partner Optimization
FA	Firefly Algorithm
EE	End-Effector
D-H matrix	Denavit-Hartenberg matrix
SCC	Source Coordinate Center

Acknowledgements

I appreciate everyone who helped me complete the thesis during my MASc studies.

I want to thank my supervisor, Dr. Jason Gu, who accepted me as his MASC student and allowed me to work on this thesis. I couldn't find this research topic without his direction and guidance, so I worked on it under his continuous professional supervision and encouragement during my MASC study time at Dalhousie.

I am grateful to my fellow researchers at the Robotics laboratory. I mainly thank Dr. Umar Farooq, Dr. Muhammad Asad, PHD student Hanxiang Zhang, and Koceila Cherfouh for supporting my thesis.

I also want to thank two professors of my committee members, Dr. Ya-Jun Pan and Dr. Kamal El-Sankary, for their valuable suggestions.

Finally, I wanted to thank my parents for their support and understanding and my wife and two lovely sons for their belief and support during my study period. Without their love, I could not finish this thesis.

CHAPTER 1 Introduction

This chapter will overview the robotic manipulator system and introduce the geomagic OMNI manipulator. Additionally, we will discuss the motivation behind our thesis. Finally, we will outline the contributions of our thesis and the structure of the thesis itself.

1.1 Robot Manipulation Systems

Robots have made a substantial contribution to the industrial world in recent decades. The fields of robotics and artificial intelligence have gained significant attention, not only in daily life but also in manufacturing. In particular, robot manipulators, or robot arms, have proven to be very useful, replacing people in complicated and repetitive jobs [1-2]. They have been used to carry heavy objects, work in high-temperature, toxic, explosive, and radioactive environments, and complete dangerous and repetitive work instead of humans. This benefit has reduced the intensity of human labour while increasing productivity. The use of industrial robots increased worldwide from 1990 to 2000 because the cost of the robots decreased continuously while the labour cost increased [3]. There are several reasons why robots have been gaining significant attention. First, they reduce labour costs. When a robot arm is in operation, only one person is needed to control or monitor the robot manipulators, reducing the number of workers and labour costs. In addition, they are more reliable and secure when performing tasks. Since the robot arm imitates human actions to complete activities, there is no risk of injury or death in case of emergency, ensuring safety. Lastly, the accuracy of the product is crucial for factories. Robot arms can reduce the error rate of products compared to humans and stop when the pre-set goal is achieved, thus increasing production efficiency [4].

The use of robot manipulator systems has shown significant progress in human healthcare. One example is the da Vinci Surgical System, shown in Figure 1 [5]. The da Vinci Surgical robot arm has gained popularity in the medical field in recent decades. This phenomenon happens because it has dramatically improved surgery accuracy and reduced patient recovery time. Additionally, it causes a much smaller traumatic surface area, allowing patients to return to their daily lives quickly. For doctors, the system increases the field of view angle, reduces hand tremors, and reduces the number of participants required during the operation, increasing efficiency and reducing labour costs. Researchers have also put effort into improving the system. Some have created a camera-robot calibration to perform automated tasks with precision and minimize accumulation error [6]. Others have used a deep learning approach to create a neural network that estimates interaction force based on the da Vinci System [7]. Yet others have designed a fuzzy dynamic surface controller on the Da Vinci System to observe and eliminate uncertainties [8].

Figure 1: The Da Vinci Surgical System

Nowadays, robot arms are playing a crucial role in the industrial field. As the Fig. 2 [9] shows.

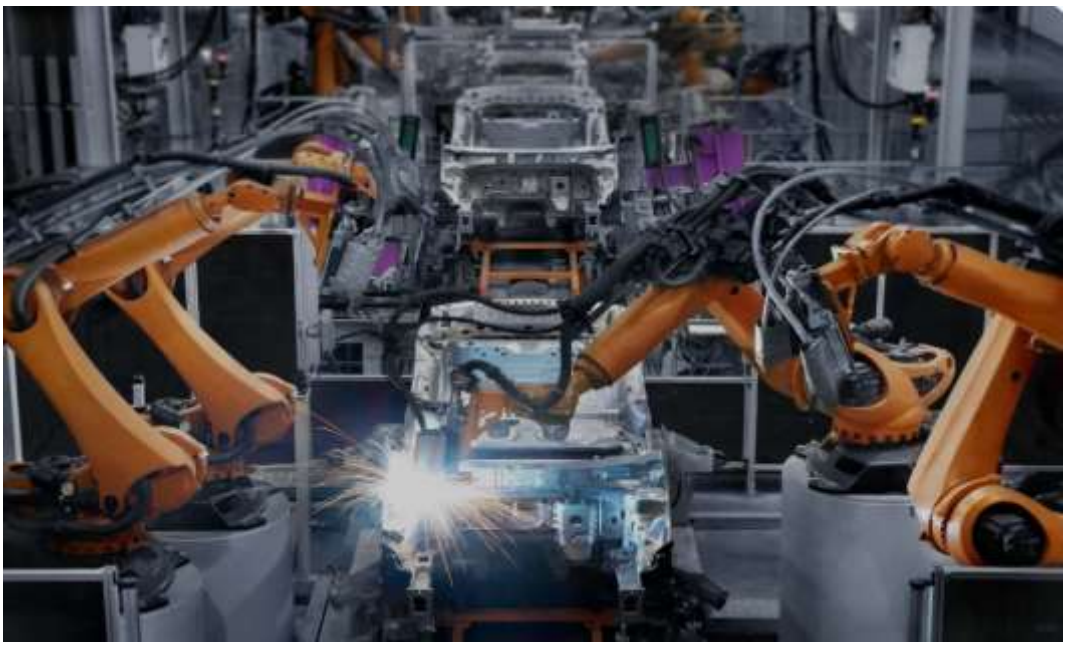

Figure 2: Example of the Industrial Robot Arm

Industrial robot manipulators can do different tasks, such as Metal processing, polishing and grinding, assembly, machine loading and unloading, palletizing/transportation, rubber/plastic, sorting, etc. Therefore, researchers conducted a study on robot control of the industrial robot arm to maximize accuracy [10]. Other researchers have used the industrial KUKA robot arm's dynamic control and identification platform to create open-source software for education and research purposes [11].

The robot arm comprises several links joined together by joints that actuators or motors can move. The joints can be revolute or prismatic, and the links can be rigid or flexible. A common way to categorize robot manipulators is by the number of freedoms (DOFs) determined by the number of joints [12]. The three examples will be shown as follows.

The Fig. 3 [13] shows a 2-DOF manipulator.

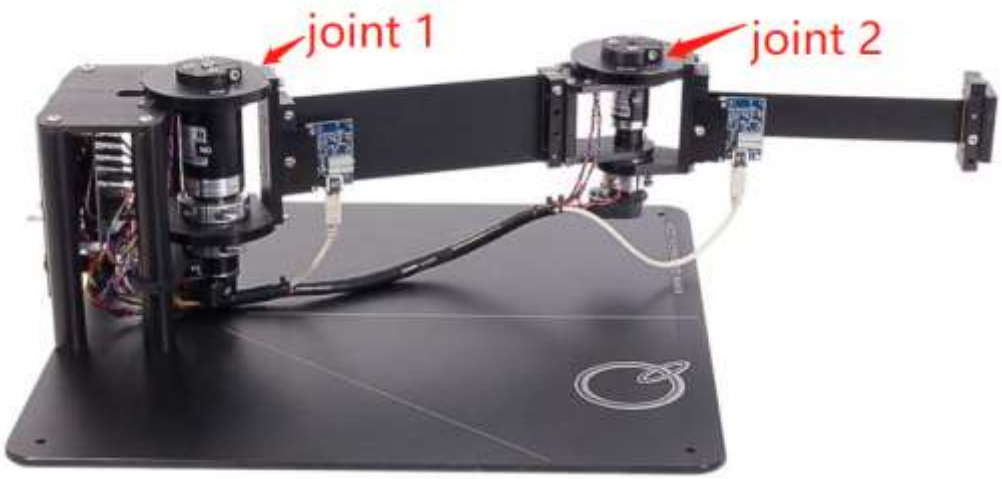

Figure 3: 2-DOF Robot Arm

Moreover, the 3-DOF arm is shown in Fig. 4 [14].

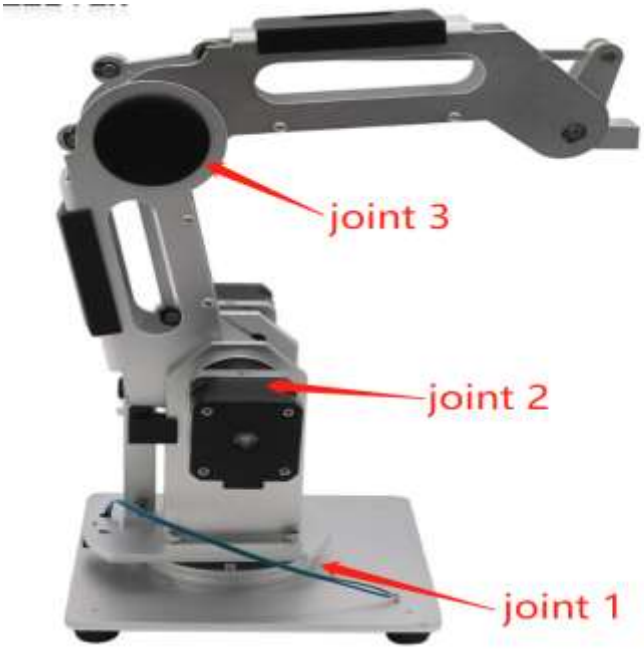

Figure 4: 3-DOF Robot Arm

Fig. 5 [15] shows a 4-DOF arm.

Figure 5: 4-DOF Robot Arm

1.2 The Geomagic OMNI Manipulator System

There are two main types of robotic manipulators: serial manipulators and parallel manipulators. Serial manipulators typically comprise links connected end-to-end, forming a kinematic chain controlled by different joints [16]. An example of a serial manipulator is the OMNI manipulator, shown in Figure 6 [17].

Figure 6: OMNI Manipulator

The OMNI manipulator consists of six joints, three of which are actuated and three that are non-actuated. Researchers are particularly interested in the three actuated joints. In Chapter 3, we will discuss the modelling of the OMNI manipulator, including the forward kinematic model of position, inverse kinematic model of position, forward kinematic model of velocity, and inverse model of velocity, in detail.

Also, the three actuated joints have a limited range, which will be discussed later.

1.3 Thesis Motivation

Robots have made a significant contribution to the industrial world today. They can effectively replace humans in dangerous situations and perform complicated and repetitive tasks. However, more accurate location monitoring and improved metaheuristic algorithm control methods must be developed to meet the required high-level demands and criteria. Robots' challenges in achieving their intended performance include time-varying variables, system uncertainties, nonlinearities, and

typical coupling effects [18]. A robot manipulator is a multi-input and multi-output (MIMO), a highly nonlinear and coupled system designed to perform tasks automatically, mimicking or reproducing human actions in a specific area [19].

Robot manipulator controllers still frequently use traditional proportional-derivative (PD) or proportional-integral-derivative (PID) algorithms despite the current control theory's effectiveness [20]. Advanced control methods, such as feedforward compensation control methods [21], computed torque methods [22], and nonlinear feedback control [23], have been developed to solve optimization problems. However, it is challenging to implement these methods in real-time due to the system's nonlinearities and uncertainties, making it challenging to find the robot's mathematical model. Recently, natural-inspired metaheuristic algorithms have gained popularity for resolving optimization issues. The social behaviour of living things in their natural habitats influenced the development of various metaheuristic algorithms, such as Particle Swarm Optimization (PSO) [24], Cuckoo Search (CS) [25], Grey Wolf Optimizer (GWO) [26], and Whale Optimization Algorithm (WOA) [27]. The reasons for the popularity of nature-inspired optimization algorithms can be summarized into three main parts: simplicity, flexibility, and the ability to avoid local minima.

Meta-heuristics are optimization algorithms based on animal and evolutionary concepts, making them easy to understand and use. They are also adaptable, which means they can solve various problems without requiring complex adjustments. Additionally, they can avoid getting stuck in local minima due to their unpredictable search behaviour.

However, although meta-heuristic algorithms can potentially solve most real-world optimization problems, the No Free Lunch theorem [28] states that no single

algorithm can handle all issues. Some algorithms perform better than others in specific class problems, while others excel in other situations. This is why we need distinct meta-heuristic algorithms for real-time issues instead of relying on a single one.

Metaheuristic algorithms have shown promising results in solving complex optimization problems in various fields. Therefore, applying these algorithms in the development of robot manipulators presents an opportunity to improve the efficiency and effectiveness of these machines.

This thesis aims to explore the application of metaheuristic algorithms, specifically beetle bee algorithms and particle swarm optimization, to enhance their tracking performance. The research will involve designing and simulating a robotic manipulator utilizing these algorithms and comparing the results with those obtained by the original beetle bee algorithm.

Overall, this thesis aims to contribute to developing an OMNI manipulator that can improve tracking performance efficiently and effectively, thereby enhancing the capabilities of these machines and their potential applications in various industries.

1.4 Contributions

As discussed above, this thesis will improve the tracking performance of the original beetle bee algorithm. The three contributions will be introduced as follows to achieve better performance compared to the original algorithm.

- The first optimization applied the incremental PID control to the step size of the proposed algorithm.
- The second optimization algorithm is combined with Beetle Swam Optimization (BSO) to achieve better tracking performance for the OMNI

manipulator.

- The last optimization is adding the angular velocity square to the objective function to achieve a more stable and faster convergence speed.

1.5 Thesis Outline

This thesis proposes three Improved Beetle Bee Algorithms with Application to OMNI Manipulator Trajectory Tracking Control. The rest of the paper is structured as follows.

Chapter 1 provides an introductory overview of the robot manipulator system, with a specific focus on the OMNI manipulator. It outlines the motivation behind the research, details the contributions of the thesis, and presents the overall structure and organization of the work.

Chapter 2 offers a comprehensive review of existing literature, covering both conventional and intelligent control algorithms relevant to robot manipulators. The review aims to contextualize the research within the broader field and highlight significant advancements and methodologies.

Chapter 3 introduces the fundamental theories pertinent to the study, including robotics methodologies and the kinematic and dynamic models specific to the OMNI manipulator.

Chapter 4 addresses the development of the improved algorithm, detailing the three key optimizations incorporated within it. The discussion focuses on the theoretical underpinnings and practical implications of these optimizations.

Chapter 5 presents the results obtained from simulations, including a detailed analysis of the methods employed and the outcomes for each case studied. The results are discussed in terms of their implications and contributions to the field.

Chapter 6 summarizes the key findings of the thesis, drawing conclusions from the research conducted. Recommendations for future research directions are offered, highlighting potential areas for further investigation and development.

CHAPTER 2 Literature Review

This chapter briefly overviews classical and intelligent control algorithms, focusing on the Beetle Antenna Search (BAS).

2.1 A State-of-the-Art Control Algorithms

Robot manipulator controllers often still need to rely on traditional proportional-derivative (PD) or proportional-integral-derivative (PID) algorithms despite the availability of newer and more effective control theory methods [29]. Advanced control methods, such as feedforward compensation control methods [30], computed torque methods [31], and nonlinear feedback control [32], are used to solve optimization problems.

2.2 Intelligent Control Algorithms

Finding the mathematical model of a robot can be a complex task due to the nonlinearities and uncertainties involved in the system. Therefore, the methods mentioned above may not be feasible in real-time situations. However, natural-inspired metaheuristic algorithms have recently gained significant attention for addressing optimization problems. These algorithms, such as Honey Bee Optimization (HBO) [33], Monkey Search (MS) [34], Dolphin Partner Optimization (DPO) [35], and Firefly Algorithm (FA) [36], are developed by observing the social behaviour of living organisms in their natural habitats. The popularity of these nature-inspired optimization algorithms can be attributed to their simplicity, flexibility, and ability to avoid local minima.

Meta-heuristics have some significant advantages. Firstly, they are based on animal and evolutionary concepts, making them easy to understand and implement. Secondly, these optimization algorithms are adaptable, meaning they can be applied

to many problems without significant modifications. Thirdly, they can avoid getting stuck in local minima due to their unpredictable search behaviour.

However, even though meta-heuristic algorithms have the potential to address most real-world issues, it's important to note that no single algorithm can handle all optimization problems. The No Free Lunch theorem states that some algorithms might perform better than others in specific optimization problems, while others might be more effective in other issues. Therefore, it's necessary to have a variety of meta-heuristic algorithms to tackle real-time problems.

The Beetle Bee Algorithm is an intelligent control algorithm that mimics the food-searching nature of beetles. Unlike other insects, beetles don't work in swarms and can hunt for food independently, reducing computational complexity and time consumption. The algorithm uses two antennae to detect the intensity of the food's scent and compare it to determine the new movement direction for the following step. By sensing the difference in smell at each antenna, the beetle can develop a map of the smell intensity of the unknown environment. This map helps search for maximum smell change, which is subsequently moved toward the destination position (food source). The Beetle Antennae Search (BAS) algorithm [37] is another name for the Beetle Bee Algorithm. BAS has been used in various real-world systems since its working process [37-38] was introduced. The working process of the original BAS is shown in Fig. 7. [39].

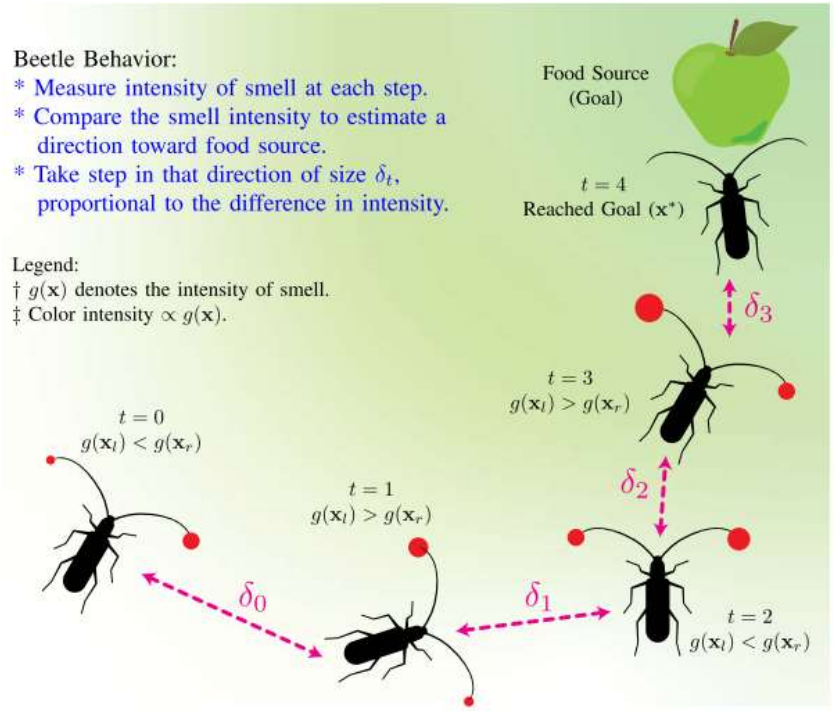

Figure 7: Illustration of the beetle's food-foraging behaviour

Firstly, the random search behaviour should be implemented as in (1),

$$\vec{b} = \frac{rands(k,1)}{\|rands(k,1)\|} \tag{1}$$

Where $rands(.)$ denotes a random function, which is a general random function to get the random value to guarantee the random search behaviour, and k represents the dimensions of the system, the "$\|.\|$" denotes the absolute value.

Since we have this random searching behaviour combined with the beetle antennae length, which shows a later function that should be sufficiently large to encompass an appropriate search region to be capable of jumping out of local minimum points so that we can avoid the local minima problem, and this feature is essential so that the robotic task can under the desired working trajectory.

Secondly, the algorithm needs to calculate the objective function values for both the left and right antennae, which are prepared to update the new angle for the next step.

Thirdly, the updated angle is used to get the new value of the objective function, and the location is updated to control the beetle and follow the desired trajectory.

CHAPTER 3 Background Theories

This chapter will introduce the background theories of the robot manipulator. This chapter includes three parts:

- Robotics terminology
- Kinematic model of robot manipulators
- Modelling of OMNI robot manipulators

3.1 Robotics Terminology

To ensure a robotic manipulator's safety, effectiveness and efficiency, it is essential to follow a methodology before designing and building the robot. The term robot in this research refers specifically to the robot manipulator. The methodology can be defined as the terminology used in this field. Robotic manipulators consist of links joined by joints, forming a kinematic chain. The robot system includes a manipulator, arm, wrist, end-effector, actuators, sensors, and controllers. Each rigid body of the robot is called a link, and two links are connected by a joint, which can be revolute (rotary) or prismatic (translatory). A manipulator is a significant robot component consisting of links, joints, and other elements. The wrist is the point in the robot's kinematic chain between the forearm and the end-effector. The end-effector is mounted on the last link and is responsible for performing the required work of the robot manipulator or arm. Actuators, which operate as drivers, are like the muscles of a robot that change its configuration. Sensors detect and collect information from the reaction between the end-effector, objects, and environment. Lastly, a controller acts like the brain of a human being, enabling the robot to perform its tasks effectively and efficiently.

3.2 Kinematic Model of Robot Manipulators

The kinematic model of robot manipulators primarily focuses on object motion rather than the force generated by movement. In robot kinematics, we study the higher-order differentiation of position, velocity, acceleration, and position variables concerning time or other variables. The typical research topics in robot kinematics are forward kinematics and inverse kinematics.

Generally speaking, forward kinematics determines the end-effector's position in the coordinates by providing a set of joint angles. It is utilized to find the position of each arm for the given joint and link parameters. The forward kinematics calculation uses the Denavit-Hartenberg (D-H) matrix [40], demonstrating the relationship among frames. There are four DH parameters: joint offset d, joint angle θ, link length a, and twist angle α. The first two parameters d_i and θ_i indicate the relative position of link $i-1$ and link i, whereas the other two parameters a_i and α_i define the size and shape of link i.

After reading the four D-H parameters for each link, the D-H table is generated., The transformation matrix can be calculated with the formula:

$$
{}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}
$$

The position change from frame $i-1$ to frame i can be calculated by multiplying the corresponding transformation matrix,

$$
[{}^{i-1}P]_{3\times1} = [{}^{i-1}_{i}T]_{3\times3}[{}^{i}P]_{3\times1} \tag{3}
$$

Multiplication of all transformation matrices can be applied to determine an overall transformation matrix,

$$ {}^{0}_{i}T = {}^{0}_{1}T\,{}^{1}_{2}T \cdots {}^{i-2}_{i-1}T\,{}^{i-1}_{i}T \quad (4) $$

An example is made by calculating the forward kinematics for a 2 DOF planar arm robot.

The first step is to assign all three labels for each robot joint. Following the rules that the Z axis is the rotation axis and applying the right-hand rule, the axes are given below in Fig. 7.

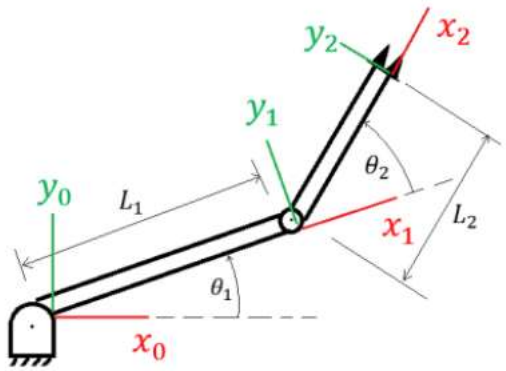

Figure 8: 2R Robot with XYZ Labels for Each Joint Assigned

The D-H Table shown in Table 3 can be constructed by observing Figure 7. Note that frame $\{0\}$ is attached to the base so that α_0 =0 and a_0 =0.

Table 1 Denavit-Hartenberg Table for Robot Manipulator

Link i	α_i	a_i	d_i	θ_i
1	0	L_1	0	θ_1
2	0	L_2	0	θ_2

Next, substitute the values into the formula for calculating the transformation matrix. The following equations can be obtained where,

$$ {}^{0}_{1}T = \begin{bmatrix} & {}^{0}_{1}R & & {}^{0}_{1}Q \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & L_1 c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & L_1 s\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

$$ {}^{1}_{2}T = \begin{bmatrix} & {}^{1}_{2}R & & {}^{1}_{2}Q \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & L_2 c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & L_2 s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6} $$

$$ {}^{0}_{2}T = {}^{0}_{1}T\,{}^{1}_{2}T = \begin{bmatrix} c\theta_{12} & -s\theta_{12} & 0 & L_1 c\theta_1 + L_2 c\theta_{12} \\ s\theta_{12} & c\theta_{12} & 0 & L_1 s\theta_1 + L_2 s\theta_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7} $$

Thus, the Cartesian position and orientation of the last joint concerning the base frame {0} can be represented by the following rotation matrix and translation matrix.

$$ {}^{0}_{2}R = \begin{bmatrix} c\theta_{12} & -s\theta_{12} & 0 \\ s\theta_{12} & c\theta_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix}, {}^{0}_{2}Q = \begin{bmatrix} L_1 c\theta_1 + L_2 c\theta_{12} \\ L_1 s\theta_1 + L_2 s\theta_{12} \\ 0 \end{bmatrix} \tag{8} $$

As for the inverse kinematics, which gives the position of the end-effector to find the corresponding joint angles, The inverse kinematics is used to find all joint parameters for a given end-effector, which contains the position information of the last joint with respect to the base. However, unlike the forward kinematic, which has a unique solution, inverse kinematics can have multiple solutions. This can be incredibly complicated for robots with a high degree of freedom.

For a 2 DOF robot, an inverse kinematic may have two solutions. In other words, two different situations of arm position have the same end effector, as shown in Fig. 9. Assume the hip joint is located at (x, y)

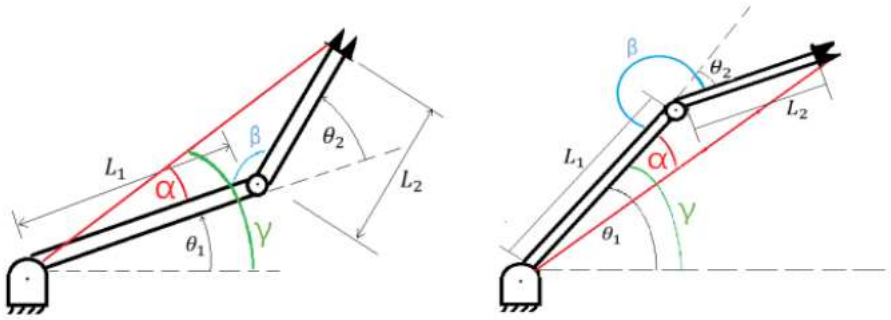

Figure 9: Two Solutions for an Inverse Kinematics

The relationship between angles α, β, γ, and two joint angles can be observed as follows,

Solution 1 (Fig. 9 Left):

$$\theta_1 = \gamma - \alpha, \theta_2 = \pi - \beta \tag{9}$$

Solution 2 (Fig. 9 Right):

$$\theta_1 = \gamma + \alpha, \theta_2 = \beta - \pi \tag{10}$$

The angle γ can be calculated using arctangent,

$$\gamma = a\tan 2(y, x) \tag{11}$$

The angles α and β can be calculated using the law of cosines,

$$\begin{aligned} L_1^2 + L_2^2 - 2L_1L_2\cos\beta &= x^2 + y^2 \\ x^2 + y^2 + L_1^2 - 2L_1\sqrt{x^2 + y^2}\cos\alpha &= L_2^2 \end{aligned} \tag{12}$$

To simplify these, we have,

$$\alpha = \cos^{-1}\frac{x^2 + y^2 + L_1^2 - L_2^2}{2L_1\sqrt{x^2 + y^2}} \tag{13}$$

$$\beta = \cos^{-1}\frac{L_1^2 + L_2^2 - x^2 - y^2}{2L_1L_2} \tag{14}$$

3.3 Modelling of OMNI Robot Manipulators

The OMNI manipulator shown in Fig. 10 is a 3-DOF device that allows kinematic integration with complex virtual objects. The haptic device kinematics includes the

position and derivates to evaluate the device's performance by implementing the proposed improved Beetle Bee Algorithm.

The modelling of the OMNI manipulator includes the forward kinematic model of position, inverse kinematic model of position, forward kinematic model of velocity, and inverse model of velocity.

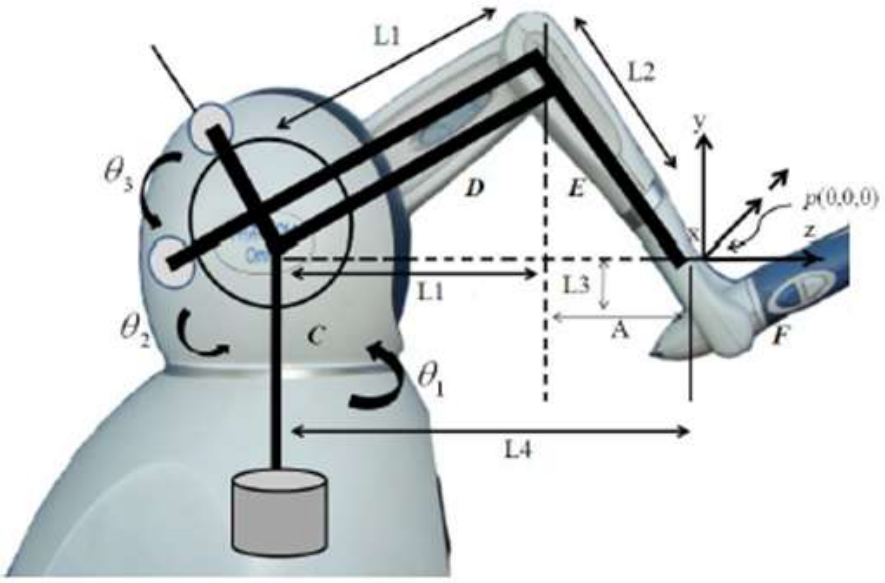

Figure 10: Modelling of OMNI Manipulator

1) Forward Kinematic Model of Position

The forward kinematic position of a manipulator is determined by the relationship between the Cartesian coordinates, which represent the position and orientation of the end-effector, and the joint space coordinates, which correspond to the joint angles. This relationship is expressed mathematically, as shown in equation (15),

$$x = f(\theta) \tag{15}$$

Where $x \in \mathbb{R}^{3\times1}$ denotes the vector of the Cartesian coordinates of the end-effector and $\theta \in \mathbb{R}^{3\times1}$ represents the vector of joint coordinates. The operational coordinates for X, Y, and Z coordinates of real-time values can be calculated from the kinematics

chain of the OMNI haptic device, and the needed variables and constants can be found in Fig. 10.

The kinematic chain of the OMNI haptic device, along with the representation of the variables and constants involved in the model, is illustrated in Fig. 10. The variables and constants are all shown in Fig. 10. The L_1 is the length of the first link, L_2 is the length of the second link, $L_1 = L_2 = 0.135$m, and A = 0.035m when L_1 and L_2 are perpendicular shows in Fig. 10, $L_4 = L_1$+A. L_3=0.025m. These parameters will be used in the calculation of the kinematic model. Calculating the kinematic chain is the position of the End-Effector (EE) from the Source Coordinate Center (SCC) to the EE of the manipulator. After performing the coordinate transformation, also known as the kinematics chain from the point (0,0,0) in Fig. 10.

The final representation of the position vector of Cartesian coordinates for the forward kinematic model is shown in (16):

$$\begin{aligned} x &= -\sin\theta_1(L_1\cos\theta_2 + L_2\sin\theta_3) \\ y &= L_3 - L_2\cos\theta_3 + L_1\sin\theta_2 \\ z &= -L_4 + \cos\theta_1(L_1\cos\theta_2 + L_2\sin\theta_3) \end{aligned} \tag{16}$$

Where θ_1, θ_2, θ_3 are the joint angles. The range of θ_1 is from $-50°$ to $55°$, θ_2 is from 0 to $105°$, and θ_3 is not in a specific range which depends on the value of θ_2 [14]. Therefore, table 2 shows the corresponding angle values with respect to the θ_2. All angles are in degrees.

Table 2 RELATIVE VALUES OF θ_3

θ_2	$\theta_{3\,minimum}$	$\theta_{3\,maximum}$
0	-20	65
15	-15	90
30	-9	105
40	0	110
50	10	112
60	20	113

80	40	114
90	50	114
105	60	110

2)Inverse Kinematic Model of Position

The inverse kinematic model of the manipulator is the computation of the given EE position to find the corresponding angle in real time. The mathematical relation is shown in (17),

$$\theta = f^{-1}(x) \tag{17}$$

3)Forward Kinematic Model of Velocity

The forward kinematic model of velocity is defined in (18):

$$\dot{x} = J\dot{\theta} \tag{18}$$

Where $\dot{x} \in \mathbb{R}^{3\times1}$ represents the velocities vector for each joint, $J \in \mathbb{R}^{3\times3}$ denotes the Jacobian matrix of the manipulator and $\dot{\theta} \in \mathbb{R}^{3\times1}$ is the joint velocities vector. After the deformation of (3) and rearranging into the matrix form [40], it becomes in (19):

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{pmatrix} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{pmatrix} \tag{19}$$

Where, $J_{11} = -(L1cos\theta_1 cos\theta_2 + L2sin\theta_3 cos\theta_1)$,

$J_{12} = L1sin\theta_1 sin\theta_2$,

$J_{13} = -L2cos\theta_3 sin\theta_1$,

$J_{21} = 0$,

$J_{22} = L1cos\theta_2$,

$J_{23} = L2sin\theta_3$,

$J_{31} = -L1cos\theta_2 sin\theta_1 - L2sin\theta_3 sin\theta_1$,

$J_{32} = -L1sin\theta_2 cos\theta_1$,

$J_{33} = L2cos\theta_3 cos\theta_1$.

4)Inverse Kinematic Model of Velocity

The inverse kinematic model of velocity is defined in (20):

$$\dot{\theta} = J^{-1}\dot{x} \tag{20}$$

Where $J^{-1} \in \mathbb{R}^{3\times3}$ represents the inverse Jacobian matrix and denoted in (21):

$$J^{-1} = \frac{adj(J)}{\det(J)} \tag{21}$$

Where $adj(J) \in \mathbb{R}^{3\times3}$ define the adjoint matrix and $det(J)$ denotes its determinant, which is described in (22):

$$\begin{aligned}\det(J) = -L_1 L_2 (&L_1 \cos\theta_2 \sin\theta_2 \sin\theta_3 \\ &+ L_1 \cos^2\theta_2 \cos\theta_3 + L_2 \sin\theta_2 \\ &- L_2 \sin\theta_2 \cos^2\theta_3 \\ &+ L_2 \sin\theta_3 \cos\theta_2 \cos\theta)\end{aligned} \tag{22}$$

And is equation being under a condition of existence $\theta_3 \neq \theta_2 + \pi/2$ which will guarantee the configuration of haptic device is in the nonsingular space.

CHAPTER 4 Improved Algorithms

This chapter will describe three proposed algorithms that can contribute to the original algorithm.

4.1 Improved Algorithm with Incremental PID Control

This part will discuss the first improved algorithm for trajectory-tracking circular and rectangular curves.

The original Beetle Antennae Search utilized quadratic optimization to minimize location tracking error, representing the difference between the reference path and the actual trajectory. Equation (23) expresses the optimization in position error as follows:

$$\begin{aligned} &\min_{\theta} g(x_r(t), \theta(t)) = e^T e \\ &s.t.\ \theta^- < \theta < \theta^+ \end{aligned} \tag{23}$$

Where $\theta^- = [\theta_1^-,\ \theta_2^-, \theta_3^-,\ \ldots \theta_m^-]^T$ denotes the minimum angle of each joint and $\theta^+ = [\theta_1^+,\ \theta_2^+, \theta_3^+,\ \ldots \theta_m^+]^T$. The tracking error e is defined as the position error between the reference curve and the real position of the end-effector. The final objective function in was using the common performance assessment criteria called integral square error (ISE), the performance of convergence was not fast enough.

In digital processors, two types of discrete PID are commonly used: positional PID and incremental PID [41]. Incremental PID is preferred over positional PID as it requires fewer memory units and calculators to store errors and parameters. The method is more straightforward, requires fewer parameters, and has a concise calculation method [42]. As the abstract states, incremental PID control adjusts the step length during each iteration. It replaces the cumulative effect by finding the increment, reducing computing performance and storage space requirements. The following are the significant steps of the proposed algorithm.

A. Step 1

Calculate the endpoint of both antennae and project them into the constrained space corresponding to the joint angle limit shown in (24).

$$\begin{aligned} \theta_L &= P(\theta_k + \lambda_k \vec{b}) \\ \theta_R &= P(\theta_k - \lambda_k \vec{b}) \end{aligned} \tag{24}$$

P(..) is the projection function and puts the endpoint of both antennae to the constrained condition concerning the objective function. And θ_L and θ_R represents projected the right and left beetle antennas. λ_k is a hyperparameter that denotes the length of the beetle antennas. θ_k is the theta value at the time t_k.

B. Step 2

Based on Step 1, we can use forward kinematics to calculate the end-effector position of both directions, which is determined by Equation 2. Then, we will find the objective function value for g_L and $g_R$which are prepared to find the new updated theta value θ_{new}, the mathematical expression is showed (25).

$$\theta_{new} = P(\theta_k - \delta_k(\lambda_k) sign(g_L - g_R)\vec{b}) \tag{25}$$

where θ_{new} is the new updated location of the end-effector. And $sign(g_L - g_R)\vec{b}$ ensures that the beetle moving direction is the small objective function value between both directions. $\delta_k(\lambda_k)$ stands for the step size. Before introducing the incremental PID control rule adjusting the step size, the idea of basic PID control law should be clarified as

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int e(t)dt + T_d \frac{de(t)}{dt}] \tag{26}$$

Where K_p is the proportional gain, T_i is the integral time constant, and T_d is the derivative time constant.

As computer control is a form of sampling control, it can only compute the control quantity based on the variation of the sampling time. It cannot continuously output the control quantity like analogue control. Because of this, we need to discretize the integral and differential terms in (26). Here are some approximations of the transformations:

$$\int e(t) \approx T\sum_{i=0}^{k} e_i, \quad \frac{de(t)}{dt} \approx \frac{e_k - e_{k-1}}{T} \tag{27}$$

Where k is the sampling time, and T is the sampling period. Then we put (27) into (26) to get the discretized PID expression in (28).

$$u_k = K_p(e_k + \frac{T}{T_i}\sum_{i=0}^{k} e_i + T_d \frac{e_k - e_{\mathrm{k-1}}}{T}) \tag{28}$$

Where u_k is the output at the k_{th} sampling time. Then, the output of the k-1 sampling time is written as

$$u_{k-1} = K_p(e_{k-1} + \frac{T}{T_i}\sum_{i=0}^{k-1} e_i + T_d \frac{e_{k-1} - e_{k-2}}{T}) \tag{29}$$

Use (28) to subtract (29) and rearrange to get the final incremental PID control rule:

$$\begin{aligned} \Delta u_k &= u_k - u_{k-1} \\ &= \mathrm{K}_p(e_k - e_{k-1}) + \mathrm{K}_i(e_k) + \mathrm{K}_d(e_k - 2e_{k-1} + e_{k-2}) \end{aligned} \tag{30}$$

Where K_p is the proportional coefficient, $\mathrm{K}_i = (\mathrm{K}_p T / T_i)$ is the integral coefficient, and $\mathrm{K}_d = ((\mathrm{K}_p T_d)/T)$ is the derivative coefficient. Resultantly, the incremental PID control is defined as

$$u_k = u_{k-1} + \Delta u_k \tag{31}$$

Consequently, incremental PID control applies to the step size δ_k in the proposed algorithm can be obtained as

$$\delta_k = \delta_{k-1} + \Delta\delta_k \tag{32}$$

C. Step 3

The third step is to use the new theta value θ_{new} to find the new objective function value g_{new}, and based on this to compare with objection value $g(x(t), \theta_k)$ from the last time constant theta value θ_k, the relation is given in (33):

$$\theta_{k+1} = \begin{cases} \theta_k, & if\ g_{new} \geq g(x(t), \theta_k) \\ \theta_{new}, & if\ g_{new} < g(x(t), \theta_k) \end{cases} \tag{33}$$

As long as the robot arm's position has been updated. The whole process will be repeated.

The improved algorithm has incorporated incremental PID control to the step size in each iteration to make the tracking progress more stable and controllable. As a result, the algorithm's convergence performance has significantly improved. In the next chapter, we will discuss the simulation results of both the original and improved algorithms in detail.

4.2 Improved Algorithm with BSO

This section will introduce the Beetle Swarm Optimization (BSO) algorithm. The BSO algorithm is derived from the classical Particle Swarm Optimization (PSO) technique, a computational optimization method used to solve practical control optimization problems [43]. It was initially proposed by Kennedy and Eberhart in 1995 [44].

Beetle Swarm Optimization is a metaheuristic algorithm inspired by the swarming behaviour of beetles. The algorithm involves a swarm of beetles, each capable of conducting exploration and exploitation. Each beetle represents a potential solution to the optimization problem and works collaboratively with other beetles to share

information. This collaborative approach enhances the stability of the optimization process and reduces the chances of falling into local minima.

From the idea of the Particle Swam Optimization, there is a swam population of n beetles and $X = (X_1, X_2, ..., X_n)$ represents the beetles in the S-dimensional space, where i th beetle denotes as $X_i = (X_{i1}, X_{i2}, ..., X_{iS})$ which means the position of i th beetle in the S-dimensional space, which also is a potential solution to the problem. In addition, the speed of the i th beetle is represented as $V_i = (V_{i1}, V_{i2}, ..., V_{iS})$. The individual best of the beetle is expressed as $P_i = (P_{i1}, P_{i2}, ..., P_{iS})$ and the global best of swam is represented as $P_g = (P_{g1}, P_{g2}, ..., P_{gS})$. The mathematical expression for updating the position of the beetle is:

$$X_{is}^{k+1} = X_{is}^{k} + \lambda V_{is}^{k} + (1-\lambda)\xi_{is}^{k} \tag{34}$$

Where $s = 1, 2, ..., S$; $i = 1, 2, ..., n$; k is the current number of iterations. V_{iS}^{k} is the speed of beetles, and ξ_{iS}^{k} denotes the increase in beetle position movement. λ is a positive constant.

The speed expression which in (34) is:

$$V_{is}^{k+1} = \omega V_{is}^{k} + c_1 r_1 (P_{is}^{k} - X_{is}^{k}) + c_2 r_2 (P_{gs}^{k} - X_{gs}^{k}) \tag{35}$$

Where c_1 and c_2 are two positive constants. r_1 and r_2 are two random functions in the range [0,1]. ω is the inertia weight. In the standard PSO algorithm, ω is a constant value. However, for this proposed algorithm, a decreasing inertia weight rule for the ω showed below:

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{K} * k \tag{36}$$

Where $\omega_{\min}$ and $\omega_{\max}$ denotes the minimum and maximum value of the inertia weight ω. K is the maximum of iterations, k is the current number of iterations. In this algorithm, the 0.9 set to $\omega_{\max}$ and 0.2 set to $\omega_{\min}$. In this case, the algorithm can search an extensive range at the beginning of the search process and find the optimal solution as fast as possible.

And for the ξ expression which defines the incremental function:

$$\xi_{is}^{k+1} = \delta^k * V_{is}^k * sign(f(X_{ls}^k) - f(X_{rs}^k)) \tag{37}$$

The searching behaviour of the right and left antenna are represented as:

$$\begin{aligned} X_{rs}^{k+1} &= X_{rs}^k - V_{is}^k * \frac{d}{2} \\ X_{ls}^{k+1} &= X_{ls}^k + V_{ls}^k * \frac{d}{2} \end{aligned} \tag{38}$$

The BSO algorithm improves trajectory tracking speed and convergence. The simulation results will be discussed in a later chapter.

4.3 Improved Algorithm with Joint Angle Velocity

In this part, the last improvement will be introduced.

The original Beetle Antennae Search was the quadratic optimization in position tracking error [40]. The expression of optimization in position error is shown below:

$$\begin{aligned} &\min_{\theta} g(x_r(t), \theta(t)) = e^T e \\ &s.t. \theta^- < \theta < \theta^+ \end{aligned} \tag{39}$$

Where $\theta^- = [\theta_1^-,\ \theta_2^-, \theta_3^-,\ \ldots\ ,\ \theta_m^-]^T$ denotes the minimum angle of each joint and $\theta^+ = [\theta_1^+,\ \theta_2^+, \theta_3^+,\ \ldots\ ,\ \theta_m^+]^T$. The tracking error e is defined as the position error between the reference curve and the real position of the end-effector. The final objective function in was using the common performance assessment criteria called Integral square error (ISE), the performance of convergence was not fast enough.

The improved Beetle Bee Algorithm is designed as below:

$$\begin{aligned} &\min_{\theta} g(x_r(t),\theta(t)) = e^T e + \dot{\theta}^T \dot{\theta} \\ &s.t. \theta^- < \theta < \theta^+ \\ &\dot{\theta}^- < \dot{\theta} < \dot{\theta}^+ \end{aligned} \tag{40}$$

Where $\dot{\theta}$ is the joint angle velocity. $\dot{\theta}^- = [\dot{\theta}_1^-,\ \dot{\theta}_2^-, \dot{\theta}_3^-,\ \ldots\ ,\ \dot{\theta}_m^-]^T$ demotes the minimum angular velocity at each joint and $\dot{\theta}^+ = [\dot{\theta}_1^+,\ \dot{\theta}_2^+, \dot{\theta}_3^+,\ \ldots\ ,\ \dot{\theta}_m^+]^T$ stands for maximum angular velocity at each joint.

Three significant steps for the proposed algorithm will be introduced as follows.

Step 1 involves projecting the endpoint of beetle antennae onto a constrained space that corresponds to the joint angle limit, as shown below:

$$\theta_L = P(\theta_k + \lambda_k \vec{b}),\ \ \theta_R = P(\theta_k - \lambda_k \vec{b}) \tag{41}$$

P(..) is the projection function and puts the endpoint of both antennae to the constrained condition concerning the objective function. And θ_L and θ_R represents projected the right and left beetle antennas. λ_k is a hyperparameter that denotes the length of the beetle antennas. θ_k is the theta value at the time t_k.

Based on step 1, we can use forward kinematics to calculate the end-effector position in both directions using equation 2. Then, we will find the objective function value for g_L and g_R which are prepared to find the new updated theta value.

$$\theta_{new} = P(\theta_k - \delta_k(\lambda_k) sign(g_L - g_R)\vec{b}) \tag{42}$$

Where θ_{new} is the new updated location of the end-effector. And $sign(g_L - g_R)\vec{b}$ ensures that the beetle moving direction is the small objective function value between both directions. $\delta_k(\lambda_k)$ stands for the step size.

As a third step, the updated theta value is used to calculate the new objective function value. Compare it to the previous theta value to determine the relationship between them. The relationship is shown below:

$$\theta_{k+1} = \begin{cases} \theta_k \ , \ if \ \ g_{new} \geq g(x(t), \theta_k) \\ \theta_{new}, if \ g_{new} < g(x(t), \theta_k) \end{cases} \tag{43}$$

After updating the robot arm's position, the process is repeated.

In the updated algorithm, the objective function now includes velocity minimization. This addition improves the stability and controllability of the tracking progress, resulting in a significant improvement in convergence performance. The following chapter will discuss a detailed comparison of simulation results between the original and updated algorithms.

CHAPTER 5 Simulation Results

This chapter presents the MATLAB simulation results for the original and improved algorithms. Additionally, the performance of convergence speed will be discussed based on the results.

To evaluate the performance of position tracking, we utilized two target trajectories based on original and improved algorithms. The first reference curve is a circular path, while the second one is a rectangular path. This section discusses the results of three different improved methods compared to the original one, which include joint angle velocity, incremental PID, and BSO methods.

5.1 Improved Algorithm with Incremental PID Control Simulation Results

5.1.1 Methods

This section presents the MATLAB simulation results for the original and improved algorithms of the incremental PID control. The performance of convergence speed will also be discussed based on the results.

Based on the original and improved algorithms, we evaluated position-tracking performance using two target trajectories: circular and rectangular paths.

5.1.2 Results and Analysis

A. Circular Trajectory Tracking Results of Original Algorithm

The Beetle Antennae Search algorithm uses the quadratic position error and calculates the step length by taking the square root of this value. The algorithm initially sets up joint angles for each joint, and the circular curve is located at y=0, causing the error difference in the y-axis to be negligible. The position error in three dimensions will be demonstrated below.

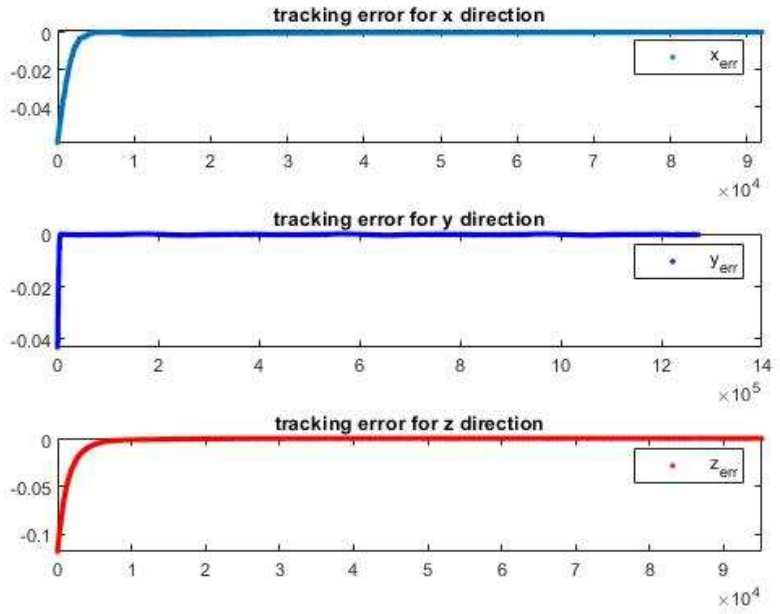

Figure 11: Circular Trajectory Tracking Error with Original Algorithm

Based on the data presented in Fig. 11, we can analyze the circular trajectory tracking performance of the original algorithm in terms of the error observed in different axes. It should be noted that the error in the y-axis is always zero and can be disregarded for further analysis. However, we can observe the error curves in the x and z directions, as shown in Fig. 3. Specifically, the error in the x-axis approaches zero after around 0.5×10^4 iterations. In contrast, on the z-axis, it takes around 0.8×10^4 iterations to reach zero difference.

B. Circular Trajectory Tracking Results of Improved Algorithm

Based on the results presented in Fig. 12, we can observe the errors for different axes for the circular trajectory tracking achieved by the improved algorithm. The incremental PID control added to the step size has significantly improved the convergence speed for both the x and z axes. This control eliminates the need for repetitive integration operations [45] by adding the step size in each iteration. The error in the x-axis drops to zero after approximately 0.1×10^4 iterations, while it takes

around 0.2×10^4 iterations in the z-axis. Hence, the convergence speed for both axes is approximately four times faster than the original algorithm.

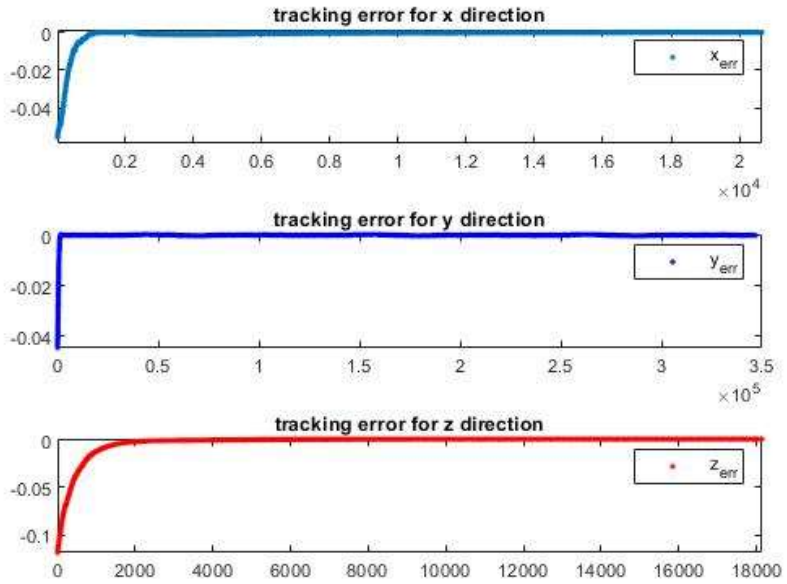

Figure 12: Circular Trajectory Tracking Error with Improved Algorithm

C. Rectangular Trajectory Tracking Results of Original Algorithm

Based on the information provided in Fig. 13, we can determine the errors for different axes in the rectangular trajectory tracking performance using the original algorithm. However, we can also observe the error curves for both the x and z directions, as shown in Fig. 5. The error in the x-axis reaches zero difference after roughly 0.4×10^4 iterations, while the error in the z-axis reaches zero difference after approximately 0.4×10^4 iterations.

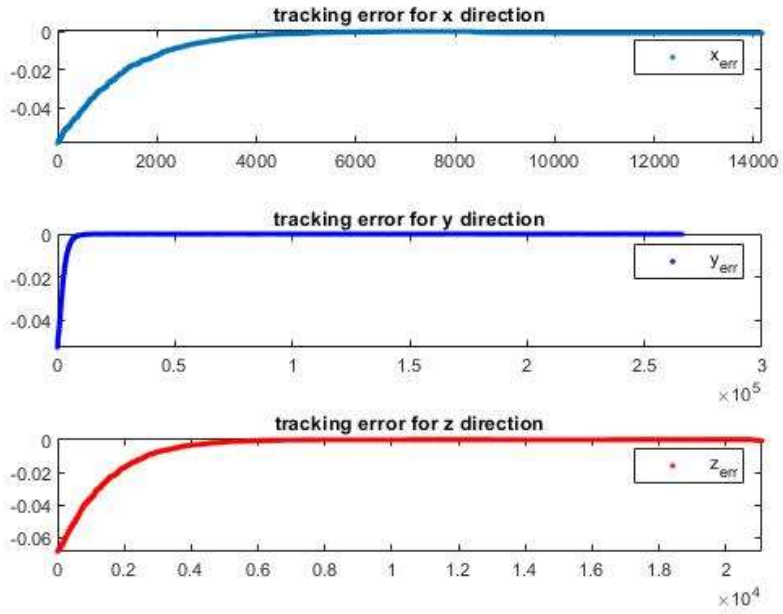

Figure 13: Rectangular Trajectory Tracking Error with Original Algorithm

D. Rectangular Trajectory Tracking Results of Improved Algorithm

Based on the results obtained from Fig. 13, it is evident that the improved algorithm successfully reduces errors in different axes in circular trajectory tracking. To achieve this, incremental PID controllers were utilized over positional PID controllers. This is because incremental PID controllers require less storage, have lower overshoot, and are more resilient than positional PID controllers [46]. The error in the x-axis is reduced to zero after approximately 0.1×10^4 iterations, while the error in the z-axis is also reduced to zero after 0.1×10^4 iterations. This means that the convergence speed for both x and z is approximately four times faster than the original algorithm.

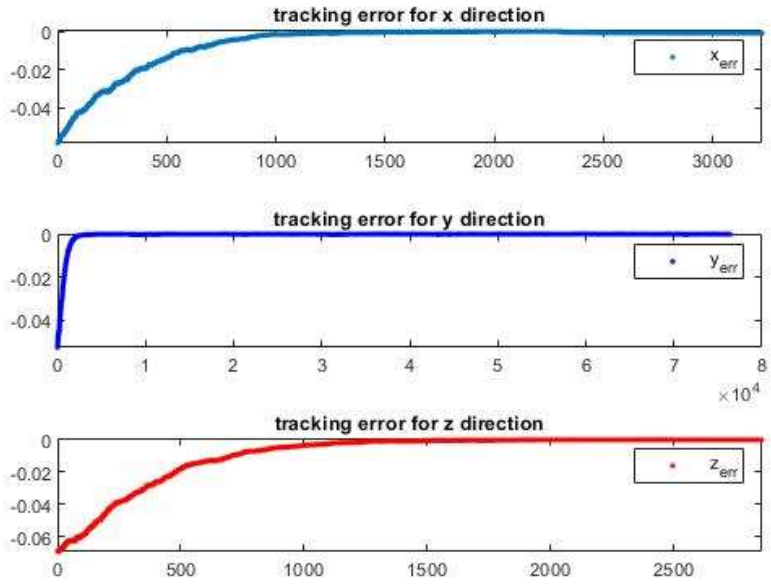

Figure 14: Rectangular Trajectory Tracking Error with Improved Algorithm

5.2 Improved Algorithm with BSO Simulation Results

5.2.1 Methods

This section presents the MATLAB simulation results for the original and improved algorithms of the BSO. The convergence speed performance will also be discussed based on the results obtained.

Using original and improved algorithms, circular and rectangular target trajectories were used to evaluate position-tracking performance.

5.2.2 Results and Analysis

A. Circular Trajectory Tracking Results of Original Algorithm

Fig. 15 illustrates the tracking error performance of the original algorithm. The graph shows that the error in the x direction reduces to zero after approximately 0.5×10^4 iterations, while the convergence time in the z direction is relatively 0.7×10^4 iterations. As previously discussed, the error in the y direction can still be considered negligible.

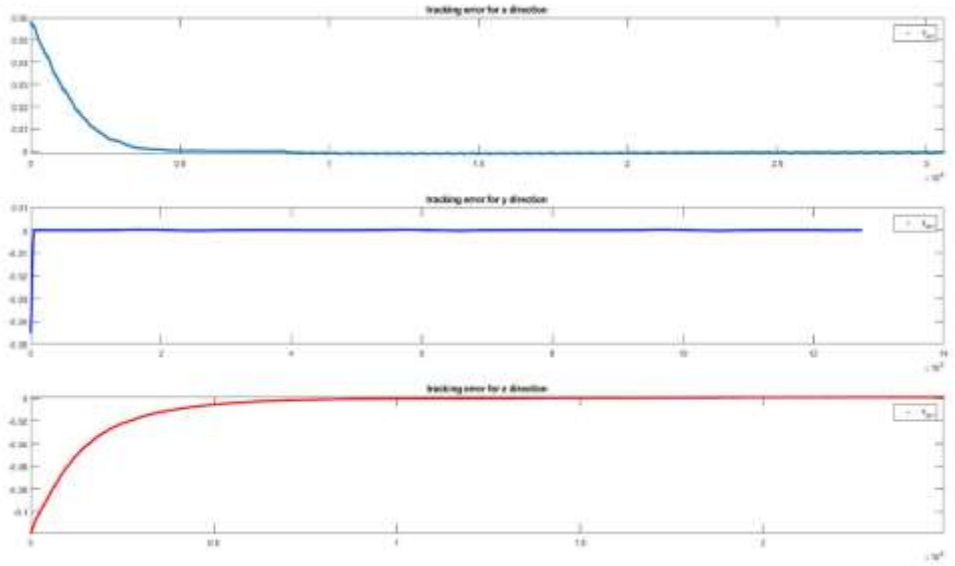

Figure 15: Circular Trajectory Tracking Error with Original Algorithm

B. Circular Trajectory Tracking Results of Improved Algorithm

Based on Fig. 16, it can be inferred that the tracking performance and convergence speed in the x and z directions have improved. This is due to the implementation of the improved algorithm that utilizes the original beetle and the beetle swarm optimization method. The swarm can communicate with one another and share information, which results in faster search speeds and helps avoid falling into local minima. The error difference in the x direction dropped to zero in only about 100 iterations, which is a significant improvement compared to the original algorithm. Additionally, the convergence speed for the error in the z direction is much faster and converges to zero in approximately 100 iterations, as opposed to the original algorithm, which takes longer.

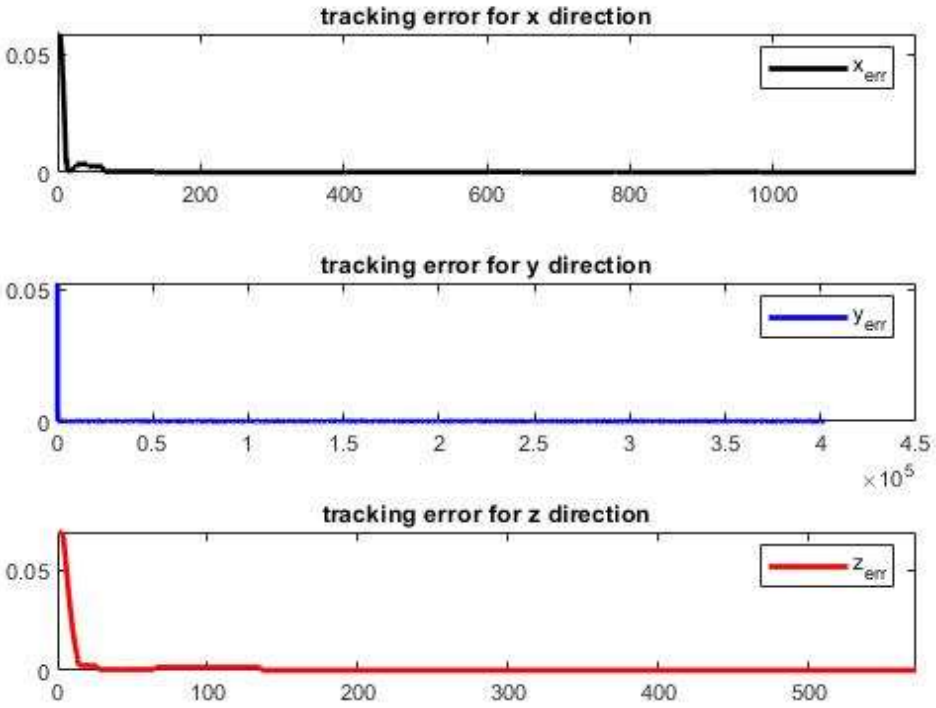

Figure 16: Circular Trajectory Tracking Error with Improved Algorithm

C. Rectangular Trajectory Tracking Results of Original Algorithm

We conducted the same simulation for the rectangular trajectory, and the results are shown in Fig. 17. The figure indicates that the error convergence in the x direction occurs in approximately 5000 iterations, while in the z direction, it occurs in almost the same amount of iterations.

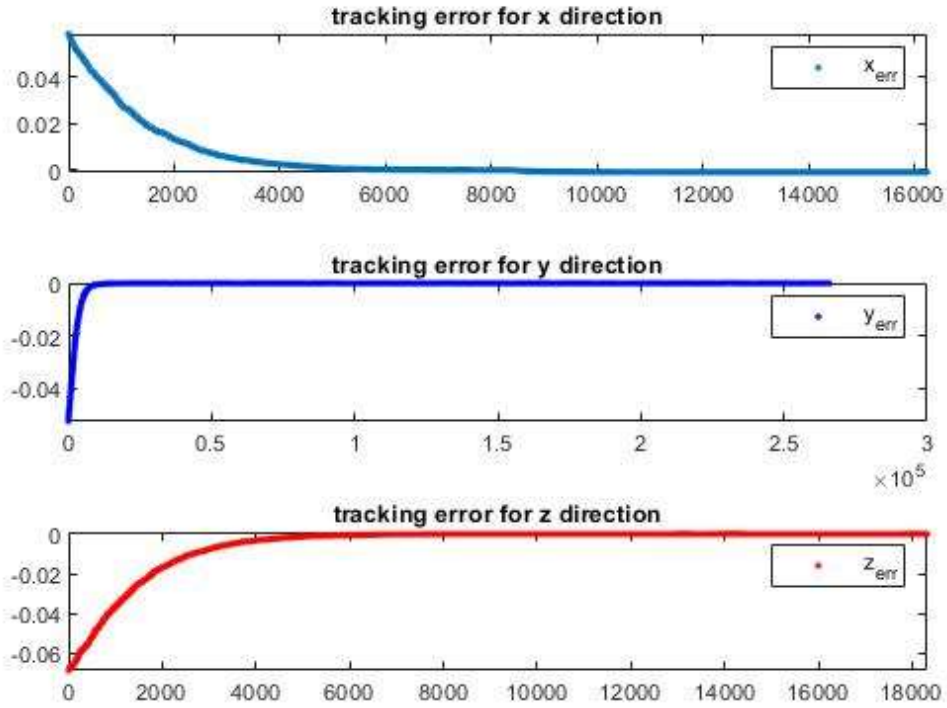

Figure 17: Rectangular Trajectory Tracking Error with Original Algorithm

D. Rectangular Trajectory Tracking Results of Improved Algorithm

As for the results of the rectangular curve from Fig. 18, after we add more beetles into the work, beetles share the information and sensitivity, and the efference will speed up. The figure implies that the error difference in the x direction drops to zero in 200 iterations, almost 20 times faster than the original algorithm. As for the z direction, the tracking error convergence to zero using approximately 100 iterations is 30 times faster than the original algorithm.

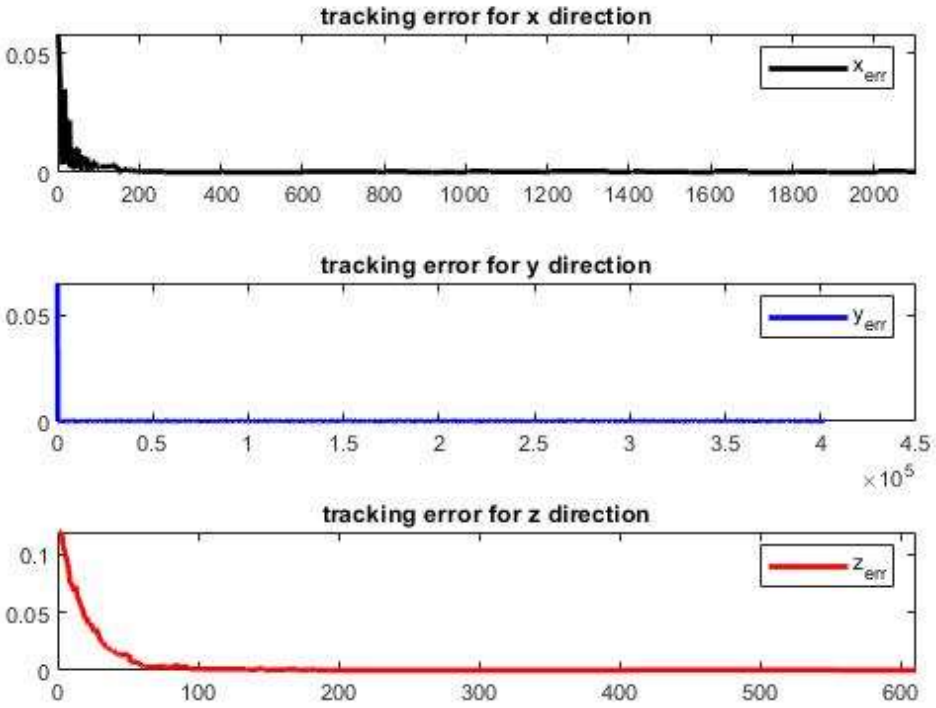

Figure 18: Rectangular Trajectory Tracking Error with Improved Algorithm

5.3 Improved Algorithm with Joint Angle Velocity Simulation Results

5.3.1 Methods

In this section, we propose an updated version of the Beetle Antennae Search Algorithm that includes velocities in joints to minimize power consumption and quadratic position error. The simulation results will be shown below.

A. Circular Trajectory Tracking Results of Original Algorithm

In this section, we present the results of the circular trajectory tracking using the original algorithm. Specifically, we will show the three-dimensional position error with respect to the initial conditions. The initial conditions correspond to the joint angles set up at the start of the experiment. The target curves, which are circular and rectangular, are located on the y=0 plane. Thus, any error

differences in the y-axis will always be zero and will be neglected in our discussion.

5.3.2 Results and Analysis

Fig. 19 shows oscillations in the X and Z directions and a severe chattering effect. A control system should be stable and able to converge towards zero or close to it, indicating that the error between the target curve and the proposed curve is minimized.

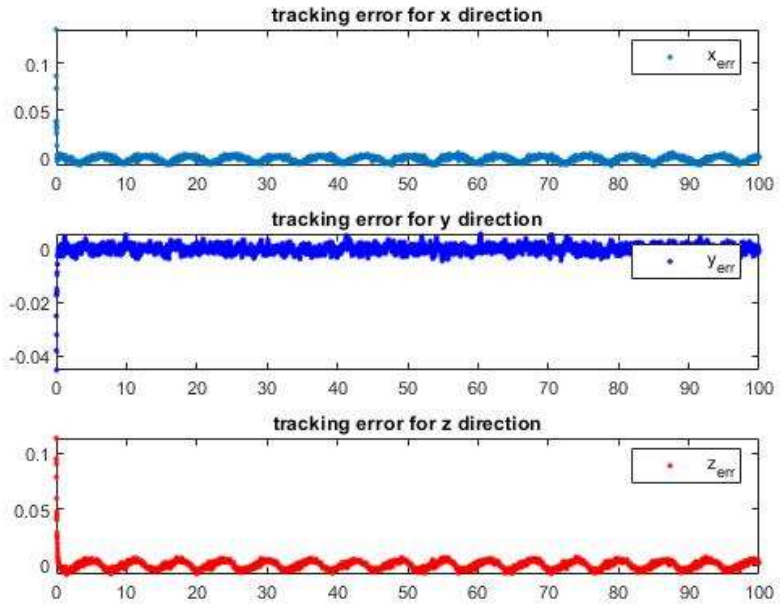

Figure 19: Circular Trajectory Tracking Error with Original Algorithm

B. Circular Trajectory Tracking Results of Improved Algorithm

Based on the information presented in Fig. 20, we can conclude that the tracking error for the y direction is negligible because the target curve is on the x-z plane. Regarding the x direction, we observe that the chattering effect is decreasing, and the tracking error is converging to zero. This occurs in approximately 1.2 seconds, which is a reasonable duration for research purposes. In the case of the z direction, the system is becoming controllable, and the tracking error is reduced to zero in approximately 1 second.

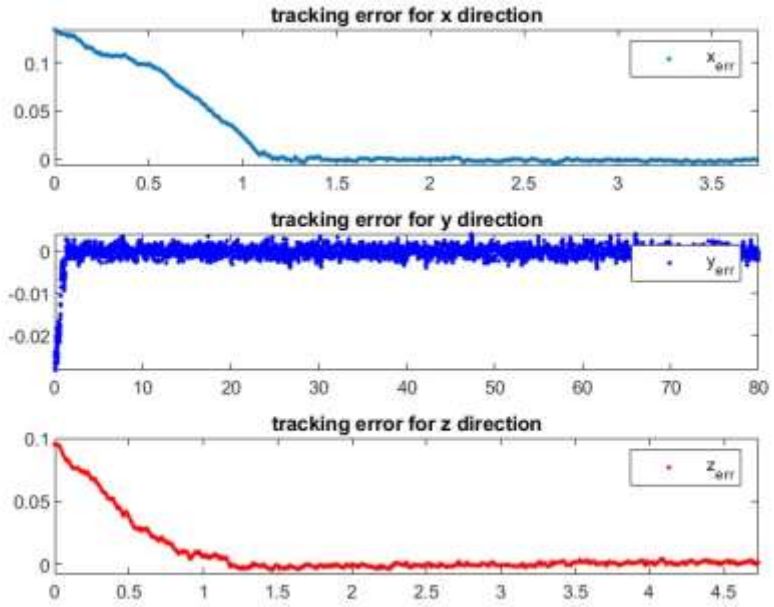

Figure 20: Circular Trajectory Tracking Error with Improved Algorithm

C. Rectangular Trajectory Tracking Results of Original Algorithm

In addition, we conducted rectangular position tracking and obtained error curves, as shown in Fig. 21. The figure indicates that the end-effector's position error reduces to nearly zero in about 10 seconds. However, it also indicates that the original algorithm needs to be more robust as the end-effector oscillates around zero.

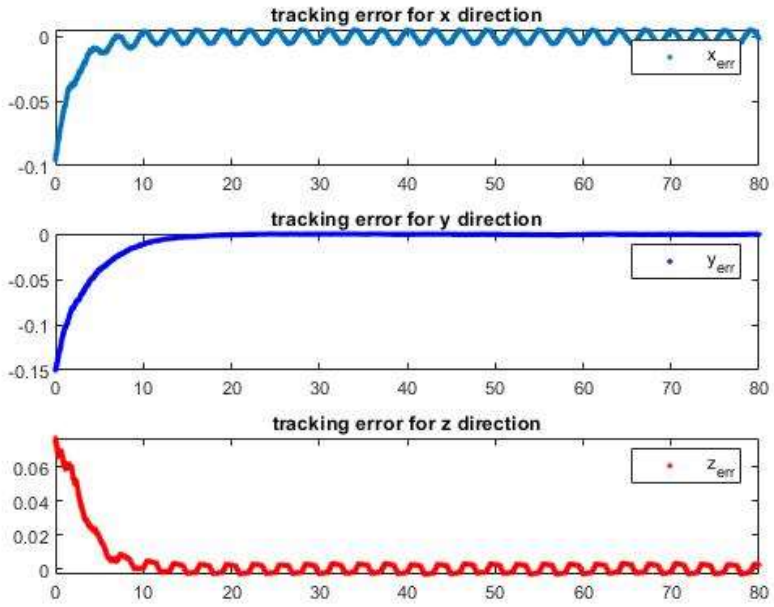

Figure 21: Rectangular Trajectory Tracking Error with Original Algorithm

D. Rectangular Trajectory Tracking Results of Improved Algorithm

Figure 22 shows that the error dynamics in the y-direction are still zero, as the trajectory remains in the x-z plane. The error-convergence performance in the x and z directions takes approximately 2 seconds, practically five times faster than the original algorithm.

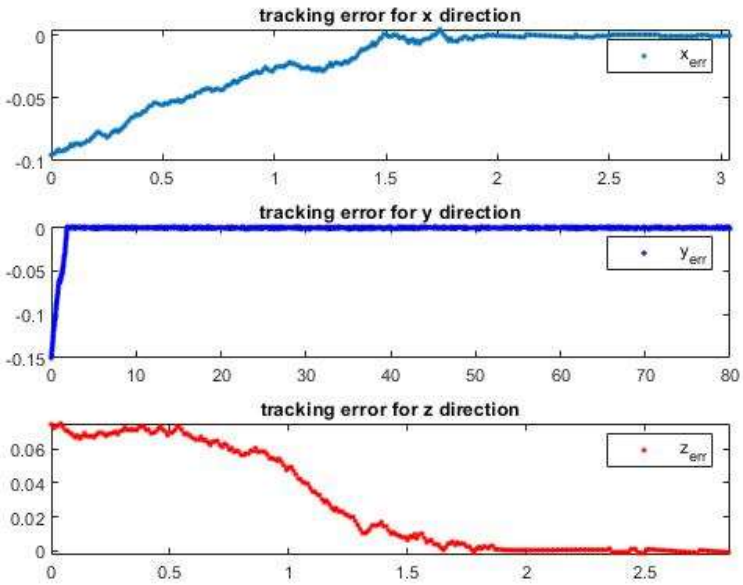

Figure 22: Rectangular Trajectory Tracking Error with Improved Algorithm

CHAPTER 6 Experimental Results

This chapter presents the experimental results obtained from evaluating the developed robotic manipulator system. This chapter will briefly introduce the Omni manipulator system, hardware setup, and the shared experimental results.

6.1 Experimentation Description

This work uses the Omni Bundle robotic system and communicates with MATLAB. The Omni Bundle is a cost-effective and efficient way to perform the control concepts of robotics and haptics. The Omni is a robot manipulator with the six revolute joints we discussed in previous sections. As we all know, only three of the joints are actuated, which is what we are interested in. The three actuated joints are J_1, J_2 and J_3 shown in Fig. 23 [47].

Figure 23: Three Actuated Joints on the Omni Manipulator

The whole process of the experiment will be divided into two main parts. The first part of the process is called Teaching Points. This part we named the teach pendant

experiment, where a set of discrete points are taught to the robot. During playback motion, the robot traverses each of the taught points. Programming the robot to do this is a three-step process. It involves creating a routine to teach the points to the robot, then creating a desired path between those points that the robot should follow, and then finally creating a routine to control the robot along that path. This step is learning how to teach Omni different points in space and then create linear trajectories between those points. However, in many applications it may be necessary that we design for the path of the end-effector and not the joints. Applications of this design are more intuitive. If the robot is required to move along

Figure 24: The Button on the Omni Manipulator

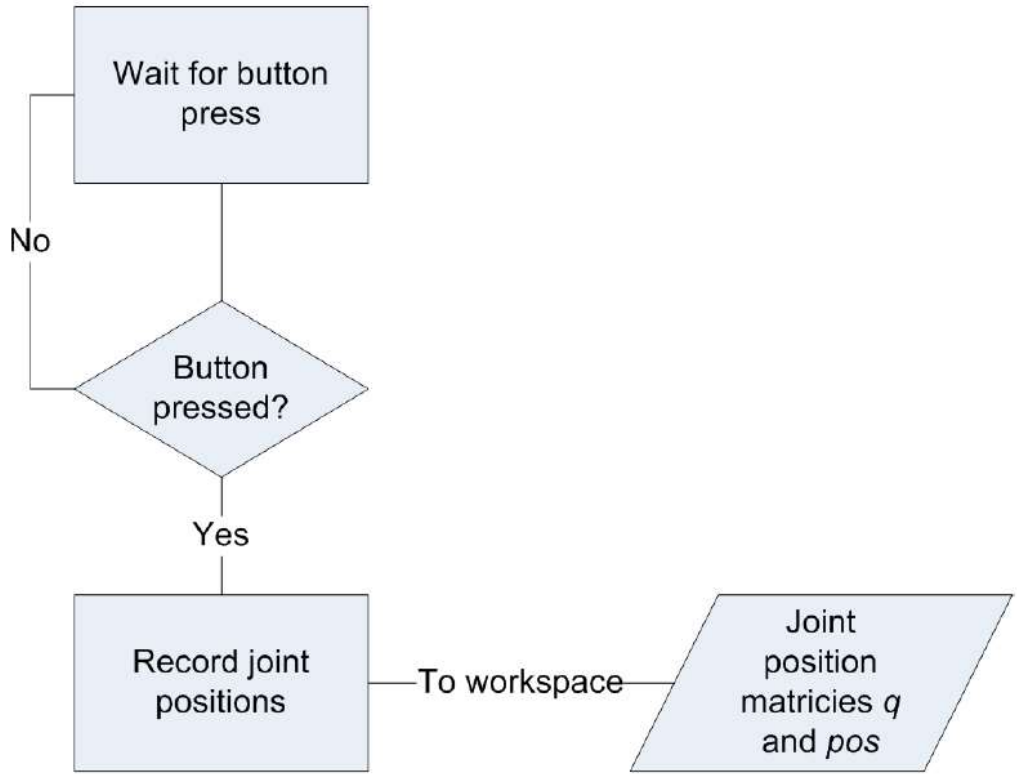

Figure 25: Teach Pendant Experiment Flowchart

a welding contour, then it may be a more intuitive motion of the end-effector and not the joints. As a result, we will observe that the end-effector moves from one point to the next. Every time the button shown in Fig. 24 is pressed, a new row is created in each of these variables to store the information. For instance, if the button is pressed four times, matrices q and pos on the workspace will be both 4x3.

The experimental flowchart is shown in Fig. 25. The flowchart illustrates the entire process for recording joint positions in response to a button press event. There are four main blocks shown in this flowchart. The initial stage is named wait for button press. This step begins in a standby mode, continuously waiting for a button press event. This loop ensures the system remains idle and conserves resources until user interaction is detected. The second stage is called button press detection. The system monitors for a button press. If no button press is detected, it loops back to the initial waiting state. This decision point is critical for ensuring that the system only proceeds

when there is active user input. The next stage is called record joint positions. Once a button press is detected, the system exits the waiting loop and proceeds to record the current joint positions. This involves capturing the joint angles and positions at the exact moment of the button press, which is essential for accurate data collection. The last stage is named save data to workspace. The recorded joint positions are then saved to the workspace in the form of matrices labelled q and pos. These matrices store the joint angles (q) and their corresponding positions (pos), providing a structured format for further analysis and processing. The significance of the flowchart is highlighted as a simple yet effective method for capturing joint position data based on user interaction. By implementing a wait mechanism for button presses, the system ensures that data is only recorded at relevant moments, minimizing unnecessary data collection and enhancing the accuracy of the recorded information. The second main phase is called Controlling the robot. The experimental flowchart is shown in Fig. 26. This process illustrates the iterative process of optimizing control gains for better tracking performance of a haptic device setup. This process leverages an optimization algorithm to refine the control parameters, ensuring improved stability and performance of the system. This process includes five steps. The first step is Pre-tuned Gains. The process starts with a set of pre-tuned gains. These initial control parameters have been manually or heuristically determined to provide a baseline performance. The second step is feeding the gains to the Optimization Algorithm. The pre-tuned gains are fed into the optimization algorithm, referred to as the "Beetle Bee" algorithm. This algorithm iteratively adjusts the control gains to improve the system's performance. The output of this step is a set of optimized gains. The next block is the newly optimized gains are applied to the system. Then the optimized gains are fed into the simulink file to control the Omni manipulator to

perform the trajectory tracking. Then the block shows the methods to check the tracking performance, which is the trajectory plot check or objective value check, and we have this named “fbest”. The system’s performance with the new gains is evaluated. This can be done by visually inspecting a trajectory plot or by checking an objective value (fbest), which quantitatively measures the system's performance. The image within the flowchart shows a setup involving the Quanser Omni Bundle haptic device and a computer displaying the evaluation results. This visual feedback is crucial for the next decision step. The last step is the stability and objective value decision point. The flowchart includes a decision point where the stability of the trajectory or the objective value (fbest) is assessed. If the trajectory is more stable or fbest is smaller (indicating better performance), the optimized gains are deemed successful, which needs to save the best objective value (fbest) and the new plot, indicating that the optimized parameters have improved the system. However, if the performance has not improved, the process loops back to the pre-tuned gains, and the optimization algorithm runs again with potentially new parameters, which means that it is a “No” decision that needs to return to the optimization algorithm step with the current or adjusted gains for further refinement. The significance of this flowchart is that it outlines a systematic approach to enhancing the performance of a haptic device through iterative optimization. Integrating both automated optimization (via the Beetle Bee algorithm) and manual verification (through trajectory plots and objective values) ensures that the gains are theoretically optimal and practically effective.

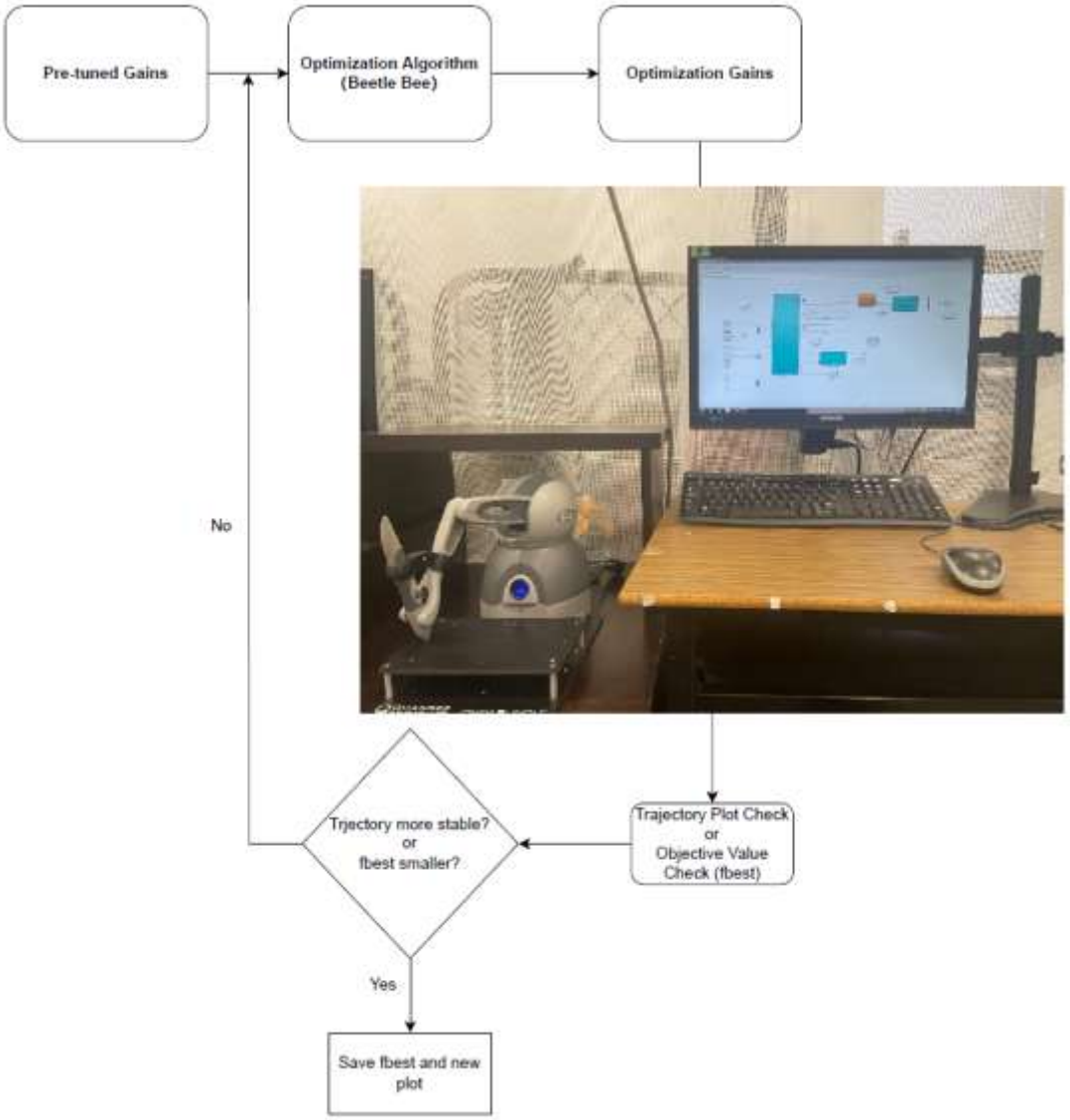

Figure 26: Controlling the Robot Experimental Flowchart

6.2 Experimentation Setup

Figure 27: Experimental Setup

This experiment setup shown in Fig. 27 gives us an operating system. This advanced experimental setup is centred around the Quanser Omni Bundle, a state-of-the-art haptic device integrated with a computer workstation. This configuration is indicative of a high-precision research environment focused on manipulation control.

The components of the setup from the image show that there is a Quanser Omni, a robust and precise haptic interface, which is the critical module of the whole setup placed on the left side of the workstation. This device allows users to interact with virtual environments or control remote systems with exceptional precision and realism. It is widely used in research fields requiring detailed manipulation and touch feedback. Another main module is the computer workstation. A monitor displays a graphical interface related to the control and data acquisition for the haptic device.

And also the supporting infrastructure, such as a study desk and appropriate cable management, to ensure a clutter-free and efficient working environment.

Overall, the Quanser Omni experimental setup represents a highly specialized and professional environment designed for trajectory-tracking research in the field of haptic technology and control systems.

6.3 Experimentation Results

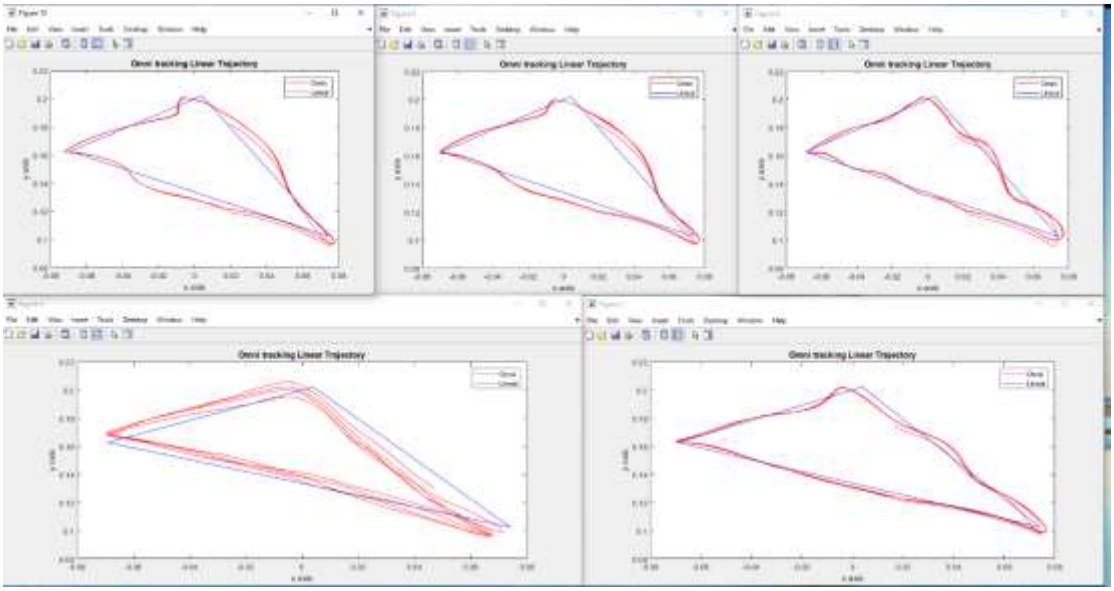

Figure 28: Tracking Performance of a Linear Triangle Curve

The obtained tracking results are shown in Fig. 28. Five figures are shown in the figure, labelled “Figure 10”, “Figure 8”, “Figure 6”, “Figure 4”, and “Figure 2”, each comparing the tracking performance of an "Omni" manipulator trajectory and a "Linear" trajectory system.

The first plot, labelled “Figure 10” at the top left, shows that the red line represents the Omni tracking curve, and the blue line expresses the linear desired curve. These two lines show notable discrepancies, especially at points of rapid direction change. The performance of this trial indicates that the Omni manipulator tracks the linear system but still exhibits significant deviations, particularly in areas with high curvature. The second plot, located at the top middle, shows that the gap between the

Omni and Linear trajectories begins to reduce. The Omni system (red) is starting to show a more consistent path closer to the reference. Improvement is noticeable in terms of the results, as the Omni manipulator tracks the trajectory more accurately than the first plot. The deviations are less pronounced, indicating better handling of direction changes and overall smoother tracking. In addition, for the third plot, which is placed at the top right, we can perceive that the Omni manipulator continues to show progress, with its trajectory more closely following the reference path compared to the previous plots. As a result, the performance improvement trend continues, with the Omni system showing reduced deviations and better alignment with the reference trajectory. The Linear system still shows larger deviations, especially in high-curvature sections. Then, for the following plot, which is at the bottom left, we notice that the trajectories for both systems are becoming more aligned, with the Omni manipulator's path nearly overlapping the reference trajectory. From the perspective of this plot, significant improvements are evident in the Omni system's tracking performance. The Omni trajectory is smoother and shows minimal deviations from the reference path. The Linear system's performance remains consistent with prior observations but is outperformed by the Omni system. From the last plot in the bottom right corner, we can observe the closest alignment between the Omni manipulator's trajectory and the reference path. Based on the analysis of this plot, the Omni manipulator achieves the best tracking performance in this trial. The trajectory is highly accurate, with minimal deviation throughout the path. The Omni system's robustness and precision in tracking are clearly demonstrated, significantly outperforming the Linear system.

In conclusion, upon completing this experiment, we can summarize the following. From “Figure 10” to “Figure 2”, the Omni manipulator's tracking performance

improves progressively with each experiment trial. Initially, the Omni system shows significant deviations, particularly in high-curvature areas. However, as the trials progress, these deviations reduce, and the Omni manipulator's trajectory becomes smoother and more closely aligned with the reference path. By “Figure 2”, the Omni manipulator demonstrates superior tracking accuracy, robustness, and precision, clearly outperforming the Linear system in all key performance aspects. This trend indicates effective adaptation and optimization of the Omni manipulator's control mechanisms over successive trials.

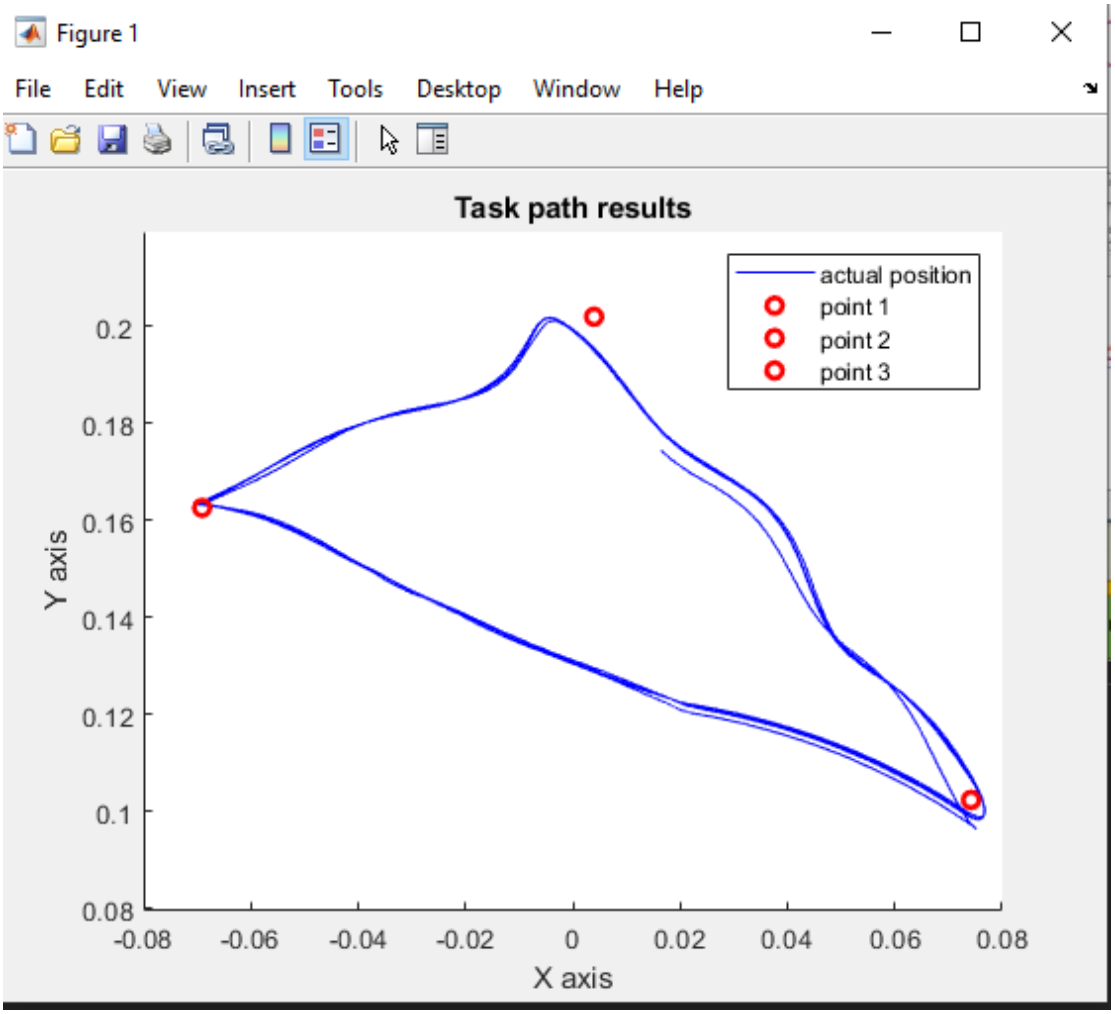

Figure 29: Task Path Results

From Fig. 29, we can observe that the ability to track predefined trajectories accurately is critical in the realm of Omni manipulation and control. This figure

shows the tracking performance of an Omni manipulator system when provided with three pre-taught points. The objective is to evaluate how these predefined points influence the manipulator's ability to follow a complex trajectory.

We can notice that three key points were selected along the desired trajectory, marked as point 1, point 2, and point 3. These points serve as reference landmarks for the manipulator. Subsequently, the Omni manipulator will follow the trajectory while utilizing the pre-taught points as guides. Thereafter, the actual position of the manipulator was recorded throughout the trajectory, and the results were plotted for analysis. The manipulator begins the trajectory with a close alignment to point 1, indicating a strong initial adherence to the pre-taught path. Afterward, the manipulator successfully reaches point 2, maintaining a consistent path with minimal deviation. This demonstrates the system's ability to handle the complexity and elevation changes in the trajectory. Ultimately, the manipulator accurately reaches point 3, completing the trajectory with precision. This indicates effective control and stability throughout the entire path.

In the final analysis, implementing pre-taught points in trajectory tracking for the Omni manipulator proves to be highly effective. The pre-taught points are reliable references, guiding the manipulator through complex paths with enhanced accuracy and stability.

CHAPTER 7 Conclusion and Future Work

7.1 Conclusion

This thesis presents several improved Beetle Bee Algorithms for position tracking control of OMNI manipulators in simulation using MATLAB. The first proposed algorithm adds incremental PID control to the step size in each iteration. This approach provides increment to the control system without an integral effect, effectively avoiding the problem of critical saturation. As a result, the system becomes more stable when unknown disturbances are introduced, making it more robust. The second improved algorithm combines beetle swarm optimization to become a hybrid algorithm to improve tracking performance. The last proposed algorithm adds the angular velocity to the objective function to enhance the trajectory tracking performance. The proposed algorithms rely on values from the end-effector by using forward kinematics based on the BAS algorithm. The simulation was performed in different scenarios, such as circle and rectangle trajectories. These tracking simulations on the OMNI manipulator prove that the proposed algorithm performs better in terms of convergence speed. In order to validate the effectiveness of these proposed optimization algorithms, real-world experiments were conducted on the Omni robotic manipulator system. The experimental results confirmed that, with successive iterations, the parameters were continually optimized, yielding increasingly stable and accurate trajectory tracking relative to the reference path. These findings underscore the proposed optimization strategies' potential to significantly enhance the Omni robotic systems' performance in trajectory-tracking tasks.

7.2 Future Work

While the current study has yielded promising results, several avenues for future research could further enhance the performance and applicability of the optimization algorithms for trajectory tracking in the Omni robotic systems.

As part of our future work, one potential way to refine the optimization algorithms is to further enhance the convergence speed and accuracy. We suggest modifying the proposed algorithms to experiment with better hyperparameters and pre-tuned parameters, improving the tracking performance.

Additionally, investigating real-time optimization strategies and adaptive control mechanisms to address dynamic environments and unforeseen disturbances would be crucial for practical applications of the Omni robotic manipulator system in complex scenarios.

Moreover, it is suggested that comparative studies be conducted with other state-of-the-art optimization algorithms and control strategies to gain valuable insights for further enhancing the trajectory tracking accuracy and efficiency of the Omni robotic manipulator system. Evaluating the position-tracking performance of both the original algorithm and the enhanced algorithms on a manipulator with a higher degree of freedom (DOF) is recommended to enable comparison with the Omni system and potential publication in other conferences.

Next, it is imperative to explore real-time optimization strategies and adaptive control mechanisms to address dynamic, unpredictable environments and unforeseen disturbances. This is especially relevant for implementing the Omni robotic manipulator system in complex scenarios. This approach can be further optimized and applied to various robotic applications research may explore the optimization of

the number and placement of pre-taught points to further refine trajectory tracking capabilities.

Furthermore, it is crucial to explore real-time optimization strategies and adaptive control mechanisms structures to address dynamic, unpredictable environments and unforeseen disturbances. This is especially pertinent for the practical implementation of the Omni robotic manipulator system in complex scenarios.

Last but not least, to tackle complex systems, such as those encountered in PhD studies, we propose to build a master-slave OMNI system. This system can execute trajectory tracking using our developed and tested algorithms. We aim to achieve optimal performance and efficiency through this approach.

Bibliography

[1] B. Xiao and S. Yin, “Exponential tracking control of robotic manipulators with uncertain dynamics and kinematics,” IEEE Transactions on Industrial Informatics, vol. 15, no. 2, pp. 689–698, 2019.

[2] X. P. Shi and S. R. Liu, “A Survey of Trajectory Tracking Control for Robot Manipulators,” Control Engineering of China, vol. 18, no. 1, pp. 116-122, 2011. [3] Craig, J.J. (2005) *Introduction to robotics: Mechanics and control*. Upper Saddle River, N.J: Pearson/Prentice Hall.

[3] L.M. Capisani, A. Ferrara, L. Magnani, Second order sliding mode motion control of rigid robot manipulators, in: Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 2007, pp. 12–14.

[4] *OFweek*. Available at: https://robot.ofweek.com/2021-09/ART-8321200-8300-30526867.html, 2021.

[5] C. Molnár, T. D. Nagy, R. N. Elek and T. Haidegger, “Visual servoing-based camera control for the da Vinci Surgical System,” 2020 IEEE 18th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2020, pp. 107-112, doi: 10.1109/SISY50555.2020.9217086.

[6] O. Özgüner et al., “Camera-Robot Calibration for the Da Vinci Robotic Surgery System,” in IEEE Transactions on Automation Science and Engineering, vol. 17, no. 4, pp. 2154-2161, Oct. 2020, doi: 10.1109/TASE.2020.2986503.

[7] N. Tran, J. Y. Wu, A. Deguet and P. Kazanzides, “A Deep Learning Approach to Intrinsic Force Sensing on the da Vinci Surgical Robot,” 2020 Fourth IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, 2020, pp. 25-32, doi: 10.1109/IRC.2020.00011.

[8] M. H. Hamedani *et al*., “Robust Dynamic Surface Control of da Vinci Robot Manipulator Considering Uncertainties: A Fuzzy Based Approach,” *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, Tehran, Iran, 2019, pp. 418-423, doi: 10.1109/ICRoM48714.2019.9071876.

[9] China, R.D.K. 17 Industrial Robot Applications for Smart Manufacturing, *RoboDK Blog*. Available at: https://robodk.com/cn/blog/ndustrial-robot-applications/, 2022.

[10] M. Abdelaal, "A Study of Robot Control Programing for an Industrial Robotic Arm," *2019 6th International Conference on Advanced Control Circuits and Systems (ACCS) & 2019 5th International Conference on New Paradigms in Electronics & information Technology (PEIT)*, Hurgada, Egypt, 2019, pp. 23-28, doi: 10.1109/ACCS-PEIT48329.2019.9062878.

[11] L. Salameen, A. Estatieh, S. Darbisi, T. A. Tutunji and N. A. Rawashdeh, "Interfacing Computing Platforms for Dynamic Control and Identification of an Industrial KUKA Robot Arm," *2020 21st International Conference on Research and Education in Mechatronics (REM)*, Cracow, Poland, 2020, pp. 1-5, doi: 10.1109/REM49740.2020.9313878.

[12] M.W, Spong and M, Vidyasagar, *Robot Dynamics and control*. New York: Wiley, 2004.

[13] *2 DOF serial flexible link Quanser*. Available at: https://www.quanser.com/products/2-dof-serial-flexible-link/, 2021.

[14] J. S. Martin and G. T. Phedes , " Proceedings of the Third Workshop on Virtual Reality Interactions and Physical Simulations", VRIPHYS 2006.

[15] *Qarm Quanser*. Available at: https://www.quanser.com/products/qarm/, 2021.

[16] R.N, JAZAR, "Robot Components," in *Theory of applied robotics: Kinematics, dynamics, and control*. New York: Springer, 2010.

[17] A. J. Silva, O. A. D. Ramirez, V. P. Vega and J. P. O. Oliver, "PHANToM OMNI Haptic Device: Kinematic and Manipulability," *2009 Electronics, Robotics and Automotive Mechanics Conference (CERMA)*, Cuernavaca, Mexico, 2009, pp. 193-198, doi: 10.1109/CERMA.2009.55.

[18] L.M. Capisani, A. Ferrara, L. Magnani, Second order sliding mode motion control of rigid robot manipulators, in: Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 2007, pp. 12–14.

[19] F. Loucif, S. Kechida, and A. Sebbagh, "Whale optimizer algorithm to tune PID controller for the trajectory tracking control of robot manipulator," Journal of the Brazilian Society of Mechanical Sciences and Engineering, vol. 42, no. 1, 2019.

[20] Y. Su, P. C. Müller and C. Zheng, "Global Asymptotic Saturated PID Control for Robot Manipulators," in *IEEE Transactions on Control Systems Technology*, vol. 18, no. 6, pp. 1280-1288, Nov. 2010, doi: 10.1109/TCST.2009.2035924.

[21] Y. Zhang, C. Wu and S. Li, "Research on compound control algorithm for motion control system," 2017 29th Chinese Control And Decision Conference (CCDC), Chongqing, 2017, pp. 5035-5040, doi: 10.1109/CCDC.2017.7979388.

[22] E. Rastogi and L. B. Prasad, "Comparative performance analysis of PD/PID computed torque control, filtered error approximation based control and NN control for a robot manipulator," 2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON), Allahabad, 2015, pp. 1-6, doi: 10.1109/UPCON.2015.7456706.

[23] M. Reyhanoglu, "Feedback control of a flexible joint robot," 1997 European Control Conference (ECC), Brussels, 1997, pp. 3194-3199, doi: 10.23919/ECC.1997.7082602.

[24] R. Tamias *et al*., "Particle Swarm Optimization Algorithm for Optimizing Item Arrangements in Storage Warehouse," *2021 3rd International Conference on Electronics Representation and Algorithm (ICERA)*, Yogyakarta, Indonesia, 2021, pp. 167-172, doi: 10.1109/ICERA53111.2021.9538775.

[25] A. Mallick, S. Roy, S. S. Chaudhuri and S. Roy, "Study of parametric optimization of the Cuckoo Search algorithm," Proceedings of The 2014 International Conference on Control, Instrumentation, Energy and Communication (CIEC), 2014, pp. 767-772, doi: 10.1109/CIEC.2014.6959194.

[26] A. A. Musa, S. Hafiz Imam, A. Choudhary and A. P. Agrawal, "Parameter Estimation of Software Reliability Growth Models: A Comparison Between Grey Wolf Optimizer and Improved Grey Wolf Optimizer," *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2021, pp. 611-617, doi: 10.1109/Confluence51648.2021.9377194.

[27] Y. Ling, Y. Zhou and Q. Luo, "Lévy Flight Trajectory-Based Whale Optimization Algorithm for Global Optimization," in IEEE Access, vol. 5, pp. 6168-6186, 2017, doi: 10.1109/ACCESS.2017.2695498.

[28] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," in IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67-82, April 1997, doi: 10.1109/4235.585893.

[29] Á. Jimenez-Uribe, L. F. Serna-Hernández, J. M. Hernández-Paredes and B. Muñoz-Barrón, "Comparative Study of a PID and PD Control Bounded by Hyperbolic Tangent Function in Robot 3 DOF," 2015 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE), Cuernavaca, Mexico, 2015, pp. 199-204, doi: 10.1109/ICMEAE.2015.35.

[30] G. Wang and N. Liu, "Design and Implementation of a New Approach Based on the Error Feedforward Compensation for Motion Controller," 2006 Chinese Control Conference, Harbin, China, 2006, pp. 1579-1584, doi: 10.1109/CHICC.2006.280759.

[31] E. Rastogi and L. B. Prasad, "Comparative performance analysis of PD/PID computed torque control, filtered error approximation based control and NN control for a robot manipulator," 2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON), Allahabad, 2015, pp. 1-6, doi: 10.1109/UPCON.2015.7456706.

[32] M. Reyhanoglu, "Feedback control of a flexible joint robot," 1997 European Control Conference (ECC), Brussels, 1997, pp. 3194-3199, doi: 10.23919/ECC.1997.7082602.

[33] M. T. Vakil-Baghmisheh and M. Salim, "A modified fast marriage in honey bee optimization algorithm," *2010 5th International Symposium on Telecommunications*, Tehran, Iran, 2010, pp. 950-955, doi: 10.1109/ISTEL.2010.5734159.

[34] M. Zein, A. E. Hassanien, A. Badr and T. -H. Kim, "Human Activity Classification Approach on Smartphone Using Monkey Search Algorithm," 2015 Seventh International Conference on Advanced Communication and Networking (ACN), Kota Kinabalu, Malaysia, 2015, pp. 84-88, doi: 10.1109/ACN.2015.31.

[35] Y. Shiqin, J. Jianjun and Y. Guangxing, "A Dolphin Partner Optimization," 2009 WRI Global Congress on Intelligent Systems, Xiamen, China, 2009, pp. 124-128, doi: 10.1109/GCIS.2009.464.

[36] S. Goel and V. K. Panchal, "Performance evaluation of a new modified firefly algorithm," Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, Noida, India, 2014, pp. 1-6, doi: 10.1109/ICRITO.2014.7014717.

[37] X. Jiang and S. Li, "BAS: Beetle Antennae Search Algorithm for Optimization Problems", International Journal of Robotics and Control, vol. 1, no. 1, p. 1, 2018.

[38] A. H. Khan, S. Li and X. Luo, “Obstacle Avoidance and Tracking Control of Redundant Robotic Manipulator: An RNN-Based Metaheuristic Approach,” in IEEE Transactions on Industrial Informatics, vol. 16, no. 7, pp. 4670-4680, July 2020, doi: 10.1109/TII.2019.2941916.

[39] A. Khan, X. Cao, S. Li and C. Luo, “Using Social Behavior of Beetles to Establish a Computational Model for Operational Management”, IEEE Transactions on Computational Social Systems, vol. 7, no. 2, pp. 492-502, 2020.

[40] L. Wu, R. Crawford and J. Roberts, “An Analytic Approach to Converting POE Parameters Into D–H Parameters for Serial-Link Robots,” in IEEE Robotics and Automation Letters, vol. 2, no. 4, pp. 2174-2179, Oct. 2017, doi: 10.1109/LRA.2017.2723470.

[41] M. Liu, H. Zhang, Y. Zhang and C. Yuan, “Design and Performance Analysis of ZYNQ Based Incremental PID-PWM Controller,” 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), 2022, pp. 1123-1126, doi: 10.1109/EEBDA53927.2022.9744964.

[42] L. Y. and S. S, "Design of flow cytometer liquid circuit control system based on incremental PID algorithm", Journal of Physics: Conference Series, vol. 1633, pp. 012001, 2020.

[43] W. Chu, X. Gao and S. Sorooshian, "Fortify particle swarm optimizer (PSO) with principal components analysis: A case study in improving bound-handling for optimizing high-dimensional and complex problems," *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans, LA, USA, 2011, pp. 1644-1648, doi: 10.1109/CEC.2011.5949812.

[44] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95-international conference on neural networks, vol. 4, pp. 1942–1948, IEEE, 1995.

[45] H. Niu, X. Liao, Y. Dai, C. Hu, L. Zhang and S. Chen, “A Multi-factor Control System in Greenhouse Based on The Internal Incremental PID Algorithm,” 2021 International Conference on Electronic Information Technology and Smart Agriculture (ICEITSA), 2021, pp. 99-105, doi: 10.1109/ICEITSA54226.2021.00028.

[46] X. Guo, Z. Li and G. Sun, “The Robot Arms Control Based on RBF with Incremental PID and Sliding Mode Robustness,” 2019 WRC Symposium on Advanced Robotics and Automation (WRC SARA), 2019, pp. 97-102, doi: 10.1109/WRC-SARA.2019.8931971.

[47] J., Apkarian, P., Karam, D., Crymble, A., Abdossalami, & R., Samra, STUDENT WORKBOOK Omni Bundle Robotics Experiment. Quanser Inc, 2021.

Appendix A Author's Publications

1. X. Zhang, J. Gu, M. U. Asad, U. Farooq and G. Abbas, "Beetle Bee Algorithm Applied to Trajectory Tracking Control of OMNI Manipulator," 2022 International Conference on Emerging Trends in Electrical, Control, and Telecommunication Engineering (ETECTE), Lahore, Pakistan, 2022, pp. 1-5, doi: 10.1109/ETECTE55893.2022.10007292.
2. X. Zhang, J. Gu, M. U. Asad, and U. Farooq. "Beetle Bee with Incremental PID Control Algorithm Applied to Trajectory Tracking Control of OMNI Manipulator," The 10th International Workshop on Soft Computing Applications (SOFA 2022). Arad.
3. X. Zhang, J. Gu, M. U. Asad, U. Farooq and K. Cherfouh, "Beetle Bee With BSO Algorithm Applied to Trajectory Tracking Control of OMNI Manipulator," (submitted to IEEE ETECTE 2024).
4. K. Cherfouh, J. Gu, Ali J.-F. and X. Zhang "Geometrical Parameter Optimization of Double Rotor Axial-Flux Permanent Magnet Synchronous Motor," accepted for presentation at 2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). Kingston, August, 2024.
5. M. U. Asad, J. Gu, U. Farooq, Khurram, Q. K., K. Cherfouh, and X. Zhang "Design of Limited System Inspired Intelligent Controller with Application to Unmanned Surface Vehicles," accepted for presentation at The OCEANS September, 2024. Halifax.