

Republic of Iraq
Ministry of Higher Education
and Scientific Research
University of Babylon
College of Engineering



Vision-Based Control System For a Robotic Manipulator

A Thesis

*Submitted to the College of Engineering of
University of Babylon in Partial Fulfillment of the
Requirements for the award of Degree of Master of
Science in Mechanical Engineering
(Applied Mechanics)*

By

Firas Subhy Hameed Ahmed

2009 A.M

1430 A.H

Abstract

Robots are electromechanical devices that are used by humans to perform repetitive processes that are required to be very accurate because humans cannot repeat the same task with the same performance indefinitely. The motion of the robot is decided by a controller that sends orders to the actuators of the robot to move according to a feedback signal. Usually, for repetitive tasks, the motion of the robot is controlled by a fixed set of feedback signals sent to the robot by a computer program designed for this purpose. This means that the program is not adaptive to the changes of the parameters and accidental errors that can happen during the operation.

The robot gains more flexibility to adapt these errors when it is supplied with the capability of interacting with its environment. Vision-based control systems provide the robot with the capability of (seeing) the working space and to respond according to the changes in the environment in which the robot is installed.

In this work, an eye-in-hand vision based control system was simulated to drive a four-link robotic arm. A dynamic model for this type of robots was derived. From this model, the forces and torques acting on the robotic arm were calculated. Many trajectories of gradual increase in complexity were assigned to the model to evaluate the torques required at each joint. The inverse kinematics that corresponds to each trajectory was determined by using a new method for this robotic arm. This method was accurate and fast in obtaining the position profile required to enable the robot to follow a desired trajectory.

Many digital image processing algorithms were implemented to detect an object of interest that moves within the view of the camera. These algorithms are responsible of generating the error signals required to drive the robot.

A software-implemented PID controller was used to drive the robotic arm. This controller receives, as inputs, the error signals generated by the digital image processing algorithms and transforms them into electrical signals to drive each joint according to the required path.

All the algorithms that were used to build the model were implemented by using the Visual Basic 6.0 programming language.

LIST OF CONTENTS

Chapter 1 Introduction

1.1. Types of Robots	1
1.2. Visual Servoing Systems	5
1.3. Robotic Manipulators Control	5
1.4. The Objective of the Present Work	7

Chapter 2 Literature Review

2.1. General	8
2.2. Vision-Based Control Systems	8
2.3. Inverse Kinematics	15
2.4. Summery	17

Chapter 3 Theoretical Work

3.1. General	19
3.2. Mathematical Model of the Dynamic System	19
3.3. Derivation of the Overall Mass Moment of Inertia of a Robotic Arm	24
3.4. Kinematics of Robotic Manipulator	29
3.4.1. Forward Kinematics	30
3.4.2. Inverse Kinematics	30
3.5. Digital Image Processing.....	35
3.5.1. Image Enhancement	36
3.5.1.1. Image Sharpening	36
3.5.1.1.A. Image Differentiation	36
3.5.1.1.B. High-Pass Filtering	38
3.5.1.2. Image Smoothing	38
3.5.1.2.A. Random Noise	38
3.5.1.2.B. Noise As a Pattern	39
3.5.2. Image Segmentation	40
3.5.2.1. Segmentation Based on Color Information	40

3.5.2.2. Segmentation by Region Growing	40
3.5.2.3. Segmentation by Edge Detection	41
3.5.2.3.A. Gradient-Based Edge Detection	41
3.5.2.3.B. Laplacian Edge Detector	42
3.5.2.3.C. Best Fit Edge Detector	42
3.5.3. Image Analysis	43
3.5.3.1. Object Detection By Template Matching.....	44
3.5.3.2. Object Detection By Hough Transform	44

Chapter 4 Simulation and Control of the Robotic Arm

4.1. Introduction.....	48
4.2. Visual Servoing	48
4.2.1. Image Pre-Processing	49
4.2.2. Object Recognition	50
4.2.2.1. Hough Transform	51
4.2.2.2. Template Matching	54
4.3. The PID Controller	54
4.4. DC Motor Control	55
4.5. Solution of the Inverse Kinematics Problem	58
4.6. Simulation of the Path Planning Method	60
4.7. Simulation of the Robotic Arm Motion	62
4.8. Calculation of the Forces Acting On the Robot During the Motion	65

Chapter 5 Results and Discussion

5.1. Kinematics	69
5.2. Dynamics	86
5.3. Visual Servoing	97
5.4. The PID Controller	108

Chapter 6 Conclusions and Suggestions For Future Work

6.1. Conclusions.....	113
6.2. Suggestions for Future Work	114

References	116
Appendix A	123
Appendix B	129

Chapter One

Introduction

Introduction

Today industrial robots have become indispensable in industries whose products are massive and of high quality. They replace humans in many applications. For example, they perfectly accomplish repetitive tasks like electronic components planting, where humans cannot compete with them. Also, robots can replace humans in going to hard-to-reach areas where dangerous environments are there work-space. Hazardous material handling, mines and explosives detection and disarming and working in radio-active regions are common examples. Automobile assembly process is highly dependant upon robotic arms for mass production. Even quality-control made use of robots to accept or refuse products. Here they adopt TV cameras to monitor the products coming out of the production line and use intelligent image processing techniques to enable the robotic quality controller to make a decision to accept or refuse a product sample.

The field of robotics is a multi-discipline engineering field. It incorporates electrical, electronics, and mechanical engineering. This makes it a fertile research area in engineering.

1.1. Types of Robots

There are many types of robots in service today each type is defined by the geometry it takes. Frank L. Lewis et al [1] classified the robots as explained below:

a- Articulated Arms: These robots have only revolute axes. They can have up to six joints (six degrees of freedom). Of these joints, the second and the third joints are co-planar and move mutually to produce motion in the vertical plane. The first axis is the base and moves around a vertical axis to allow the arm to sweep the whole assembly around. The last three joints are used for the

orientation of the end effector. Figure (1-1) shows an example of a robot of this type.



Figure (1-1) articulated arm robot.

b- Type I SCARA: This type of robots usually has four axes. The first axis is a revolute joint at the base for assembly rotation. The second axis is a prismatic joint to provide a vertical movement of the last two revolute joints which are coplanar and move in a horizontal plane. This robot is used in the assembly of electronic components and devices and can be used for the assembly of small to medium sized mechanical assemblies. Figure (1-2) shows one robot of this type.

c- Type II SCARA: This type of robots also has four axes but differs from TYPE I in that the first axis is prismatic and is responsible for lifting the assembly in the vertical direction and the second axis rotates the last two coplanar in the horizontal plane, see figure (1-3).



Figure (1-2) type-I SCARA robot.



Figure (1-3) type-II SCARA robot.

d- Cartesian Coordinates Robots: This type of robots uses orthogonal prismatic axes, referred to as x , y and z , to their end-effector or pay load through

their rectangular work space, as shown in figure (1-4). Additional revolute axes can be added to the end-effector for orientation purposes. They are commonly used for material handling applications where large loads are to be handled. They can also be used for arc welding applications, water jet cutting and inspection of large complex precision parts.

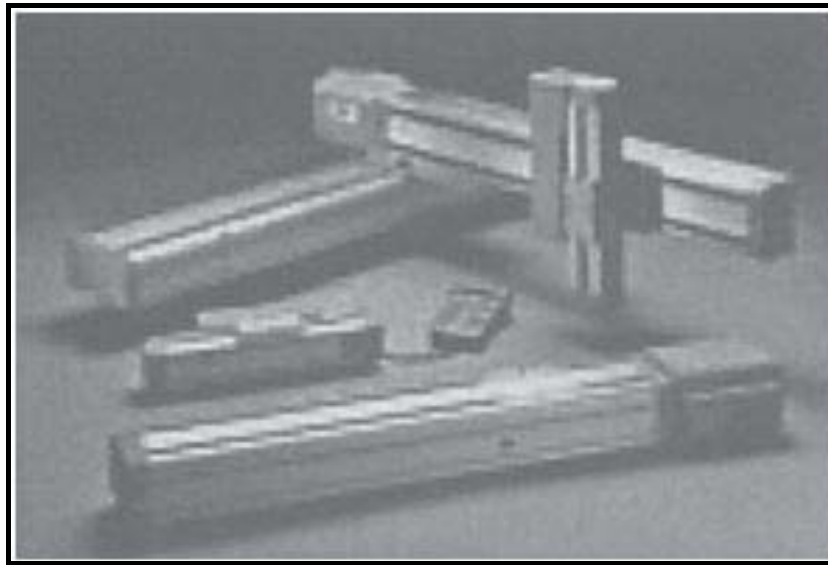


Figure (1-4) a Cartesian coordinates robot

e- Spherical and Cylindrical Coordinates Robots: The first two axes of the spherical coordinate robot are revolute and orthogonal to each other. The third axis provides prismatic radial extension. The result is a spherical coordinate system with spherical work space. The first axis of the cylindrical robot is a revolute base rotation. The second and third axes are prismatic, resulting in cylindrical work space.

f- Parallel-Link Manipulators: These robots generally have three or six links in parallel and each link is attached to a fixed base from one end and to a moving platform from the other end. These robots have greater stiffness and precision than the serial-link robots because in the later, the error is compound as one moves from the base outwards. These robots have been used in

machining and surface finishing of precision industrial and aerospace components.

1.2. Visual Servoing Systems

Visual servoing systems are categorized by Danica Kraig and Henrik L. Christensen [2] with respect to the relative position of the end-effector position and the position of the visual sensor (the camera or cameras) as:

a- Monocular eye-in-hand configuration: In this configuration one camera is located on the end-effector. The feedback visual data is a 2D image and all the measurements are based on this 2D information.

b- Monocular stand-alone configuration: In this configuration one camera is located at a fixed location relative to the robot and all measurements are referenced to this location.

c- Binocular eye-in-hand configuration: In this configuration two cameras are located in a stereoscopic setup. Different from the monocular eye-in-hand configuration, this configuration allows 3D measurements. This means that the distance of the object from the robot is measurable and the movement along the optical axis of the camera, based on these 3D measurements, is possible.

d- Binocular stand-alone configuration: Just like the previous configuration, this configuration also makes use of the advantage of the stereoscopic 3D measurement. The stereoscopic setup is located at a fixed point relative to the robot and all the measurements are referenced to this location.

e- Redundant camera system: This configuration can be considered as a hybrid configuration. It incorporates the previously mentioned configurations and the accuracy is highly increased on the expense of long processing time since the data to be analyzed are acquired from *many* cameras.

1.3. Robotic Manipulator Control

Robots are basically positioning and handling devices. Therefore a useful robot is the one that is able to control its movement and the interactive forces

and torques between the robot and its environment [1]. To control a robot, a mathematical model that describes the behavior of the robot as a dynamic system is needed to be derived and some kind of intelligence to act on the model. The mathematical model is obtained from the basic physical laws governing its motion. The intelligence requires sensory capabilities and means for acting and reacting to the sensed variables. These actions and reactions are the basic features used for the controller design.

Depending upon the type of the robot and the task it was designed to do, many types of controllers are available, for example, a *Proportional plus Derivative* (PD) controller can be used for robot manipulators that work in a horizontal plane. This type of controllers is simple to implement and can give accurate end-effector positioning but, as previously mentioned, it only works for planar robots [3]. Thus, for accurate positioning of a general 3D workspace robot manipulator, an integral part must be added and a *Proportional plus Integral plus Derivative* (PID) controller results. PID controllers are widely used in robot control with very good accuracy. In general, these controllers can be implemented either as a built-in controller, that is, a fixed hardware controllers attached to the robotic system, or, using a personal computer, these controllers may be software-implemented controllers. The later are mostly preferred because of there flexibility to modification and simplicity in implementation as software algorithm.

In this work, a vision-based control scheme was presented for a four-link, articulated joints robotic manipulator. This robotic arm is the Mitsubishi RM-501 robot which is located at the Robotics Lab that belongs to the department of mechanical engineering, college of engineering at the University of Babylon. The dynamics and the kinematics of this robotic arm were studied.

This study will help in developing a modern controller for this robot to increase its ability to do more general tasks. The study of the robot's inverse kinematics helps in transforming each workspace location into its corresponding

robot joint space vector. The study of the dynamics of this type of robots helps in determining the forces and torques acting on the robot while it performs some specific task.

The Objective of the Present Work

- i)** Designing a vision-based controller for the Mitsubishi RM-501 robotic arm using image processing techniques to enable the robotic arm to track the motion of an object in front of the camera. The eye-in-hand configuration was used.
- ii)** Deriving a dynamic model for this robot using Lagrange-Euler approach.
- iii)** Studying the robotic arm kinematics and presenting a new method for solving the inverse kinematics problem.

Chapter Two

Literature Review

Literature Review

2.1. General

Vision guided robotics has been one of the major research issue for decades ago. Visual servoing is concerned with several fields of research including vision systems, robotics and automatic control. However, without a sophisticated method for solving the inverse kinematics of the robotic arm under study, the control process of the robotic arm becomes impossible because the solution of the inverse kinematics problem is the key for determining the control signals to the robotic arm actuators at each joint. Because of this rigid relation between the control of any robotic system and its inverse kinematics, this chapter was divided into two main sections, the vision-based controllers and the methods traditionally used to solve the robotic arm inverse kinematics.

2.2. Vision-Based Control Systems

Bernard Espiau et al [4] presented a vision-based control system. The goal was to position the camera with respect to a target. They only considered the case where the vision-based task consisted in regulating the camera around the desired target position and ignored the trajectory through which the end-effector traces to reach the desired position.

Friedrich Lange et al [5] proposed a predictive control approach. In each image, two points were localized to predict the next control time step of the desired path. This was used for a learned feed forward controller to follow the desired trajectory.

Rahul Singh et al [6] proposed an approach for vision-based control system. In particular the problem of grasping different objects using a robotic arm with a single camera mounted on the end-effector. In this approach the recognition of the

object and thereby the identification of its grasp position, the subsequent translational and rotational pose alignment of the manipulator with the object and its movement in depth are controlled by using image morphing. They used a framework where the image of each object at a grasping pose was stored in a database. Given an unknown object in the workspace, its identity is established by morphing its contours to the shapes stored in the database and using a quantification of the morph as a dissimilarity measure. From the morph, a sequence of virtual images is obtained which describes the progressive transformation of the input image to the template image. These images were used to guide the eye-in-hand robotic system to attain the desired orientation and height from which the grasp can be executed.

Philippe Martinet and Jean Gallice [7] showed that a 3D visual sensor extracting 3D features at video rate can be used. They developed a theoretical model for the pose parameters which simplified the control process. They also showed the controllability and stability of this control system and found the optimal control law which allows the sensor to follow a straight line during servoing. The control law was dependant only upon the desired and current locations of the observed object. This means that only the pose estimation algorithm had to be modified.

D. Kragic and H. Castleman [8] presented a system that used a three-camera vision system for grasping purpose. Two cameras were arranged as a stereo vision system with parallel optical axis. The third camera was used to observe the robot workspace vertically from above. Color-based segmentation method used to determine the position of the object in the image.

Geraldo Selviera et al [9] addressed the issue of lateral control of an outdoor robotic airship while allowing a predefined path using visual servoing techniques. The vision-based control application was based on an interaction matrix

related to line features extracted from on-board camera as feedback signals. An output controller for the visual servo control was designed using the lateral dynamic model of the airship.

Mana Saedan and H. Ang Jr. [10] investigated the relative target-object pose estimation for vision-based control. A closed form target pose estimation algorithm was developed and implemented. Also a PI-based visual control was designed to minimize the effects of errors in the parameters of the camera. This vision-based control algorithm was implemented on a seven-degree-of-freedom industrial robot.

Christoph Theis et al [11] described a straight forward technique for tracking a human hand based on images acquired by an active stereo camera system. In this man-machine interaction control system, color-based image segmentation method was used to detect the human hand. Then a recognition algorithm was adopted to decide to what object the hand was pointing. Finally the robot was servoed to grasp that object.

Tom Drummond [12] presented a framework for three-dimensional tracking system. Graphical rendering technology was combined with active contour tracking to create a wire-frame tracking system. It operated in real-time at video frame rate of 25 Hz. It was based on an internal CAD model of the object to be tracked. The visible edge features were identified online at each frame and correspondences were found in the video feed. The researchers described how this tracking system can be extended for tracking purposes in complex configurations such as a multi-camera tracking system.

Robert Mahony et al [13] were concerned with choosing image features for image-based visual servo control and how this choice would influence the closed-loop dynamics of the system. They focused on the depth axis control of a visual

servo system and compared their approach with various approaches of other researchers.

H. Hector et al [14] described a strategy for computing the motions of a mobile robot operating in a 2D workspace without a prior knowledge of the target's intention or the distribution of the obstacles in the scene. The proposed algorithm governed the motion of the observer based on current measurement of the target's position and the location of the local obstacles. The algorithm was also used to compute the continuous control law based on a description of the geometric arrangement between the target and the observer's visible region.

Yan Fei Liu et al [15] studied the timing model for a robotic manipulator control. By timing model they referred to the delays between reality (when the part is sensed), through data processing (applying image processing algorithms to locate the part), through arrival (when the robot reaches the desired goal). In their study they introduced a timing model where sensing and control operated asynchronously. They applied this model to the RX-130 industrial robot manipulator by using six cameras for sensing and the main controller was the MV-19.

Tomoyuki Shiozaki and Toshiyuki Murakami [16] proposed a control model for a wheeled mobile robot without changing the camera view point in a hand-eye system. The camera was fixed at an initial position for a target object by visual servoing. Then the mobile robot switched back within reach of the hand-eye system. With this system the mobile robot was free to move and this movement did not influence the line of sight of the camera on board it.

Aman Betal et al [17] considered the camera-space position and orientation regulation problem for the camera-in-hand problem via visual servoing in the presence of parametric uncertainty associated with the robot dynamics and the camera system. Specifically, an adaptive robot controller was developed to force

the end-effector of a robot manipulator to move such that the position and orientation of an object are regulated to a desired position and orientation in the camera-space, despite parametric uncertainty throughout the entire robot camera system. An extension was also provided that illustrated how slight modifications can be made to the camera-in-hand control law to achieve adaptive position and orientation tracking of the end-effector in the camera-space for a fixed-camera configuration. Simulation results were provided to illustrate the performance of the adaptive camera-in-hand controller.

Ahmet Denker and Tugrul Adiguzel [18] presented a solution for a robotic manipulation problem in industrial settings. The problem was sensing objects on a conveyor belt, identifying the target, planning and tracking an interception trajectory between end effector and the target. Such a problem could be formulated as combining object recognition, tracking and interception. For this purpose, the researchers integrated a vision system to the manipulation system and employed tracking algorithms. The control approach was implemented on a real industrial manipulation setting, which consisted of a conveyor belt, objects moving on it, a robotic manipulator, and a visual sensor above the conveyor. The trajectory for robotic interception at a rendezvous point on the conveyor belt was analytically calculated. Test results showed that tracking the target along this trajectory resulted in interception and grabbing of the target object.

L. Peterson et al [19] introduced the Smart Cars Project at the Australian National University/National ICT Australia, together with a discussion and an example of a driver assistance system. They presented a framework for interactive driver assistance systems that included techniques for fast-speed sign detection and classification, obstacle detection and tracking applied to pedestrian detection, and lane departure warning. In addition, the driver's actions were monitored. The

integrated system used information extracted from the road scene (speed signs, position within the lane, relative position to other cars, etc.) together with information about the driver's state, such as eye gaze and head pose, to issue adequate warnings. A touch screen monitor was used to present relevant information and allow the driver to interact with the system.

Matthew R. Parks [20] described a complete algorithm to estimate the motion of a fixed-wing aircraft given a series of digitized flight images. The algorithm was designed for fixed wing aircraft because carefully procured flight images and corresponding navigation data were available to the researcher for testing. After image pre-processing, optical flow data was determined by automatically finding and tracking good features between pairs of images. The image coordinates of matched features were then processed by an optical flow-motion estimation algorithm. Input factors were weighed to provide good testing techniques. Error analysis was performed with simulation data keeping these factors in mind to determine the effectiveness of the optical flow algorithm. The output of this program was an estimate of rotation and translation of the imaged environment in relation to the camera, and thereby the airplane. Real flight images from NASA test flights were used to confirm the accuracy of the algorithm. The estimated motion parameters were compared with recorded flight instrument data to confirm the correctness of the algorithm. The results showed that the algorithm was accurate to within a degree provided that enough optic flow feature points are tracked.

S. Voros et al [21] developed a new method to detect instruments in Laparoscopic images. This method used information on the 3D position of insertion point of an instrument into the abdominal cavity. This information strongly constrains the search for the instrument in each endoscopic image. This made it possible to detect the instrument in near real-time using shape consideration.

Yasushi Iwatani et al [22] proposed a visual tracking method which was robust to occlusion. A set of extracted image features was chosen and used to estimate all the image features in the selection step. This enabled the vision-based control system to track image features even when non-predicted occlusion occurs. Four cameras were used each two of them were connected to a computer. The two computers (called client computers) were linked to a server computer. The client computers executed image feature extraction by binarizing two of the of the images at each time step. Image features were extracted in windows and the estimate was made by the server computer. The control signals are supplied by the server computer through a digital-to-analogue converter.

Hesheng Wang et al [23] presented an image-based visual servoing system for a robotic manipulator with an eye-in-hand setup. The parameters of the camera were not calibrated and the 3D coordinates of the features (which were points and lines) were unknown. They showed that it was possible to parametrize the close loop dynamics of the system even when these parameters are unknown. An algorithm was developed to make estimations of the unknown parameters. By minimizing the errors between the real and estimated projections of the features on multiple images captured during the motion of the robot, this algorithm guarantees the convergence of the estimated parameters to the real values.

Mechael Sauvee et al [24] presented a learning algorithm that neither requires nor tries to build a 3D model of the object. Given two (or more) images of an object, the algorithm attempts to identify a few points in each image corresponding to good locations at which to grasp the object. This set of points is then triangulated to obtain a 3D location at which to attempt a grasp. This is in contrast to standard dense stereo, which tries to triangulate every single point in an image (and often fails to return a good 3D model). This algorithm for identifying grasp locations from an image was trained by means of supervised learning, using

synthetic images for the training set. The researchers demonstrated this approach on two robotic manipulation platforms. The algorithm has successfully used to grasp a wide variety of objects, such as plates, tape rolls, jugs, cell phones, keys, screwdrivers, staplers, a thick coil of wire, a strangely shaped power horn and others, none of which were seen in the training set. They also applied the method to the task of unloading items from dishwashers.

Ashutosh Saxena et al [25] investigated the control of an instrument motion based on ultrasound images through non-linear predictive control. The new scheme ensured the convergence of the instrument to the desired position and also offered the possibility of satisfying constraints such as joint limits, actuator saturation and visibility preserving. The experiments were conducted on a PA-10 robot.

2.3. Inverse Kinematics

Inverse kinematics of the serial robotic arms is a very complicated problem because it involves highly nonlinear trigonometric functions. This makes it difficult to obtain one acceptable solution among infinitely many possible solutions. In fact the design of a robotic arm is limited by the availability of an inverse kinematics solution for this robotic arm. Most researchers avoid the numerical or the analytical methods to solve the robotic arm inverse kinematics problem. Numerical approaches suffer from the drawback that they are not guaranteed to converge (if they converge at all) to a practically acceptable solution. Analytical approaches, on the other hand, involve the solution of nonlinear equations which is usually a complicated task. This has led the researchers to seek other approaches to obtain acceptable solutions for different types of robotic arms. Artificial neural networks (ANN), the Genetic Algorithms (GA) and Fuzzy Logic Reasoning are the alternatives commonly used to solve the inverse kinematics of robotic

manipulators. These methods fall in the category of Artificial Intelligence Techniques which simulate the biological information processing of human beings.

Sang-Bae Lee [26] showed that the inverse kinematics problem can be solved by using fuzzy logic reasoning instead of trying to solve the highly nonlinear equations of the robotic arm analytically. They obtained good control efficiency and sufficient precision, minimum tracking error through simulating a two-degree-of-freedom robotic manipulator.

Gianluca Antonelli et al [27] presented an algorithmic approach for solving the inverse kinematics problem for the case of joints velocity and acceleration limitations. The proposed technique was based on an inverse kinematics algorithm which enables the handling of the velocity and acceleration constraints while the robotic arm is tracking some desired path.

LI-XIN WEI et al [28] used the ANFIS (Adaptive Neuro-Fuzzy Inference Systems) neural networks to solve the robotic arm inverse kinematics problem. Simulation results indicated that this method has the advantage of faster learning rate, higher identification precision.

Eimei Oyama et al [29] proposed a novel neural network system for solving the inverse kinematics problem for robotic arms with joint displacement limits. The inverse kinematics for this type of robots is a multi-valued and discontinuous function and using a single neural network cannot solve the inverse kinematics. So, the researchers used a number of expert modules where each expert module approximates a continuous part of the inverse kinematics function. The proposed system selects one appropriate expert whose output minimizes the expected

position-orientation error of the end-effector of the robotic arm. The system can learn a precise inverse kinematics model of robotic arms with equal or more degrees of freedom than that of its end-effector.

Yugui Yang et al [30] presented a new approach to solve the inverse kinematics problem based on neural networks. The least squares method was used to obtain a set of configurations that are of minimum error then, with the application of generic algorithms, a search is done for all the optimum solutions. These optimum solutions are considered as the training set that was used to train the neural network. Simulation showed that the method was feasible and provided a solution to the inverse kinematics problem for robotic arms with any redundancy.

Wuxin Huang et al [31] presented a solution to the inverse kinematics problem based on the Immune Genetic Algorithm (IGA). In their work, the researchers combined the features of the genetic algorithms with the biological immune idea. The human immune system, as an important component of the human information processing system, has many information processing mechanisms and functions which provide important inspiration in the designation and development of new intelligent computing models. Experimental results showed that the solution of the inverse kinematics problem obtained by this method is practically accepted for its high convergence speed and high accuracy.

2.4. Summary:

Concerning the vision-based control researches, the authors did not pay enough attention to the robot inverse kinematics and focused on designing a control system that enables the robot to be visually interactive with its workspace. They used many digital image processing algorithms to accomplish this task.

The solution of the inverse kinematics of the robot is very important to accomplish the control process. The researchers assumed that the inverse kinematics of the robotic system was known in advance. In section 2.3, the traditional techniques for obtaining a solution to this problem were presented. Although they produced practically acceptable inverse kinematics solutions, they are not free of drawbacks. One of the most disadvantages of these methods is the complexity of the training phase of the networks. This phase usually takes a long period of time and the network under consideration is learnt a small number of sample data compared to the huge points accessible by the robotic manipulator within its workspace. This limits the generality of the method to solve the inverse kinematics problem for any desired path.

In the present work, a new technique was used to solve the inverse kinematics problem of a four-link robotic manipulator. In this method a geometrical approach was used to solve the inverse kinematics problem. It gathers both features of simplicity and accuracy in obtaining the solution. It can be generalized to robotic manipulators of any degree of freedom. As the results showed, it was efficient and gave accurate end-effector path following performance. Also, with the use of sophisticated computer hardware, the method can give real-time solutions to the inverse kinematics of the robotic manipulator which is a very important feature of the method for the purpose of the manipulator control process.

Chapter Three

Theoretical Work

Theoretical Work

3.1. General

Mathematical modeling is the process of building a mathematical relation that describes the system behavior, when the later is affected upon by some inputs. This mathematical relation predicts the response of the physical system to those inputs.

Mathematical models are usually derived using one of two techniques [3]

- 1- The analytical technique.
- 2- The experimental technique.

The analytical approach uses the fundamental laws of physics to describe the problem. The accuracy of the resulting system is highly dependent upon the assumptions made by the designer to simplify the original physical system. The more realistic assumptions, the more accuracy is gained. However, since the models are usually highly non-linear differential equations, a close form solution is impossible to be achieved without such assumptions. So, an optimization must be made between the accuracy of the resulting model and the simplicity of the solution of the differential equations that describe the model.

In the experimental approach, on the other hand, the designer sends certain inputs to the system and measures the response. According to the data collected, the designer can describe the system using a model that responds the same way to similar inputs. Because the resulting model was chosen depending on a limited data set, the system overall performance is more inaccurate than the system derived by the first approach. However, this technique has the advantage of being much easier and quicker to obtain than the analytical approach.

3.2. Mathematical Model of the Dynamic System

For control design purposes, it is necessary to have a mathematical model that best describes the dynamical behavior of a system. In this section the

dynamic equations of motion for an n-degree-of-freedom robot manipulator have been derived. The approach is to derive the Kinetic and Potential energy functions of the robotic manipulator and then the Lagrangian was used to derive the required equations of motion of the dynamic system. The Lagrangian is given by equation (3.1) as [3]:

$$L(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q}) \quad \text{..... (3.1)}$$

Where L is the Lagrangian, T is the Kinetic energy function and U is the Potential energy function.

The equations of motion describing the behavior of the robotic manipulator can be derived using the so called Lagrange-Euler equation [3]:

$$\frac{d}{dt} \left[\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} \right] - \frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \mathbf{q}} = \boldsymbol{\tau} \quad \text{..... (3.2)}$$

Where $\mathbf{q}$, $\dot{\mathbf{q}}$ are the $[n \times 1]$ vectors of joint angles and joint angular velocities, respectively and $\boldsymbol{\tau}$ is the vector of the external torques necessary to drive each joint.

The dynamic model of a robot is a set of ordinary non-linear differential equations. This set has the general form [3]:

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \quad \text{..... (3.3)}$$

Where M is the mass matrix of the system,

C is the matrix of the centrifugal and coriolis torques, and

$\mathbf{g}$ is the vector of gravitational torques.

The set of the equations of motion for the robotic manipulator in matrix form in equation (3.3) can be derived using equation (3.1) and equation (3.2) as described below:

Consider the four-link robotic manipulator that is shown in figure (3-1). A set of four ordinary non-linear differential equations were derived to describe the behavior of the system. The Kinetic and Potential energy functions were derived for each link individually and then the Lagrange-Euler equation (3.2)

was used to derive the system's equations of motion. The method of derivation is shown below:

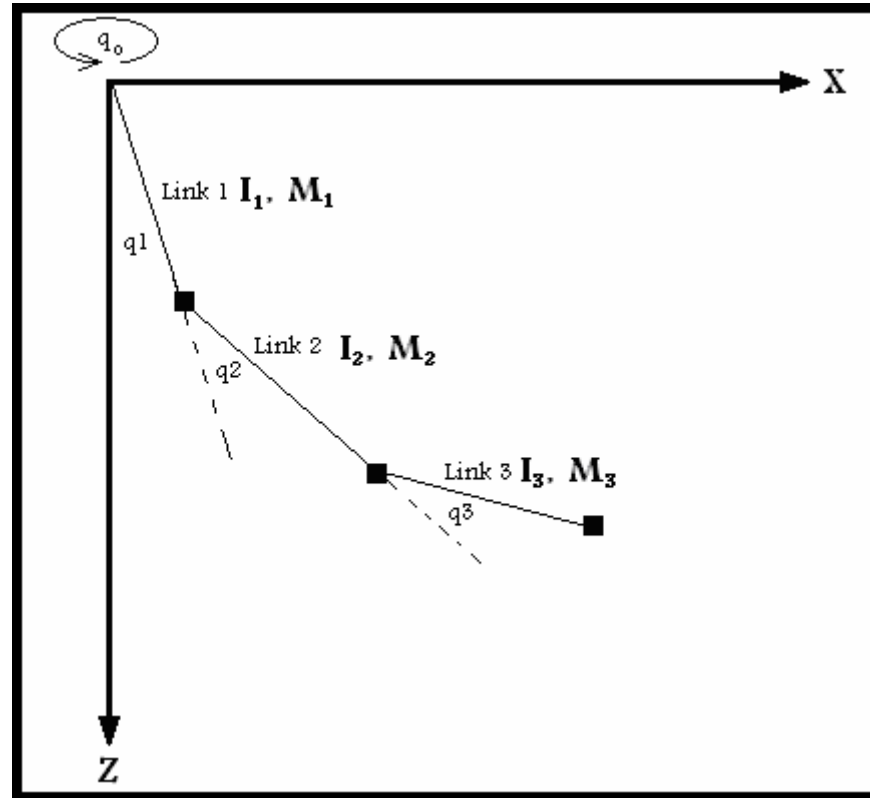


Figure (3-1) schematic diagram of a 4-DOF robot manipulator

The base (link 0), which is taken as the datum for the calculations, is free to rotate about the generalized Z - axis. So, it has only a Kinetic energy function (the Potential energy function is zero). For the other links, in general, the i^{th} link rotates about its axis of rotation.

The method of deriving the Potential and Kinetic energy functions is illustrated in appendix A. According to this method, the Kinetic and Potential energy functions for the four links were found to be:

- Kinetic Energy Function for the base (link #0): The base has only a rotational kinetic energy and it is given by:

$$T_o = \frac{1}{2} I_{ass} \dot{q}_o^2 \quad \dots\dots\dots (3.4)$$

$$U_o = 0 \quad \dots\dots\dots (3.5)$$

- Kinetic & Potential Energy functions for link #1:

$$T_1 = \frac{1}{2} m_1 l_{c_1}^2 \dot{q}_1^2 + \frac{1}{2} I_1 \dot{q}_1^2 \quad \dots\dots\dots (3.6)$$

$$U_1 = -m_1 l_{c_1} g \cos(q_1) \quad \dots\dots\dots (3.7)$$

- Kinetic & Potential Energy for Link #2:

$$T_2 = \frac{1}{2} m_2 l_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 l_{c_2}^2 (\dot{q}_1 + \dot{q}_2)^2 +$$

$$m_2 l_1 l_{c_2} \dot{q}_1 (\dot{q}_1 + \dot{q}_2) \cos(q_2) + \frac{1}{2} I_2 (\dot{q}_1 + \dot{q}_2)^2 \quad \dots\dots\dots (3.8)$$

$$U_2 = -m_2 l_1 g \cos(q_1) - m_2 l_{c_2} g \cos(q_1 + q_2) \quad \dots\dots\dots (3.9)$$

- Kinetic & Potential Energy for Link #3:

$$T_3 = \frac{1}{2} m_3 l_1^2 \dot{q}_1^2 + \frac{1}{2} m_3 l_2^2 (\dot{q}_1 + \dot{q}_2)^2 +$$

$$\frac{1}{2} m_3 l_{c_3}^2 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3)^2 +$$

$$m_3 l_1 l_2 \dot{q}_1 (\dot{q}_1 + \dot{q}_2) \cos(q_2) +$$

$$m_3 l_1 l_{c_3} \dot{q}_1 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \cos(q_2 + q_3) +$$

$$m_3 l_2 l_{c_3} (\dot{q}_1 + \dot{q}_2) (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \cos(q_3) +$$

$$\frac{1}{2} I_3 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3)^2 \quad \dots\dots\dots (3.10)$$

$$U_3 = -m_3 l_1 g \cos(q_1) - m_3 l_2 g \cos(q_1 + q_2)$$

$$-m_3 l_{c_3} g \cos(q_1 + q_2 + q_3) \quad \dots\dots\dots (3.11)$$

- The total Kinetic and Potential energy functions of the robotic arm are:

$$T_{total} = T_o + T_1 + T_2 + T_3 \quad \dots\dots\dots (3.12)$$

$$U_{total} = U_o + U_1 + U_2 + U_3 \quad \dots\dots\dots (3.13)$$

After the substitution of equations (3.12) and (3.13) in equation (3.1), the equations that describe the behavior of the 4-DOF robotic arm were found as:

$$\boxed{I_{ass} \ddot{q}_o + \frac{\partial I_{ass}}{\partial t} \dot{q}_o = \tau_o} \quad \dots\dots\dots (3.14)$$

$$\begin{aligned} & [m_1 l_1^2 + I_1 + m_2 l_1^2 + m_2 l_{c_2}^2 + 2 m_2 l_1 l_{c_2} \cos(q_2) + I_2 + m_3 l_1^2 + m_1 l_2^2 + I_3 + \\ & m_3 l_{c_3}^2 + 2 m_3 l_1 l_2 \cos(q_2) + 2 m_3 l_1 l_{c_3} \cos(q_2 + q_3) + 2 m_3 l_2 l_{c_3} \cos(q_3)] \ddot{q}_1 + \\ & [m_2 l_{c_2}^2 + m_2 l_1 l_{c_2} \cos(q_2) + I_2 + m_3 l_2^2 + m_3 l_{c_3}^2 + m_3 l_1 l_2 \cos(q_2) + m_3 l_1 l_{c_3} \cos(q_2 + q_3) + \\ & 2 m_3 l_2 l_{c_3} \cos(q_3) + I_3] \ddot{q}_2 + \\ & [m_3 l_{c_3}^2 + m_3 l_1 l_{c_3} \cos(q_2 + q_3) + m_3 l_2 l_{c_3} \cos(q_3) + I_3] \ddot{q}_3 \\ & - 2 m_2 l_1 l_{c_2} \sin(q_2) \cdot \dot{q}_1 \cdot \dot{q}_2 - m_2 l_1 l_{c_2} \sin(q_2) \cdot \dot{q}_2^2 - m_3 l_1 l_2 \sin(q_2) \cdot \dot{q}_2^2 - 2 m_3 l_1 l_{c_3} \sin(q_2 + q_3) \cdot \dot{q}_1 \cdot (\dot{q}_2 + \dot{q}_3) \\ & - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \cdot \dot{q}_2 \cdot (\dot{q}_2 + \dot{q}_3) - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \cdot \dot{q}_3 \cdot (\dot{q}_2 + \dot{q}_3) - 2 m_3 l_2 l_{c_3} \sin(q_3) \cdot \dot{q}_1 \cdot \dot{q}_3 \\ & - 2 m_3 l_2 l_{c_3} \sin(q_3) \cdot \dot{q}_2 \cdot \dot{q}_3 - m_3 l_2 l_{c_3} \sin(q_3) \cdot \dot{q}_3^2 + \\ & \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_1} + [m_1 l_{c_1} + m_2 l_1 + m_3 l_1] g \sin(q_1) + [m_2 l_{c_2} + m_3 l_2] g \sin(q_1 + q_2) + \\ & m_3 l_{c_3} g \sin(q_1 + q_2 + q_3) = \tau_1 \end{aligned} \quad \dots\dots (3.15)$$

$$\begin{aligned} & [m_1 l_1^2 + I_1 + m_2 l_1^2 + m_2 l_{c_2}^2 + 2 m_2 l_1 l_{c_2} \cos(q_2) + I_2 + m_3 l_1^2 + m_1 l_2^2 + I_3 + \\ & m_3 l_{c_3}^2 + 2 m_3 l_1 l_2 \cos(q_2) + 2 m_3 l_1 l_{c_3} \cos(q_2 + q_3) + 2 m_3 l_2 l_{c_3} \cos(q_3)] \ddot{q}_1 + \\ & [m_2 l_{c_2}^2 + I_2 + m_3 l_2^2 + m_3 l_{c_3}^2 + 2 m_3 l_2 l_{c_3} \cos(q_3) + I_3] \ddot{q}_2 + \\ & [m_3 l_{c_3}^2 + m_3 l_2 l_{c_3} \cos(q_3) + I_3] \ddot{q}_3 \\ & - m_2 l_1 l_{c_2} \sin(q_2) \cdot \dot{q}_1 \cdot \dot{q}_2 - m_3 l_1 l_2 \sin(q_2) \cdot \dot{q}_1 \cdot \dot{q}_2 - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \cdot \dot{q}_1 \cdot (\dot{q}_2 + \dot{q}_3) \\ & - 2 m_3 l_2 l_{c_3} \sin(q_3) \cdot \dot{q}_1 \cdot \dot{q}_3 - 2 m_3 l_2 l_{c_3} \sin(q_3) \cdot \dot{q}_2 \cdot \dot{q}_3 - m_3 l_2 l_{c_3} \sin(q_3) \cdot \dot{q}_3^2 \\ & - m_2 l_1 l_{c_2} \sin(q_2) \cdot \dot{q}_1 \cdot (\dot{q}_1 + \dot{q}_2) - m_3 l_1 l_2 \sin(q_2) \cdot \dot{q}_1 \cdot (\dot{q}_1 + \dot{q}_2) - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \cdot \dot{q}_1 \cdot (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \\ & \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_2} - m_2 l_{c_2} g \sin(q_1 + q_2) - m_3 l_2 g \sin(q_1 + q_2) - m_3 l_{c_3} g \sin(q_1 + q_2 + q_3) = \tau_2 \end{aligned} \quad \dots (3.16)$$

$$\begin{aligned}
& [m_3 l_{c_3}^2 + m_3 l_1 l_{c_3} \cos(q_2 + q_3) + m_3 l_2 l_{c_3} \cos(q_3) + I_3] \ddot{q}_1 + \\
& [m_3 l_{c_3}^2 + m_3 l_2 l_{c_3} \cos(q_3) + I_3] \ddot{q}_2 + \\
& [m_3 l_{c_3}^2 + I_3] \ddot{q}_3 \\
& - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \cdot \dot{q}_1 \cdot (\dot{q}_2 + \dot{q}_3) - m_3 l_2 l_{c_3} \sin(q_3) \cdot \dot{q}_1 \cdot \dot{q}_3 - m_3 l_2 l_{c_3} \sin(q_3) \cdot \dot{q}_2 \cdot \dot{q}_3 \\
& - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \cdot \dot{q}_1 \cdot (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) - m_3 l_2 l_{c_3} \sin(q_3) \cdot (\dot{q}_1 + \dot{q}_2) \cdot (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \\
& + \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_3} - m_3 l_{c_3} g \sin(q_1 + q_2 + q_3) = \tau_3
\end{aligned}
\tag{3.17}$$

3.3. Derivation of the Overall Mass Moment of Inertia of a Robotic Arm

Equations (3.14), (3.15), (3.16) and (3.17) describe the motion of the robotic arm in the 3D work space. They contain the parameter of the mass moment of inertia of the robotic arm assembly about the axis of rotation of the base. It is designated as (I_{ass}) and its value is changing instantaneously at each time step during the motion. This is caused by the continuous change in the configuration of assembly of the links during the movement of the robotic arm around the Z-axis. This effect is shown in the figure (3-2). In the present work, an expression for (I_{ass}) was derived by calculating the mass moment of inertia for each link about the axis of rotation of the base [32], then the total mass moment of inertia of the whole assembly was calculated as the summation of the mass moment of inertia of all the links.

Consider the figure (3.3) below. For each link the mass moment of inertia can be formulated as follows:

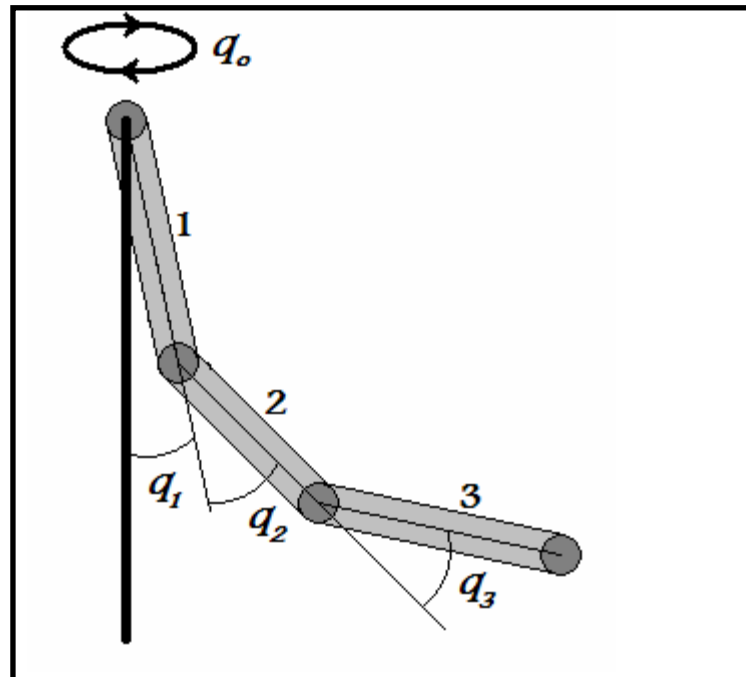


Figure (3-2) robotic arm linkage

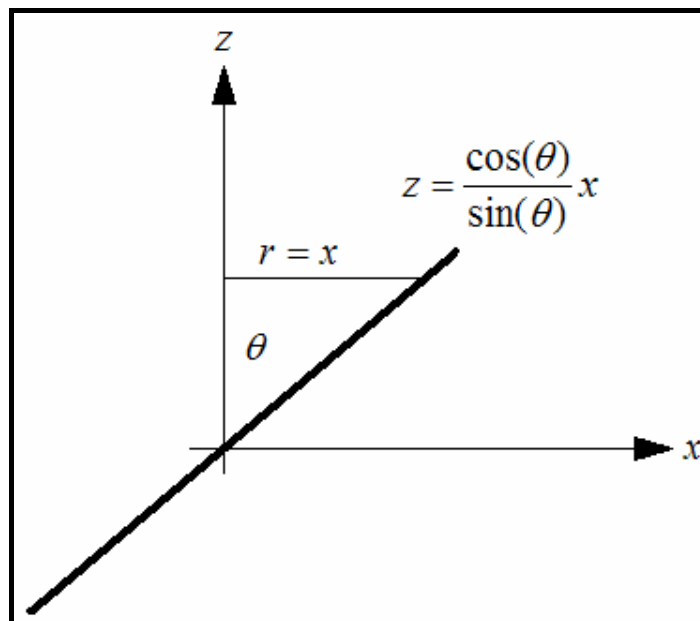


Figure (3-3) single link configuration

Taking an element on the link of mass (dm) per unit length, from figure (3-3):

$$\begin{aligned}
 dI &= r^2 dm \\
 &= x^2 \frac{M}{L} dl \quad \dots\dots\dots (3.18)
 \end{aligned}$$

$$\begin{aligned}
 dl &= \sqrt{1 + \left(\frac{dz}{dx}\right)^2} dx \\
 &= \sqrt{1 + \left(\frac{\cos(\theta)}{\sin(\theta)}\right)^2} dx \\
 &= \frac{1}{\sin(\theta)} dx
 \end{aligned}
 \quad \dots\dots\dots (3.19)$$

$$\begin{aligned}
 dI &= x^2 \cdot \frac{M}{L} \cdot \frac{1}{\sin(\theta)} dx \\
 I &= 2 \int_0^{\frac{L}{2}\sin(\theta)} x^2 \cdot \frac{M}{L} \cdot \frac{1}{\sin(\theta)} dx \\
 &= \frac{2}{\sin(\theta)} \cdot \frac{M}{L} \left[\frac{x^3}{3} \right]_0^{\frac{L}{2}\sin(\theta)} \\
 &= \frac{1}{12} ML^2 \sin^2(\theta)
 \end{aligned}
 \quad \dots\dots\dots (3.20)$$

Using the parallel axis theorem,

$$\begin{aligned}
 I_z &= \frac{1}{12} ML^2 \sin^2(\theta) + M \left(\frac{L}{2} \sin(\theta) \right)^2 \\
 &= \frac{1}{3} ML^2 \sin^2(\theta)
 \end{aligned}
 \quad \dots\dots\dots (3.21)$$

The mass moment of inertia for the n^{th} link about the Z-axis of the base (I_{nz}), after applying the above procedure, can be found to be:

- For link #1:

$$I_{1z} = \frac{1}{3} m_1 l_1^2 \sin^2(q_1)$$

- For link #2:

$$I_{2z} = \frac{1}{12} m_2 l_2^2 \sin^2(q_1 + q_2) + m_2 \left(\frac{l_2}{2} \sin(q_1 + q_2) + l_1 \sin(q_1) \right)^2$$

- For link #3:

$$\begin{aligned}
 I_{3z} &= \frac{1}{12} m_3 l_3^2 \sin^2(q_1 + q_2 + q_3) + \\
 &\quad m_3 \left(\frac{l_3}{2} \sin(q_1 + q_2 + q_3) + l_2 \sin(q_1 + q_2) + l_1 \sin(q_1) \right)^2
 \end{aligned}$$

- And for the whole assembly,

$$I_{ass} = I_{1z} + I_{2z} + I_{3z} \dots\dots\dots (3.22)$$

Equation (3.22) represents the instantaneous mass moment of inertia for the robotic arm about the axis of rotation of the base (Z -axis). Clearly it is a function of the joint angles which are changing at each time step during the movement of the robotic arm in the general 3D work space.

From the above equations, the partial derivatives appeared in the equations of motion of the robotic arm can be formulated as:

$$\frac{\partial I_{1z}}{\partial q_1} = \frac{2}{3} m_1 l_1^2 \sin(q_1) \cdot \cos(q_1)$$

$$\begin{aligned} \frac{\partial I_{2z}}{\partial q_1} = & \frac{1}{6} m_2 l_2^2 \sin(q_1 + q_2) \cdot \cos(q_1 + q_2) + \\ & 2m_2 \left(\frac{l_2}{2} \sin(q_1 + q_2) + l_1 \sin(q_1) \right) \cdot \\ & \left(\frac{l_2}{2} \cos(q_1 + q_2) + l_1 \cos(q_1) \right) \end{aligned}$$

$$\begin{aligned} \frac{\partial I_{3z}}{\partial q_1} = & \frac{1}{6} m_3 l_3^2 \sin(q_1 + q_2 + q_3) \cdot \cos(q_1 + q_2 + q_3) + \\ & 2m_3 \left(\frac{l_3}{2} \sin(q_1 + q_2 + q_3) + l_2 \sin(q_1 + q_2) + l_1 \sin(q_1) \right) \cdot \\ & \left(\frac{l_3}{2} \cos(q_1 + q_2 + q_3) + l_2 \cos(q_1 + q_2) + l_1 \cos(q_1) \right) \end{aligned}$$

$$\frac{\partial I_{ass}}{\partial q_1} = \frac{\partial I_{1z}}{\partial q_1} + \frac{\partial I_{2z}}{\partial q_1} + \frac{\partial I_{3z}}{\partial q_1} \dots\dots\dots (3.23)$$

$$\frac{\partial I_{1z}}{\partial q_2} = 0$$

$$\begin{aligned} \frac{\partial I_{2z}}{\partial q_2} = & \frac{1}{6} m_2 l_2^2 \sin(q_1 + q_2) \cdot \cos(q_1 + q_2) + \\ & 2m_2 \left(\frac{l_2}{2} \sin(q_1 + q_2) + l_1 \sin(q_1) \right) \cdot \left(\frac{l_2}{2} \cos(q_1 + q_2) \right) \end{aligned}$$

$$\begin{aligned} \frac{\partial I_{3z}}{\partial q_2} = & \frac{1}{6} m_3 l_3^2 \sin(q_1 + q_2 + q_3) \cdot \cos(q_1 + q_2 + q_3) + \\ & 2m_3 \left(\frac{l_3}{2} \sin(q_1 + q_2 + q_3) + l_2 \sin(q_1 + q_2) + l_1 \sin(q_1) \right) \cdot \\ & \left(\frac{l_3}{2} \cos(q_1 + q_2 + q_3) + l_2 \cos(q_1 + q_2) \right) \end{aligned}$$

$$\frac{\partial I_{ass}}{\partial q_2} = \frac{\partial I_{1z}}{\partial q_2} + \frac{\partial I_{2z}}{\partial q_2} + \frac{\partial I_{3z}}{\partial q_2} \quad \dots\dots\dots (3.24)$$

$$\frac{\partial I_{1z}}{\partial q_3} = 0$$

$$\frac{\partial I_{2z}}{\partial q_3} = 0$$

$$\begin{aligned} \frac{\partial I_{3z}}{\partial q_3} = & \frac{1}{6} m_3 l_3^2 \sin(q_1 + q_2 + q_3) \cdot \cos(q_1 + q_2 + q_3) + \\ & 2m_3 \left(\frac{l_3}{2} \sin(q_1 + q_2 + q_3) \right) \cdot \left(\frac{l_3}{2} \cos(q_1 + q_2 + q_3) \right) \end{aligned}$$

$$\frac{\partial I_{ass}}{\partial q_3} = \frac{\partial I_{1z}}{\partial q_3} + \frac{\partial I_{2z}}{\partial q_3} + \frac{\partial I_{3z}}{\partial q_3} \quad \dots\dots\dots (3.25)$$

$$\frac{\partial I_{1z}}{\partial t} = \frac{2}{3} m_1 l_1^2 \sin(q_1) \cdot \cos(q_1) \cdot \dot{q}_1$$

$$\begin{aligned} \frac{\partial I_{2z}}{\partial t} = & \frac{1}{6} m_2 l_2^2 \sin(q_1 + q_2) \cdot \cos(q_1 + q_2) \cdot (\dot{q}_1 + \dot{q}_2) + \\ & 2m_2 \left(\frac{l_2}{2} \sin(q_1 + q_2) + l_1 \sin(q_1) \right) \cdot \\ & \left(\frac{l_2}{2} \cos(q_1 + q_2) \cdot (\dot{q}_1 + \dot{q}_2) + l_1 \cos(q_1) \cdot \dot{q}_1 \right) \end{aligned}$$

$$\begin{aligned} \frac{\partial I_{3z}}{\partial t} = & \left(\frac{1}{6} m_3 l_3^2 \sin(q_1 + q_2 + q_3) \cdot \cos(q_1 + q_2 + q_3) \cdot (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \right) + \\ & 2m_3 \left(\frac{l_3}{2} \sin(q_1 + q_2 + q_3) + l_2 \sin(q_1 + q_2) + l_1 \sin(q_1) \right) \cdot \\ & \left(\frac{l_3}{2} \cos(q_1 + q_2 + q_3) \cdot (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) + l_2 \cos(q_1 + q_2) \cdot (\dot{q}_1 + \dot{q}_2) \right. \\ & \left. + l_1 \cos(q_1) \cdot \dot{q}_1 \right) \end{aligned}$$

$$\frac{\partial I_{ass}}{\partial t} = \frac{\partial I_{1z}}{\partial t} + \frac{\partial I_{2z}}{\partial t} + \frac{\partial I_{3z}}{\partial t} \quad \dots\dots\dots (3.26)$$

The substitution of equations (3.23) through (3.26), which represent the partial derivatives of (I_{ass}), into equations (3.14) through (3.17) completes the description of the dynamic model of the robotic arm.

3.4. Kinematics of Robotic Manipulators

Robot kinematics is the study of the motion of robots. In a kinematics analysis the position, velocity and acceleration of all the links are calculated without considering the forces that caused this motion. The relationship between motion and the associated torques (or forces) is studied in robot dynamics. Robot kinematics can be divided in serial manipulator kinematics, parallel manipulator kinematics, mobile robot kinematics and humanoid kinematics [33].

Serial robotic manipulator kinematics is concerned with the following two issues:

3.4.1. Forward Kinematics

Forward kinematics is also known as direct kinematics. In forward kinematics, the length of each link and the angle of each joint are given and the position of any point in the work volume of the robot should be calculated.

From Figure (3-4), the coordinates of point E are given by the equations below [28]:

$$X = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3) \quad \dots\dots\dots (3.27)$$

$$Y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) + l_3 \sin(q_1 + q_2 + q_3) \quad \dots\dots\dots (3.28)$$

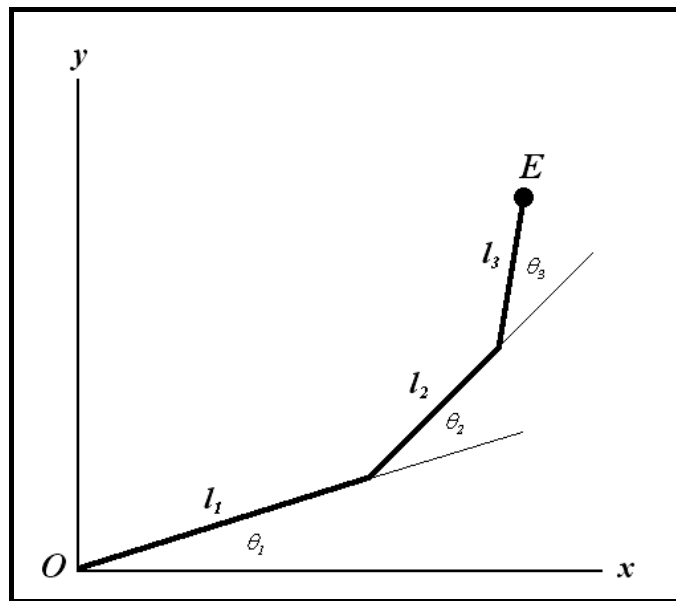


Figure (3-4) a 3 DOF robotic arm extended in plane.

Knowing the values of the joint angles q_1, q_2 and q_3 is sufficient to define exactly where the point E existed.

3.4.2. Inverse Kinematics

For the robotic manipulator to do some task, it must reach a specific point (or follow a specific path) in the space. This point (or points) may be located anywhere in the working space of the robot. So, for each point, the robot must

be reconfigured to reach that point. This is known as the inverse kinematics of the robotic manipulator.

The solution of the inverse kinematics problem is, generally, not always possible because it involves nonlinear trigonometric functions. This usually causes many possible solutions and, accordingly, many possible configurations (or postures) to be existed. This gives rise to the problem of which posture the robot should take to reach a given point in the space. As an example, let's take the case of a two-link planar robotic arm shown in figure (3-5).

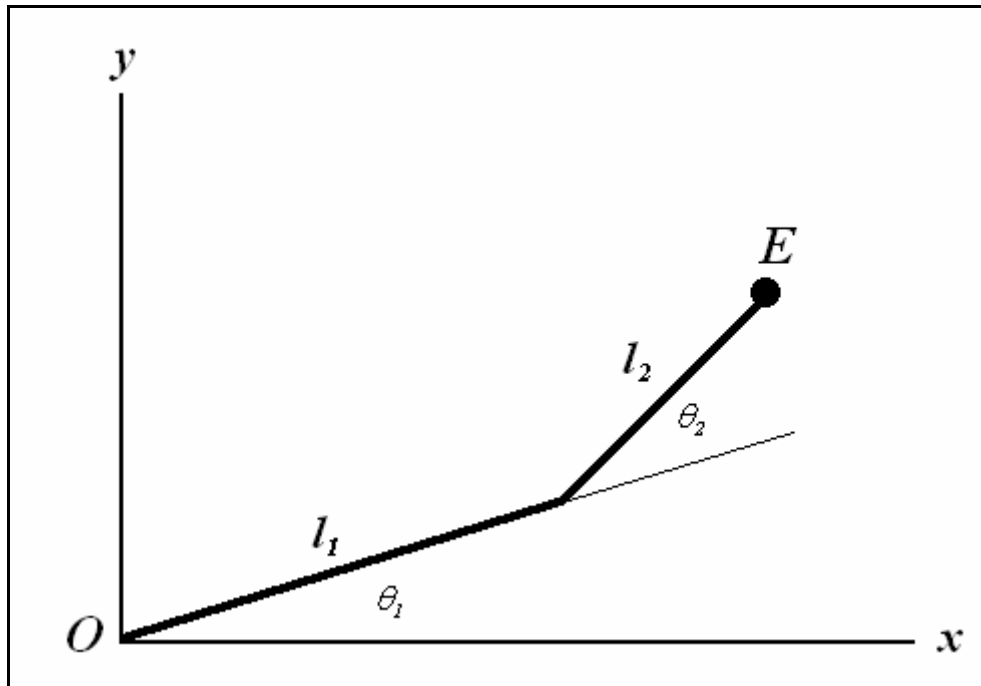


Figure (3-5) two-link robotic arm.

The kinematics equations of the end point E are:

$$X = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \quad \dots\dots\dots (3.29)$$

$$Y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \quad \dots\dots\dots (3.30)$$

With the aid of the equations $\cos^2(\theta_i) + \sin^2(\theta_i) = 1$, $i = 1, 2$, the solution is [28]:

$$\cos(q_1) = \frac{1}{X^2 + Y^2} \left[X(l_1 + l_2 \lambda) \pm Y l_2 (1 - \lambda^2)^{1/2} \right] \quad \dots\dots\dots (3.31)$$

$$\sin(q_1) = \frac{1}{X^2 + Y^2} \left[Y(l_1 + l_2 \lambda) \mp X l_2 (1 - \lambda^2)^{1/2} \right] \quad \dots\dots\dots (3.32)$$

$$\cos(q_2) = \lambda \quad \dots\dots\dots (3.33)$$

$$\sin(q_2) = \pm (1 - \lambda^2)^{1/2} \quad \dots\dots\dots (3.34)$$

Where

$$\lambda = \frac{1}{2l_1 l_2} \left[(X^2 + Y^2) - (l_1^2 + l_2^2) \right] \quad \dots\dots\dots (3.35)$$

Thus, two solutions arise for this simple planar robotic arm. The problem becomes more complicated when the robot was constructed with more links than two.

Another way to solve the inverse kinematics problem is the numerical method. Suppose that the kinematics equations describing a six-degree-of-freedom robotic arm are:

$$\begin{aligned} X &= f_1(q_1, \dots\dots\dots, q_6) \\ Y &= f_2(q_1, \dots\dots\dots, q_6) \\ Z &= f_3(q_1, \dots\dots\dots, q_6) \end{aligned} \quad \dots\dots\dots (3.36)$$

Where X , Y and Z are the desired end-effector position in the work space. It is possible to find a linear approximation to the set of nonlinear functions describing the inverse kinematics by taking small changes in the joint angles.

This results in the following expressions [34]:

$$\delta X = \frac{\partial f_1}{\partial q_1} \delta q_1 + \dots\dots\dots + \frac{\partial f_1}{\partial q_6} \delta q_6 \quad \dots\dots\dots (3.37)$$

$$\delta Y = \frac{\partial f_2}{\partial q_1} \delta q_1 + \dots\dots\dots + \frac{\partial f_2}{\partial q_6} \delta q_6 \quad \dots\dots\dots (3.38)$$

$$\delta Z = \frac{\partial f_3}{\partial q_1} \delta q_1 + \dots\dots\dots + \frac{\partial f_3}{\partial q_6} \delta q_6 \quad \dots\dots\dots (3.39)$$

These equations can be written as:

$$\Delta \mathbf{X} = \mathbf{J} \Delta \mathbf{q} \quad \dots\dots\dots (3.40)$$

The matrix **J** is known as the jacobian matrix [4] which is a matrix of partial derivatives:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \dots & \frac{\partial f_1}{\partial q_6} \\ \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial q_2} & \dots & \frac{\partial f_2}{\partial q_6} \\ \frac{\partial f_3}{\partial q_1} & \frac{\partial f_3}{\partial q_2} & \dots & \frac{\partial f_3}{\partial q_6} \end{bmatrix} \dots \dots \dots (3.41)$$

The jacobian matrix behaves very much like the first derivative of a function in one variable. For multiple-variable function, the Taylor's series becomes:

$$\mathbf{X} + \Delta \mathbf{X} \approx \mathbf{f}(\mathbf{q}) + \mathbf{J}(\mathbf{q}) \Delta \mathbf{q} \dots \dots \dots (3.42)$$

Using the Newton-Raphson method, equation (3.42) can be reformulated as:

$$\mathbf{f}(\mathbf{q} + \mathbf{h}) = \mathbf{f}(\mathbf{q}) + \mathbf{J}(\mathbf{q}) \cdot \mathbf{h} \dots \dots \dots (3.43)$$

When the vector of joint angles (**q**) is a solution (a root) for equation (3.43) then **f(q)**= 0 and the error (**h**) may be found to be:

$$\mathbf{h} \approx \mathbf{J}^{-1}(\mathbf{q}) \cdot \mathbf{f}(\mathbf{q} + \mathbf{h}) \dots \dots \dots (3.44)$$

The error (**h**) is defined as the difference between the angle vector in the i^{th} iteration and that in the $(i+1)^{st}$ iteration. That is:

$$\mathbf{h}^{(i)} = \mathbf{q}^{(i)} - \mathbf{q}^{(i+1)} \dots \dots \dots (3.45)$$

Thus the iteration scheme, equation (3.45), becomes:

$$\mathbf{q}^{(i+1)} = \mathbf{q}^{(i)} - \mathbf{J}^{-1}(\mathbf{q}^{(i)}) \cdot \mathbf{f}(\mathbf{q}^{(i)}) \dots \dots \dots (3.46)$$

Equation (3.46) is the Newton-Raphson formula for a function with multiple variables.

The above numerical technique gives one solution for each new end-effector position. Hence, numerical solution has an advantage over the analytic solution in that the later gives many solutions for the same end-effector location. However, a serious drawback of using numerical methods is that the resulting sequence of solutions for the sequence of end-effector locations is random. This,

consequently, results in unstable performance of the robotic arm and the path planning techniques are no longer applicable.

Neural networks approach is sometimes used to solve the inverse kinematics problem [35]. In this approach, a training data in the form of pairs of the end-effector's position and orientation and the corresponding joint values is generated. This data is used for training the NN with the position and orientation vector as the input and the corresponding joint values as the desired output.

The disadvantage of this method is that the training process is not so easy to implement because large amount of end-effector's position and orientation data is required, especially when the workspace is large. This limits the performance of the robot to a limited number of paths (one path per task).

In the present work a new technique is used to solve the inverse kinematics problem. It is based upon the equations describing the geometry of the robotic arm. This technique has proved its efficiency in overcoming the drawbacks of both the analytic and numerical solution methods. It produced one joint space vector for each end-effector position and, when a path was desired to be followed, the method gave a smooth transition between every two consecutive joint space configurations. A comparison between this method for solving the inverse kinematics problem and the numerical solution of the inverse kinematics is given in chapter four (section 4.4).

The solution of the inverse kinematics problem produces a vector of joint angles that constitutes the joint space vector that corresponds to the given general or work space coordinate vector $[X, Y, Z]$. If a path was to be followed, a set of numerical values for each joint angle, called the position profile, is generated. This position profile results from the solution of the inverse kinematics of the robotic manipulator for each point on the desired path. From the position profile for each joint a velocity and acceleration profiles are derived using the finite differences as shown in equation (3.42) and equation (3.43), respectively.

$$\dot{q} = \frac{dq}{dt} = \frac{q_{i+1} - q_i}{\Delta t} \quad \dots\dots\dots (3.47)$$

$$\ddot{q} = \frac{d^2q}{dt^2} = \frac{q_{i+1} - 2q_i + q_{i-1}}{(\Delta t)^2} \quad \dots\dots\dots (3.48)$$

Where (Δt) is the time step of the motion.

When q , $\dot{q}$ and $\ddot{q}$ are substituted in the dynamic equations of motion derived in the previous section (equations 3.14 through 3.17) the torque required from each actuator can be found.

In the present work, a web camera is used as a visual feedback sensor. It was used to measure the displacement of a moving object using image processing and image analysis algorithms for object detection. These displacement values were fed to a subroutine that performs the solution of the inverse kinematics problem. This subroutine decides what joint space values the robotic arm should take to reach the given location. These joint space values are fed to the digitally implemented PID controller. The motion of the actuators is decided according to the control signals sent by the PID controller.

3.5. *Digital Image Processing*

Digital image processing aims to change the image from a degraded and noisy form into a form useful for the observer (human or a computer). In the following, the major image processing techniques used in this work are discussed. In general, most of image processing techniques fall into three important categories Image Enhancement, Image Segmentation and Image Analysis.

3.5.1 Image Enhancement

At the instant the image is taken, there are usually many sources of image degradations. Noise is unavoidable in almost all imaging devices with different levels and distribution patterns. This is highly dependant upon the hardware itself. Also, the camera may be out of focus causing the image to be blurred. Blur may occur as a result of moving objects pictured by low speed cameras. Even the illumination of the scene may cause problems in the distribution of brightness over the image. Another source of image degradation is the process of digitization itself, incorrect sampling and quantization will definitely distort the image [36].

All these problems may be resolved using some algorithms that deal with each problem independently.

3.5.1.1 Image Sharpening

Image degradation mostly involves blurring [36]. The effect of blurring is to spread the color information over larger areas inside the image. Mathematically, this is modeled as an integration operation. This suggests that one may perform a differentiation operation to reverse the integral effect of blurring.

3.5.1.1. A. Image Differentiation

To differentiate an image, a suitable partial derivative operator is applied over the entire image. The effect of this operator should be the same in all directions, that is, the operator is a direction-invariant, or isotropic operator. One of the most popular operators is the gradient operator. It is defined as below:

For any surface $f(x, y)$ the gradient $G(f(x, y))$, or for short $G(f)$, is given by the following equation [31]:

$$G(f) = \frac{\partial f}{\partial x} \mathbf{i} + \frac{\partial f}{\partial y} \mathbf{j} \quad \dots\dots\dots (3.49)$$

The length of this vector is

$$|G(f)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \dots\dots\dots (3.50)$$

For digital images, the above equations must be in the discrete form that is:

$$\frac{\partial f}{\partial x} \approx f(x+1, y) - f(x, y) = \Delta_x f(x, y) \quad \dots\dots\dots (3.51)$$

$$\frac{\partial f}{\partial y} \approx f(x, y+1) - f(x, y) = \Delta_y f(x, y) \quad \dots\dots\dots (3.52)$$

The gradient has a magnitude of

$$|G(f)| = \left\{ [\Delta_x f(x, y)]^2 + [\Delta_y f(x, y)]^2 \right\}^{1/2} \quad \dots\dots\dots (3.53)$$

And a direction of

$$\theta = \tan^{-1} \left(\frac{\Delta_y f(x, y)}{\Delta_x f(x, y)} \right) \quad \dots\dots\dots (3.54)$$

If the point (x, y) lies in a uniform brightness region, then $|G(f)|$ will have a small or zero value but when the point (x, y) lies near an edge, then the value of $|G(f)|$ will be large. So, adding the value of $|G(f)|$ to the original image will emphasize edges and leave flat regions unaffected. However, care must be taken when dealing with noisy images because the differentiation process involves emphasizing high rate of change in brightness which is a property of a noisy point. So, noise will be stronger and the image is further damaged. Usually, the process of differentiation is preceded by the process of noise removal, described later, to reduce this effect.

3.5.1.1. B. High-Pass Filtering

In images, the unit of frequency is always the inverse of the unit of distance used in the picture plane, the pixel. Hence, the frequency is measured in $(\text{pixel})^{-1}$ or cycles-per-pixel [38].

The analogue image is sampled at the interval of one pixel to produce the digital version of it. This means that to perfectly reconstruct the original analogue image, the spatial frequencies must be less than or equal to (0.5), which is the case for our image. The highest representable spatial frequency in a quantized image is an alternating sequence of black and white which has a maximum value of (0.5). Thus the scales of the frequency components in x and y directions run from -0.5 to 0.5 cycles per pixels [38].

The transition between different gray levels in the image is the source of high frequency in the spatial frequency domain. So, if we want to sharpen an image, we must emphasize these frequencies via suitable filter, the high-pass filter. One high-pass filter is given by the equation [38]

$$H(\omega_x, \omega_y) = 2 - [\cos(\pi \omega_x) \cdot \cos(\pi \omega_y)]^n \quad \dots\dots\dots (3.55)$$

Where the exponent (n) determines how fast the transition of the transfer function from its minimum value to its maximum value.

3.5.1.2. Image Smoothing

Usually, image smoothing is concerned with noise removal. There are many types of noise encountered in images; among them are Random Noise and Distributed-Pattern Noise.

3.5.1.2. A. Random Noise

This type of noise occurs as isolated, high contrast, randomly-distributed pixels over the entire image. It is commonly called "salt-and-pepper" noise [6].

The classical method for removing this type of noise is the image averaging method. In this technique, see figure (3-6), the value of each pixel in the processed image is the result of taking the *local average* color of the neighboring pixels in the original image [39], that is

$$P'_0 = \frac{1}{4}(P_1 + P_2 + P_3 + P_4) \quad \dots\dots\dots (3.56)$$

This method is effective for high signal-to-noise-ratio images but is has almost no effect when the signal-to-noise ratio is low. In such a case, averaging process is performed for a sequence of images for the same scene and the problem of severely-noised images is overcome. In mathematics

$$P'_0 = \frac{1}{N} \sum_{i=1}^N \left[\frac{1}{4}(P_{i,1} + P_{i,2} + P_{i,3} + P_{i,4}) \right] \quad \dots\dots\dots (3.57)$$

N is the number of frames taken for the scene.

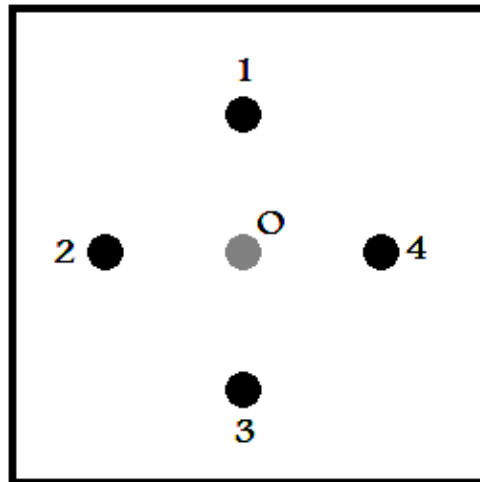


Figure (3-6) central pixel and its four neighbors

3.5.1.2. B. Noise As a Pattern

When the noise presents as a pattern, it can be removed by modeling it mathematically and extracting it from the image to restore the non-noisy image.

$$F(x, y) = S(x, y) + N(x, y) \quad \dots\dots\dots (3.58)$$

F: is the noisy image function

S: is the perfect image function

N : is the noise pattern

This is, however, not always possible because the pattern may be too complicated to be formulated as a closed form function $N(x, y)$. Noise patterns can exist as repeated lines with some known frequency, such a noise can be removed by transforming the image into the frequency domain by and, using a suitable band-reject filter, the noise can be removed [36].

3.5.2. Image Segmentation

Image segmentation means partitioning the image into separate regions each of which corresponds to an object in the scene. In what follows a brief discussion of the most frequently used methods for image segmentation is presented [36].

3.5.2.1. Segmentation Based on Color Information

This is the easiest way to decide which pixels belong to what object. In this method a scanning process is performed over the entire image and recording all the colors in it. Then lookup operation is performed to determine what label should a pixel have to define its belongingness to one object in the scene.

3.5.2.2. Segmentation by Region Growing

The previous method for image segmentation suffers from the fundamental problem that the spatial information is ignored when classifying the pixels. For instance, two separate regions (or objects) can be considered as one when they have the same color which is clearly not the case. Another example when the image contains isolated noise or small color changes within the object may cause points inside the object to be incorrectly classified.

Region growing is a bottom-up segmentation approach in which a *seed point* within the region of interest is selected and the adjacent pixels are added to the region as long as these pixels satisfy certain measures [39]. At the end, a

single connected region in the image is extracted. The selection of the seed point may be accomplished either manually or automatically depending on some expected properties of the region of interest.

3.5.2.3. Segmentation by Edge Detection

The boundaries of an object are of particular interest because they contain all the information needed for defining an object among the other objects in the scene. The scale, orientation and position of the object are familiar features to be extracted for the late object detection process. So, isolating these boundary points in an image is one of the most important goals in digital image processing [36].

An edge point is defined to be that point at which abrupt change in brightness occurs. This suggests that an edge detector should be of differential nature. This is generally the case.

3.5.2.3. A. Gradient-Based Edge Detection

In section (3.4.1.1.A) the definition of the gradient was

$$G(f) = \frac{\partial f}{\partial x} \mathbf{i} + \frac{\partial f}{\partial y} \mathbf{j} \quad \dots\dots\dots (3.59)$$

Where f is the image function. The magnitude of the gradient is:

$$|G(f)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \dots\dots\dots (3.60)$$

When the value of $|G(f)|$ is above some given threshold, the function can be considered discontinuous and the point (x, y) can be labeled as an edge point.

This technique works efficiently when the boundaries are of high contrast, but for gradual color transitions, poor edges are detected depending upon the value of the threshold selected. That is why care must be taken when selecting a threshold value.

3.5.2.3.B. Laplacian Edge Detector

For a given function $f(x, y)$ the Laplacian is given by [37]

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \dots\dots\dots (3.61)$$

When $f(x, y)$ is an image function

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} &= f(x+1, y) - 2f(x, y) + f(x-1, y) \\ \frac{\partial^2 f}{\partial y^2} &= f(x, y+1) - 2f(x, y) + f(x, y-1) \end{aligned} \dots\dots\dots (3.62)$$

So, the Laplacian becomes

$$\nabla^2(f(x, y)) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \dots (3.63)$$

To use the Laplacian, we simply convolve the mask

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

With the entire image.

3.5.2.3. B. Best Fit Edge Detection

In this technique, a plane $f(x, y) = ax + by + c$ is fitted to four image pixels $f(x, y)$, $f(x+1, y)$, $f(x, y+1)$ and $f(x+1, y+1)$ then the error in fit is minimized by the minimum square error formula given by [36]:

$$\begin{aligned} \varepsilon &= [ax + by + c - f(x, y)]^2 + [a(x+1) + by + c - f(x+1, y)]^2 + \\ &\quad [ax + b(y+1) + c - f(x, y+1)]^2 + [a(x+1) + b(y+1) + c - f(x+1, y+1)]^2 \dots\dots (3.64) \end{aligned}$$

Differentiating ($\mathcal{E}$) with respect to a , b and c and setting the resulting expressions to zero and solving for a , b and c the results are

$$\begin{aligned} a &= \frac{f(x+1,y) + f(x+1,y+1)}{2} - \frac{f(x,y) + f(x,y+1)}{2} \\ b &= \frac{f(x,y+1) + f(x+1,y+1)}{2} - \frac{f(x,y) + f(x+1,y)}{2} \quad \dots (3.65) \\ c &= \frac{1}{4} [3f(x,y) + f(x+1,y) + f(x,y+1) - f(x+1,y+1) - a \cdot x - b \cdot y] \end{aligned}$$

The gradient of the plane $f(x, y) = ax + by + c$ has the magnitude of

$$|G(f)| = (a^2 + b^2)^{1/2} \quad \dots\dots\dots (3.66)$$

3.5.3. Image Analysis

This part of image processing is concerned with describing the objects present in an image [40]. It makes use of computer-based methods to analyze and classify the objects in the scene.

Many applications benefit from these techniques, among them is the biomedical science. When a large number of microscope slides are to be examined visually for clinical and research purposes automatic analysis become of vital importance for both time saving and accurate results. Ground-Penetrating Radar (GPR) data are processed and analyzed using image analysis techniques to detect under ground objects. Another application is the intelligent automobiles. The visual analyzer replaces the human driver in driving the automobile by analyzing each picture taken from the accompanying camera and make decisions for safe driving. Robot manipulators can benefit extremely from a visual feedback sensory. The manipulator can, with the aid of the visual controller, decide what object to pick and where to place it. This application is what this work is concerned with.

3.5.3.1. Object Detection by Template Matching

A template is a small picture contains the image of the object to be detected. This template is moved over the entire image and, at each location, a measure of match is taken and the likelihood of the presence of the object in the image is decided [40].

As a measure of match, the following relation is used

$$R = \iint_{\ell} (f - t)^2 \quad \dots\dots\dots (3.67)$$

Where (f) is the picture, (t) is the template and (ℓ) is the image region.

From the above relation

$$R = \iint f^2 - 2 \iint f \cdot t + \iint t^2 \quad \dots\dots\dots (3.68)$$

The first and last integrals are constants everywhere, so, the middle integral is of interest, it is called the cross-correlation between (t) and (f). For digital images, summation is used instead of integrals, so

$$R(m, n) = \sum_i \sum_j f(i, j) \cdot t(i - m, j - n) \quad \dots\dots\dots (3.69)$$

The picture and the template are said to be similar when (R) is large and vice versa.

3.5.3.2. Object Detection Using Hough Transform

When the object to be detected can be expressed in a closed form equation, Hough transform is the most powerful method to detect it. Circles, ellipses and lines and any set of points that can be fitted into an equation can uniquely be identified and detected [40].

To explain how Hough transform method works, let's consider the simplest case, that is, the line detection case. Any line in the (x - y) plane can be expressed as

$$y = ax + b \quad \dots\dots\dots (3.70)$$

Where (a) and (b) are constants.

When a set of points is extracted from an image, the coordinates (x, y) of all these points are known but the constants (a, b) must be evaluated to decide whether or not the given set of points represents a line. This can be done by transforming the above equation from its spatial space into what is called the parameter space [40]. So, the above equation can be re-written as

$$b = y - a x \quad \dots\dots\dots$$

(3.71)

Now, any given image point in (x, y) coordinates is transformed into a line in the parameter space with x and y are the parameters for that line. This is shown in the figure (3-7).

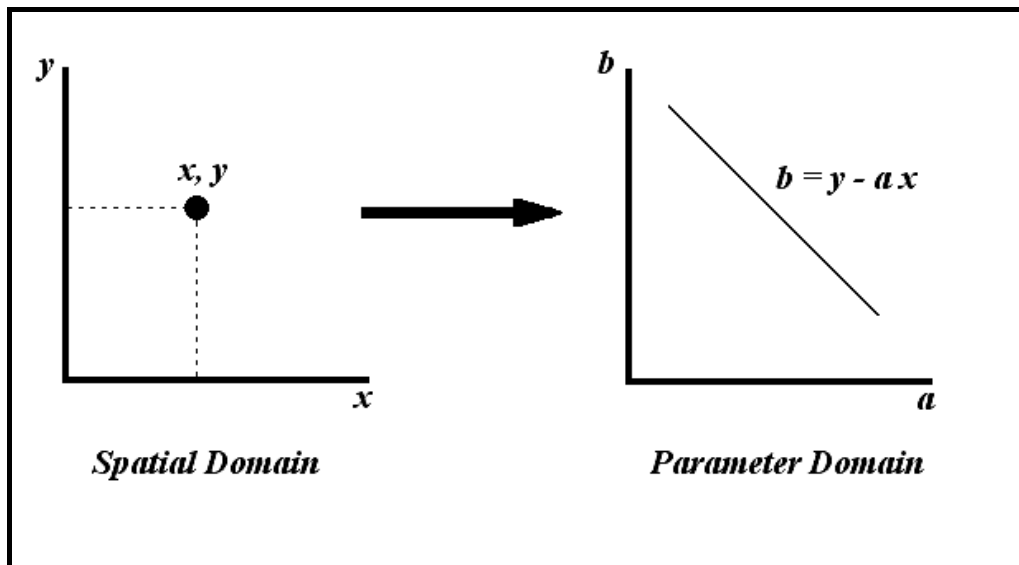


Figure (3-7) transformation of a spatial domain point into a parameter domain line.

For a set of points aligned as a line, the lines $b = y_n - a x_n$ all intersect at the parameter space coordinate (a, b), as shown in figure (3-8).

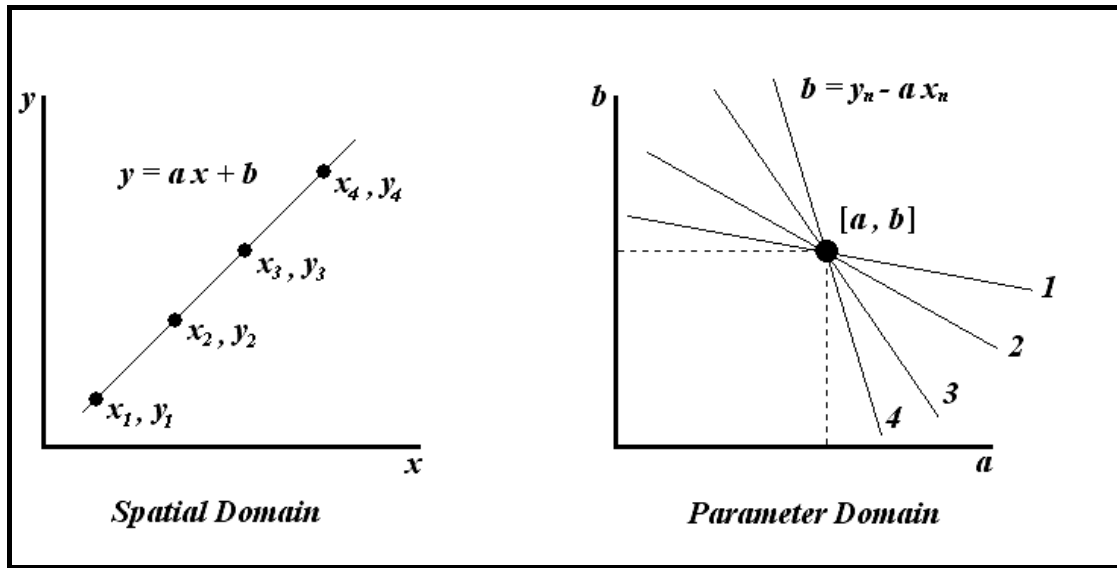


Figure (3-8) the set of points in the spatial domain transformed into a set of lines in the parameter domain.

Using the parameter domain as an accumulator, the point $[a, b]$ can be detected by taking the maximum value of the accumulator.

Here is the algorithm for detecting a line using Hough transform.

For each image point (x_n, y_n) do the following:

For $a = \text{minvalue}$ to maxvalue

$$b = y_n - a x_n$$

$$A(a, b) = A(a, b) + 1$$

Next a

Next Point

After the scanning of the whole image is done, the accumulator array $[A]$ will contain a maximum value. The location of this maximum value lies at the coordinates (a) and (b) of the accumulation array and it is the required pair of

the parameters of the line representing the set points in the image (or the spatial domain).

Chapter Four

Simulation and Control of the Robotic Arm

Simulation and Control of the Robotic Arm

4.1. Introduction

In this chapter a detailed explanation of the algorithms and tools used to simulate a robotic arm vision-based controller is presented. As mentioned earlier, visual servoing control systems can be classified into two categories according to the way used to install the camera within the control system [2]. A Stand-Alone servoing system is the one in which the camera is installed in a fixed position in the work space of the robot and the position of the camera is assumed to be the reference and, when compared to the object location, the robot responds accordingly. The second category is the Eye-In-Hand servoing system in which the camera is located on-board the robot, usually at the end-effector, in this case all the calculations of the object position in the 3D work space are based on the measurements made for the image space which is referenced at the end-effector of the robot.

The present work is concerned with the second category, that is, the Eye-in-hand controller. An ordinary web camera is used as the visual feedback sensor and it is supposed to be located at the end-effector of the robotic arm. Image processing techniques were used to analyze the visual feedback information. The robotic arm used for simulation is the RM-501 robot which is a small size, light weight industrial robotic arm. It has four links: the base (link #0), link #1, link #2 and link #3. These links are modeled as simple rods and each of them is driven by a DC servomotor. The DC motor model is taken from [41].

4.2. Visual Servoing

The term visual servoing means that a robot is ordered to move in its workspace according to a visual feedback data (an image) which usually holds information about an object to be tracked. This information can be extracted

from the input image by using digital image processing algorithms designed for this purpose. These algorithms are called object recognition processes. Object recognition process is usually accomplished in two stages; the pre-processing stage and the recognition stage.

4.2.1. Image Pre-Processing

In the pre-processing stage, the image is processed with the fundamental image processes. These include:

a- Image Segmentation Algorithms: These algorithms were used to isolate all the objects in the captured image (including the background) and each object pixels were stored in a separate layer. If each object has a unique color, then simple color-based image segmentation can be easily used. However, when two or more objects have the same color, then color-based segmentation method cannot be applied. Instead, a more general technique can be used, that is, the region growing technique. This technique is described in chapter three and it was used in this work.

b- Edge Detection Algorithms: These algorithms were used to extract the points of interest in the image. The purpose of using edges for object detection has two advantages. The first advantage is to reduce the number of points to be processed in the image and hence speeding up the processing. The second advantage is that the resulting edge image is a binary image, that is, an image with only two colors, white (1) and black (0). This makes the successor processes independent from the color information and only the geometry of the object is considered for recognition purposes.

c- The Pose of the Object: The location of an object and its orientation constitute the pose information of that object. This information is very important to be determined for each object in the scene. However, the location of the object is easy to be determined in terms of its center of gravity. It can be calculated using the following algorithm:

```

10 read an object pixel  $P(X,Y)$ ;
     $n=n+1$ ;
     $sum\_X = sum\_X + X$ ;
     $sum\_Y = sum\_Y + Y$ ;
    if more pixels remain unread goto 10
     $Centre\_X = sum\_X / n$ ;
     $Centre\_Y = sum\_Y / n$ ;
    Define the object location to be  $L(Centre\_X, Centre\_Y)$ ;
End.

```

The orientation of the object is more complicated to be determined. In this work the Best Eigenvector Fit (BEF) method [38] was used. This method is originally used for line fitting purposes but in this work it was used for obtaining the orientation of an object. BEF method is described in appendix B.

Determining the orientation of the object is very important because it is impractical to save so many templates for the same object each with different orientation. This results in huge data to be stored and, accordingly, reduces the speed of object detection process and slows down robot speed of motion. Using the BEF method enables the end-effector to align with the object in the scene which makes the matching process much more efficient.

What was said above about the importance of object orientation also applies to object scale. To solve this problem, the template was saved in a minimum size. This made it possible to reduce the scale of the suspected object into that of the template and then the matching process begins.

4.2.2. Object Recognition

In this work, Hough transform was used to detect a circle and an ellipse, while the template matching was used for detecting an arbitrarily-shaped object.

4.2.2.1. Hough Transform

To detect a circle with a given feature (the radius of the circle R), Hough transform algorithm was used to seek for any set of points in the given scene that were distributed in a form of a circle with the desired feature (R). Any other circles with a radius different from (R) are ignored. At the end of processing, the algorithm returned the coordinates of the centre of the desired circle.

The algorithm for detecting a circle with a given radius (R) using Hough transform is described below:

The equation with radius R and a centre located at $C(h, k)$ is

$$(x-h)^2 + (y-k)^2 = R^2 \quad \text{..... (4.1)}$$

In the above equation x, y and R are known and it is required to determine the corresponding h and k which constitute the coordinates of the centre of the desired circle. To use this equation in Hough transform, it must be transformed in to the *parameter space*. This is shown in equation (4.2) below.

$$h = x \mp [R^2 - (y-k)^2]^{1/2} \quad \text{..... (4.2)}$$

In equation (4.2), x and y are the coordinate of a pixel in the image while h and k are the centre of the circle with radius R . If the scene contains the desired circle, then h and k must be located inside the image area. This means that

$$0 \leq h \leq W$$

$$0 \leq k \leq H$$

W and H are the width and the height of the image, respectively.

Hough transform algorithm can be written as:

- **Define the array $A[W, H]$; ' The accumulation array**
- **Input the desired feature R ; ' The radius**
- **For all the image pixels $P(x, y)$ do**
- **For $k = 0$ to H**

$$h = x \mp [R^2 - (y - k)^2]^{1/2}$$

$$A[h, k] = A[h, k] + 1$$

Loop

- **Loop**
- **Search for the maximum value in $A[W, H]$;**
- **The maximum value is located at $A[h, k]$;**
- **END.**

After completing this algorithm, the accumulation array will contain numerical values. From these values the maximum is identified and the desired location of the circle corresponds to the coordinates of this maximum value in the accumulation array. As an example, a circle of radius (40 pixels) was detected using Hough transform, see figure (4-1). In figure (4-2) the corresponding accumulation array and the maximum value are shown. The dark circle on the accumulation refers to the location at which the maximum value of the array presents.

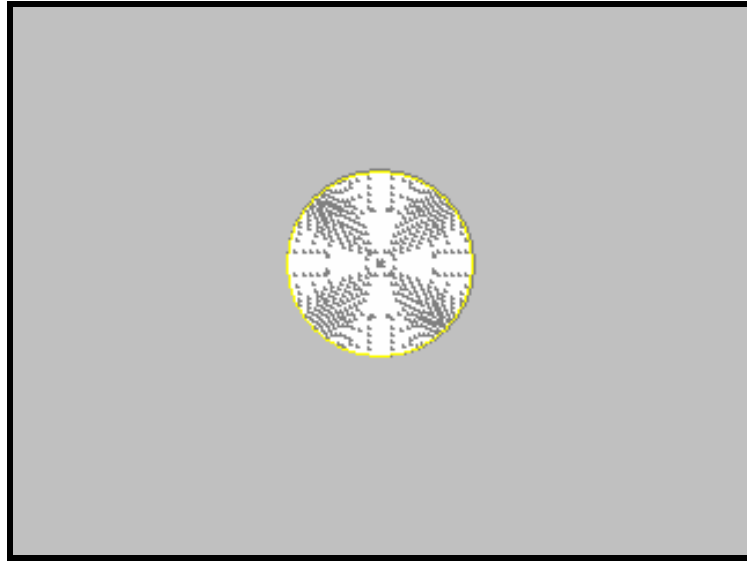


Figure (4-1) the circle to be detected with HT.

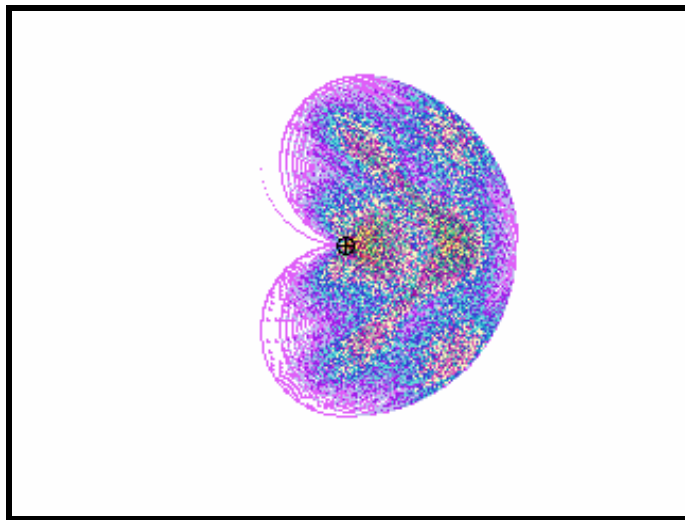


Figure (4-2) the accumulation array representation for the circle of Figure (4-1).

Similarly, an ellipse with known values for the major and the minor axes can be detected. The only difference is the equation used for accumulation. The ellipse is defined with the following equation:

$$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1 \quad \dots\dots (4.3)$$

By rearranging equation (4.3), the accumulation equation for the ellipse can be determined to be:

$$h = x \pm \left[a^2 - \left(\frac{a}{b} \right)^2 (y-k)^2 \right]^{1/2} \quad \text{..... (4.4)}$$

4.2.2.2. Template Matching

Template matching technique is much easier to implement than Hough transform. It is fully explained in chapter three (section 3.4.3.1).

4.3. The PID Controller

A proportional plus integral plus derivative controller (or a PID controller) is a feedback control mechanism widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired set point by responding with a corrective action that can adjust the process accordingly. The formula describing this type of controllers, in the time domain, is given by [42]:

$$m(t) = K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int_0^t e(t) dt \quad \text{..... (4.5)}$$

and it has a corresponding transfer function :

$$\frac{M(s)}{E(s)} = K_p + K_d s + \frac{K_i}{s} \quad \text{..... (4.6)}$$

Where K_p represents the proportional constant, K_d represents the derivative constant and K_i represents the integral constant. These constants can be evaluated experimentally using an approach called the PID tuning procedure [42].

Since a digital computer is used as the main control unit, the PID controller was implemented digitally to fit into the system. The algorithm for a computer-based PID controller is given below:

Implementation of the PID Algorithm

```
00  start;
01  set  previous_error = 0;
02  set I = 0;
03  set error = set_point - actual_position;
04  set P = Kp * error;
05  set I = I + Ki * error * dt;
06  set D = (Kd / dt) * ( error - previous_error );
07  set output = P + I + D;
08  set previous_error = error;
09  goto 03
10  END.
```

4.4. DC Motor Control

To control a DC motor, a variable voltage must be generated and applied to power the motor. There are two main sources of voltage supply for a motor: the linear power supply, and the switching power supply. The linear power

supply technique uses power transistors acting in their linear mode to provide a smooth and continuously adjustable output voltage depending upon load drive requirements. In this mode, the transistors are used as variable resistors, conducting according to the level of the input signal applied. This method of control suffers from the high heat dissipation by the transistors during the operation [43].

Switching power supply techniques use transistors to switch the voltage through to the motor in a series of pulses that applies an *average* voltage to the motor. Two switching methods are commonly used, *pulse-width modulation* (PWM), and *pulse-frequency modulation* (PFM) [43].

Pulse-width modulation is the more common method used for motor control. This method uses a constant period (T) between sequential pulses, while the width (w) of the pulses is altered (duty cycle) to change the effective or average voltage supplied to the motor. See figure (4-3).

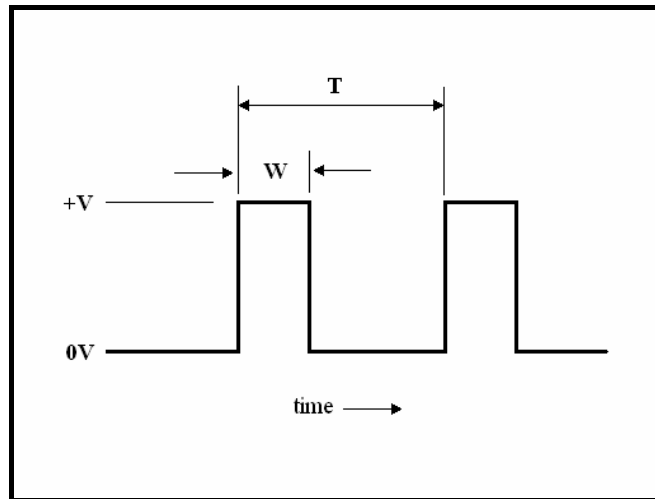


Figure (4-3) PWM method for a DC motor control.

Pulse-frequency modulation varies the frequency of pulses having a constant pulse width (t) to control the average voltage applied to the motor, see figure (4-4).

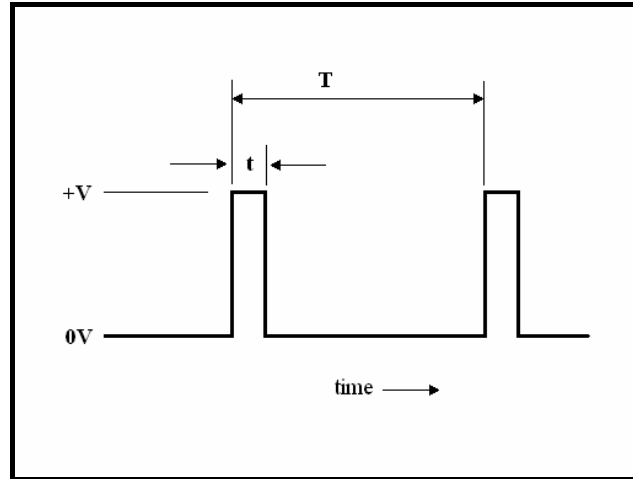


Figure (4-4) PFM method for a DC motor control.

The PID controller's output is either a pulse width (w) when the PWM method is used or a pulse frequency (T) when the PFM method is used. However, the effective voltage that corresponds to a pulse width (or a pulse frequency) is obtained through experimental testing and it is dependant upon the DC motor type used and the loading conditions applied on it.

The block diagram of the control system for a DC motor is shown in figure (4-5).

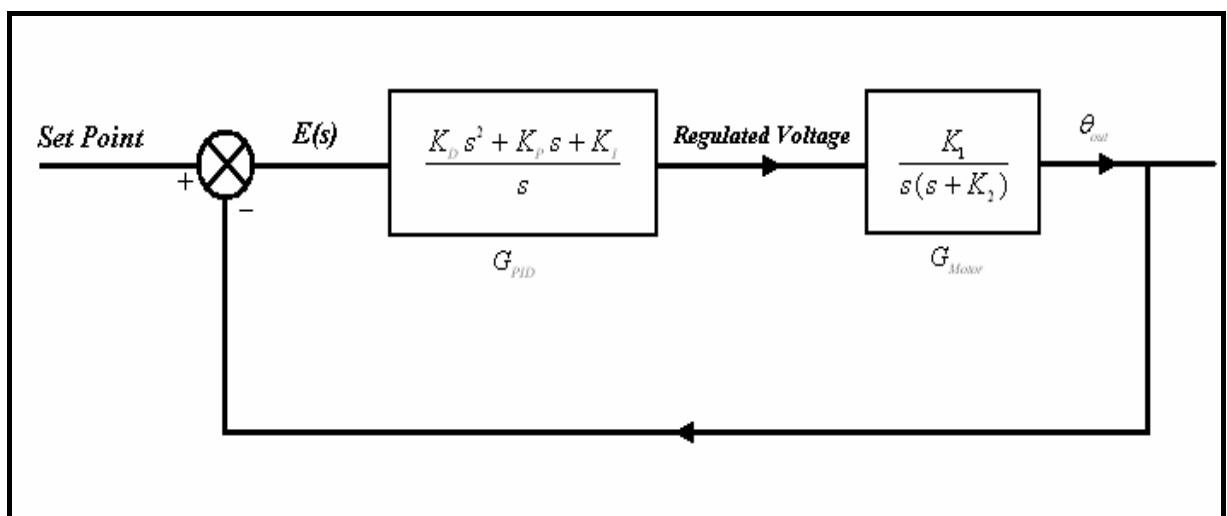


Figure (4-5) DC servomotor control system.

4.5. Solution of the Inverse Kinematics Problem (IKP)

The solution of the inverse kinematics problem is the key for a robot control task. Without such a solution one has no idea about the spatial configuration the robotic arm should take in order to reach a specific point in the robot's workspace and, accordingly, the controller has no measure according to which it sends the control signals to the actuators of the robotic arm, see section (3.3.2).

In the present work, an efficient method was used to solve the inverse kinematics problem. It is a geometrical method that generates a set of solutions for the same end-effector point and, according to some set of rules, selects the best solution. These rules are basically dependant upon the workspace available, the obstacles the robot should avoid and the dynamical and power saving considerations. This method is illustrated in figure (4-6) and figure (4-7). As shown in the figure (4-6), a path is specified for the robotic arm to follow. This path is shown as a set of dark points. In figure (4-7) the output from the inverse kinematics problem solution algorithm is shown. This method gave a good accuracy in solving the inverse kinematics problem, and when compared with the numerical solution shown in figure (4-8), the efficiency of this method is obvious. This method can be generalized for robotic arms that have more than three links with the same efficiency.

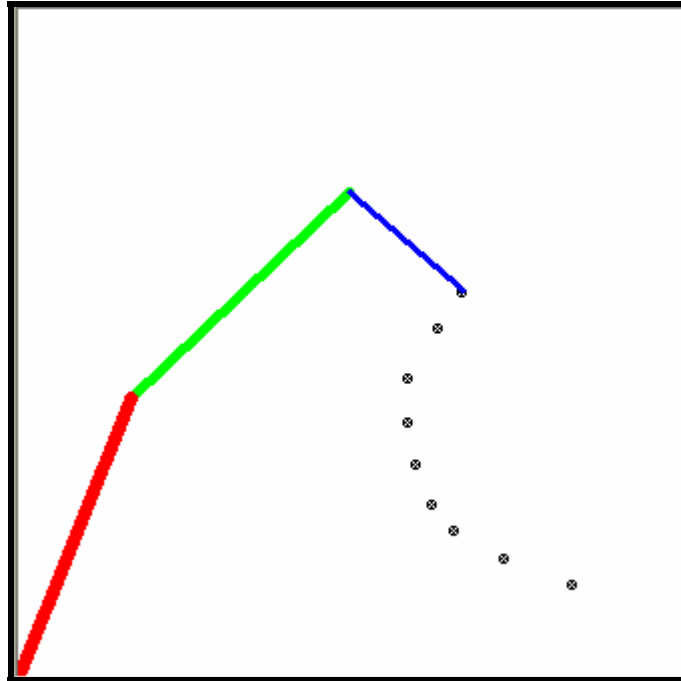


Figure (4-6) the required path for the robotic arm to follow.

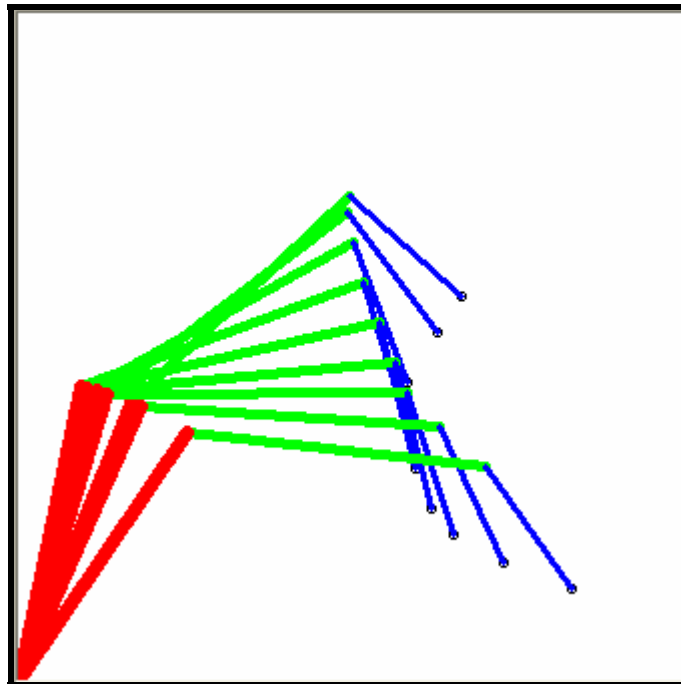


Figure (4-7) solution of the inverse kinematics problem to the given path in figure (4-6).

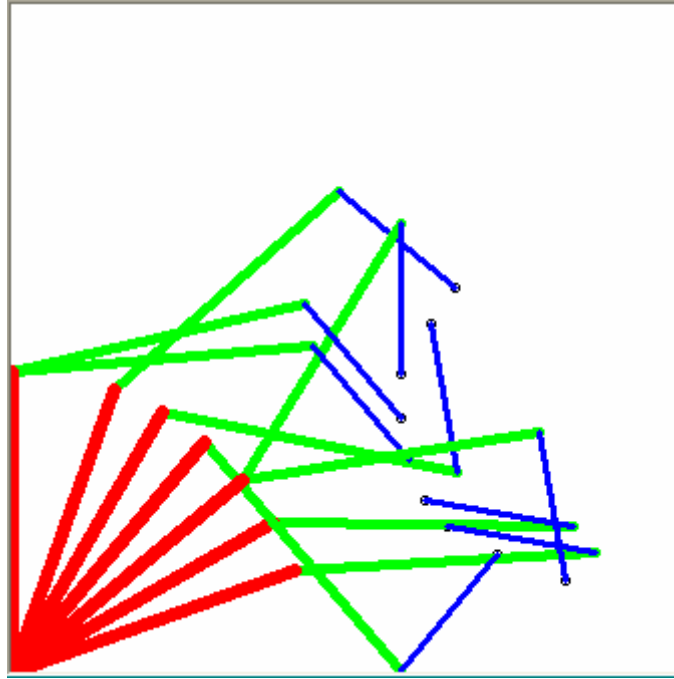


Figure (4-8) numerical solution of the inverse kinematics problem to the given path in figure (4-6).

4.6. Simulation of the Path Planning Method

To move an object from rest to a specified velocity, the inertia force has a maximum value at the instant of motion initiation [44]. This requires a corresponding high acceleration to overcome this inertia force. This also applies to the torques required from an actuator in a robotic arm. An actuator is connected to the link of the robot and maximum torque is required at the beginning of the movement of the actuator. When the actuator approaches the desired position it must start deceleration to achieve zero speed at the vicinity of the setpoint to minimize the error and avoid high overshoot. In figure (4-9), the acceleration, velocity and position profiles are shown for moving an object from rest and then braking it at the desired position.

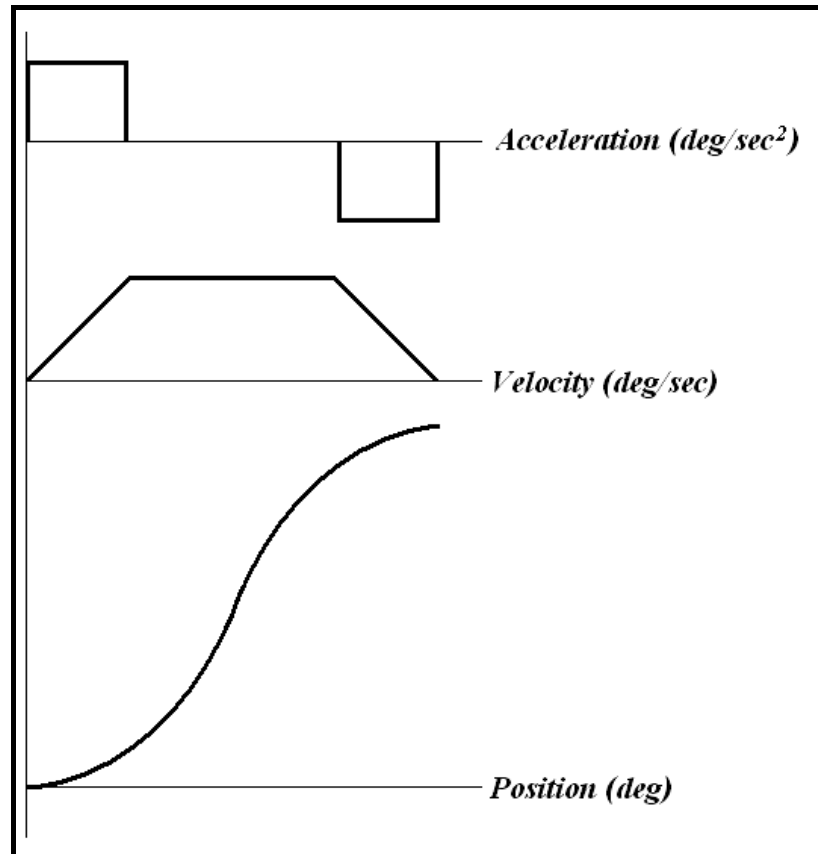


Figure (4-9) acceleration, velocity and position profiles for a path-independent controller output.

This method of control is useful when the path of approach to the desired end-effector's position is immaterial. When a path is to be followed one cannot expect, using the above method, that the end-effector would follow the required path. In order to use this method the path of the moving object must be subdivided into a set of key points, known as *via points* [34], and the profiles shown in figure (4-9) are applied between any two consecutive via points. The more via points are selected the less path following error is obtained. In industry, the path followed by the end-effector is predictable and the vision-based controller is used as a fine-tuning stage to achieve accurate end-effector positioning. In the worst case in a visual servoing system, the path can be arbitrary if the system tracks an object that moves in random motion. This makes the control process more difficult because at each time step, during the

motion of the object, a digital image must be taken and processed as fast as possible. Generally, image processing algorithms are slow which makes the motion of the robot slow as well. Accordingly the motion of the object is recorded only at discrete points and the polygon connecting these points is assumed to be the desired path. This means that the path connecting any two consecutive locations is a *line segment* as shown in the figure (4-10).

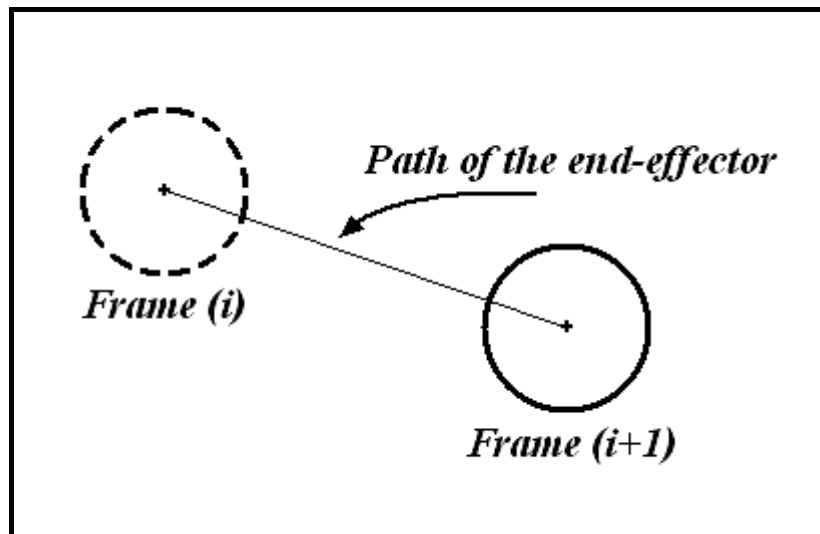


Figure (4-10) the path followed by the end-effector between two consecutive object locations.

4.7. Simulation of the Robotic Arm Motion

The robotic arm used in the simulation is constructed of four links: the base (link #0), link #1, link #2 and link #3, see figure (4-11). Each of these links is powered by a DC servomotor that is controlled by the PID controller implemented digitally in the simulation program. The camera was mounted at the end of link #3.

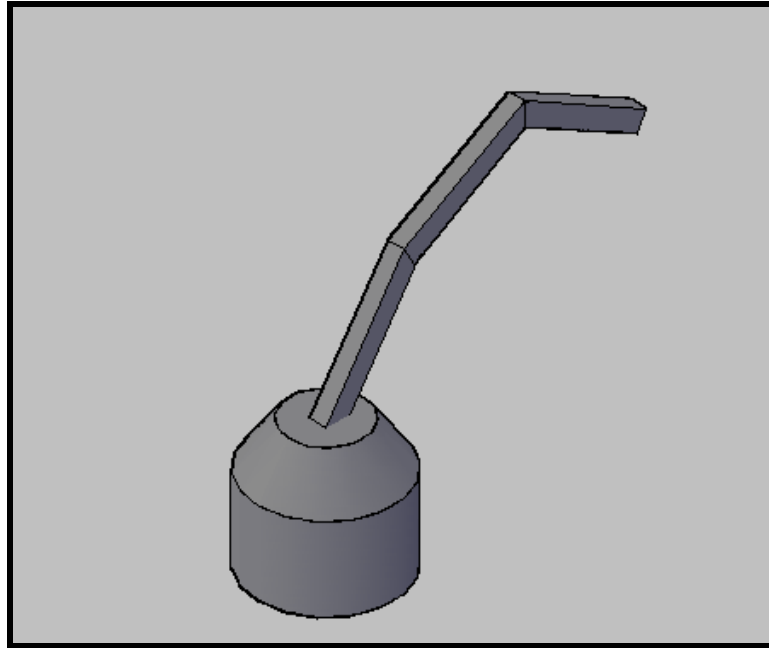


Figure (4-11) the robotic arm used for simulation.

The program is built according to the algorithm below:

Visual Servoing Algorithm

```
00  start;
01  input home position vector (initial joint space data);
02  input the robotic arm physical properties;
03  call the image processing routines to locate the object of interest;
04  calculate the error signal;
05  if error = 0 then goto 03;
06  call the IKP subroutine to evaluate the desired position profile;
07  calculate the velocity and acceleration profiles;
08  apply the PID algorithm to drive the robotic arm to the desired position;
09  goto 03;
10  END.
```

According to the algorithm, the program is initialized by setting the home position vector ($\mathbf{q}_{home}$) which the starting position of the robotic arm. Then the physical parameters that the robotic arm has, such as the mass, length and moment of inertia for all the links are fed to the program. After this the subroutine for initializing the web camera is called, this is the trigger for the beginning of the closed loop vision-based control system.

The close loop visual control system is composed of a web camera as the feedback sensor that takes digital images to the scene in which an object of interest is moving. This information (the image) is sent to the block of routines that perform all the digital image processing and object recognition techniques necessary to measure the current location of the moving object (x_{object}, y_{object}). An error value is calculated that represents the difference between the reference image position (x_o, y_o) and the position measured for the location of the object as shown in figure (4-12). The reference image position is chosen to be the center point of the image.

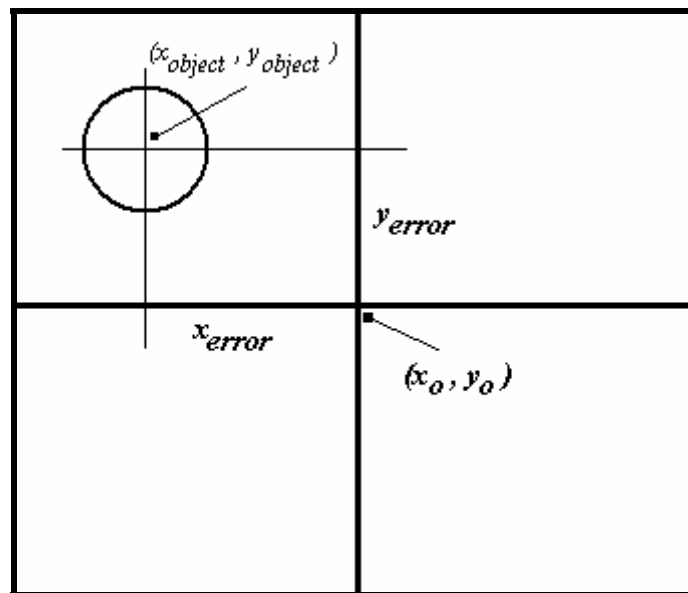


Figure (4-12) reference image point.

$$x_{error} = x_{object} - x_o \quad \text{..... (4.7)}$$

$$y_{error} = y_{object} - y_o \quad \text{..... (4.8)}$$

The path between the reference image position and the measured object position, as mentioned earlier, is a line segment joining the two points, see figure (4-8). Along this line segment the inverse kinematics solution method is used to calculate the position profile in which a motion history for each joint is specified. From this position profile a velocity and acceleration profiles are generated using equations (3.47) and (3.48), respectively. These equations are repeated below:

$$v_i = \frac{d q_i}{d t} \quad \text{..... (4.9)}$$

$$a_i = \frac{d^2 q_i}{d t^2} \quad \text{..... (4.10)}$$

These equations can be approximated by a finite difference formulation as:

$$v_i = \frac{q_{i+1} - q_i}{\Delta t} \text{ and}$$

$$a_i = \frac{q_{i+1} - 2q_i + q_{i-1}}{(\Delta t)^2}$$

Where (i) is a running index and (Δt) is the time step of motion. The above algorithm is repeated as long as the object is moving. When the object stops, no error signal is sensed and the algorithm terminates.

4.8. Calculation of the Forces Acting On the Robot During the Motion

When the position profile is obtained, it becomes easy to determine all the forces acting on the robotic arm during the motion by using the system's set of equations of motion derived in chapter three. Substituting the values of the position profile of the joint space in equations (3-14), (3-15), (3-16) and (3-17)

with the aid of equations (4-24) and (4-25) the values of the forces are as follows:

1. Inertia Forces (torques)

These forces can be obtained by multiplying the system mass matrix with the acceleration vector of each joint. That is:

$$\{F\} = \begin{Bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \end{Bmatrix} = [M] \{\ddot{\theta}\} = \begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_0 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{Bmatrix} \quad \text{..... (4-11)}$$

Where

F_i ($i = 0$ to 3) is the inertia force at the i^{th} joint.

$\ddot{\theta}_i$ is the acceleration at the i^{th} joint.

M_{ij} are the elements of the system's mass matrix in which

$$M_{01} = M_{02} = M_{03} = 0$$

$$M_{10} = M_{20} = M_{30} = 0$$

$$M_{00} = I_{ass}$$

$$M_{11} = \begin{bmatrix} m_1 l_{c_1}^2 + I_1 + m_2 l_1^2 + m_2 l_{c_2}^2 + 2m_2 l_1 l_{c_2} \cos(q_2) + I_2 + m_3 l_1^2 + m_1 l_2^2 + I_3 \\ m_3 l_{c_3}^2 + 2m_3 l_1 l_2 \cos(q_2) + 2m_3 l_1 l_{c_3} \cos(q_2 + q_3) + 2m_3 l_2 l_{c_3} \cos(q_3) \end{bmatrix}$$

$$M_{22} = [m_2 l_{c_2}^2 + I_2 + m_3 l_2^2 + m_3 l_{c_3}^2 + 2m_3 l_2 l_{c_3} \cos(q_3) + I_3]$$

$$M_{33} = [m_3 l_{c_3}^2 + I_3]$$

$$M_{12} = M_{21} = \begin{bmatrix} m_2 l_{c_2}^2 + m_2 l_1 l_{c_2} \cos(q_2) + I_2 + m_3 l_2^2 + m_3 l_{c_3}^2 + m_3 l_1 l_2 \cos(q_2) + m_3 l_1 l_{c_3} \cos(q_2 + q_3) + \\ 2m_3 l_2 l_{c_3} \cos(q_3) + I_3 \end{bmatrix}$$

$$M_{13} = M_{31} = [m_3 l_{c_3}^2 + m_3 l_1 l_{c_3} \cos(q_2 + q_3) + m_3 l_2 l_{c_3} \cos(q_3) + I_3]$$

$$M_{23} = M_{32} = [m_3 l_{c_3}^2 + m_3 l_2 l_{c_3} \cos(q_3) + I_3]$$

2. Coriolis Forces (torques)

These forces are obtained, in general, by multiplying the coriolis matrix [C] with the joint space velocity vector. Since the equations of motion of the system are highly non-linear it is difficult to assign the matrix [C] a specific form. So, the coriolis forces are obtained directly from the equations of motion. These forces are given by:

$$F_{C_0} = \frac{\partial I_{ass}}{\partial t} \dot{q}_o$$

$$\begin{aligned} F_{C_1} = & -2m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_1 \dot{q}_2 - m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_2^2 - m_3 l_1 l_2 \sin(q_2) \dot{q}_2^2 - 2m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_1 (\dot{q}_2 + \dot{q}_3) \\ & - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_2 (\dot{q}_2 + \dot{q}_3) - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_3 (\dot{q}_2 + \dot{q}_3) - 2m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_1 \dot{q}_3 \\ & - 2m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_2 \dot{q}_3 - m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_3^2 + \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_1} \end{aligned}$$

$$\begin{aligned} F_{C_2} = & -m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_1 \dot{q}_2 - m_3 l_1 l_2 \sin(q_2) \dot{q}_1 \dot{q}_2 - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_1 (\dot{q}_2 + \dot{q}_3) \\ & - 2m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_1 \dot{q}_3 - 2m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_2 \dot{q}_3 - m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_3^2 \\ & - m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_1 (\dot{q}_1 + \dot{q}_2) - m_3 l_1 l_2 \sin(q_2) \dot{q}_1 (\dot{q}_1 + \dot{q}_2) - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_1 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \\ & + \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_2} \end{aligned}$$

$$\begin{aligned} F_{C_3} = & -m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_1 (\dot{q}_2 + \dot{q}_3) - m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_1 \dot{q}_3 - m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_2 \dot{q}_3 \\ & - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_1 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) - m_3 l_2 l_{c_3} \sin(q_3) (\dot{q}_1 + \dot{q}_2) (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \\ & + \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_3} \end{aligned}$$

3. The Gravitational Forces (torques)

No specific form of the gravity matrix is available due to the high non-linearity of the equations of motion of the robotic system. However, the values of the gravitational forces are extracted from the four differential equations and are given by the following equations:

$$G_0 = 0$$

$$\begin{aligned} G_1 = & [m_1 l_{c_1} + m_2 l_1 + m_3 l_1] g \sin(q_1) + [m_2 l_{c_2} + m_3 l_2] g \sin(q_1 + q_2) + \\ & m_3 l_{c_3} g \sin(q_1 + q_2 + q_3) \end{aligned}$$

$$G_2 = -m_2 l_c g \sin(q_1 + q_2) - m_3 l_2 g \sin(q_1 + q_2) - m_3 l_{c_3} g \sin(q_1 + q_2 + q_3)$$

$$G_3 = -m_3 l_{c_3} g \sin(q_1 + q_2 + q_3)$$

The torque at each joint during the motion is the algebraic summation of the above force vectors.

Since no specific values available for the properties of the links of the robotic arm under consideration, the following values referenced to unity were used in the simulation program:

Link #1

Length = 1

Distance of the c.g. from the joint = 0.5

Mass = 1

Moment of inertia about the joint = 1

Link #2

Length = 1

Distance of the c.g. from the joint = 0.5

Mass = 0.75

Moment of inertia about the joint = 0.75

Link #3

Length = 0.5

Distance of the c.g. from the joint = 0.25

Mass = 0.5

Moment of inertia about the joint = 0.5

The mass, length and moment of inertia of Link#0 (i.e. the base) were neglected.

Chapter Five

Results and Discussion

Results and Discussion

In this chapter the results of applying the algorithms described in the previous chapters are presented. The results were obtained from the simulation program which was built using *Visual Basic 6.0* programming language.

The behavior of a robotic manipulator can be fully described by a set of second order, highly non-linear, ordinary differential equations. This is generally the case for any dynamic system. Solving the differential equations for a general dynamic system requires that the external forces (or torques) applied to the system must be known and the solution returns the displacements (or angular positions) that correspond to the input forces. However, in robotics the situation is reversed. The robot cannot be allowed to move randomly, instead, it must move according to the commands that makes it follow a pre-specified path which is dependant upon the task that the robot is designed to accomplish. This means that the position profile must be known in advance and the unknown is the vector of torques required from each actuator in order to follow the required path.

The dynamic system simulated in this work belongs to a four-link robotic arm. This means that this dynamic system is expressed by four highly nonlinear ordinary differential equations. These equations were derived in chapter three (section 3.2).

5.1. Kinematics

The kinematics of any linkage is concerned with the displacement of each joint of the assembly without dealing with the forces (or torques) that caused these displacements. The dynamic model built in this work assumed a four-link robotic arm. This type of robots gives infinite number of solutions to the inverse kinematics problem. However, the inverse kinematics problem was solved by using another technique which was dependant upon the geometry of the robotic

arm. This made it possible to express each location in the workspace in terms of the corresponding joint space vector. The method showed very good efficiency in obtaining a solution in the form of position profiles that correspond to the paths that the robot should follow. Other researchers solved this problem using numerical, analytical and neural networks techniques which have many problems in obtaining the acceptable results.

The objective of solving the inverse kinematics problem is to find the vector of the angular displacements of the joints of the robotic arm for a given (desired) end-effector position. When the robot tracks a path a position profile, which represents all the values of the joint angular displacements at each point on the desired path during the motion, is generated. Figure (5-1) shows a 3D circular path that was fed to the program as the desired path to be followed by the robotic arm. The figure shows the circle in both XY and XZ planes. Figure (5-1a) illustrates the position profile that corresponds to the given 3D circular path. It shows the relationship between the angular displacement of each joint and the time during which the tracking of the path completes. To test whether or not these angular displacements are correct, the forward kinematics relations were used to calculate the end-effector position (in the workspace domain). Applying the forward kinematics relations produced the correct 3D circular path.

The geometry of the given path is simple and hence the position profile obtained shows that each joint has experienced smooth angular displacements as the path was traced. The values of the angular velocity and the angular acceleration at each joint during the motion are dependant upon the time step (Δt) between any two consecutive configurations. If (Δt) is small (i.e. fast robot movement), then high velocity and acceleration values are obtained. This means that when the robot is forced to move rapidly along a specified path, the corresponding dynamic forces acting on the robot will be high. The maximum path tracking speed is limited by two factors, the strength of the robotic system

against these forces and the stability of the system during the movement. These two factors are important criteria in the design stage of any robotic system.

Assuming one second between every two consecutive points on the position profile ($\Delta t = 1$ sec), the velocity profile and the acceleration profile were generated using the first and the second time derivative of each joint displacement respectively. The velocity and acceleration profiles are shown in figure (5-1b) and figure (5-1c), respectively.

Another example was the plane circular path described in figure (5-2). The inverse kinematics solution for this path, shown in figure (5-2a), had the same overall smooth behavior of the joints during the motion when compared to the 3D circular path. The difference in the shape is due to the difference in the geometry of the two paths. The velocity and acceleration profiles for (Δt) of one second are shown in figures (5-2b) and (5-2c), respectively.

More complex paths were tested. These paths are a 3D conical spiral and a 3D hemispherical spiral as shown in figures (5-3) and (5-4), respectively. The complexity of the geometry of the given paths resulted in severe fluctuative position profile of all the joints of the robotic arm. Figures (5-3a) and (5-4a) show the position profiles that correspond to these two paths. Severe fluctuations in joints displacements have serious drawback from system stability point of view. When the links were heavy or when the end-effector was heavily loaded, high overshoot at each of the joints will result and an accumulated error in the path tracking process will exist. This can be avoided by choosing heavy duty actuators to bear these severe loading conditions. On the other hand, frequent fluctuation can induce vibrations in one (or more) of the links which produces unstable performance of the system. Figures (5-3b) and (5-4b) represent the velocity profiles for these two paths with respect to time. As is shown, more complex behavior for these velocity profiles exists. This is due to the large measured changes in the corresponding position profiles during the

time interval between any two points. This also applies to the acceleration profiles for these paths presented in figures (5-3c) and (5-3d), respectively.

From the above discussion, a robotic arm cannot be assigned any path to follow without taking into account the dynamic properties of the robot otherwise the robot will fail in accomplishing the task required since tracking accuracy is the key for acceptable performance of the robot. The simulation of the dynamics of the robotic arm plays a major roll in deciding whether a path is accurately trackable by the robotic arm or not. The closer the model to the real physical system was, the better decision is made.

However, the method used in this work for solving the inverse kinematics problem gave fast solutions and excellent accuracy in following any specified path regardless of its complexity. It also gave a smooth transition between any two consecutive configurations as the path was traced. As a comparison, the time consumed to obtain the inverse kinematics solution of the complicated hemispherical spiral path by using the new method was less than 6 minutes. George-Christopher Vosniakos and Zenon Kannas [45] presented a solution for the inverse kinematics problem based on the Genetic Algorithms. This approach was time consuming even for simple paths. The authors tested the method for short line segment path along the X -axis and a circular path. In both cases the desired solution was obtained after 28 hours and 40 minutes.

Although numerical solutions are fast in obtaining a possible kinematic configuration, they lag an important condition for acceptable robot performance, that is, the smooth transition between any two consecutive solutions. This is illustrated by comparing the numerical solution of the inverse kinematics of the 3D hemispherical path with the solution by using the new method for the same path. Figures (5-4a) and (5-4aa) show a comparison between the position profiles by using the two methods, respectively. Similarly, figures (5-4b) and (5-4bb) compare the velocity profiles, and figures (5-4c) and (5-4cc) compare the acceleration profiles. As shown in figure (5-4aa), the position

curves of all the joints are very far from being smooth and the maximum angular displacement was raised to a value of more than (131 degrees) after it was about (96 degrees). Also, the maximum velocity value became more than (47 deg/sec) after it was only (6 deg/sec). The acceleration maximum value was (66.33 deg/sq.sec) after it was only (4.67 deg/sq.sec). This has its effect on the dynamic performance of the robotic arm because this random-like behavior raises the values of the forces acting on the robotic arm, as explained in section 5.2 below, which requires an extra amount of power. On the other hand, this random-like behavior can induce unaccounted for vibrations in one or all the links of the robotic arm which means inaccurate path following and unstable performance of the robotic arm.

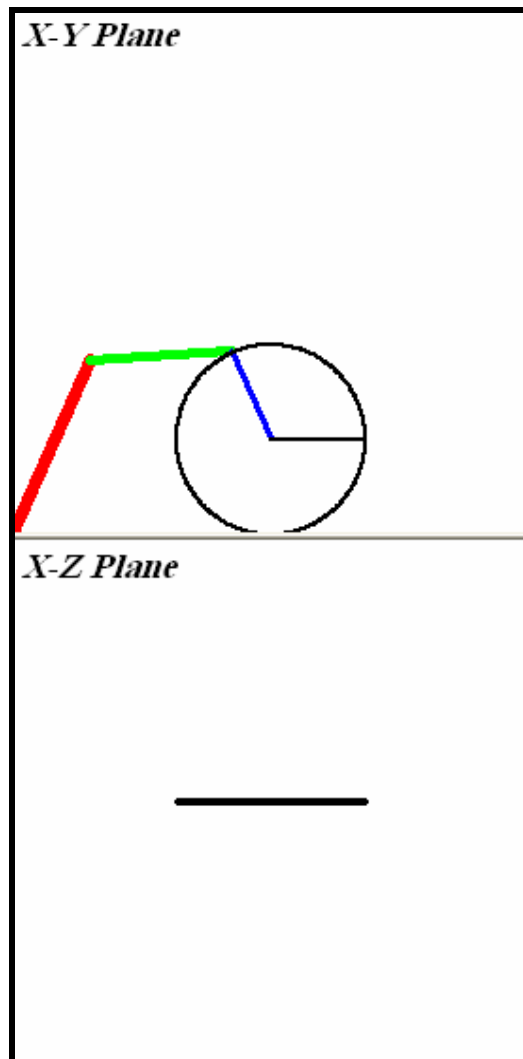
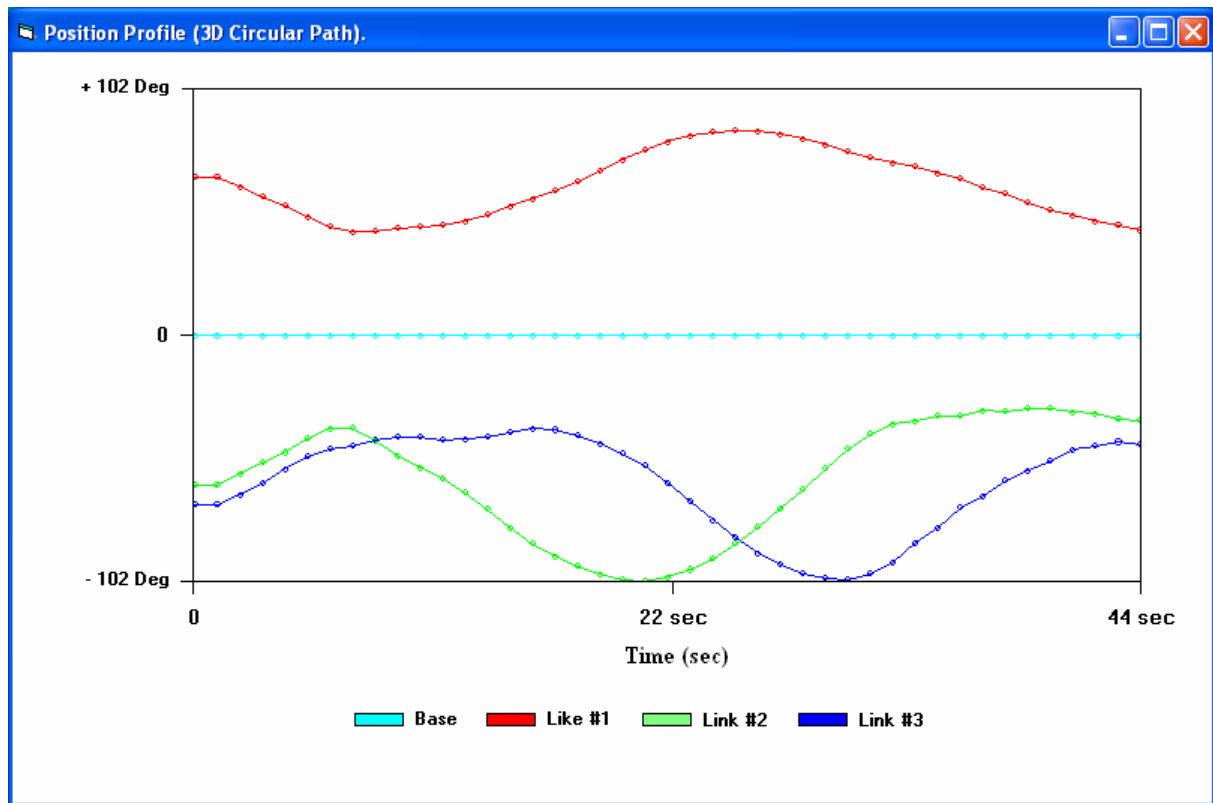
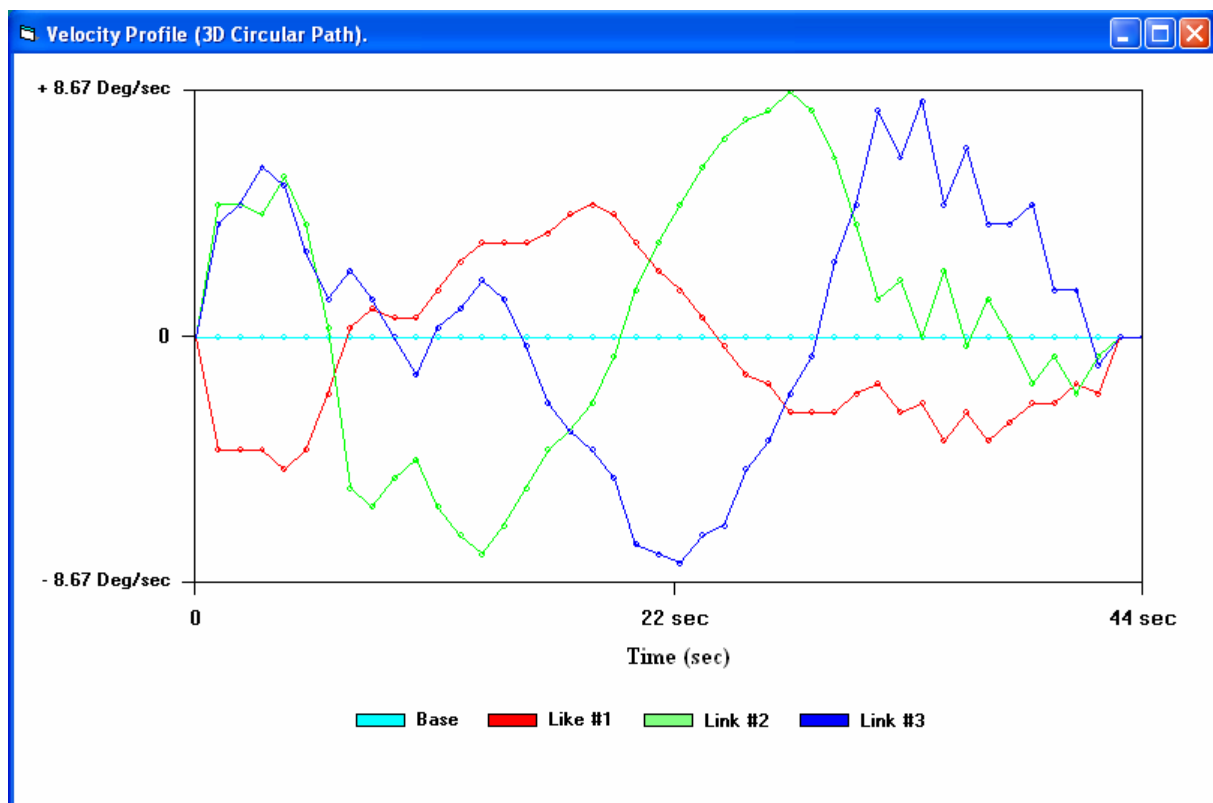


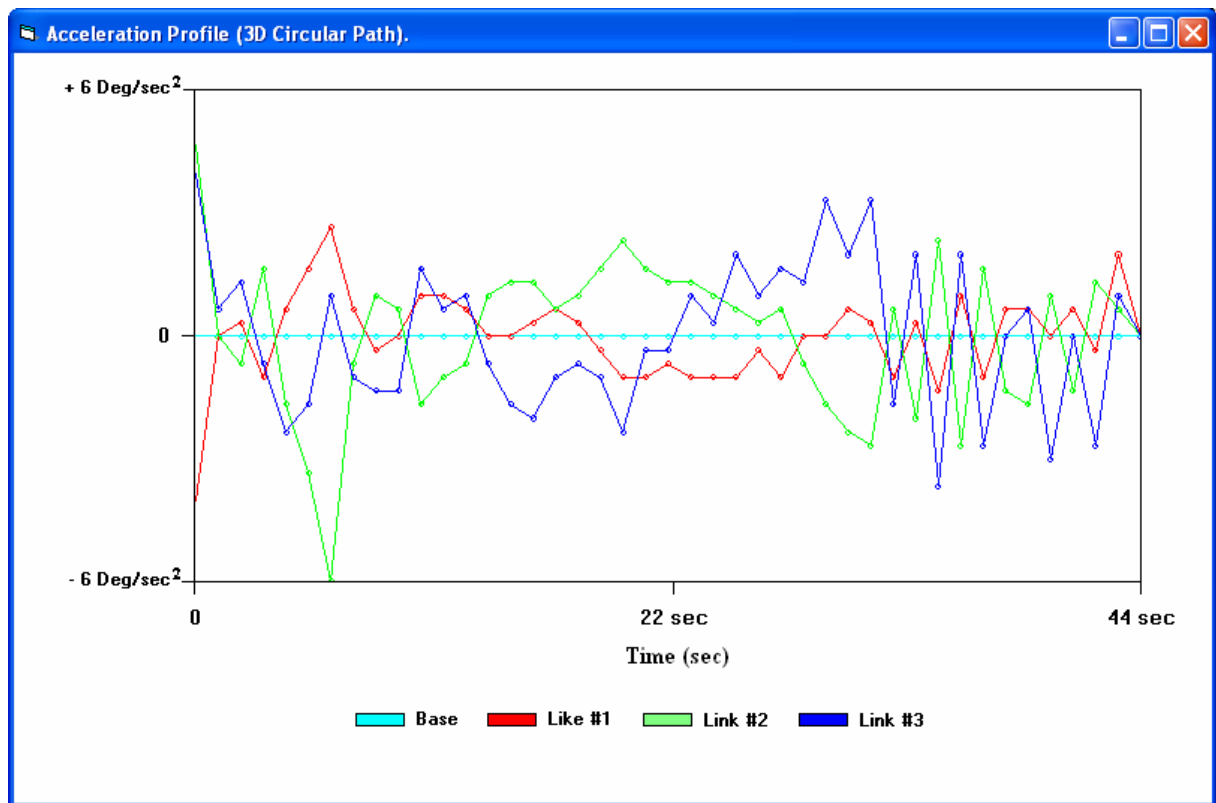
Figure (5-1) 3D circular path.



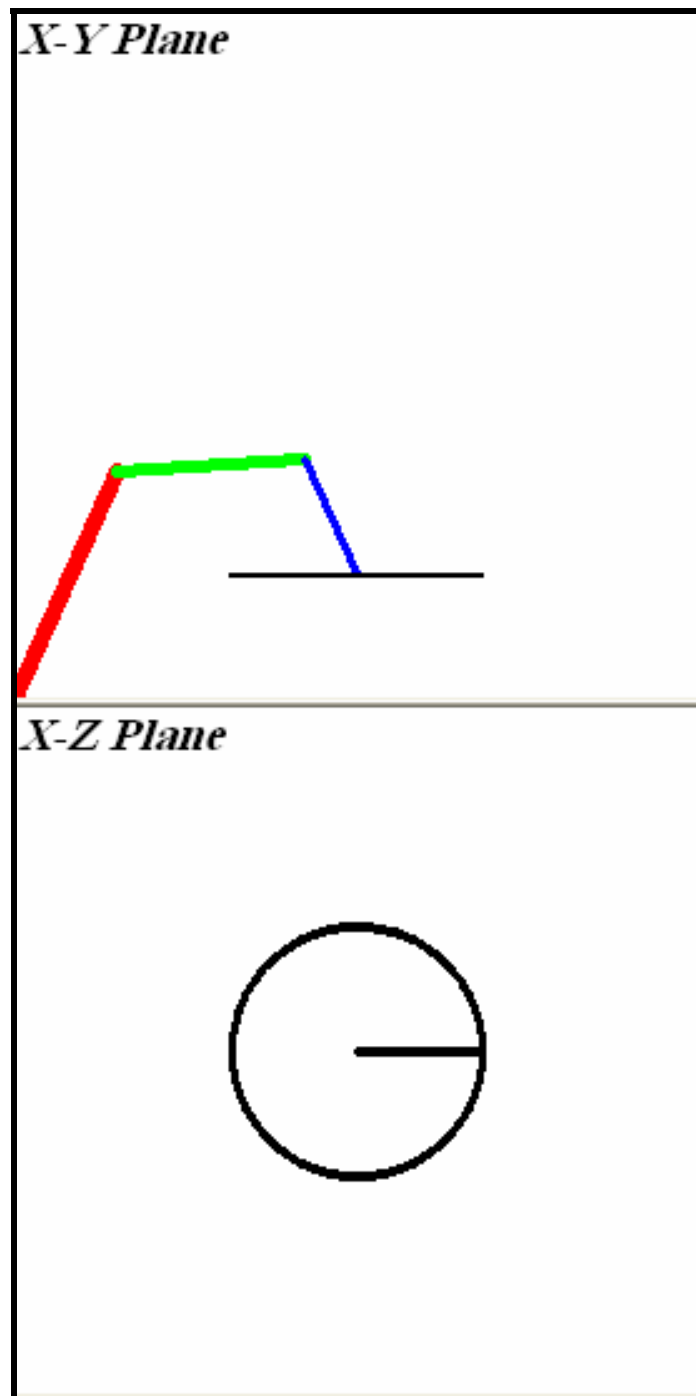
Figure(5-1a) Position Profile [3D Circular Path].



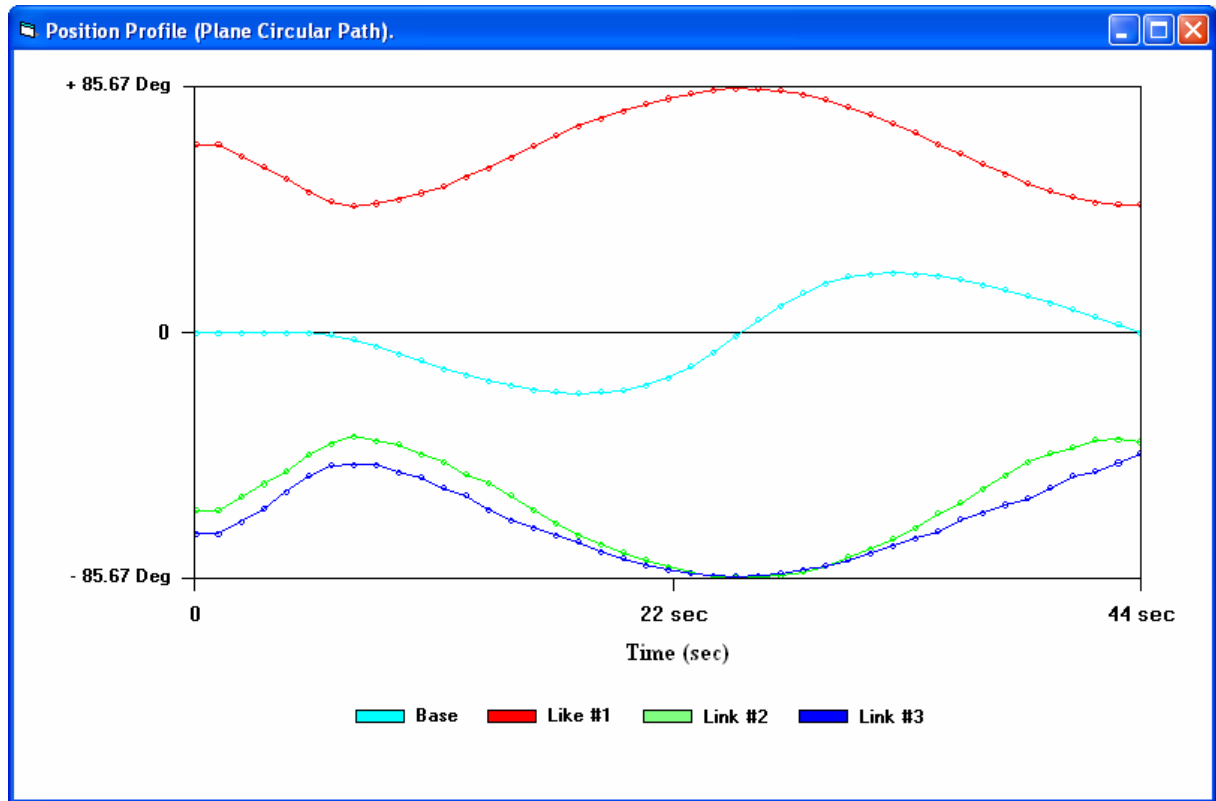
Figure(5-1b) the velocity profile [3D Circular Path].



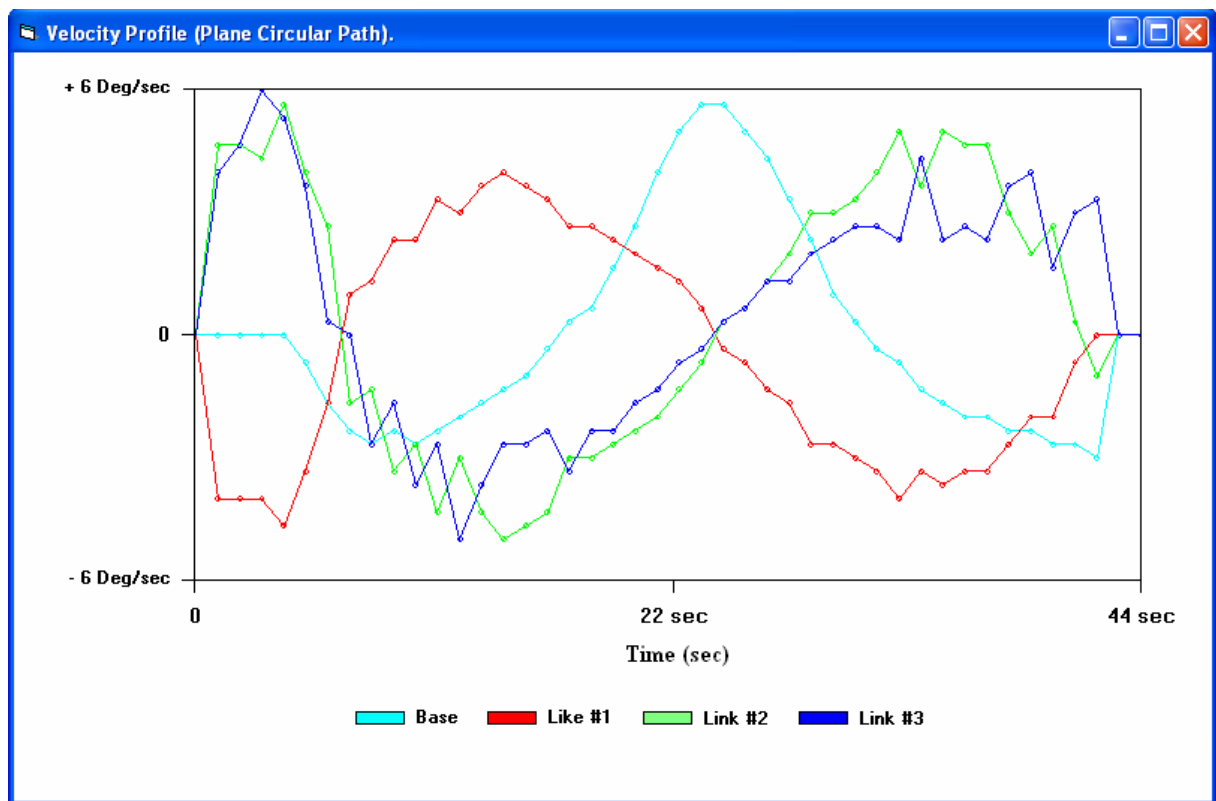
Figure(5-1c) acceleration profile [3D Circular Path].



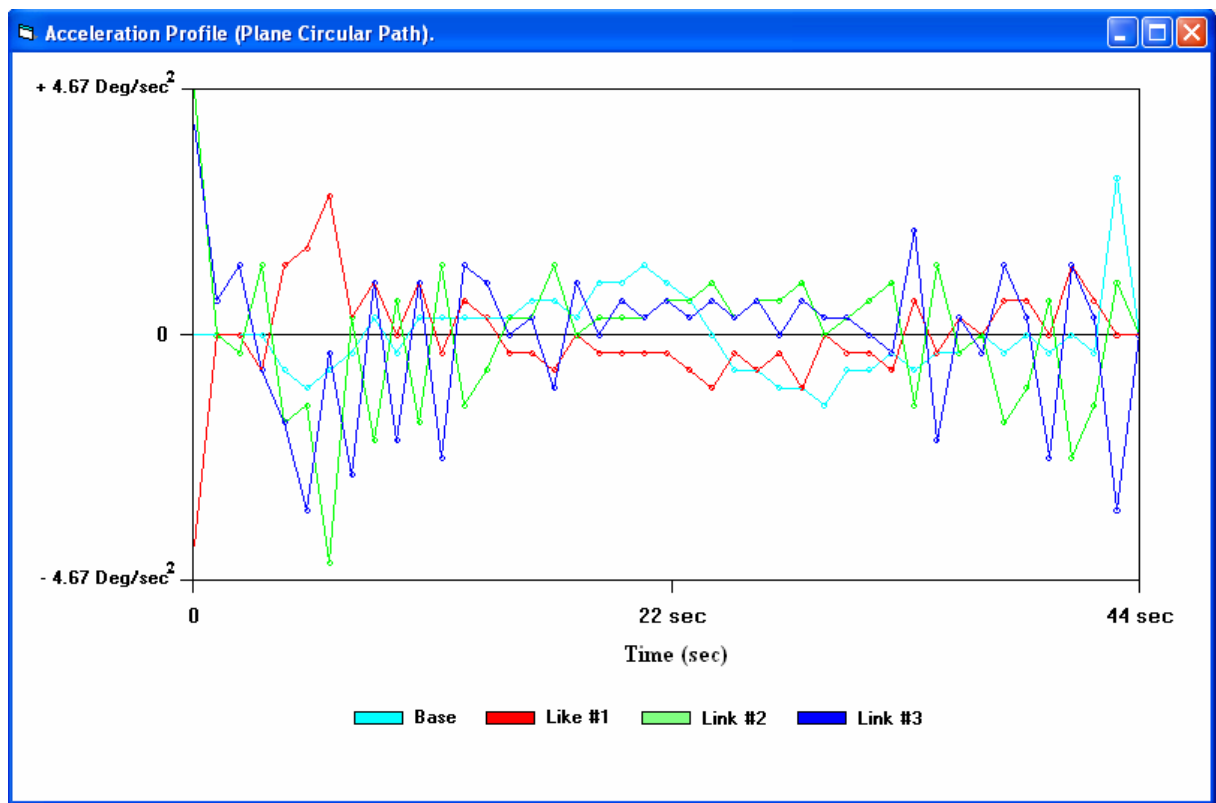
Figure(5-2) plane circular path.



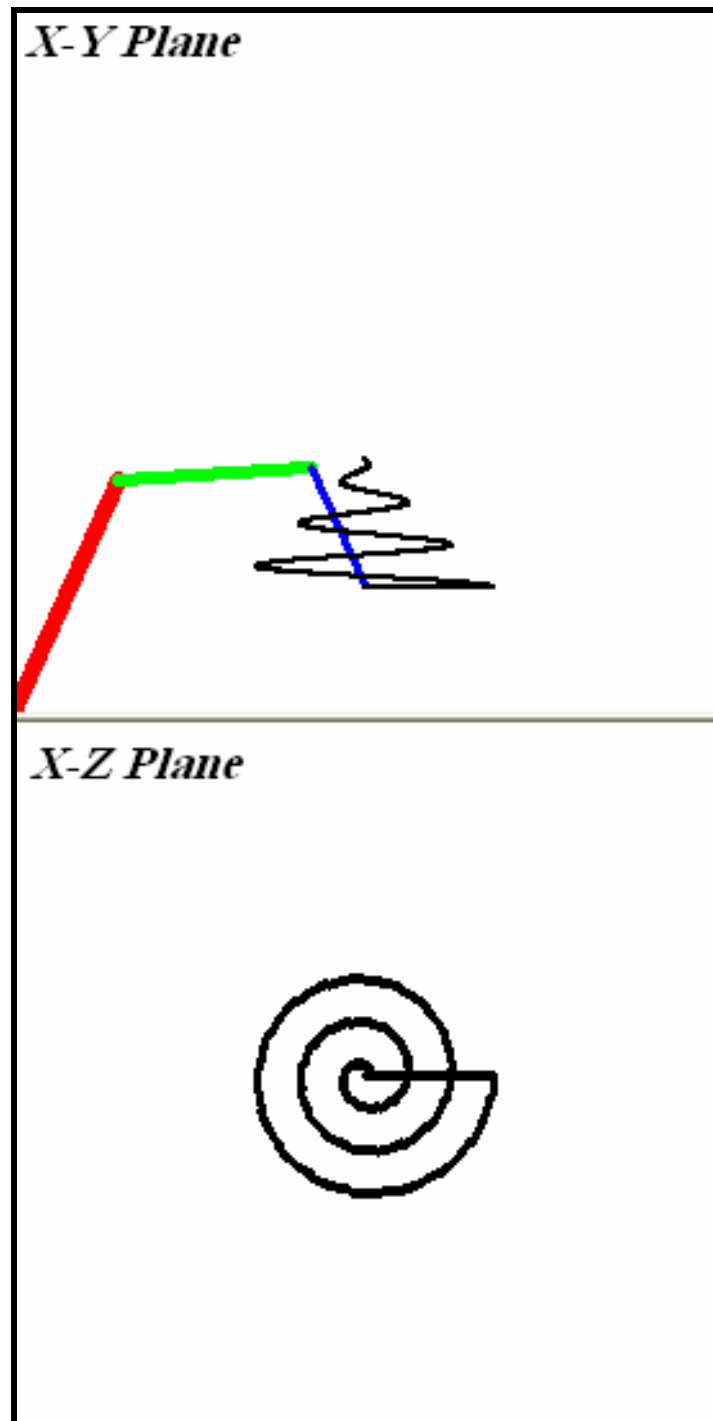
Figure(5-2a) position profile [Plane Circular Path].



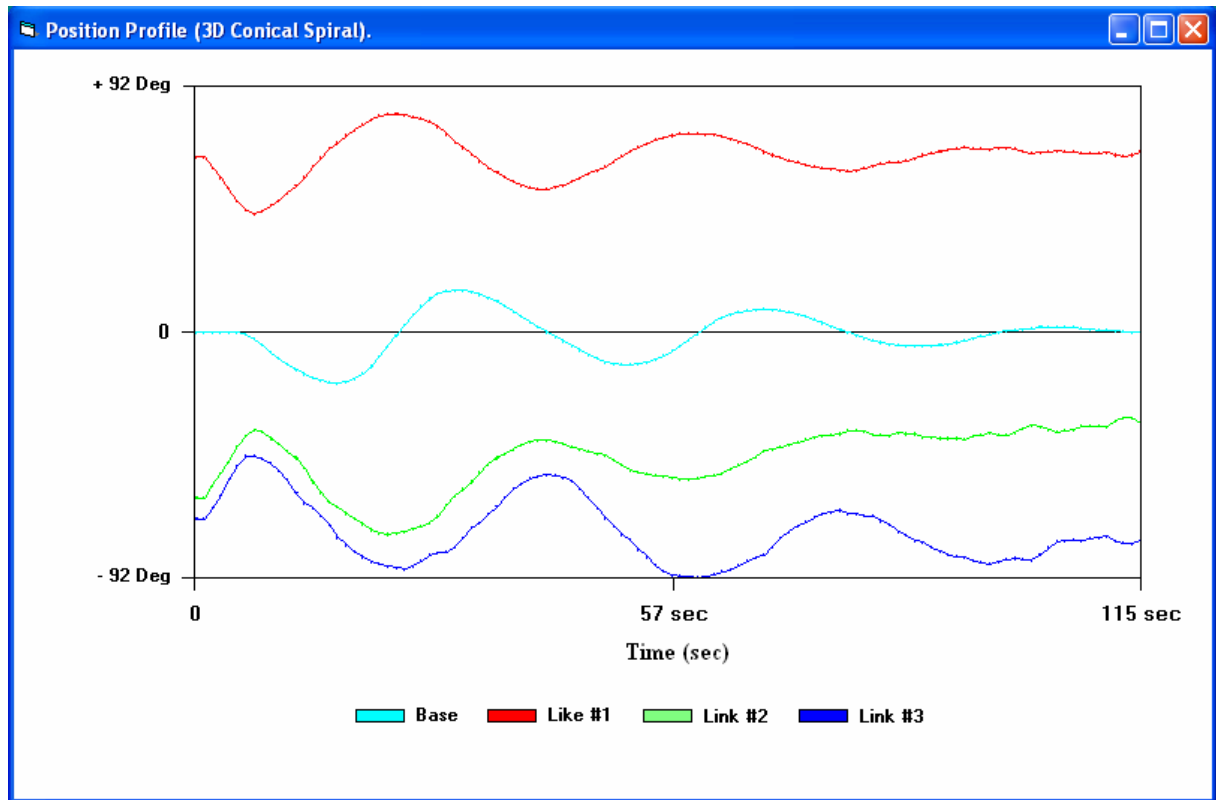
Figure(5-2b) velocity profile [Plane Circular Path].



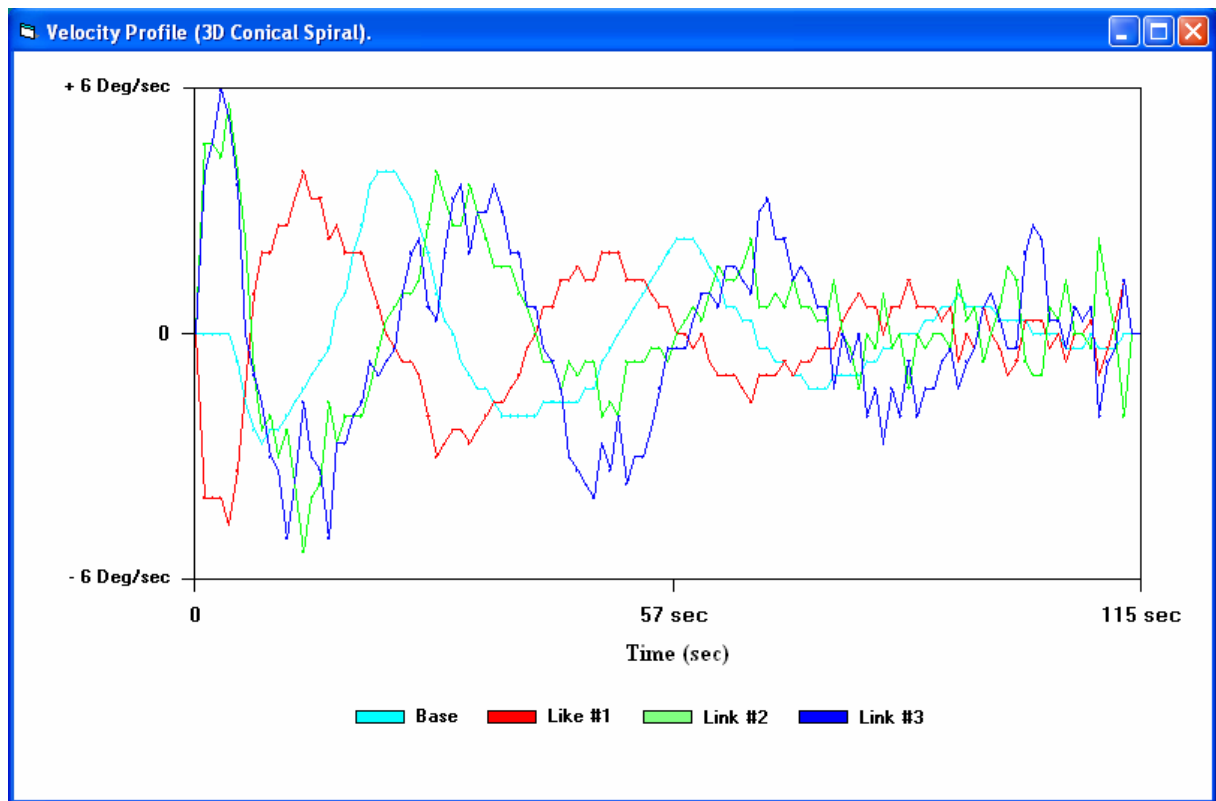
Figure(5-2c) acceleration profile [Plane Circular Path].



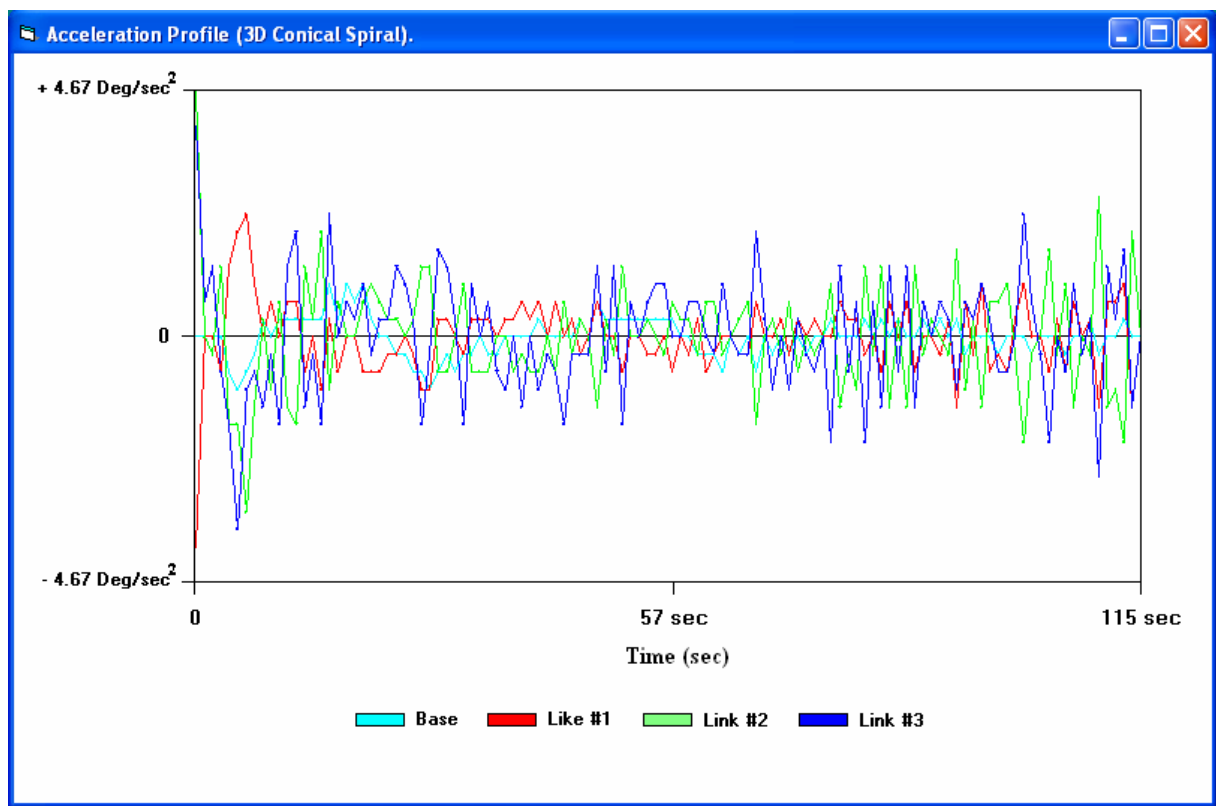
Figure(5-3) 3D conical spiral path.



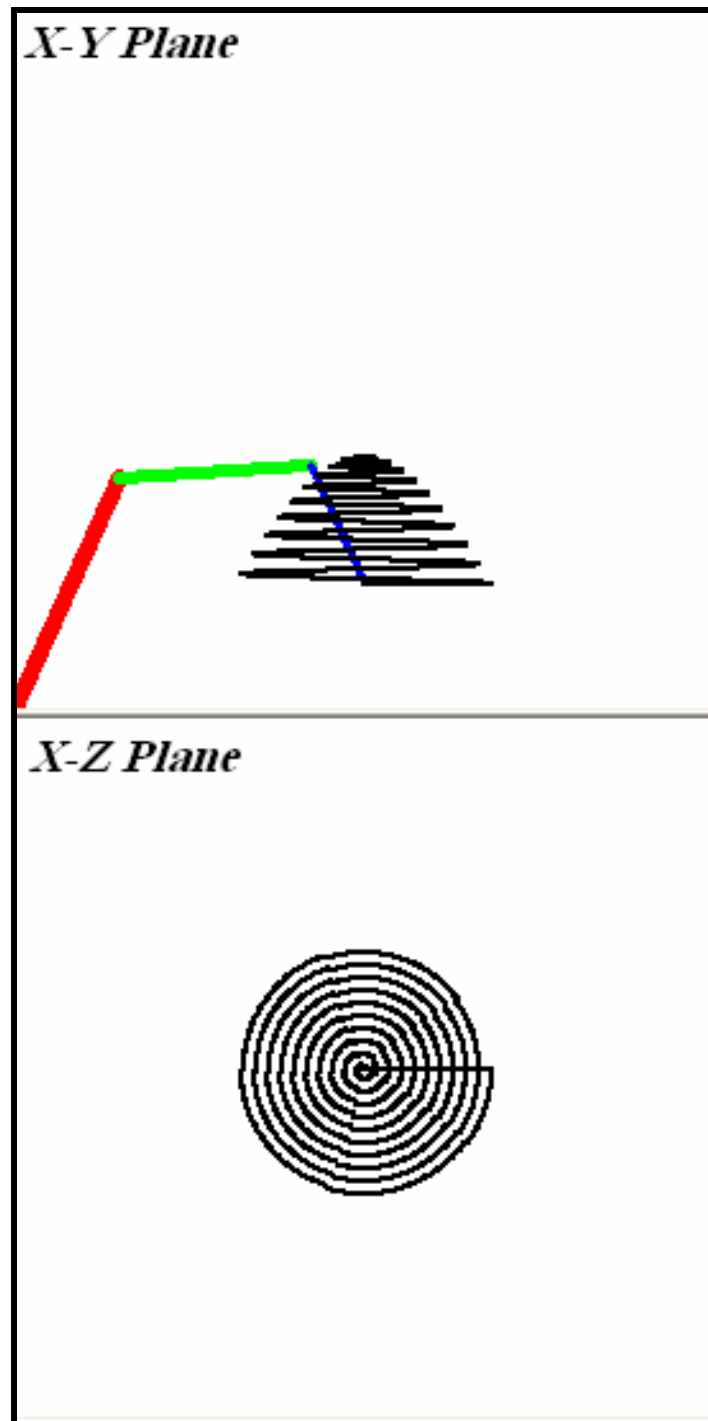
Figure(5-3a) position profile [3D Conical Spiral Path].



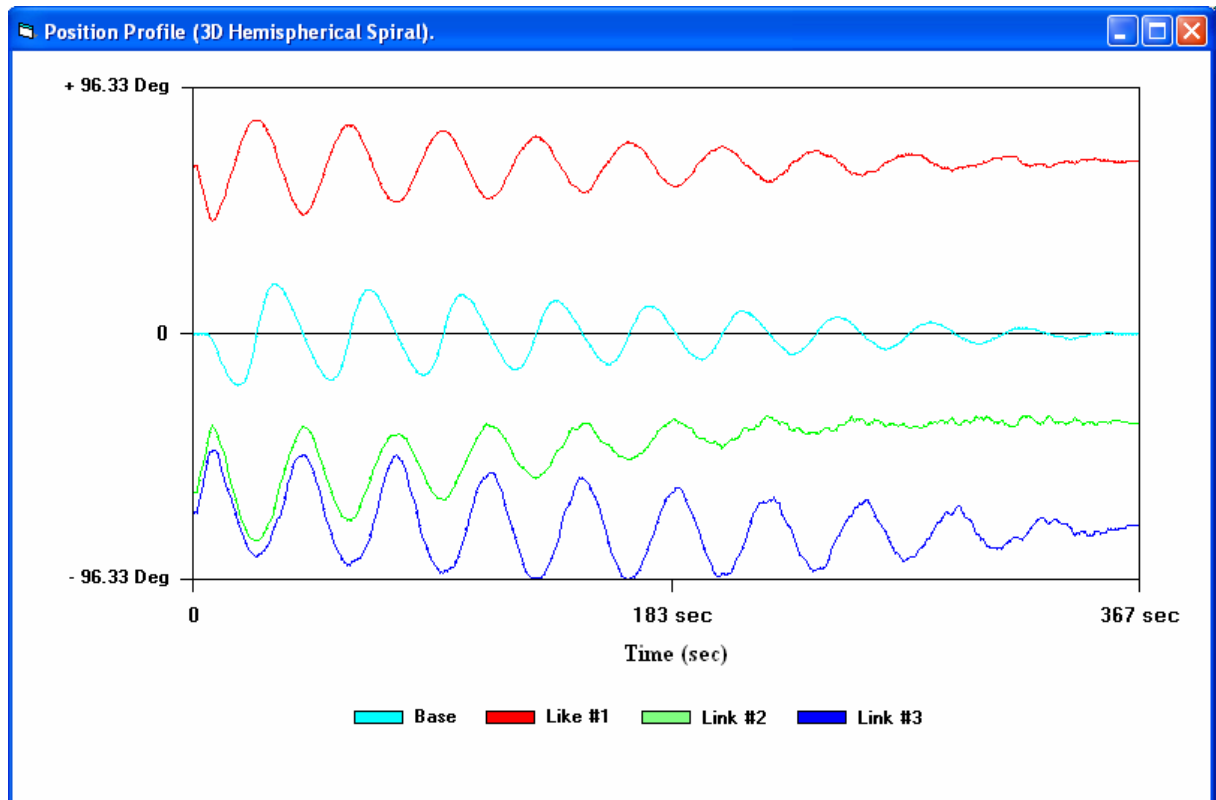
Figure(5-3b) velocity profile [3D Conical Spiral Path].



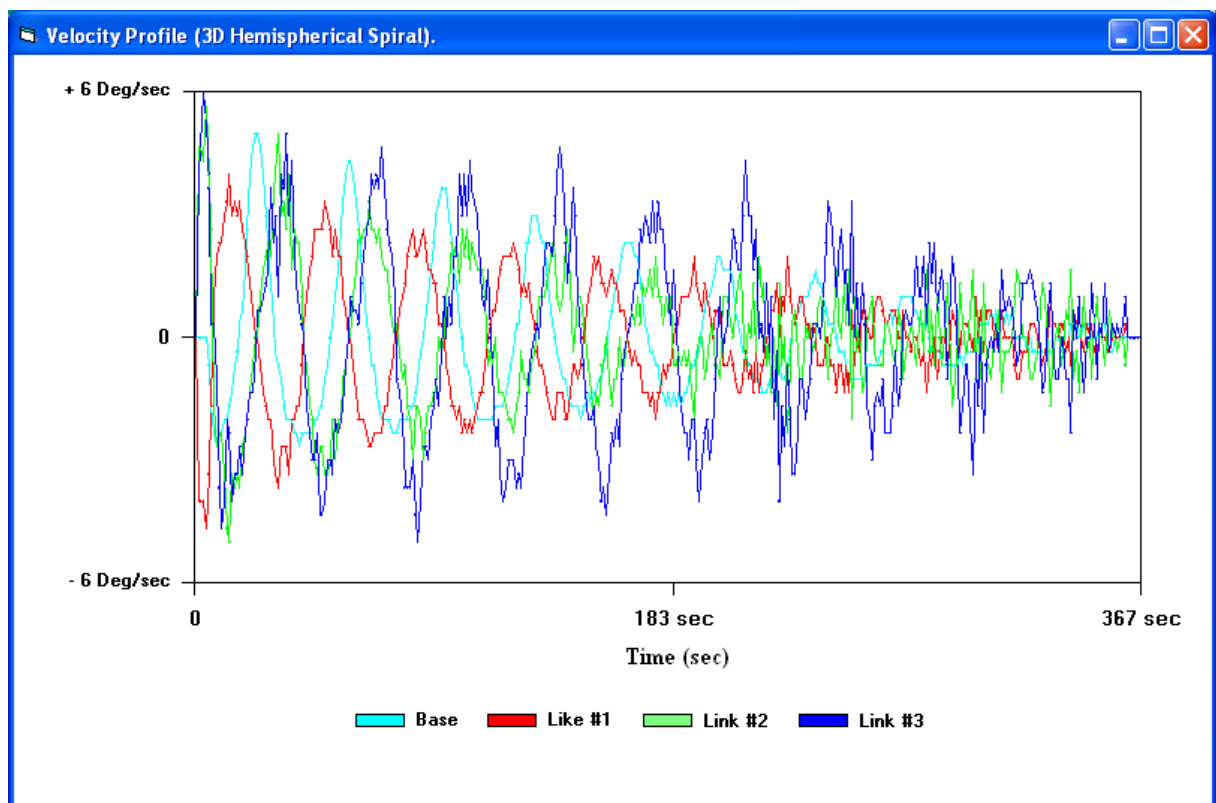
Figure(5-3c) acceleration profile [3D Conical Spiral Path].



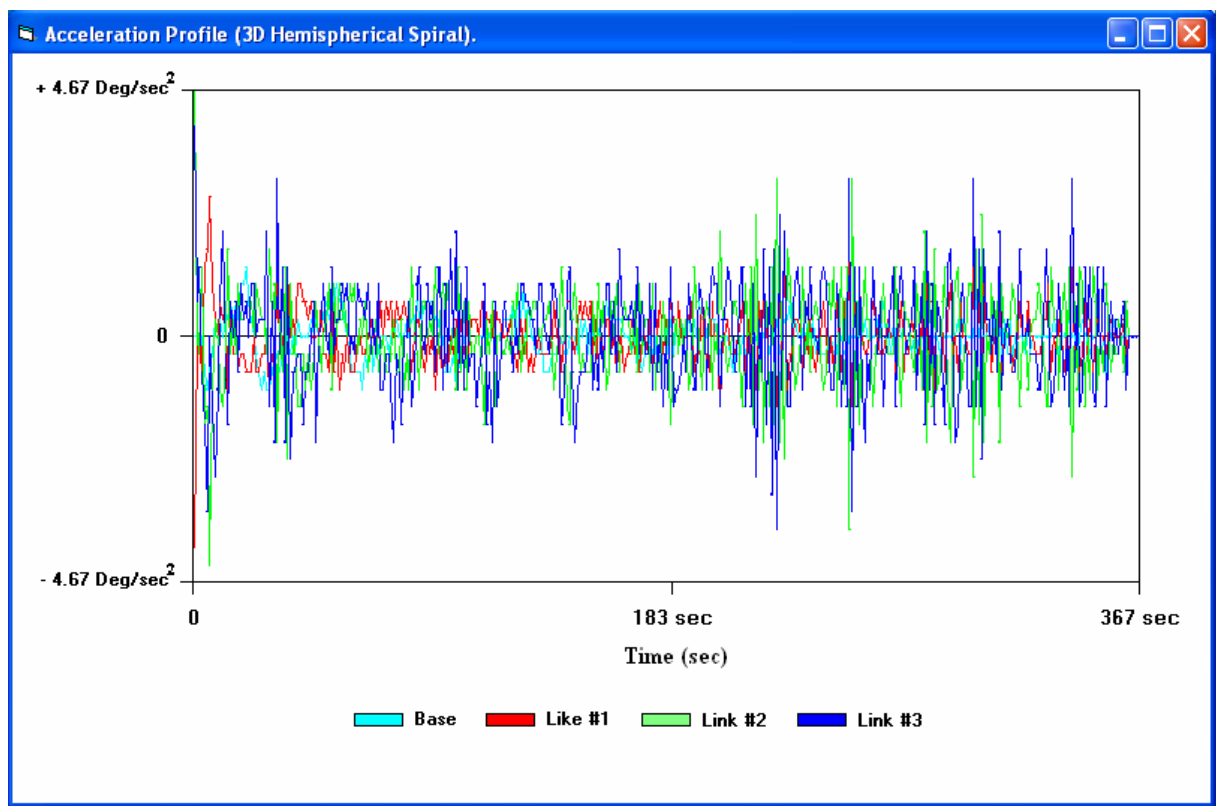
Figure(5-4) Hemispherical spiral path.



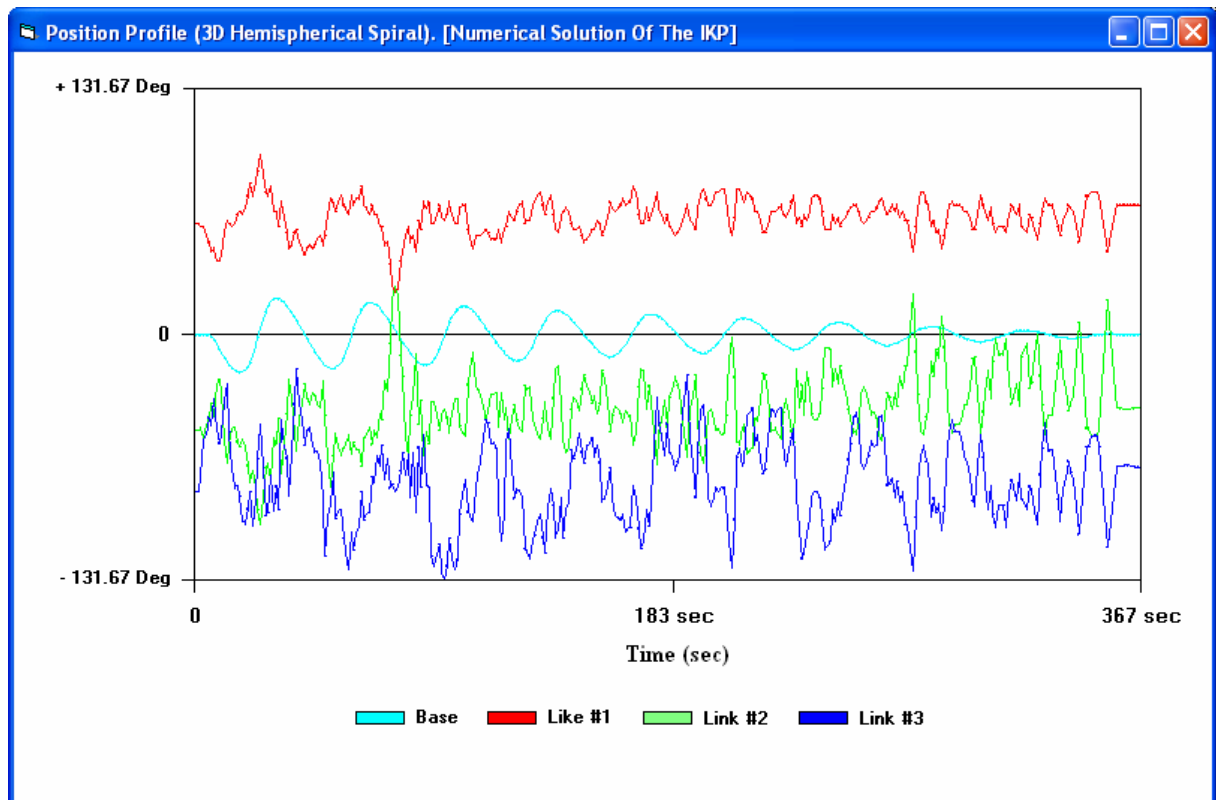
Figure(5-4a) position profile [Hemispherical Spiral Path].



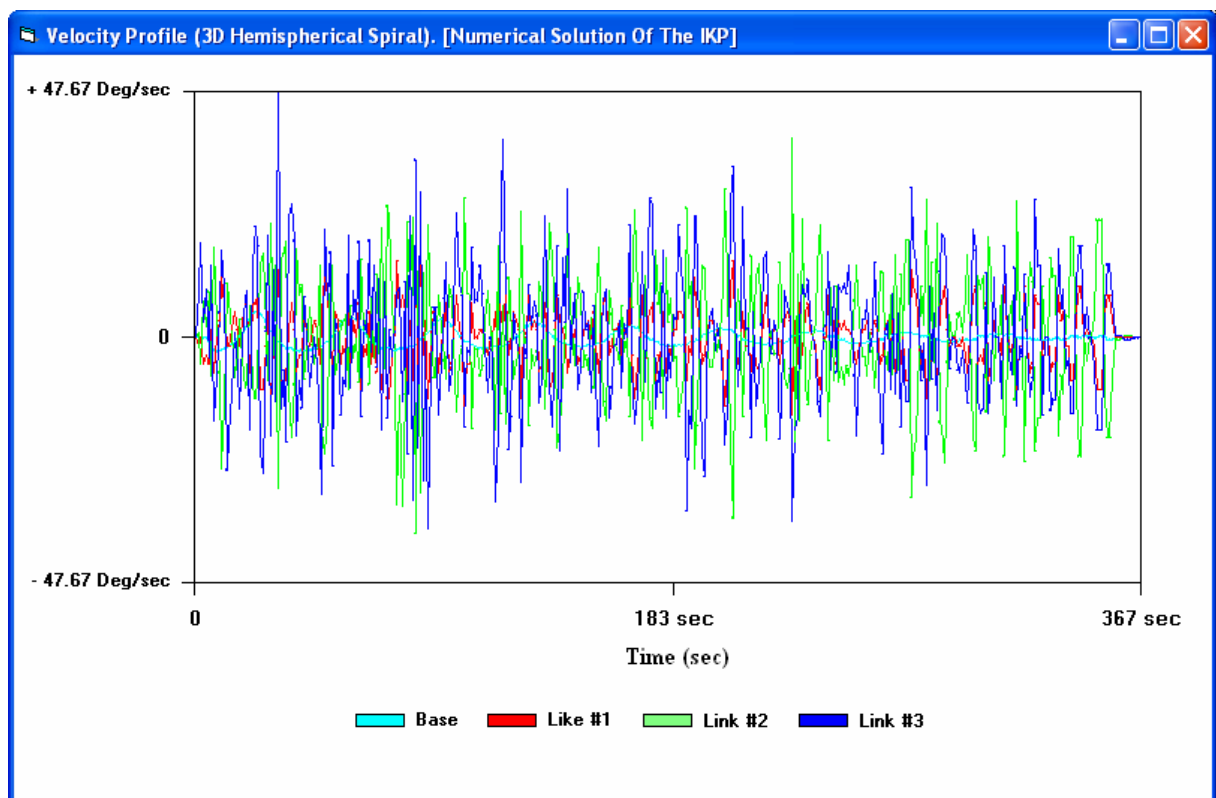
Figure(5-4b) velocity profile [Hemispherical Spiral Path].



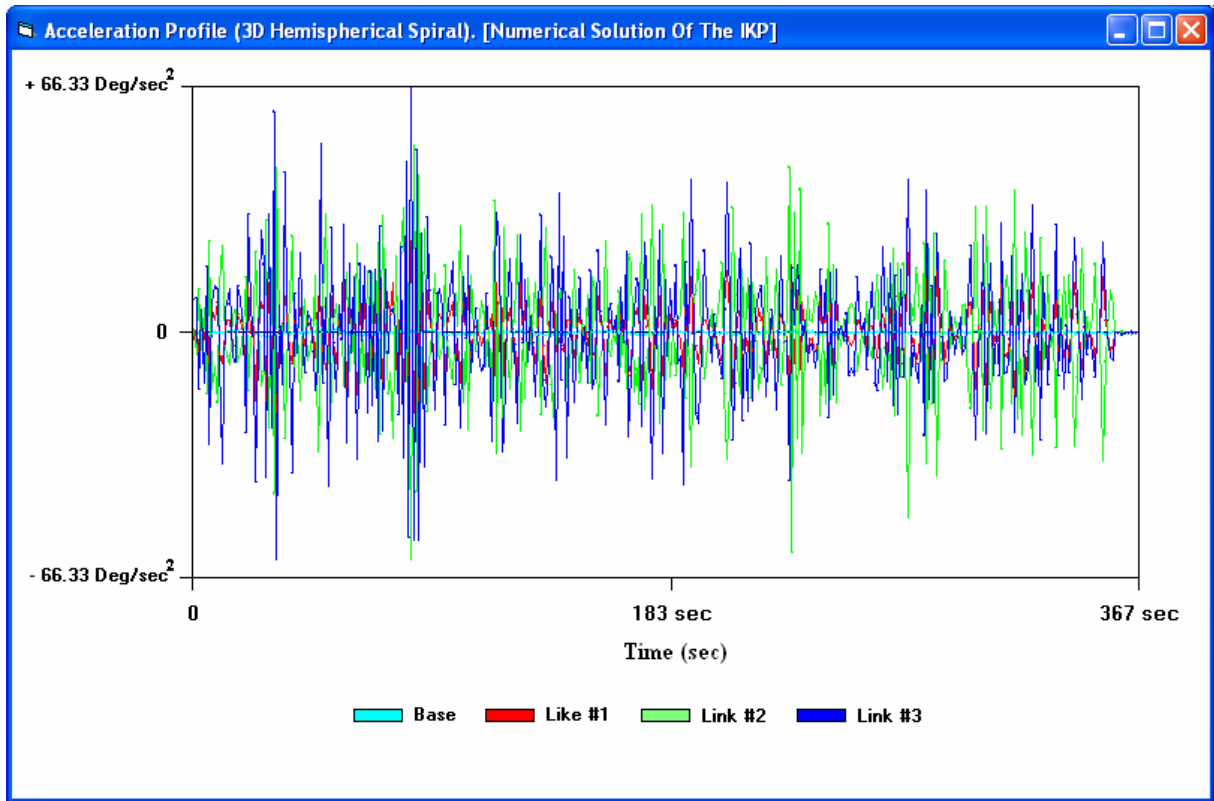
Figure(5-4c) acceleration profile [Hemispherical Spiral Path].



Figure(5-4aa) Position profile [Hemispherical Spiral Path], numerical solution.



Figure(5-4bb) velocity profile [Hemispherical Spiral Path], numerical solution.



Figure(5-4cc) Acceleration profile [Hemispherical Spiral Path], numerical solution.

5.2. Dynamics

The model that was built in this work simulated the dynamics of the four-link robotic arm. From this model the forces acting on the robotic arm were obtained. These forces include: the inertia forces, the coriolis forces, and the gravitational forces.

Knowing the above force vectors is sufficient to calculate the torque required at each joint because the torque at a joint is the algebraic sum of the inertia, coriolis and gravitational forces at that joint. These forces are highly non linear because the equations expressing them contain trigonometric functions of the angular displacement of the joints of the robot. This makes the profiles obtained for these forces look periodical and, as mentioned in the previous section, care must be taken about the loading conditions and the tracks assigned to the robotic arm to avoid unstable performance of the system during the

process of path tracking. The evaluation of these forces at each joint in every time step is important to help the designer choose the suitable type of the actuator at each joint to handle these force fluctuations.

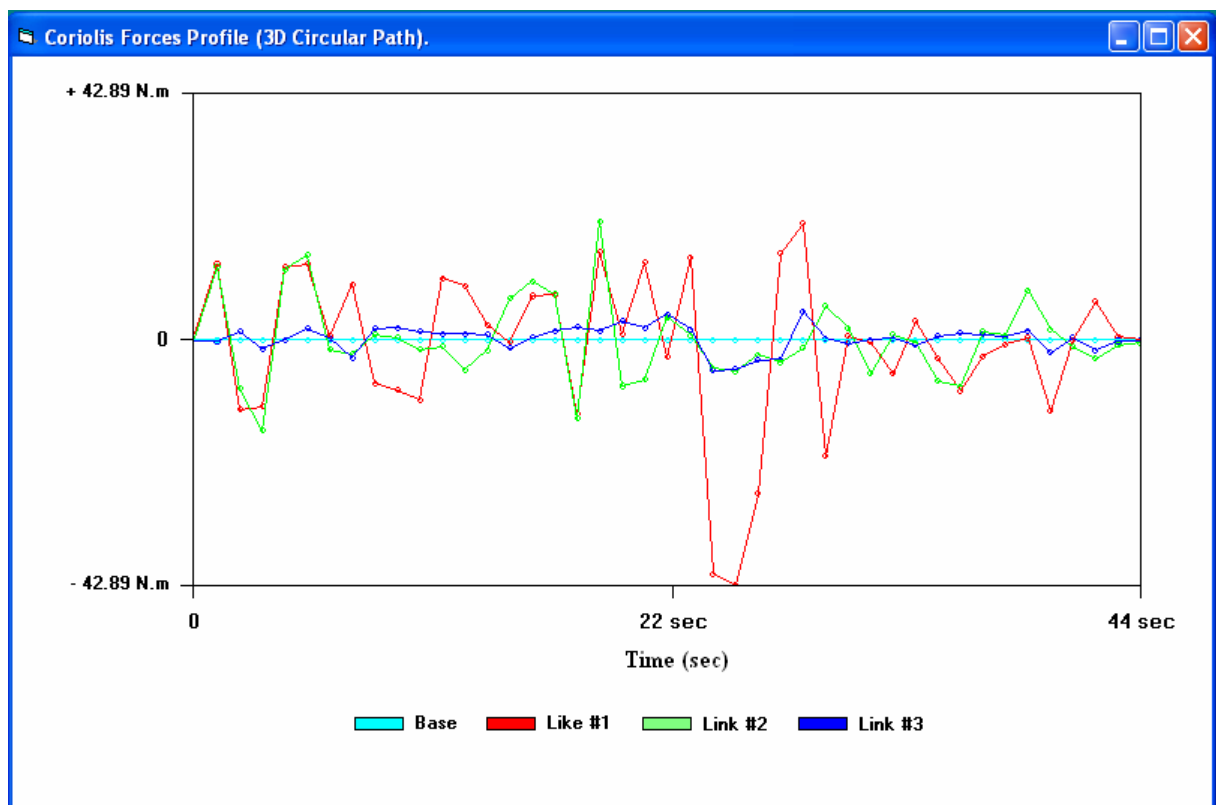
Figures (5-5), (5-8), (5-11), and (5-14) present relation between the coriolis forces and the time of path tracking for the four example paths. The behavior of the resulting profiles looks almost the same with little difference. This is due to the geometrical differences corresponding to each path.

Figures (5-6), (5-9), (5-12), and (5-15) show the relation between the gravitational forces at each joint and the time duration of path tracking for the four paths selected. The conical spiral and hemispherical spiral profiles gave higher forces during the motion. This is due to the abrupt changes in the angular displacement at each joint as the robot traces these trajectories.

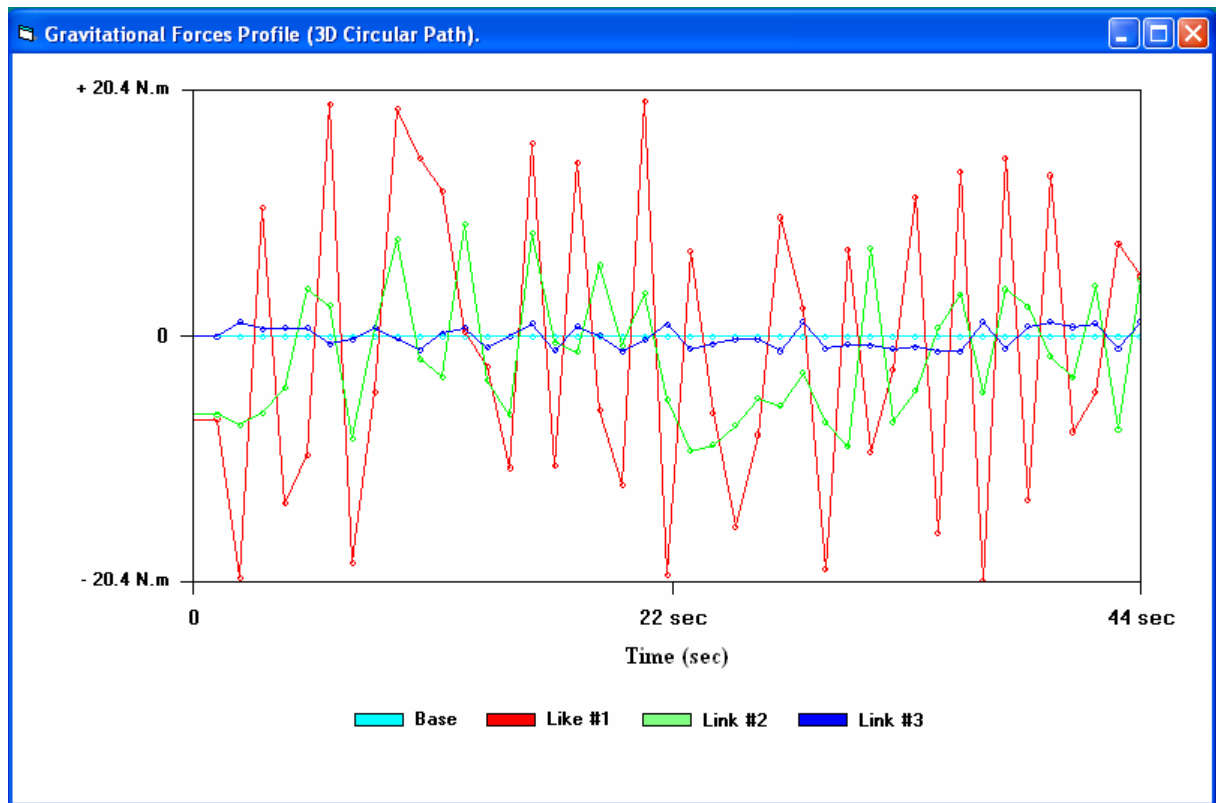
Figures (5-7), (5-10), (5-13), and (5-16) show the torques required at each joint during the motion in the form of the torque profile for each trajectory. As is shown, the value of the torque obtained with the circular path is higher than the torques on for the other paths.

A comparison is made between the values of the dynamic forces acting on the robotic arm when the new method for solving the inverse kinematics and the values of these forces when the numerical method for obtaining the inverse kinematics solution was used. As mentioned earlier, a numerical method produces a random-like sequence of solutions for a given path. This has its effect on the dynamic forces acting on the robotic arm, which is shown in figures (5-14a), (5-15a), and (5-16a). From these figures, the maximum value of the coriolis forces was raised to more than (390 N.m) after it was only (15.77 N.m), the gravitational forces have changed from (25.72 N.m) to (26.57 N.m). Accordingly, the maximum torque was raised to a value of about (462 N.m) after it was slightly above (33 N.m). This makes the new method of solving the robotic arm inverse kinematics a competitive to the commonly used methods used for the same purpose.

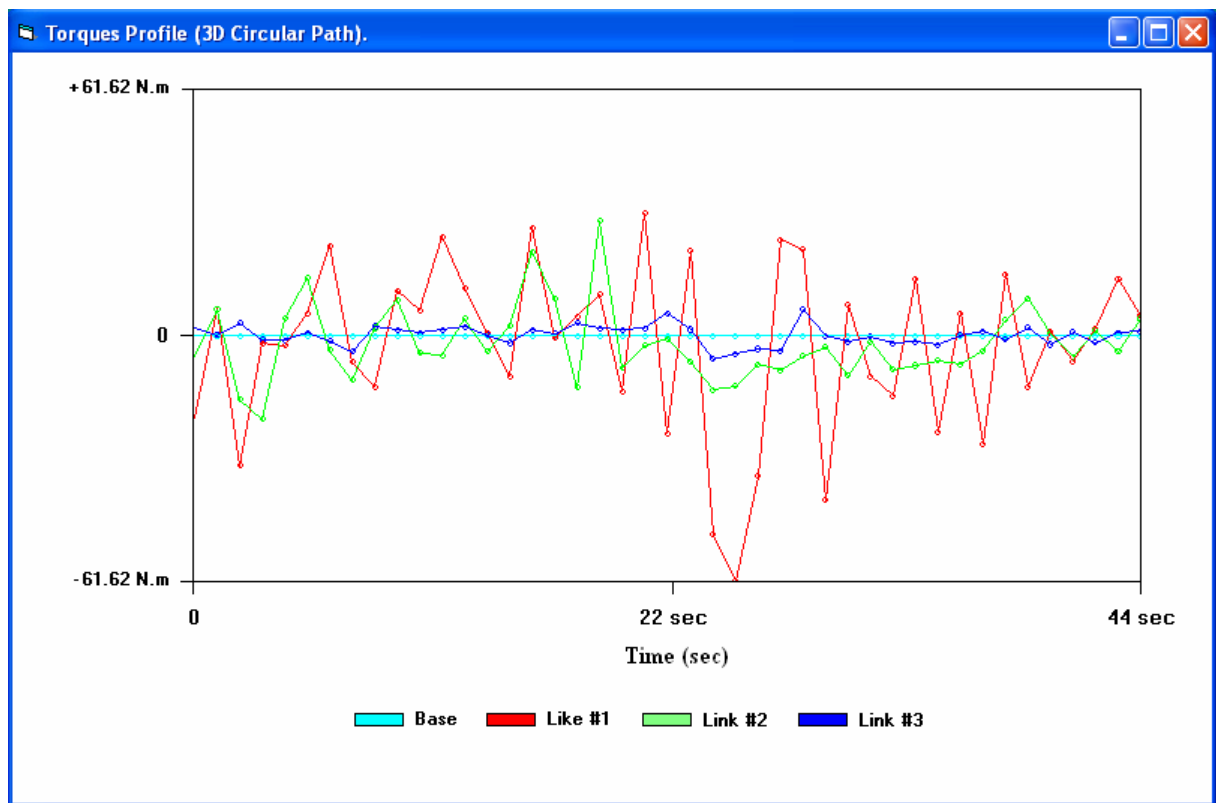
Rafael M. Inigo and James S. Morton [46] proposed an algorithm for simulating the dynamics of a two-link robotic manipulator. They built their model using the Lagrange-Euler formulation. The authors solved the inverse kinematics of this simple robotic arm using the transformation matrices to evaluate the joint space for a desired end-effector position. They used the pseudo-inertia matrices to represent the inertia of the assembly. However, they did not present any data to show the performance of their path tracking method. So, there is no guarantee that the robot will track a given path with smooth movement, since numerical solutions were adopted. Also, the use of transformation matrices (which are in terms of trigonometric functions) complicates the analysis, specially, for more complex robotic arm configuration.



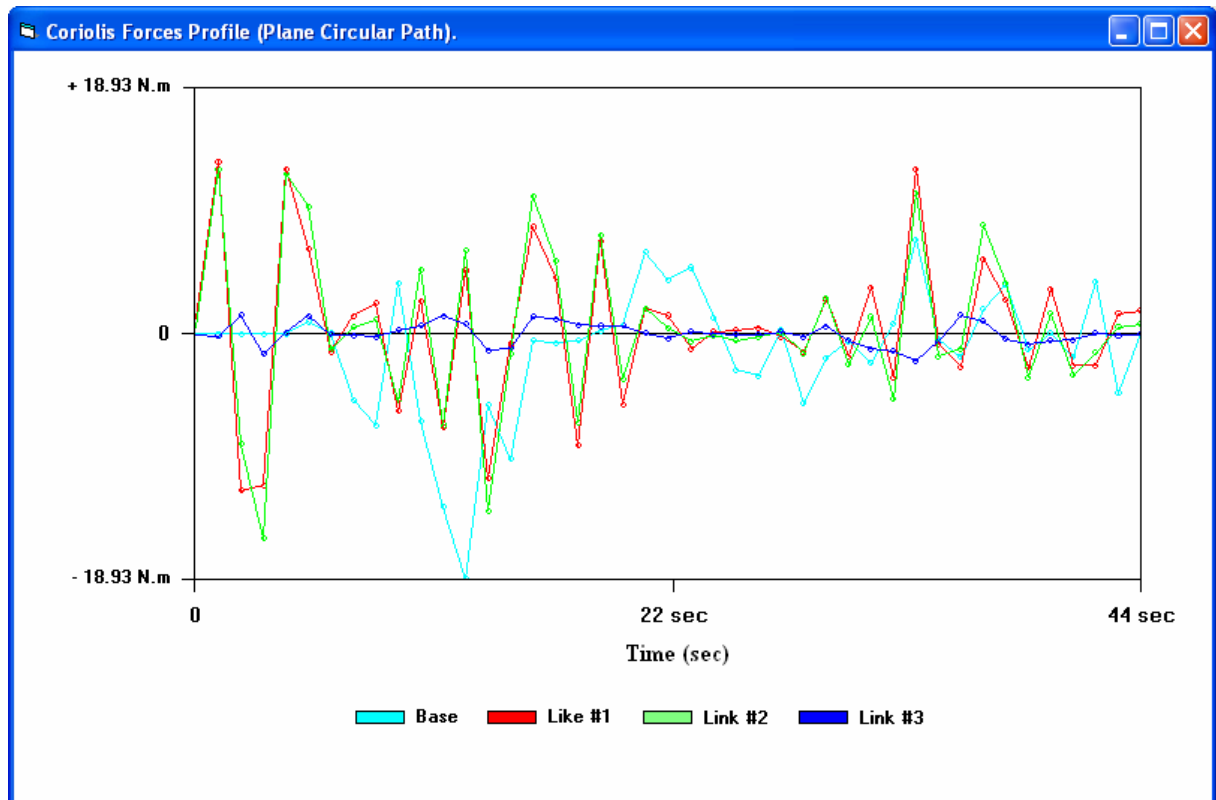
Figure(5-5) centrifugal & coriolis forces profile [3D Circular Path].



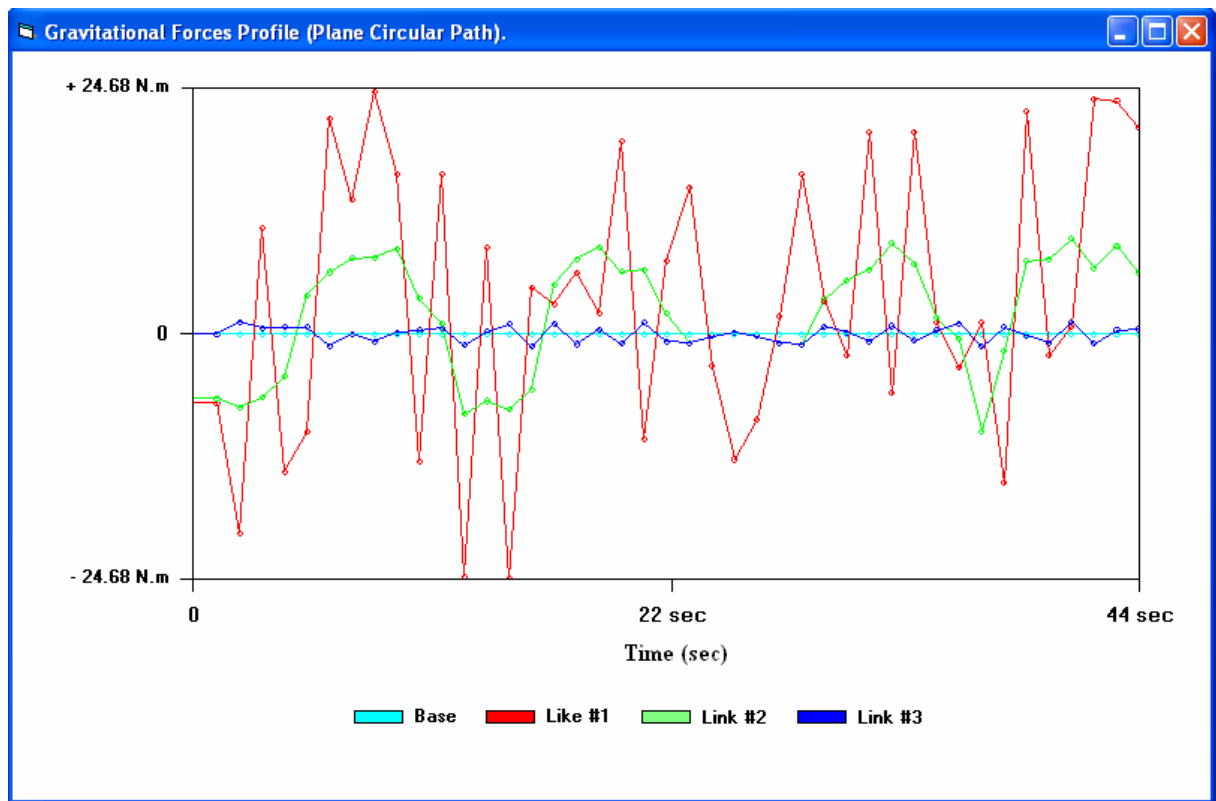
Figure(5-6) gravitational forces profile [3DCircular Path].



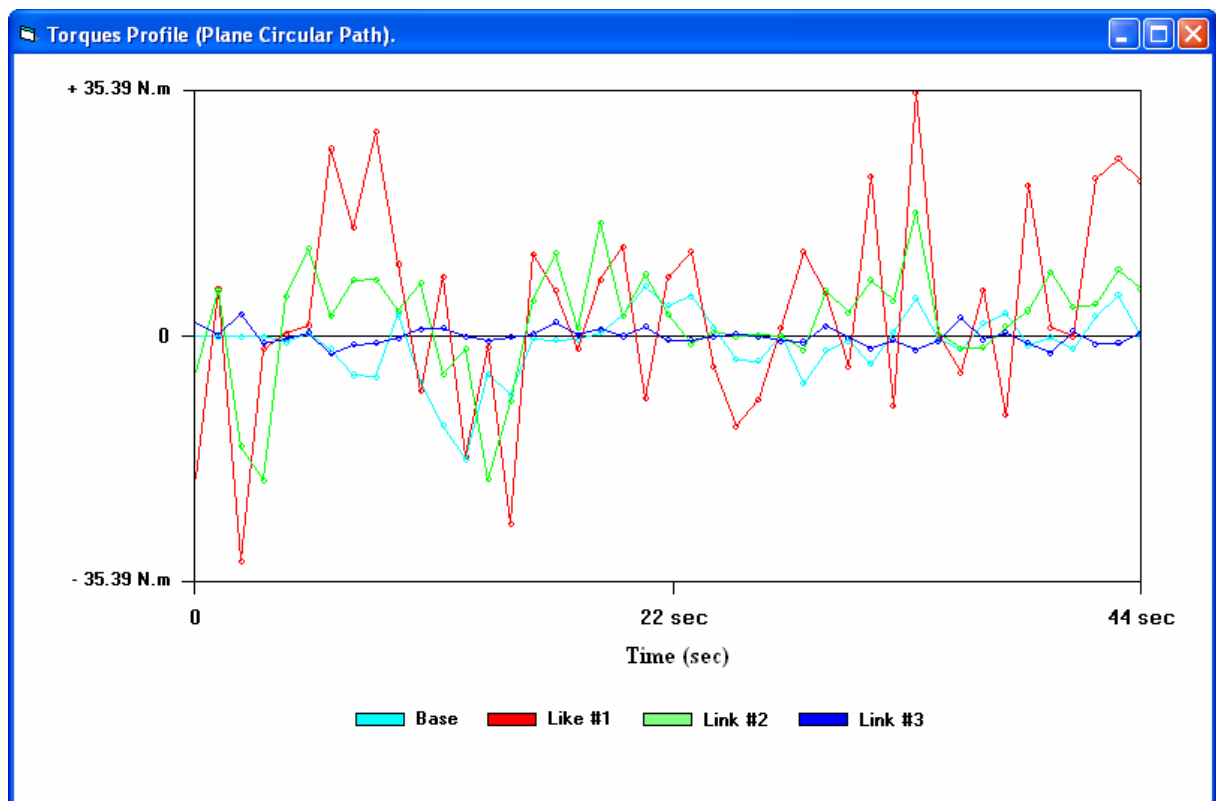
Figure(5-7) torques profile [3D Circular Path].



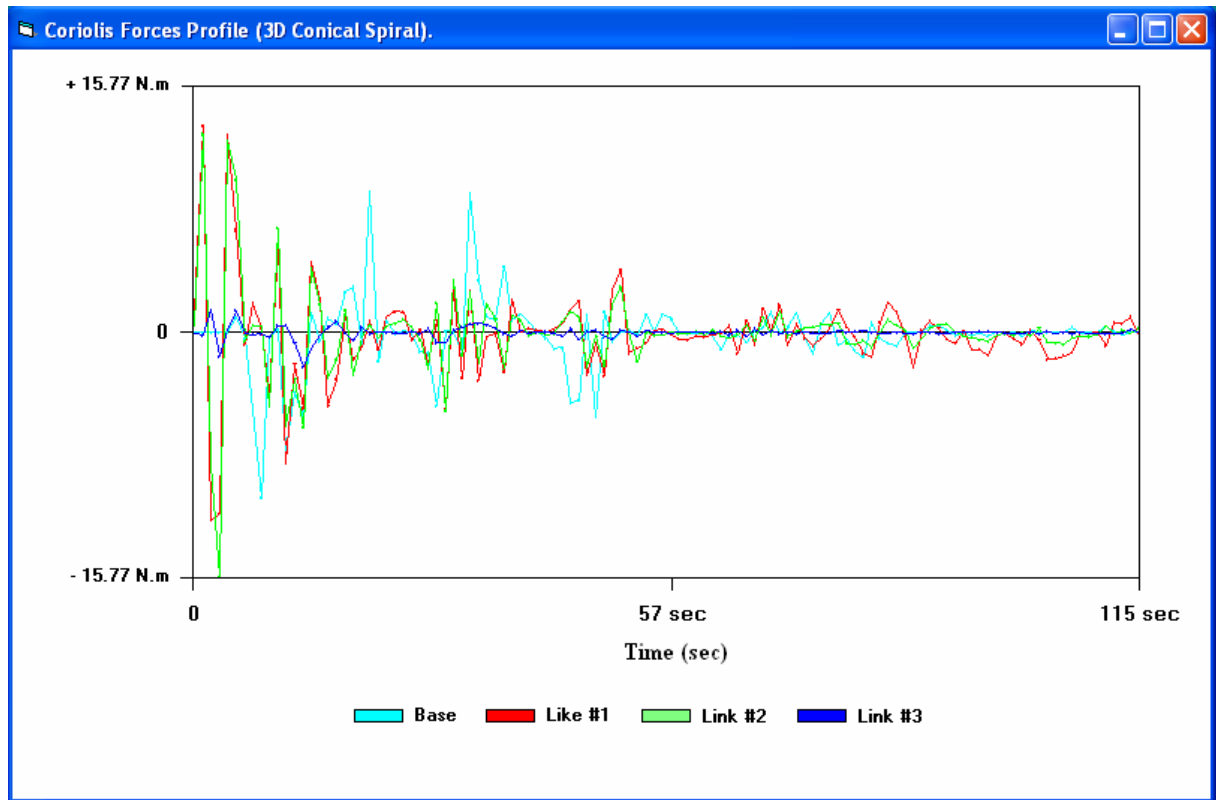
Figure(5-8) centrifugal & coriolis forces profile [Plane Circular Path].



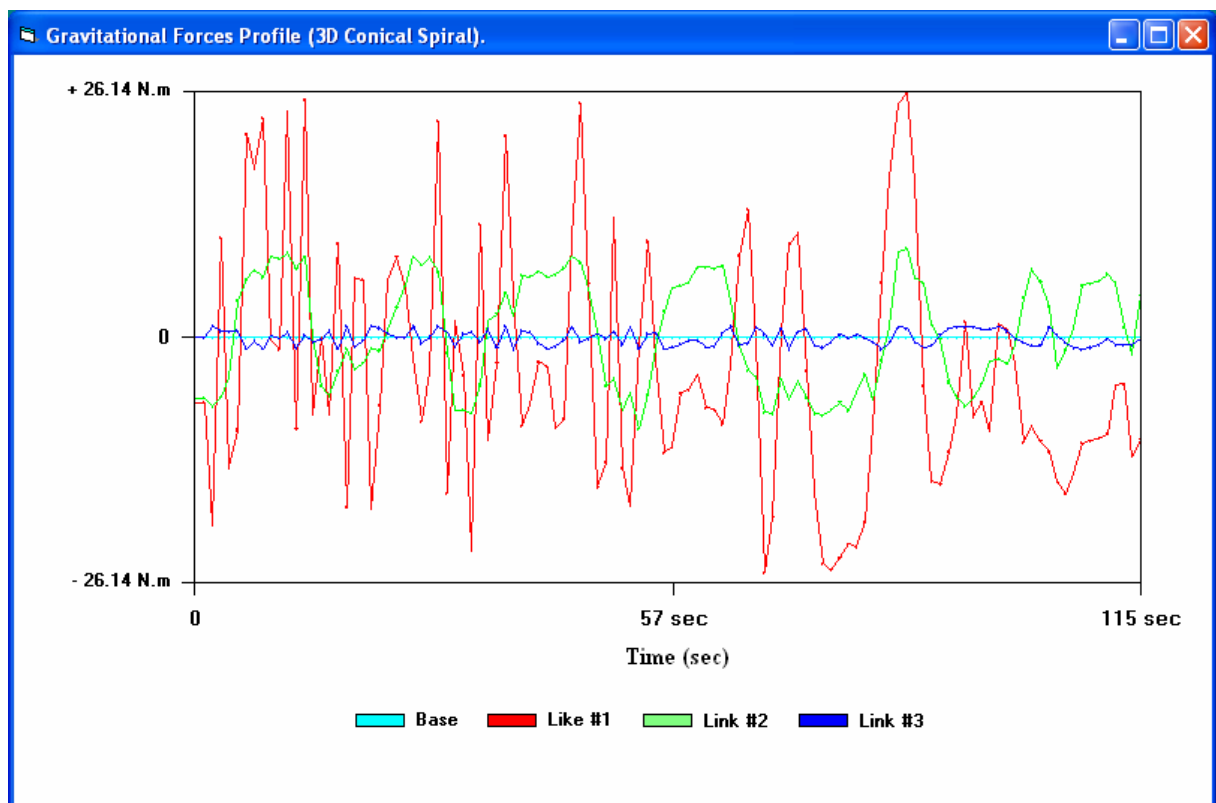
Figure(5-9) gravitational forces profile [Plane Circular Path].



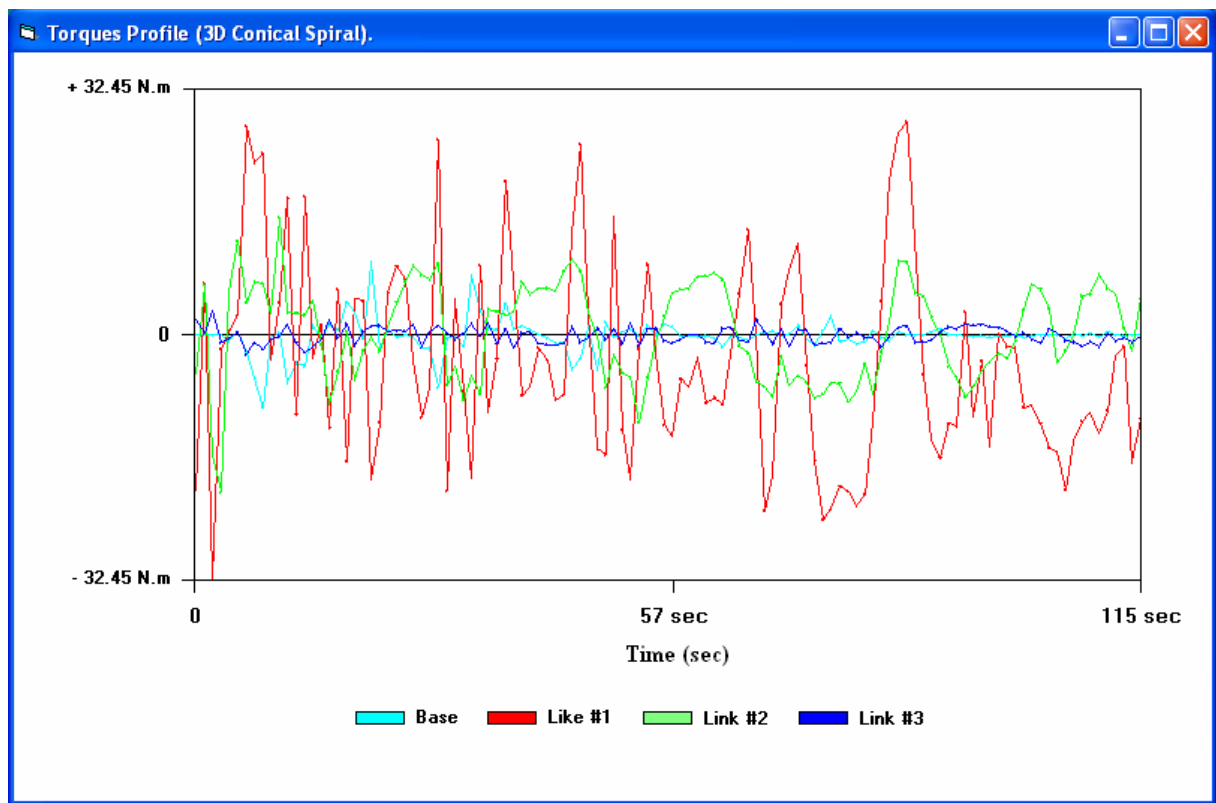
Figure(5-10) torques profile [Plane Circular Path].



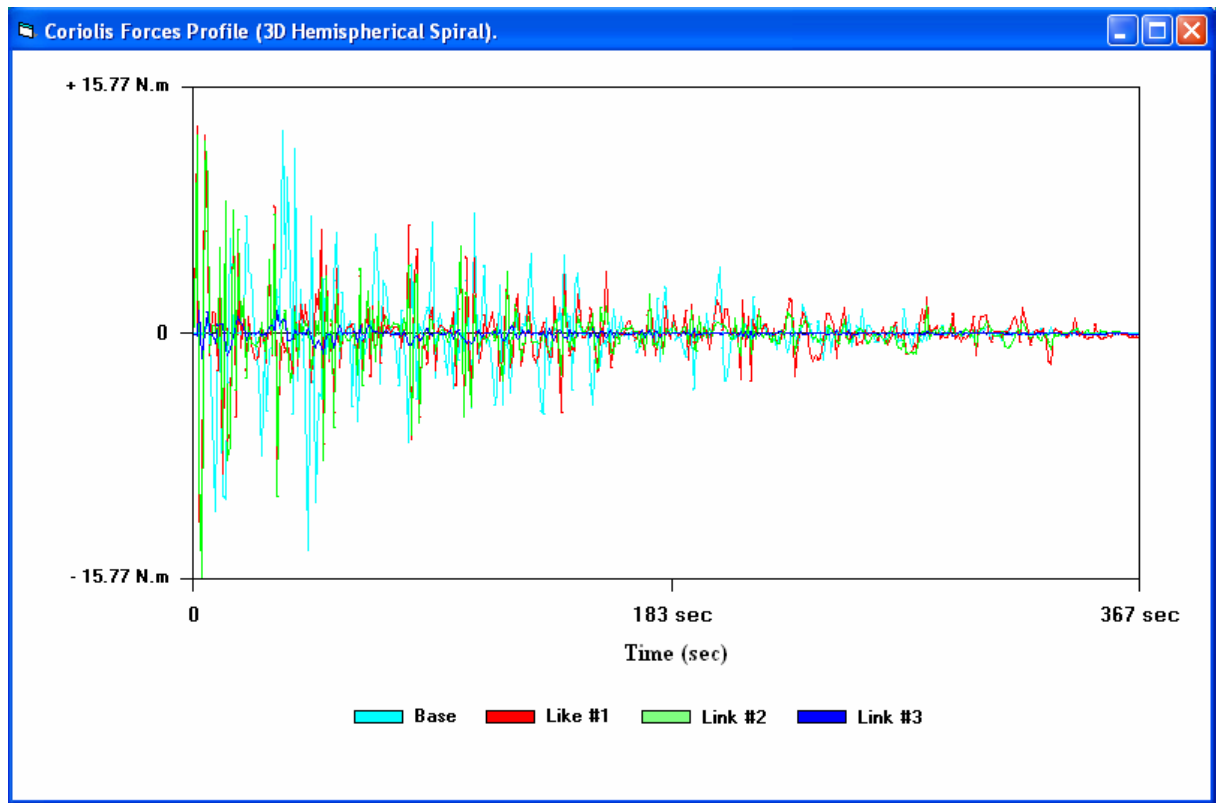
Figure(5-11) centrifugal & coriolis forces profile [3D Conical Spiral Path].



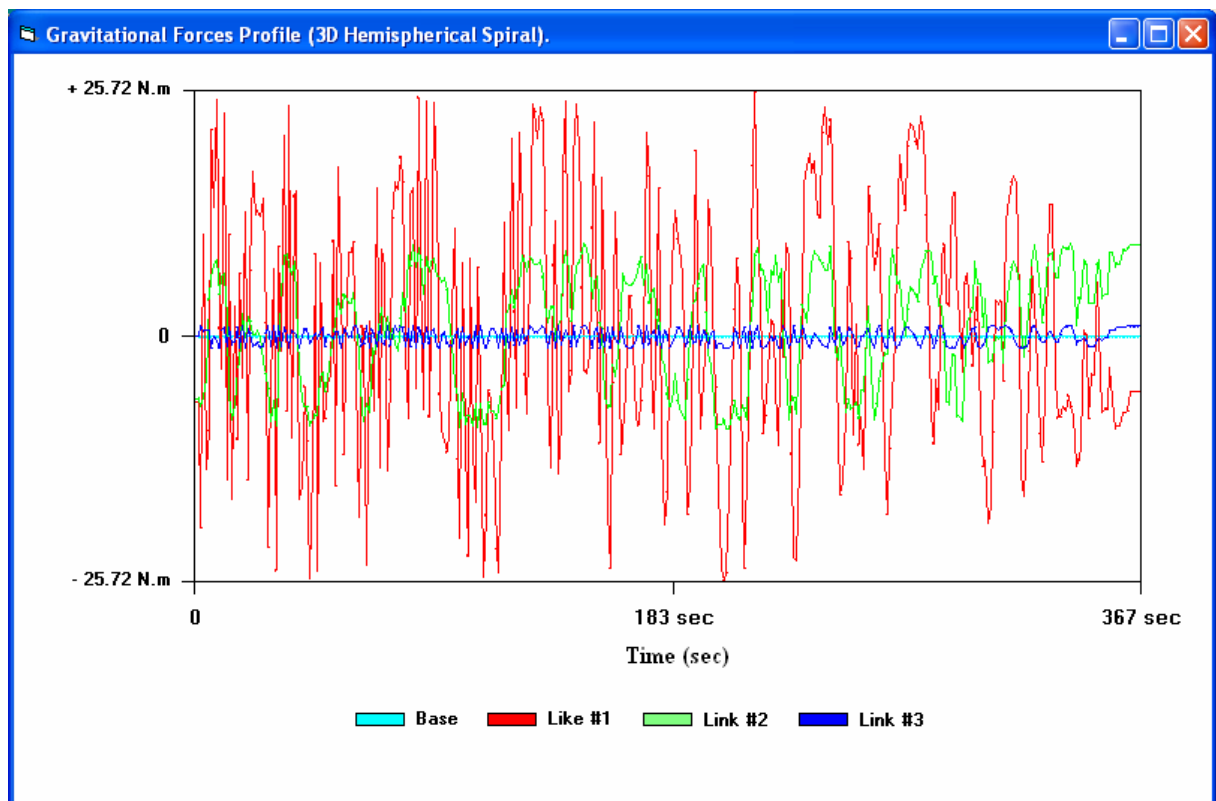
Figure(5-12) gravitational forces profile [3D Conical Spiral Path].



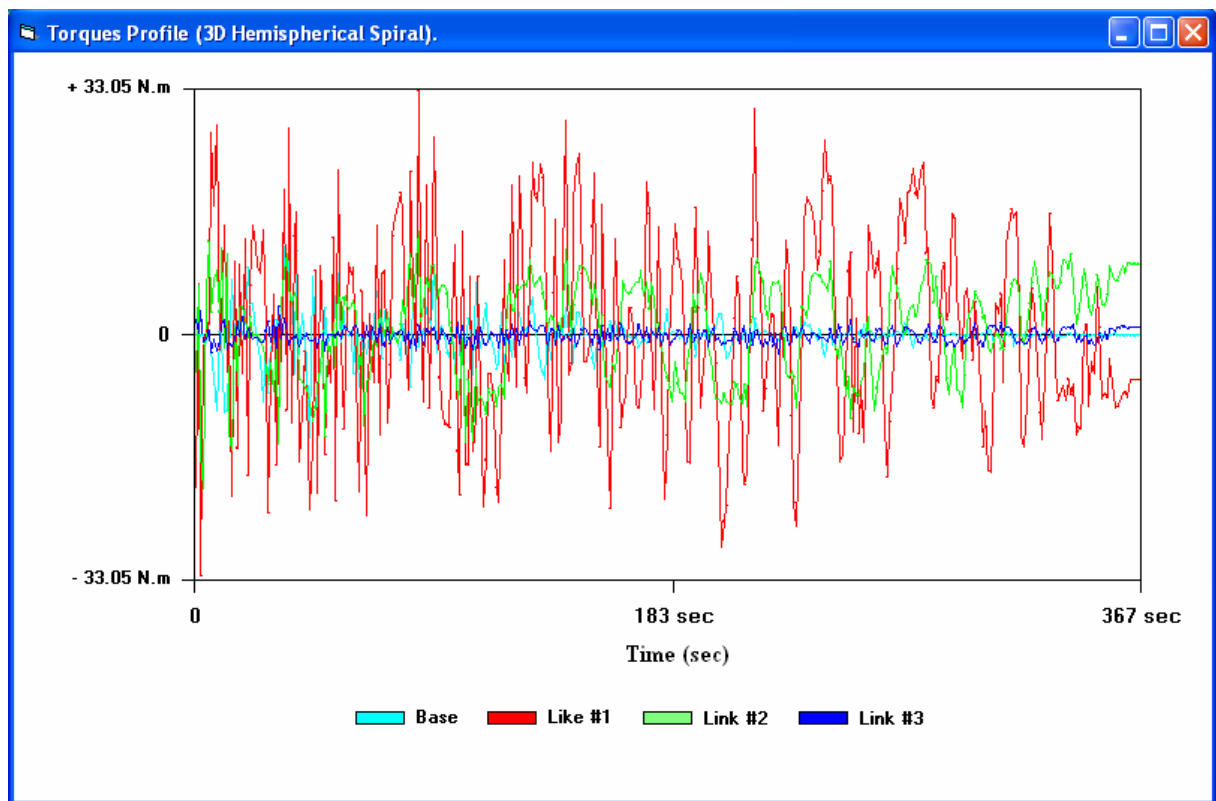
Figure(5-13) torques profile [3D Conical Spiral Path].



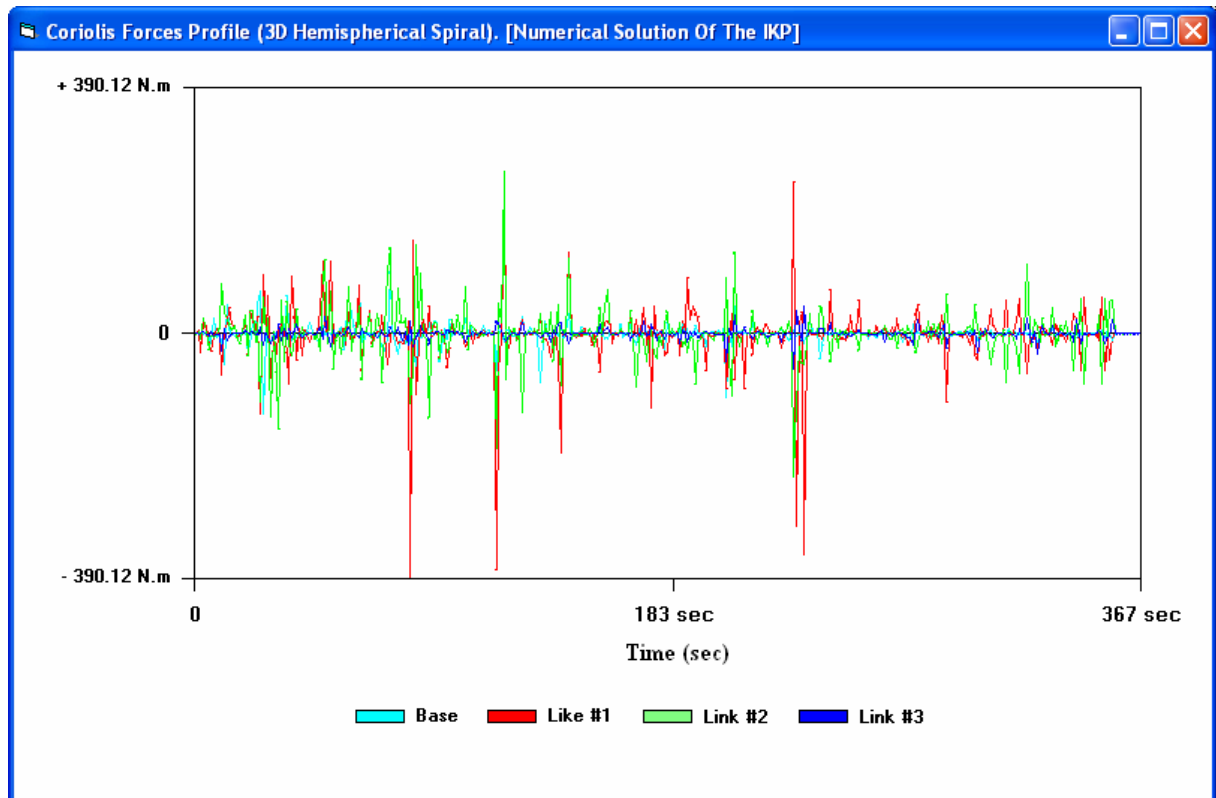
Figure(5-14) centrifugal & coriolis forces profile [Hemispherical Spiral Path].



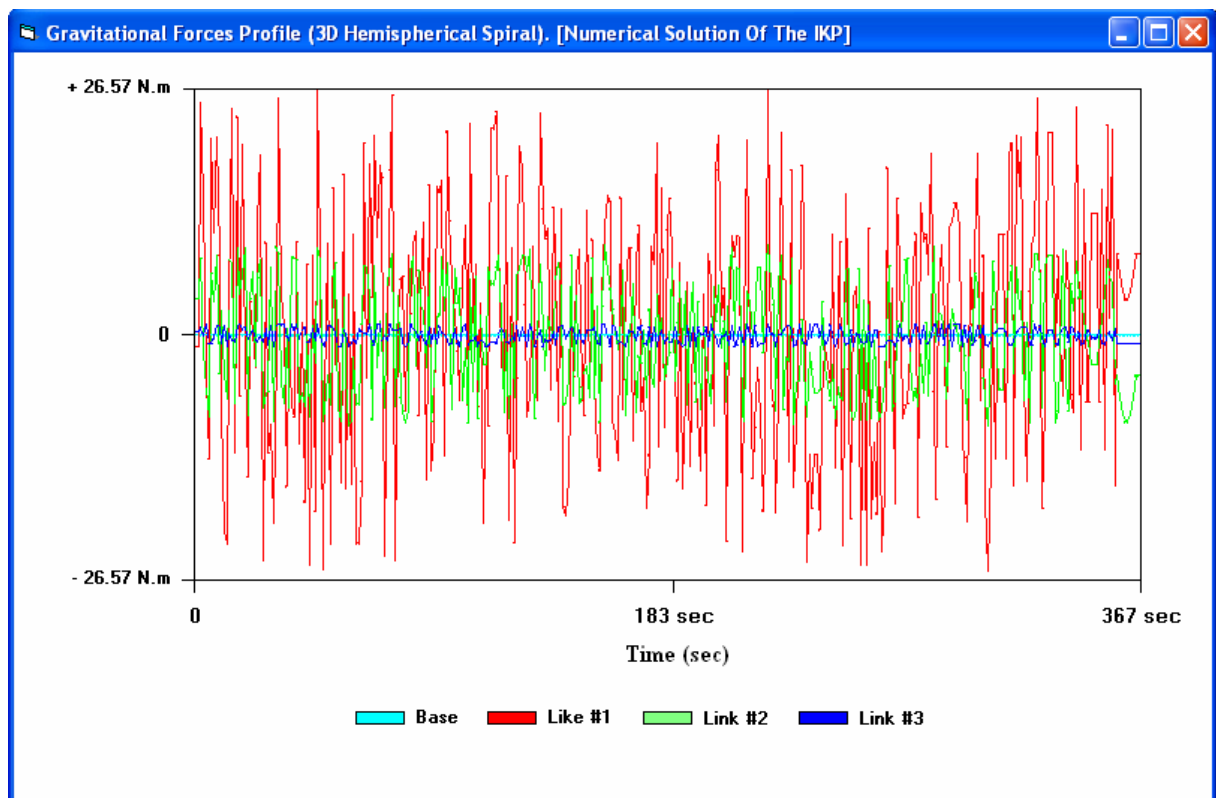
Figure(5-15) gravitational forces profile [Hemispherical Spiral Path].



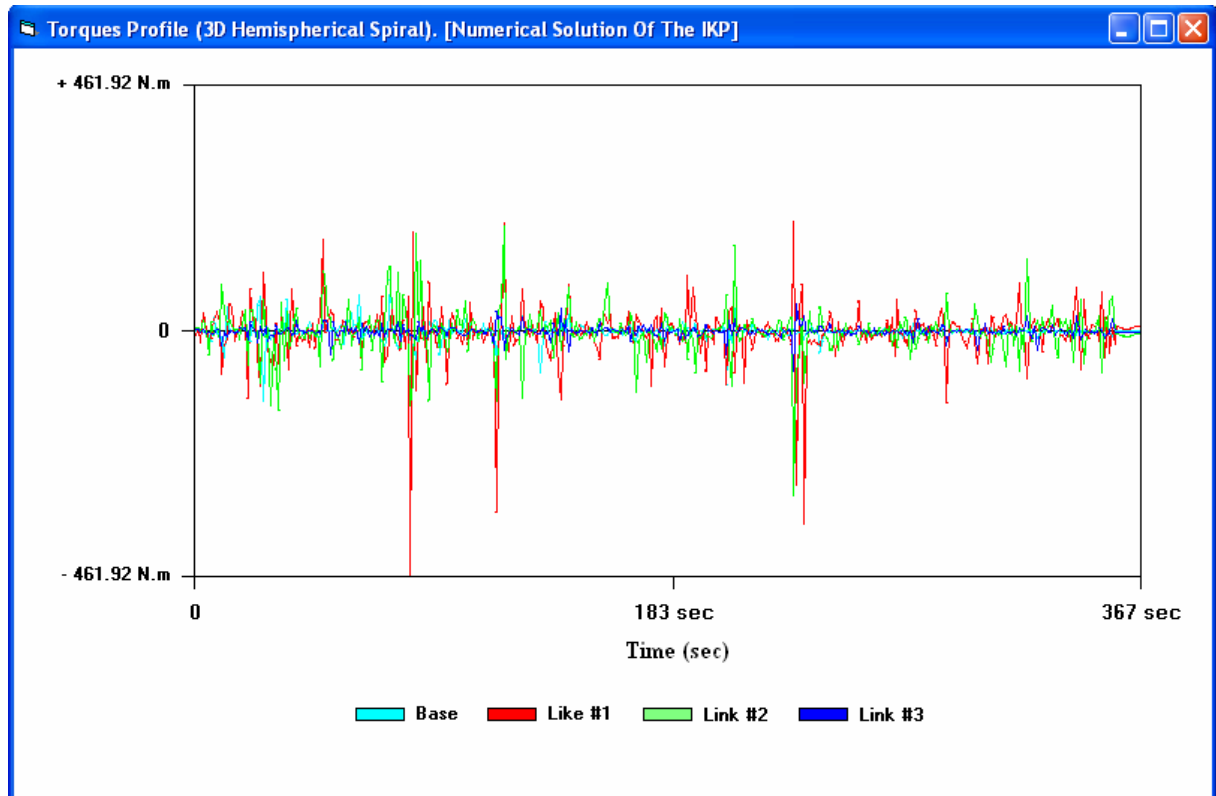
Figure(5-16) torques profile [Hemispherical Spiral Path].



Figure(5-14a) coriolis forces profile [Hemispherical Spiral Path], numerical solution.



Figure(5-15a) gravitational forces profile [Hemispherical Spiral Path], numerical solution.



Figure(5-16a) torques profile [Hemispherical Spiral Path], numerical solution.

5.3. Visual Servoing

To drive a robot depending on visual feedback data, an object recognition algorithm must be used. Many methods can be used to accomplish this task. However, in this work two types of algorithms were used. These two types are the template matching algorithm and the Hough transform algorithm. When the object of interest can be described by a closed form equation (like a circle, an ellipse, etc.), Hough transform is the best choice to be adopted because one of the most important features of Hough transform is its efficiency even if the signal-to-noise ratio of the captured image was relatively low. The second method, that is the template matching method, is more general for the detection of complex shaped objects but it is not efficient for noisy images. This part of the program is responsible of driving the robotic arm through the measurement of the visual feedback signal. This measurement is accomplished by using a set

of digital image processing algorithms. These algorithms are continuously applied to each frame received from the visual sensor (the web camera). The first subroutine applied was `analyze_image()` which returns a new version of the image which is an array in which each object, including the background, is stored in a separate layer. For each layer the edge detector subroutine named `detect_edges()` was applied to reduce the number of points of interest for further processing. The angle of rotation of the object with respect to the camera is calculated by the subroutine `find_orientation()`. The angle evaluation is accomplished using the Best Eigenvector Line Fitting method described in appendix B.

At this stage the image is transformed into layers each of which contains an outline (the edges) of the object it contains. The algorithms applied until now on the image are pre-processing algorithms. The aim of applying these algorithms is to prepare the image for the next phase of processing, that is, the object recognition algorithms.

In this work two objects were used for testing the program, a circular object and an arbitrarily shaped object. The subroutine used to detect the circle is `detect_circle()`. Figure (5-17) shows the image captured by the camera just before the entrance of the circular object into the view of the camera, so, no action was taken since the program did not detect any circular object, hence, no feedback signal is generated.

Figure (5-18) shows the first entrance of the circular object into the scene. The first arrival of the object caused the `detect_circle()` to respond and to indicate that the object of interest was present in the scene. Accordingly the subroutine responded and measured the centre of the circle. The error feedback signal was in the form of the line segment between the centre of the image and the measured centre of the circle. Then the robot should start to track the detected object along this line segment. The robot controller must be provided with the data required to drive the robot along this line. This data is obtained by

calling the subroutine IKP() which produces the inverse kinematics solution in the form of a position profile. From the position profile, both the velocity and the acceleration profiles are calculated. The subroutine find_torque() was then applied to determine the required torque at each joint to drive the robot according to the specified line segment. Figure (5-19) shows the 7th sample frame and the path that was generated during the motion of the object. This procedure was repeated to drive the robotic arm as long as long as the detect_circle() subroutine was detecting that the circle was moving. Figure (5-20) shows the final position of the circle. This was the last image frame captured by the web camera. At this instant no feedback signal was generated because detect_circle() detected no movement of the circle and hence the robotic arm stops. When the circular object stops, detect_circle() continues to report that the circle was existed but at the same location every measurement time, hence, no error signal was generated.

detect_circle() subroutine was one of many other subroutines that were written in *Visual Basic 6.0* programming language. These subroutines were used to simulate the kinematics and the dynamics of the robotic arm. detect_circle() together with IKP() generate a position profile for each line segment on the path of the moving object which means a torque at each joint is calculated in the form of profiles for each segment on the path.

The simulation program was general in tracking any trajectory within the available working space, and also it has the ability to detect any desired object using its image processing subroutines.

The second object used for testing was an arbitrarily shaped object, so, the template matching technique was used to detect this object. The subroutine that performs the template matching technique for detection was detect_object(). In figure (5-21) the frame just before the entrance of the object into the view of the camera is shown. No error signal was generated and the robotic arm continues to take its home position. Figure (5-22) shows the first arrival of the

object and, accordingly, the first error signal was measured in the form of the line segment shown. In figure (5-23), the 7th frame captured by the web camera is shown. This figure also shows the track of the object up to the instant of capturing this frame. The robot continued to follow the object because while the object was moving, `detect_object()` was continuously generating error signals that were used to drive the robotic arm. The program terminates when the object stopped in the location shown in figure (5-24) because `detect_object()` produced zero error signal which indicated that the object had stopped. The images captured by the web camera were continuously processed and continuously indicate that the object is present in the scene but the error signals are generated only when the object changes its location in a frame with respect to the previous frame. When the object stops no error signal can be generated and hence no change in the configuration of the robot is required. Accordingly, the torque at each joint is zero which means that the robot continues to take the last required configuration.

Position profile, velocity profile and acceleration profile for the path generated as a result of the circular object tracking are shown in figures (5-25), (5-26) and (5-27), respectively. The corresponding coriolis forces profile, gravitational forces profile and the torque profile are shown in figures (5-28), (5-29) and (5-30), respectively. Similar profiles were obtained for the tracking process of the arbitrary-shaped object.



Figure (5-17) the frame captured just before the circular object entry.



Figure (5-18) first object arrival detection

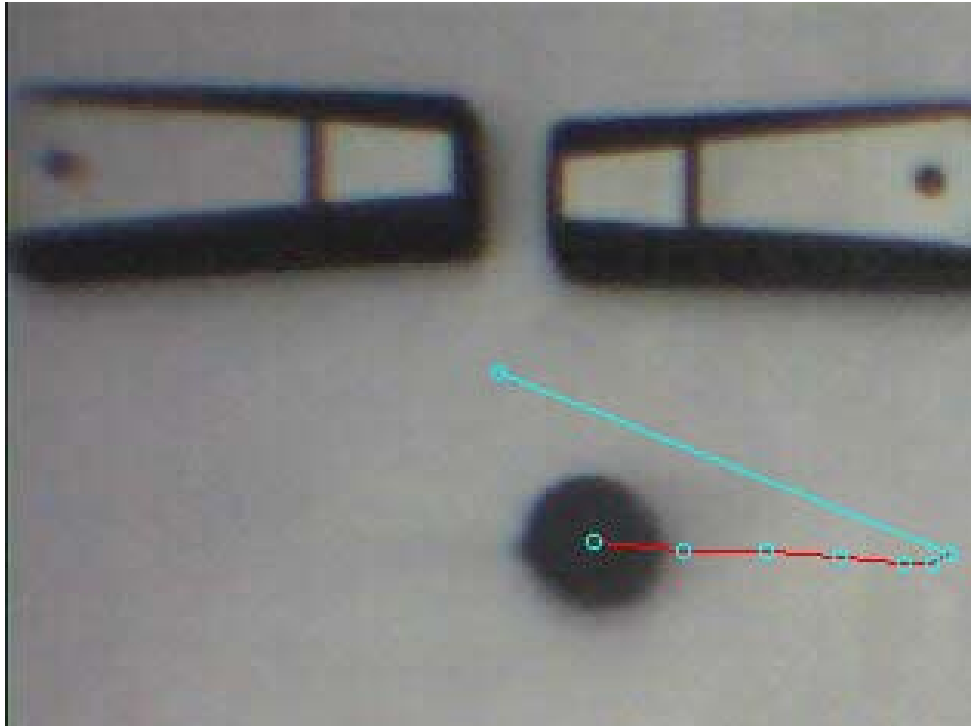


Figure (5-19) the detection of the path at the 7th frame.

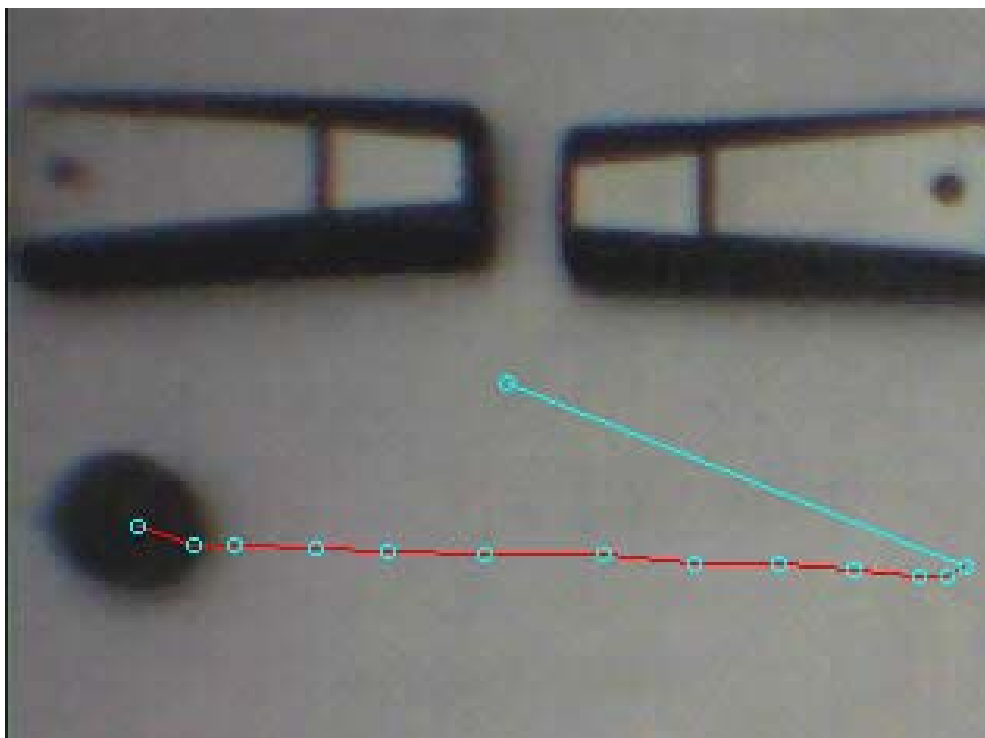


Figure (5-20) the last frame captured at which the circular object stopped.

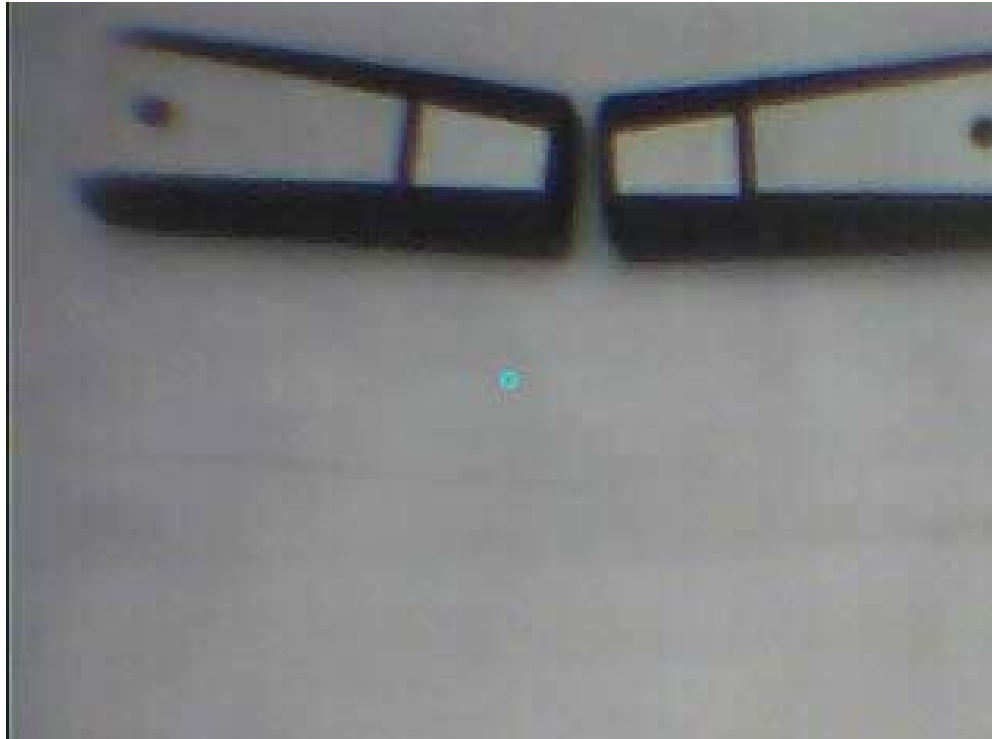


Figure (5-21) the frame captured just before the object entry.

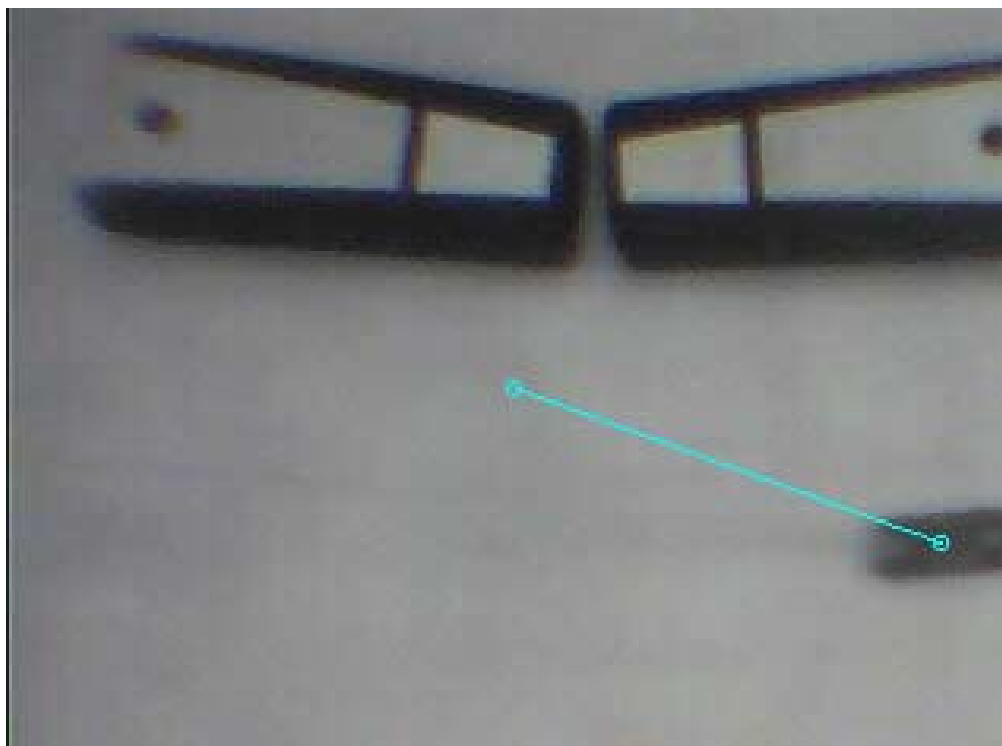


Figure (5-22) first arrival detection of the object.

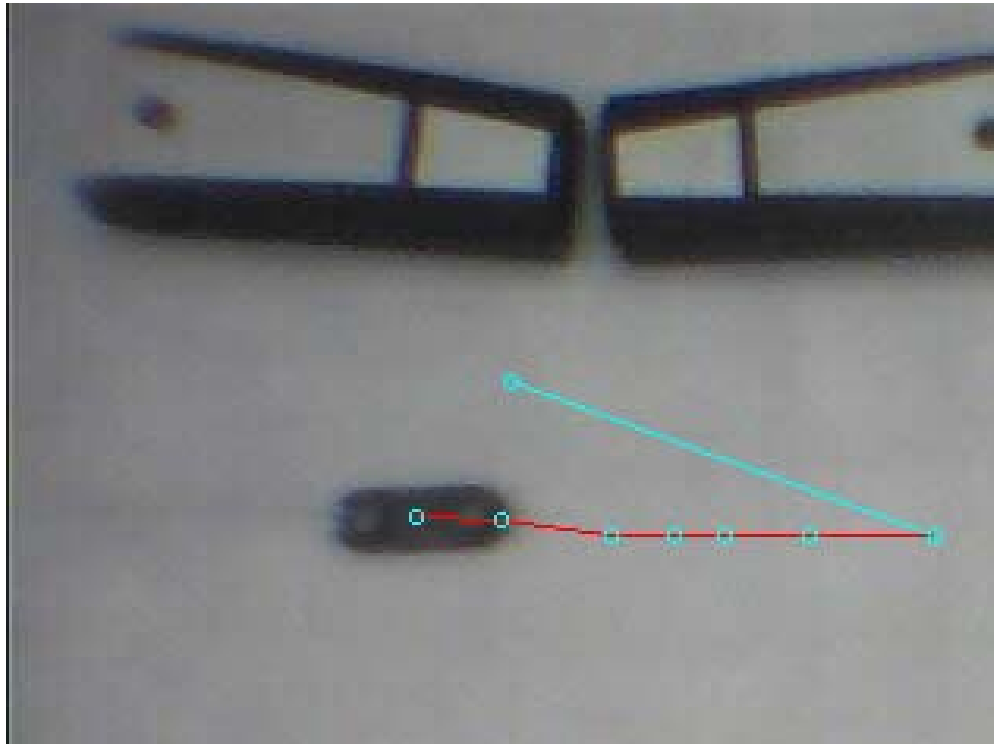


Figure (5-23) the detection of the path at the 7th frame.

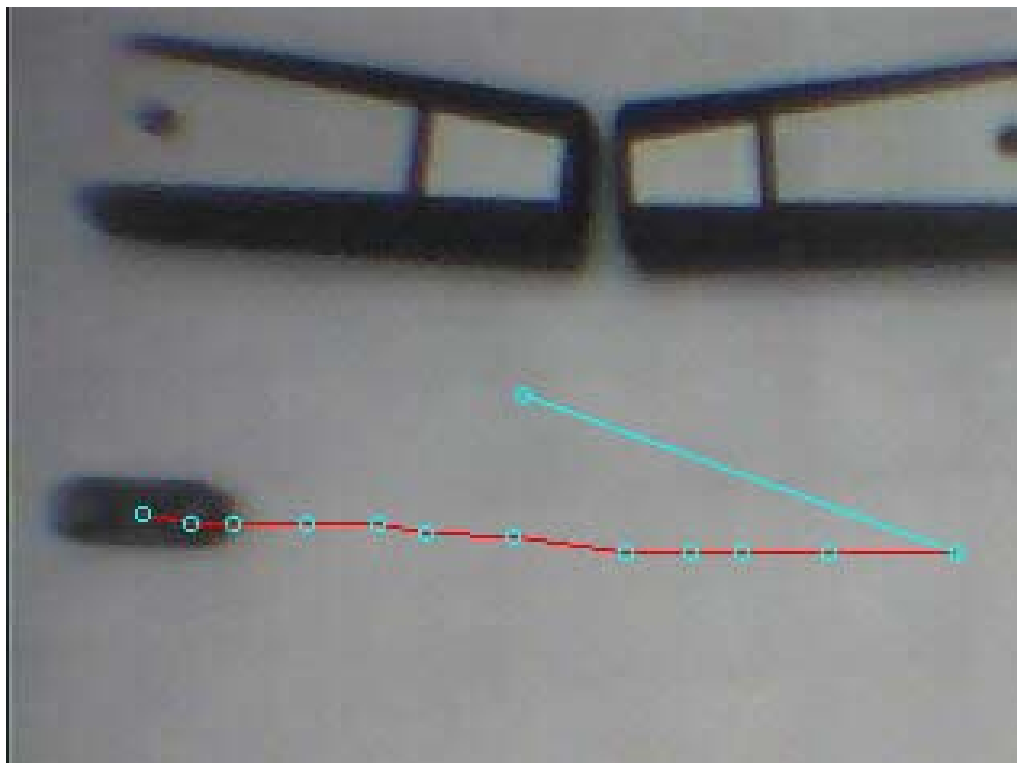


Figure (5-24) the last frame captured at which the object stopped.

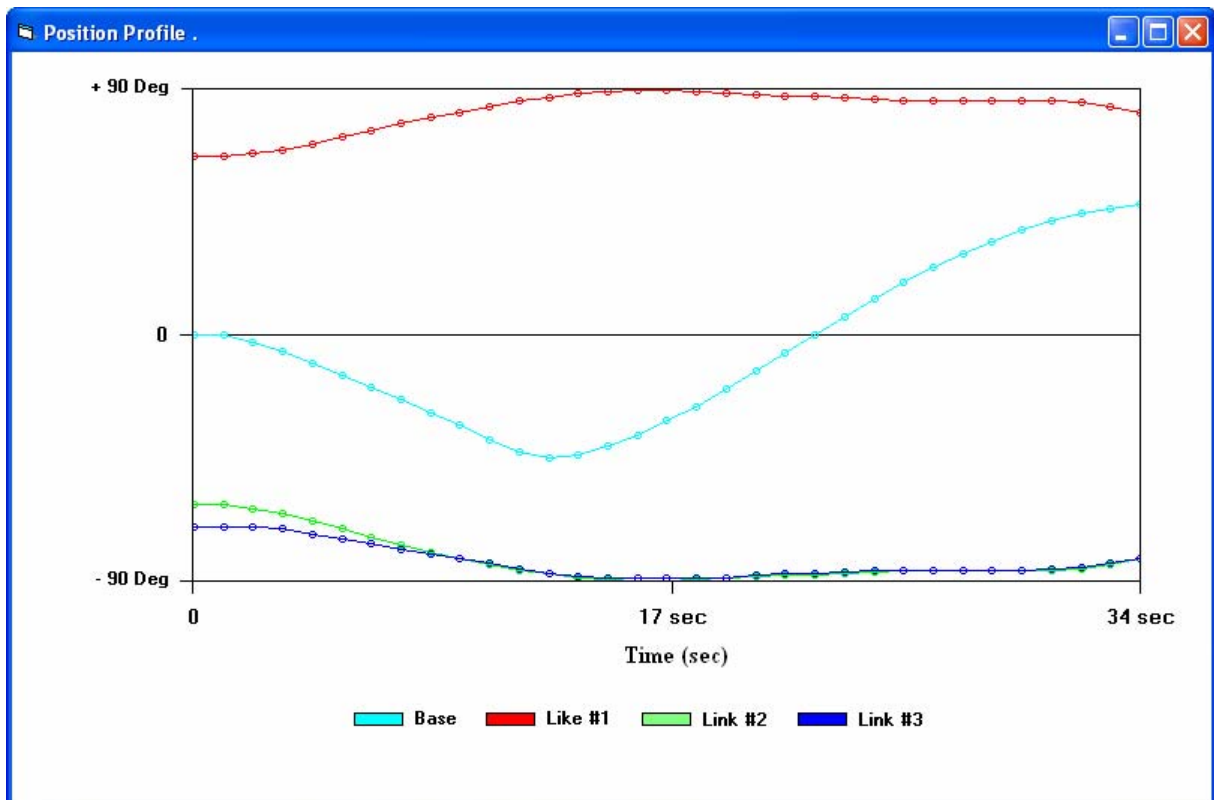


Figure (5-25) position profile for circular object tracking

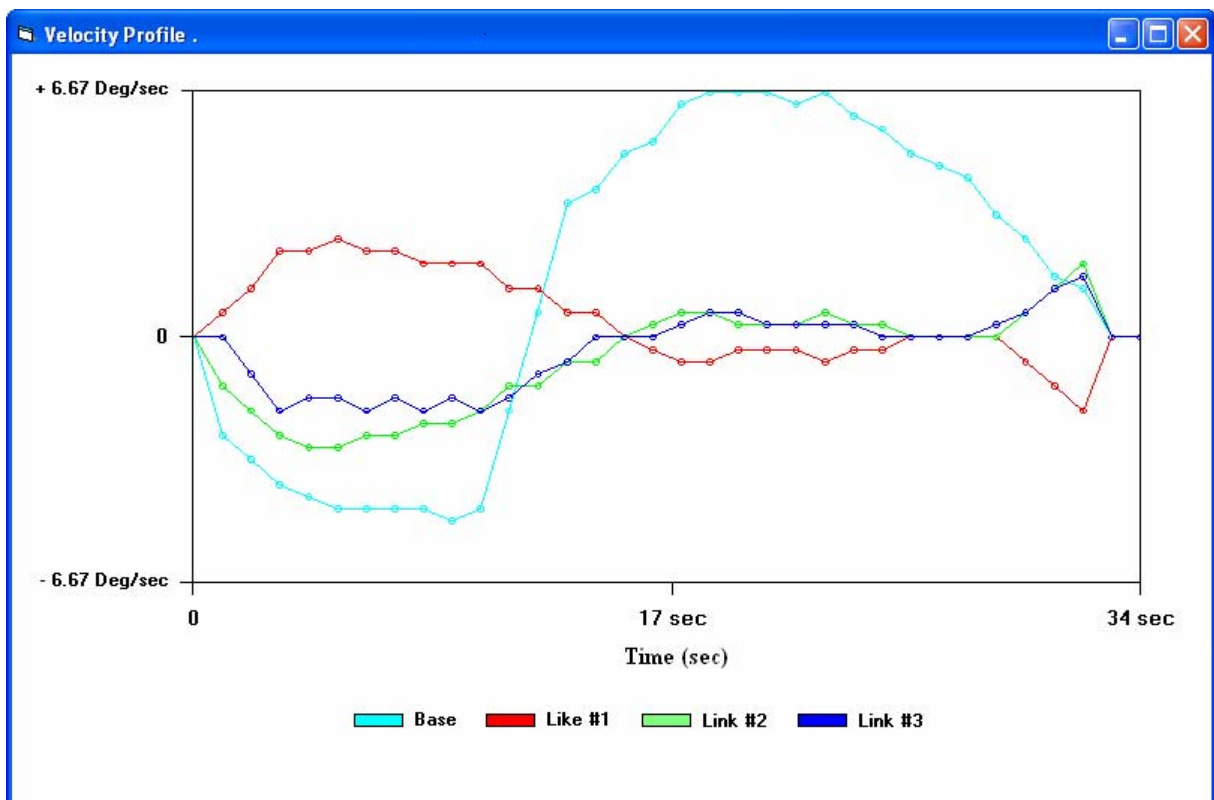


Figure (5-26) velocity profile for circular object tracking

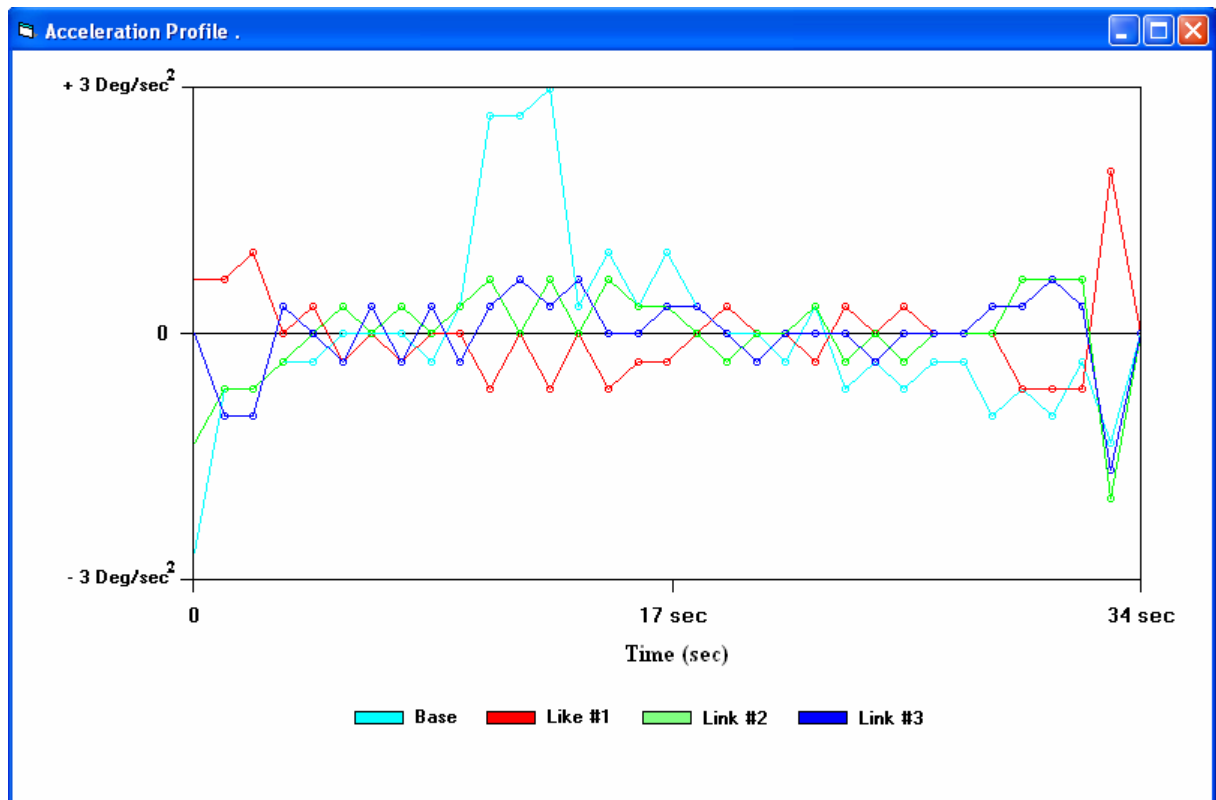


Figure (5-27) acceleration profile for circular object tracking

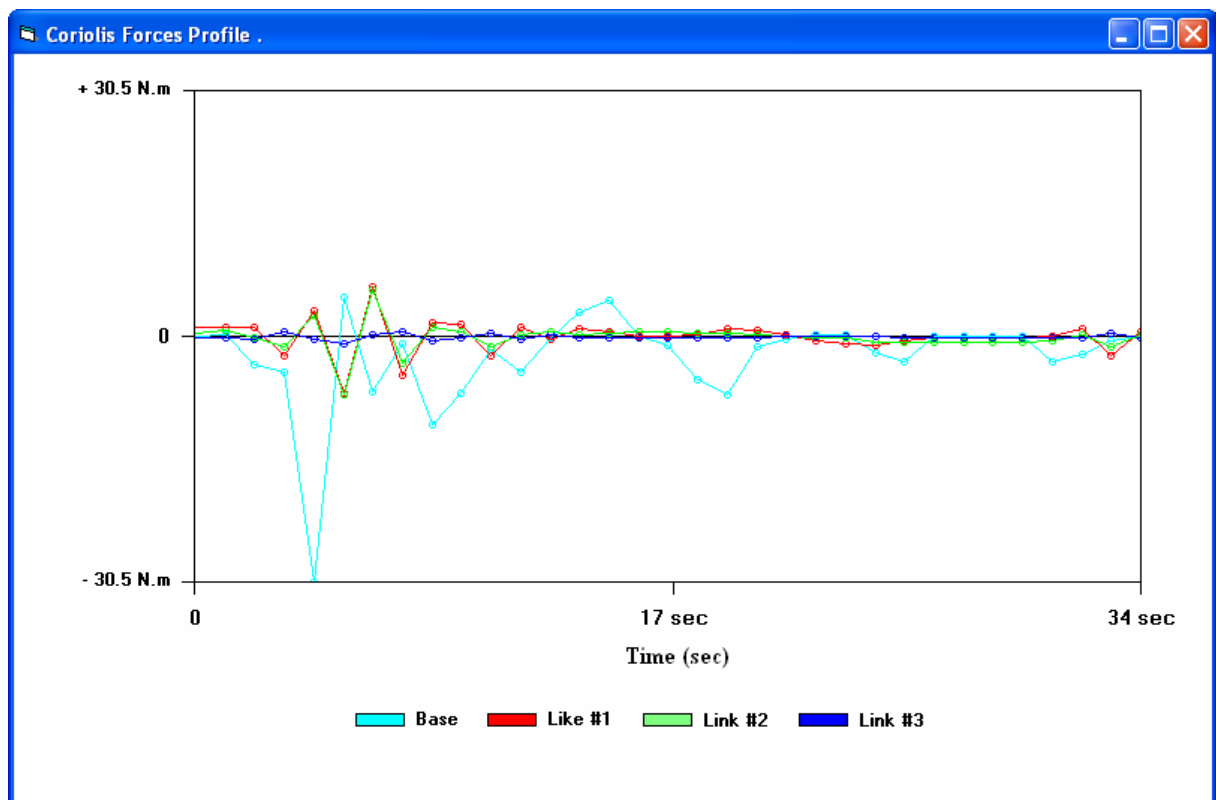


Figure (5-28) coriolis forces profile for circular object tracking

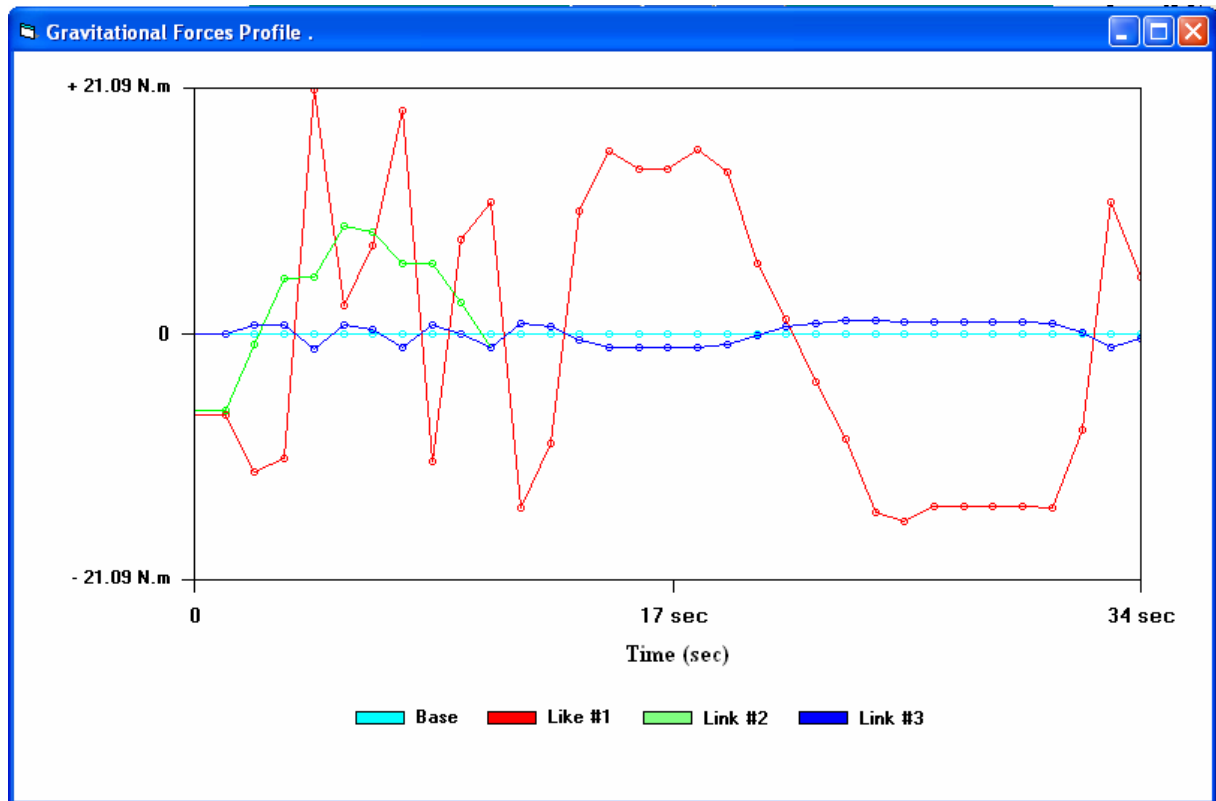


Figure (5-29) gravitational forces profile for circular object tracking

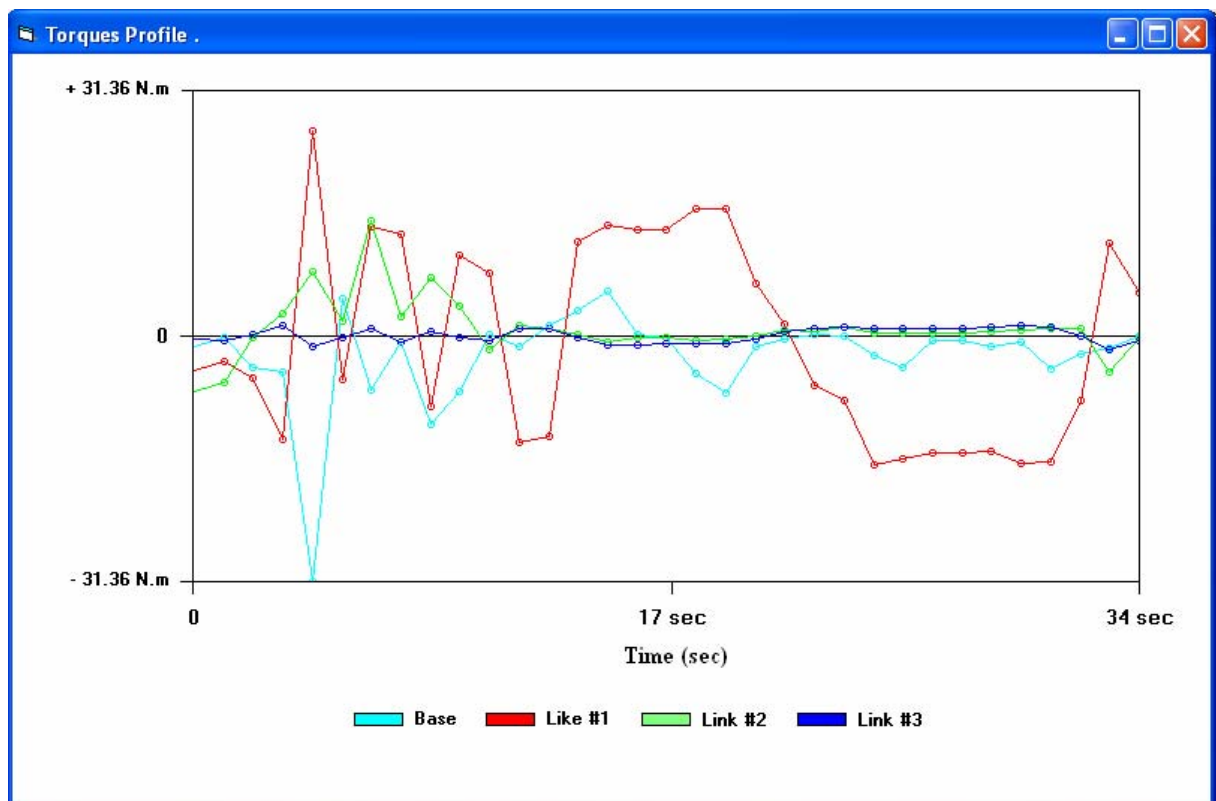


Figure (5-30) torques profile for circular object tracking

5.4. The PID Controller

In the present work, a PID controller was used for servoing the robotic arm which was built as a subroutine in the simulation program. PID() is the subroutine that performs as the PID controller. It calculates the required electrical signal to drive each actuator on the robotic arm for a given error feedback signal which is derived from the position profile for a given trajectory.

The model which was discussed in chapter four and shown in figure (5-31) was assumed to represent the dynamics of each actuator on the robotic arm.

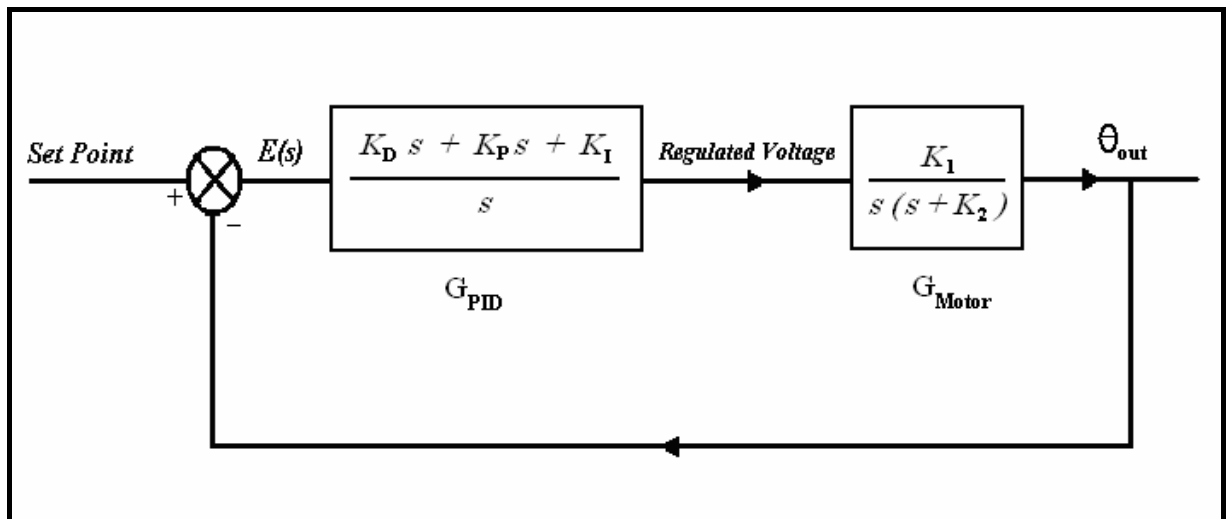


Figure (5-31) DC servomotor control system.

To evaluate the constants of the PID controller, a *Visual Basic* program was designed for this purpose. The scroll bars were used to easily adjust the values of the PID controller and the values of each constant was displayed on the corresponding text box. A step input was entered to the system and the corresponding output was evaluated.

A unit step input of one degree (or one radian ... etc) was applied to the system and the response (angular position) of the DC servomotor without a PID controller is shown in figure (5-32). As expected, for a first order system, zero steady state error is obtained for a unit step input. Although such a zero steady

state error response is highly desired for a robotic system, the overall response is slow. The system reached its steady state after 20 seconds, which is a very long period of time causing the performance of the robotic arm to be undesirably slow.

To improve the response of this system, the PID controller with $K_p = 25$, $K_i = 0.5$ and $K_d = 0$ was applied to the system. With its derivative part inactive, the PID controller acts as a PI controller. The effect of the integral part of the controller is to improve the settling time of the system. Figure (5-33) shows the effect of the PID controller with the above values. It is clear that the PID controller shortened the settling time to become 7.5 seconds in stead of the original 20 seconds for the system without a PID controller. However, the improvement of the steady state response was on the account of the transient response because high overshoot (about 35% percentage overshoot) and oscillatory response resulted. This is not acceptable for accurate path tracking performance. To improve the transient response the derivative part of the PID controller was activated by making $K_d = 5$ as shown in figure (5-34). Although the percentage overshoot was reduced to about 15%, the settling time still long (3.5 seconds). Figure (5-35) shows that by making $K_d = 10$, the settling time became 2.5 seconds but the transient response still suffering from a 5% percentage overshoot.

However, the desired response can be achieved by continuously varying the constants of the PID controller. In figure (5-36) a good response of the system is shown for the given values of the PID controller's constants. The rise time was 0.15 seconds and the maximum overshoot was (0.6%).

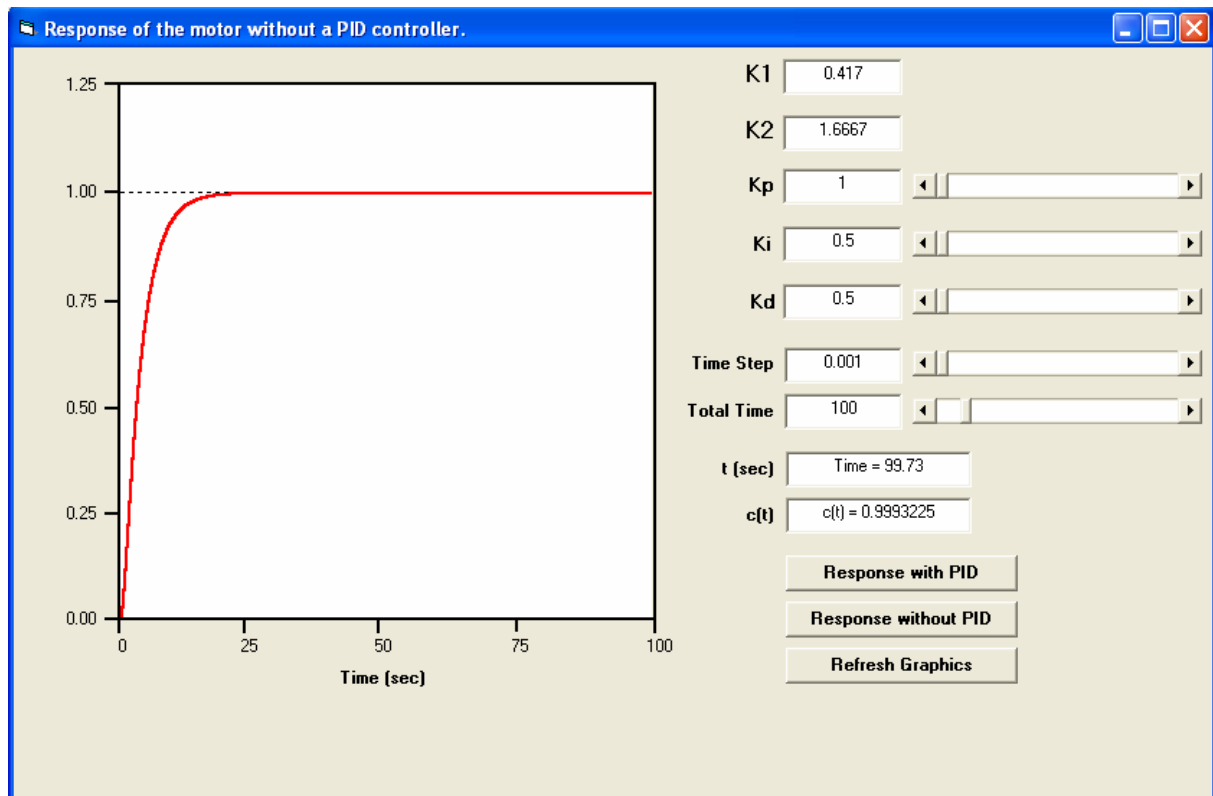


Figure (5-32) system response without a PID controller.

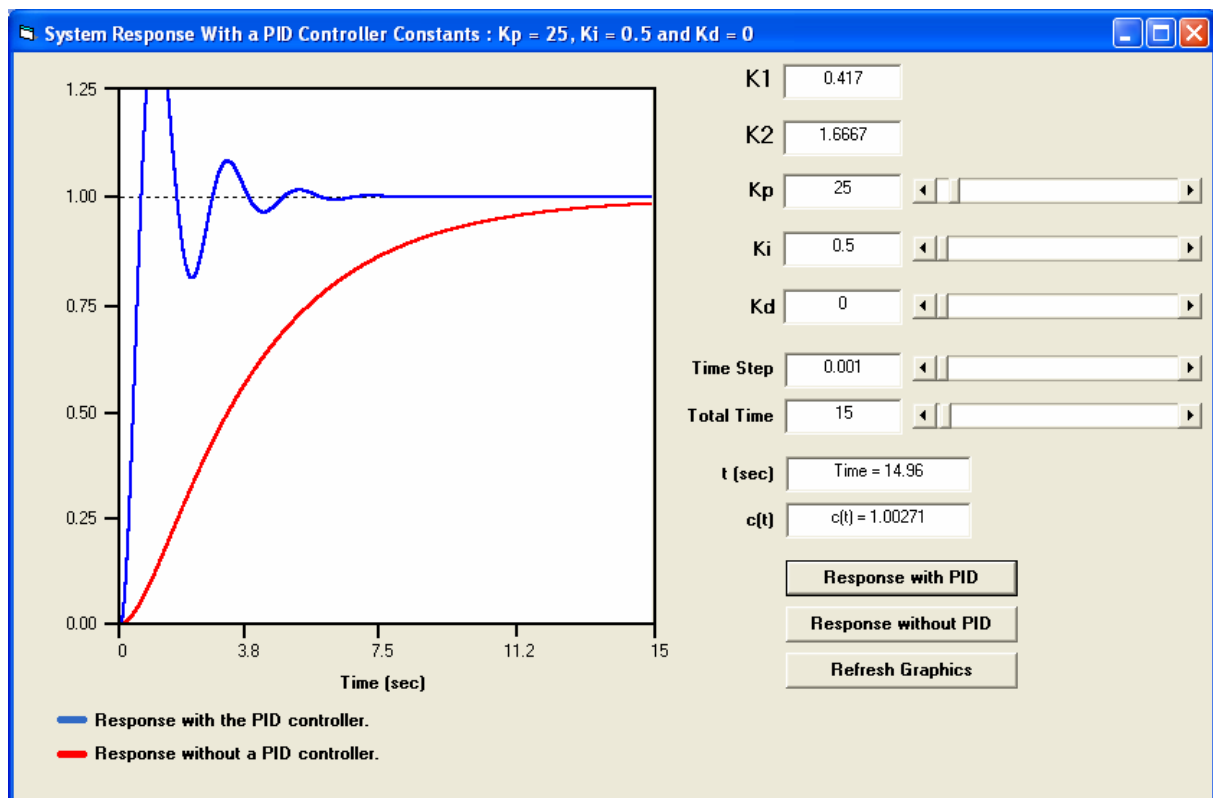


Figure (5-33) PID controller : $K_p = 25$, $K_i = 0.5$ and $K_d = 0$.

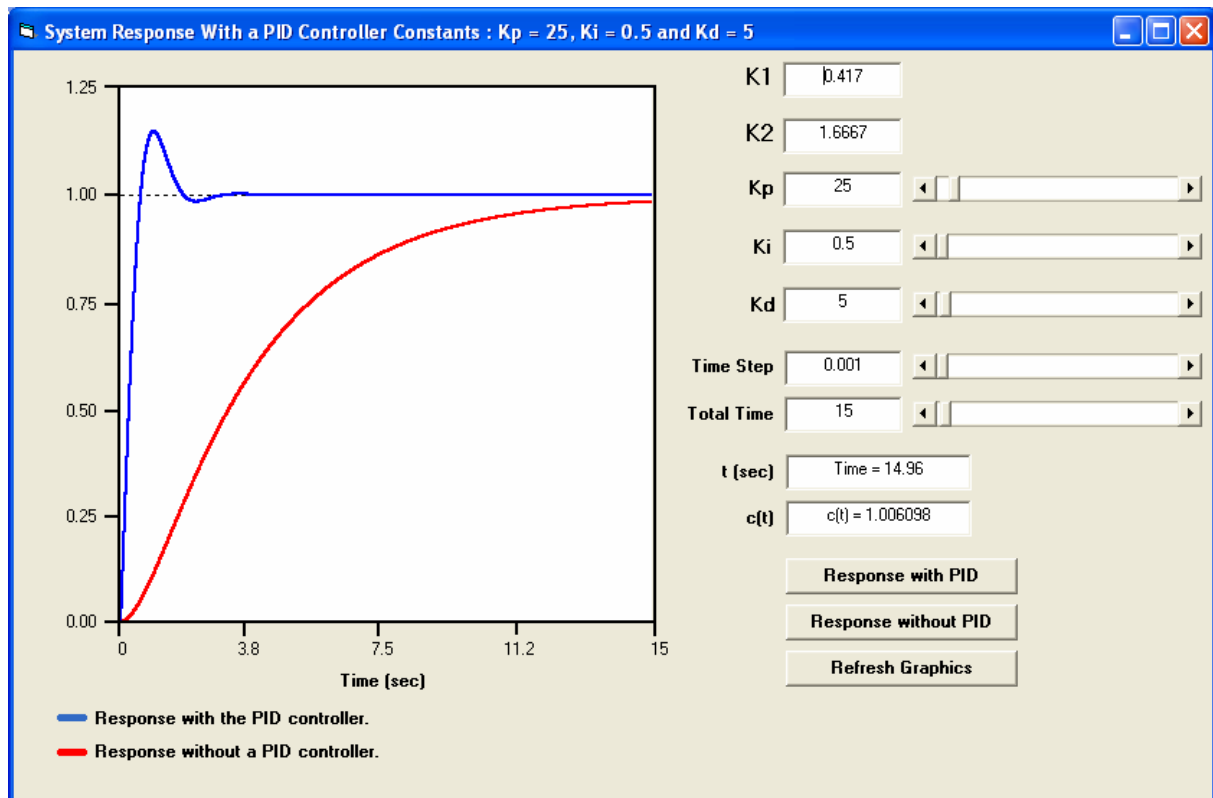


Figure (5-34) PID controller : $K_p = 25$, $K_i = 0.5$ and $K_d = 5$.

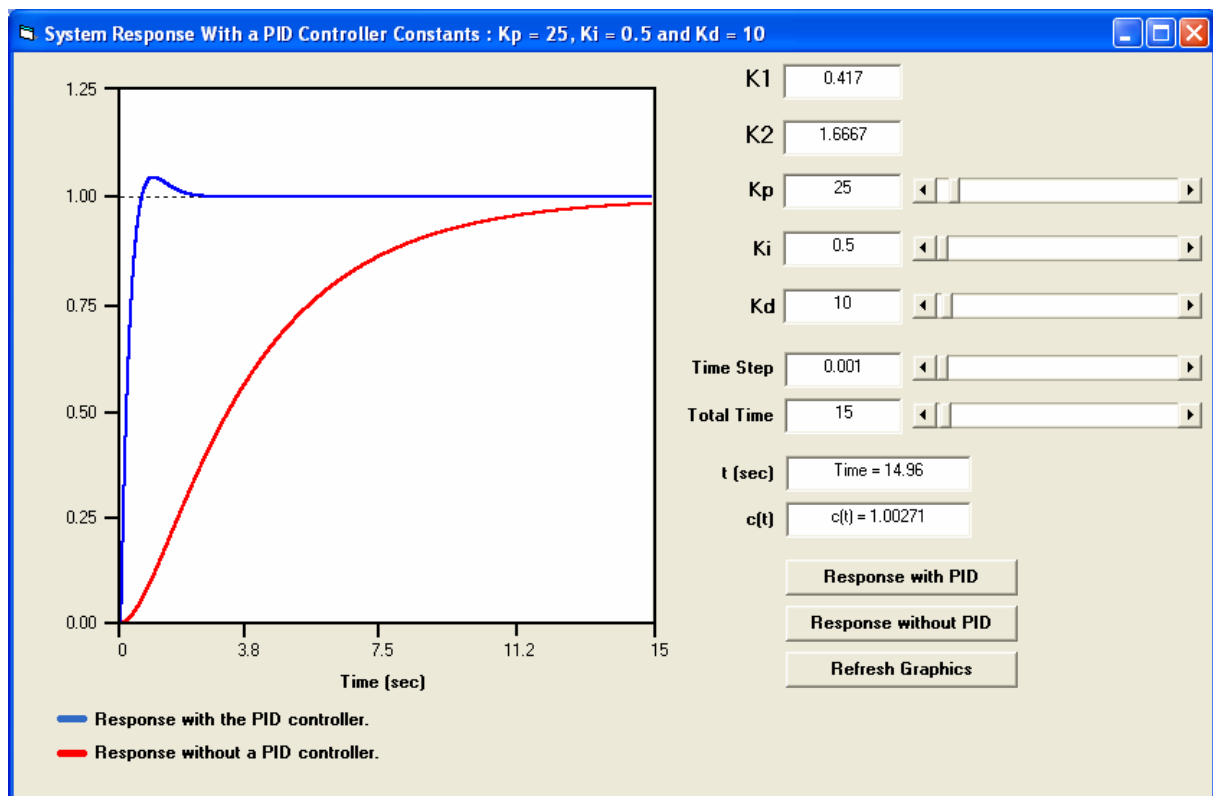


Figure (5-35) PID controller : $K_p = 25$, $K_i = 0.55$ and $K_d = 10$.

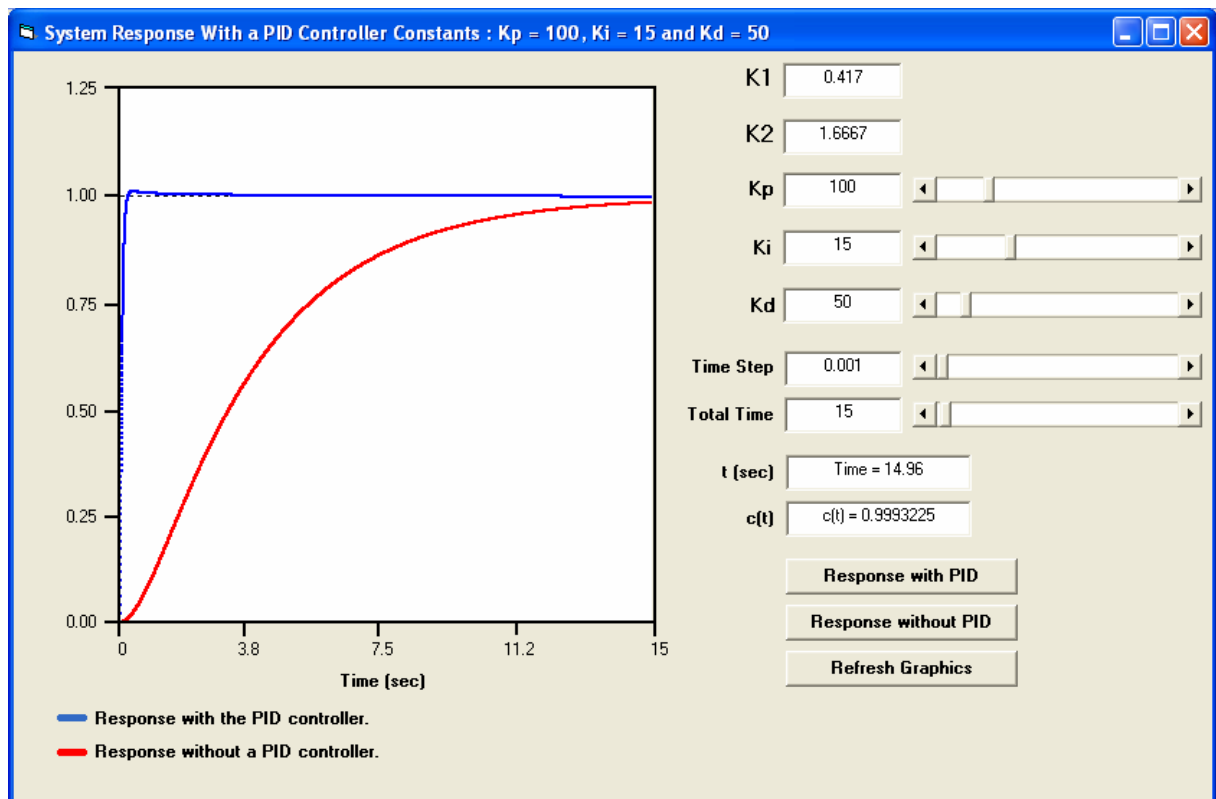


Figure (5-36) PID controller : $K_p = 100$, $K_i = 15$ and $K_d = 50$.

Chapter Six

Conclusions and Suggestions for Future Work

Conclusions and Suggestions for Future Work

6.1. Conclusions

An eye-in-hand vision-based control system was built to drive a four-link robotic manipulator according to the movement of an object of interest that moves in front of the web camera which was held on the end effector of the robotic arm. Also, a dynamic model that describes the dynamics of this type of robotic systems was built. Referring to the previous discussions presented in chapter five, the following main conclusions can be drawn:

- 1- Vision-based control systems give the robot extra capabilities to handle the objects within the working space of the robot because they enable the robotic system (to see) the environment in which it is placed in.
- 2- The location of the object of interest is determined by using a set of digital image processing algorithms. These algorithms are usually relatively slow which makes the overall performance of the robotic system to become slow accordingly.
- 3- To evaluate the dynamic forces acting on the robotic manipulator, a dynamic model must be derived. The complexity of the dynamic model is highly related to the assumptions made for building the model and the degree of the geometric complexity of the robotic arm to be manufactured. With minimum assumptions made, the model will approach the behavior of the real system.

The dynamic model for a robotic arm is a set of non-linear and highly coupled ordinary differential equations. An n -link robotic arm is described by a set of (n) differential equations.

4- The inverse kinematics of the robotic arm must be fully determined in advance, that is, for each work space position there must be a corresponding robotic arm joint space vector. For a trajectory following purposes, a position profile in joint space must be determined for each desired path. When this position profile is fed to the dynamic model of the robotic arm, the torque required by each actuator at any instant during the movement of the robotic arm is evaluated. This means that the knowledge of the inverse kinematics is the key to drive the robotic arm along the desired trajectory.

5- For accurate path following motion, a PID controller must be used to drive the robotic manipulator. With its integral part, the PID controller eliminates the steady-state errors in the angular positions of the servomotors of the robotic arm. By its derivative part, the PID controller improves the transient response characteristics, such as, minimum percentage over shoot, minimum rise time, etc. The desired overall response is achieved by determining the suitable value for the constants of the PID controller through the use of the standard tuning procedures.

6.2. Suggestions for Future Work

As related to the present work, the following suggestions for future work developments forwards:

1- Implementing this work experimentally by using an interfacing hardware between the computer and the robotic arm.

2- Designing and implementation of a vision-based control system to drive flexible robots. An eye-in-hand visual servoing system *cannot* be used with such type of robots because of the vibrating behavior of the robot, specially, at the

end-effector location. A stand-alone visual servoing system is the best choice for a flexible robot vision-based controller.

References

References

- [1] Frank L. Lewis, Darren M. Dawson and Chaouki T. Abdallah, "Robot Manipulator Control, Theory and Practice", Second Edition, Marcel Dekker Press, Inc. 2004.
- [2] Danica Kragic and Henrik L. Christensen, "Survey on Visual Servoing for Manipulation", Centre of Autonomous Systems, Numerical Analysis and Computer Science, Sweden, 2002.
- [3] Kelly R., Santibanez V. and Loria A., "Control of Robotic Manipulators in Joint Space", First Edition, Springer-Verlag London Press Limited, 2005.
- [4] Bernard Espiau, Francois Chaumette and Patrick Rives, "A New Approach to Visual Servoing", IEEE Journal on Robotics and Automation. Vol. 8, No. 3, June 1992.
- [5] Friedrich Lange, Patrick Wunsch and Gerhard Hirzinger, "Predictive Vision-Based Control of High Speed Industrial Robot", Published by the Institute of Robotics and System Dynamics, Wessling, Germany, 1998.
- [6] Rahul Singh, Richard M. Voyles, Donald Littau and Nikolaos P. Papanikolopoulos, "Grasping Real Objects Using Virtual Images", Proceedings of The 37th IEEE Conference on Decision and Control. Tampa, Florida, USA, Dec 1998.
- [7] Philippe Martinet and Jean Gallice, "Position-Based Visual Servoing Using a Non-linear Approach", Proceedings of IEEE International Conference on Intelligent Robots and Systems, 1999.

- [8] Kragic D. and Castleman H., "Using Redundant and Coarsely Calibrated Vision System For 3D Grasping", Proceedings of The Computational Intelligence For Modeling, Control and Automation, IOS Press, 1999.
- [9] Geraldo Selviera, Reginaldo J., Carvalho H., Marconi K. Madrid, Samuel S. Bueno and Patrick Rives, "Lateral Control of an Aerial Unmanned Robot Using Visual Servoing Techniques", Second Workshop on Robot Motion and Control, Oct 2001.
- [10] Mana Saedan and Ang H. Jr., "3D Vision-Based Control on an Industrial Robot", Proceedings of The IASTED International Conference on Robotics and Applications, Florida, USA, Nov. 2001.
- [11] Christoph Theis, Ioannis Iossifidis and Axel Steinhage, "Image Processing Methods for Interactive Robot Control", Institute of Neuroinformatics, Bochum, Germany, 2002.
- [12] Tom Drummond, "Real-Time Visual Tracking of Complex Structures", IEEE Journal on Pattern Analysis and Machine Intelligence (PAMI), Vol. 24, No. 7, July 2002.
- [13] Robert Mahony, Peter Corke and Francois Chaumitte, "Choice of Image Features for Depth-Axis Control in Image-Based Visual Servo Control", IEEE International Conference on Intelligent Robots and Systems, 2002.

- [14] Hector H., Gonzales Banos, Cheng Yu Lee and Jean-Claude Latombe, "Real-Time Combinatorial Tracking of a Target Moving Unpredictably Among Obstacles", Published by CS Robotics Laboratory, Stanford University, Stanford, California, USA 2002.
- [15] Yan Fei Liu, Adam W. Hoover and Ian D. Walker, "A Timing Model For Vision-Based Control in Industrial Robotic Manipulators", IEEE Transactions on Robotics, Vol. 20, No. 5, Oct 2005.
- [16] Tomoyuki Shiozaki and Toshiyuki Murakami, "Posture Control of Mobile Robots By Hand-Eye System", IEEE Conference on Control Applications, Toronto, Canada, Aug 2005.
- [17] Aman Betal, Setler P., Warren Dixon and Darren M. Dawson, " Adaptive Position and Orientation Regulation for the Camera-in-Hand Configuration", Journal of Robotic Systems, Vol. 22, No. 9, 2005.
- [18] Ahmet Denker and Tugrul Adiguzel, "Vision-Based Robotic Interception in Industrial Manipulation Tasks", International Journal of Computational Intelligence, Vol. 3, No. 4, Feb 2006.
- [19] L. Peterson, Fletcher L., Zelinski A., Barnes N. and Arnell F., "Towards Safer Roads by Integration of Road Scene Monitoring and Vehicle Control", International Journal of Robotics Research, Vol. 25, No. 1, Jan 2006.
- [20] Matthew R. Parks, "Vision-Based Self Motion Estimation in a Fixed-Wing Aerial Vehicle", M.Sc. Thesis, Faculty of Virginia Polytechnic Institute and State University, 2006.

- [21] Voros S., Long J.A. and Cinquin P., "Automatic Detection of Instruments in Laparoscopic Images: A First Step Towards High Level Command of Robotic Endoscopic Holders", *International Journal of Robotics Research*, Vol. 26, No. 11, Nov 2007.
- [22] Yasushi Iwatani, Kei Watanabe, and Koichi Hashimoto, "Visual Tracking With Occlusion Handling for Visual Servo Control", *IEEE International Conference on Robotics and Automation*, Pasadena, California, USA, May 19-23, 2008.
- [23] Hesheng Wang, Yun-Hui Liu, and Dongxiang Zhou, "Adaptive Visual Servoing Using Point and Line Features With an Uncalibrated Eye-in-Hand Camera", *IEEE Transactions on Robotics*, Vol. 24, No. 4, Aug 2008.
- [24] Mechaél Sauvee, Philippe Poignet, and Etienne Dombre, "Ultrasound Image-Based Visual Servoing of a Surgical Instrument Through Nonlinear Predictive Control", *International Journal of Robotics Research*, Vol. 27, No. 1, Jan 2008.
- [25] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng, "Robotic Grasping of Novel Objects Using Vision", *International Journal of Robotics Research*, Vol. 27, No. 2, Feb 2008.
- [26] Sang-Bae Lee, "Industrial Robotic Systems With Fuzzy Logic Controllers and Neural Networks", *International Conference of Knowledge-Based Intelligent Electronic Systems*, Adelaide, Australia, 21-23 May 1997.
- [27] Gianluca Antonelli, Stefano Chiaverini, and Giusppe Fusco, "A New On-Line Algorithm for Inverse Kinematics of Robotic Manipulators Ensuring Path

Tracking Capability Under Joint Limits", IEEE Transactions On Robotics And Automation, Vol. 19, No. 1, Feb 2003.

[28] Li-Xin Wei, Hong-Rui Wang, and Ying Li, "A New Solution For The Inverse Kinematics Of Robotic Manipulators Based On Neural Networks", International Conference of Machine Learning and Cybernetics, Xi'an, 2-5 Nov. 2003.

[29] Eimei Oyama, Taro Maeda, John Q. Gan, Eric M. Rosales, Karl F. MacDorman, Susumu Tachi and Arvin Agah, "Inverse Kinematics Learning for Robotic Arms with Fewer degrees of Freedom by Modular Neural Network Systems", IEEE International Conference on Intelligent Robots and Systems, 2005.

[30] Yugui Yang, Guangzheng Peng, Yifeng Wang, and Hongli Zhang, "A New Solution fot Inverse Kinematics of 7-DOF Manipulator Based on Neural Network", IEEE International Conference on Automation and Logistics, 18-21 of August, 2007.

[31] Wuxin Huang, Shili Tan, and Xianhua Li, "Inverse Kinematics of Compliant Manipulators Based on the Immune Genetic Algorithm", IEEE Computer Society, The Fourth International Conference on Natural Computation, 2008.

[32] Harry Benson, "University Physics", First Edition, John Wiley and Sons Publishing Incorporation, 1991.

[33] Jadran Lenarcic and Roth B., "Advances in Robot Kinematics, Mechanisms and Motion", Springer-Verlag, 2006.

- [34] Selig J. M., "Introductory Robotics", First Edition, Prentice Hall Incorporation, 1992.
- [35] Eimei Oyama, Taro Maeda, John Q. Gan, Eric M. Rosales, Karl F. MacDorman, Susumu Tachi, Arvin Agah, "Inverse Kinematics Learning for Robotic Arms with Fewer Degrees of Freedom by Modular Neural Network Systems", Published by the National Institute of Advanced Industrial Science and Technology (AIST), Japan, 2004.
- [36] Azriel Rosenfeld and Avinash Kak, "Digital Picture Processing", Academic Press Incorporation, 1976
- [37] C. Ray Wiley and Louis C. Barrett, "Advanced Engineering Mathematics", Fifth Edition, McGraw Hill Book Company, 1987.
- [38] Rechar O. Duda and Peter E. Hart, "Pattern Classification and Scene Analysis", First Edition, John Wiley and Sons Incorporation, 1973.
- [39] Kenneth R. Castleman, "Digital Image Processing", First Edition, Prentice Hall Incorporation, 1979.
- [40] "Computer Vision", Dana H. Ballard and Christopher M. Brown, Published by The British Machine Vision Association, London, 2003.
- [41] Norman S. Nise, "Control Systems Engineering", Third Edition, John Wiley and Sons Incorporation, 2000.
- [42] Katsuhiko Ogata, "Modern Control Engineering", Fourth Edition, Prentice Hall Incorporation, 2002.

- [43] Jayantha Katupitiya and Kim Bentley, “Interfacing With C Plus Plus”, Springer-Verlag Incorporation, 2006.
- [44] John Graig, “Introduction to Robotics, Mechanics and Control”, Second Edition, Addison Wesley Longman, 1989.
- [45] George-Christopher Vosniakos, ZenonKannas, " Motion coordination for industrial robotic systems with redundant degrees of freedom", Journal of Robotics and Computer-Integrated Manufacturing, Vol. 25, pp 417-431, 2009.
- [46] Rafael M. Inigro and James S. Morton, "Simulation of the Dynamics of an Industrial Robot", IEEE Journal, Vol. 34, No. 1 1991.

Appendices

A

Appendix

Derivation of the Equations of Motion of the Robotic Arm

The derivation method is based on the Lagrange-Euler method which can be stated as:

$$\frac{d}{dt} \left[\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_i} \right] - \frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial q_i} = \tau_i, \quad i=1, 2, \dots, n \quad \text{.....(A-1)}$$

where n is the number of actuators used in the robot and $L(\mathbf{q}, \dot{\mathbf{q}})$, the *Lagrangian*, which is defined as:

$$L(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q}) \quad \text{.....(A-2)}$$

$T(\mathbf{q}, \dot{\mathbf{q}})$ and $U(\mathbf{q})$ are the total Kinetic and total Potential energy functions of the dynamic system respectively. Starting from the base, the total Kinetic energy is the sum of the Kinetic energy functions of the individual links and is given in the general form $T_i = \frac{1}{2} I_i v_i^2$, where (v) is the velocity vector of the centre of mass of the i^{th} link.

1- The base (Link #0):

The kinetic energy for the base is given as:

$$T_o = \frac{1}{2} I_{ass} \dot{q}_o^2, \quad \text{.....(A-3)}$$

where I_{ass} is the assembly mass moment of inertia of the robotic arm about the axis of rotation of the base. It is calculated in chapter three in section 3.3. The base has no Potential energy function because it is usually taken as the datum for the whole assembly.

2- Link #1:

$$\begin{aligned}
x_1 &= l_{c_1} \sin(q_1) \\
\dot{x}_1 &= l_{c_1} \cos(q_1) \cdot \dot{q}_1 \\
y_1 &= -l_{c_1} \cos(q_1) \\
\dot{y}_1 &= l_{c_1} \sin(q_1) \cdot \dot{q}_1
\end{aligned}
\tag{A-4}$$

The velocity vector is:

$$\boldsymbol{\nu}_1 = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} l_{c_1} \cos(q_1) \cdot \dot{q}_1 \\ l_{c_1} \sin(q_1) \cdot \dot{q}_1 \end{bmatrix}
\tag{A-5}$$

The square of the velocity vector is:

$$\begin{aligned}
|\boldsymbol{\nu}_1|^2 &= \boldsymbol{\nu}_1^T \cdot \boldsymbol{\nu}_1 \\
&= l_{c_1}^2 \dot{q}_1^2
\end{aligned}
\tag{A-6}$$

The total Kinetic energy is:

$$\begin{aligned}
T_1 &= \frac{1}{2} m_1 \boldsymbol{\nu}_1^T \cdot \boldsymbol{\nu}_1 + \frac{1}{2} I_1 \dot{q}_1^2 \\
&= \frac{1}{2} (m_1 l_{c_1}^2 + I_1) \dot{q}_1^2
\end{aligned}
\tag{A-7}$$

The Potential energy is:

$$U_1 = -m_1 l_{c_1} g \cos(q_1)
\tag{A-8}$$

3- Link #2:

$$\begin{aligned}
x_2 &= l_1 \sin(q_1) + l_{c_2} \sin(q_1 + q_2) \\
\dot{x}_2 &= l_1 \cos(q_1) \cdot \dot{q}_1 + l_{c_2} \cos(q_1 + q_2) \cdot (\dot{q}_1 + \dot{q}_2) \\
y_2 &= -l_1 \cos(q_1) - l_{c_2} \cos(q_1 + q_2) \\
\dot{y}_2 &= l_1 \sin(q_1) \cdot \dot{q}_1 + l_{c_2} \sin(q_1 + q_2) \cdot (\dot{q}_1 + \dot{q}_2)
\end{aligned}
\tag{A-9}$$

The velocity vector is:

$$|\boldsymbol{\nu}_2|^2 = \boldsymbol{\nu}_2^T \cdot \boldsymbol{\nu}_2
\tag{A-10}$$

After making the necessary calculations, the Kinetic energy for link #2 is:

$$T_2 = \frac{1}{2} \left[m_2 l_1^2 \dot{q}_1^2 + m_2 l_{c_2}^2 (\dot{q}_1 + \dot{q}_2)^2 + m_2 l_1 l_{c_2} \dot{q}_1 (\dot{q}_1 + \dot{q}_2) \cos(q_2) + I_2 (\dot{q}_1 + \dot{q}_2)^2 \right] \dots\dots\dots (A-11)$$

The Potential energy for link#2 is:

$$U_2 = -m_2 l_1 g \cos(q_1) - m_2 l_{c_2} g \cos(q_1 + q_2) \dots\dots\dots (A-12)$$

4- Link #3:

$$\begin{aligned} x_3 &= l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) + l_{c_3} \sin(q_1 + q_2 + q_3) \\ \dot{x}_3 &= l_1 \cos(q_1) \cdot \dot{q}_1 + l_2 \cos(q_1 + q_2) \cdot (\dot{q}_1 + \dot{q}_2) + \\ &\quad l_{c_3} \cos(q_1 + q_2 + q_3) \cdot (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \end{aligned} \dots\dots\dots (A-13)$$

$$\begin{aligned} y_3 &= -l_1 \cos(q_1) - l_2 \cos(q_1 + q_2) - l_{c_3} \cos(q_1 + q_2 + q_3) \\ \dot{y}_3 &= l_1 \sin(q_1) \cdot \dot{q}_1 + l_2 \sin(q_1 + q_2) \cdot (\dot{q}_1 + \dot{q}_2) + \\ &\quad l_{c_3} \sin(q_1 + q_2 + q_3) \cdot (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \end{aligned}$$

Following the same approach, the Kinetic energy for link #3 is:

$$\begin{aligned} T_3 &= \frac{1}{2} \left[m_3 v_3^T \cdot v_3 + I_3 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3)^2 \right] \\ &= \frac{1}{2} m_3 l_1^2 \dot{q}_1^2 + \frac{1}{2} m_3 l_2^2 (\dot{q}_1 + \dot{q}_2)^2 + \frac{1}{2} m_3 l_{c_3}^2 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3)^2 + \dots\dots\dots (A-14) \\ &\quad m_3 l_1 l_2 \dot{q}_1 (\dot{q}_1 + \dot{q}_2) \cos(q_2) + m_3 l_1 l_{c_3} \dot{q}_1 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \cos(q_1 + q_2) + \\ &\quad m_3 l_2 l_{c_3} (\dot{q}_1 + \dot{q}_2) (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \cos(q_3) + \frac{1}{2} I_3 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3)^2 \end{aligned}$$

The Potential energy is:

$$U_3 = -m_3 l_1 g \cos(q_1) - m_3 l_2 g \cos(q_1 + q_2) - m_3 l_{c_3} g \cos(q_1 + q_2 + q_3) \dots\dots\dots (A-15)$$

The total Kinetic energy is:

$$T = T_o + T_1 + T_2 + T_3 \dots\dots\dots (A-16)$$

The total Potential energy is:

$$U = U_o + U_1 + U_2 + U_3 \dots\dots\dots (A-17)$$

Recalling equation (A2), the lagrangian is given by:

$$L(q, \dot{q}) = T_o + T_1 + T_2 + T_3 - (U_o + U_1 + U_2 + U_3)$$

$$\frac{\partial L}{\partial \dot{q}_o} = I_{ass} \dot{q}_o$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_o} \right) = I_{ass} \ddot{q}_o + \dot{q}_o \frac{\partial I_{ass}}{\partial t} \quad \dots\dots\dots(A-18)$$

$$\frac{\partial L}{\partial \dot{q}_1} = m_1 l_1^2 \dot{q}_1 + I_1 \dot{q}_1 + m_2 l_1^2 \dot{q}_1 + m_2 l_{c_2}^2 (\dot{q}_1 + \dot{q}_2) +$$

$$m_2 l_1 l_{c_2} (2\dot{q}_1 + \dot{q}_2) \cos(q_2) + I_2 (\dot{q}_1 + \dot{q}_2) +$$

$$m_3 l_1^2 \dot{q}_1 + m_3 l_2^2 (\dot{q}_1 + \dot{q}_2) + m_3 l_{c_3}^2 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) +$$

$\dots\dots\dots(A-19)$

$$m_3 l_1 l_2 (2\dot{q}_1 + \dot{q}_2) \cos(q_2) + m_3 l_1 l_{c_3} (2\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \cos(q_2 + q_3) +$$

$$m_3 l_2 l_{c_3} (2\dot{q}_1 + 2\dot{q}_2 + \dot{q}_3) \cos(q_3) + I_3 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3)$$

$$\frac{\partial L}{\partial \dot{q}_2} = m_2 l_{c_2}^2 (\dot{q}_1 + \dot{q}_2) + m_2 l_1 l_{c_2} \dot{q}_1 \cos(q_2) + I_2 (\dot{q}_1 + \dot{q}_2) +$$

$$m_3 l_2^2 (\dot{q}_1 + \dot{q}_2) + m_3 l_{c_3}^2 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) + m_3 l_1 l_2 \dot{q}_1 \cos(q_2) +$$

$\dots\dots\dots(A-20)$

$$m_3 l_1 l_{c_3} \dot{q}_1 \cos(q_2 + q_3) + m_3 l_2 l_{c_3} (2\dot{q}_1 + 2\dot{q}_2 + \dot{q}_3) \cos(q_3) +$$

$$I_3 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3)$$

$$\frac{\partial L}{\partial \dot{q}_3} = m_3 l_{c_3}^2 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) + m_3 l_1 l_{c_3} \dot{q}_1 \cos(q_2 + q_3) +$$

$\dots\dots\dots(A-21)$

$$m_3 l_2 l_{c_3} (\dot{q}_1 + \dot{q}_2) \cos(q_3) + I_3 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3)$$

$$\begin{aligned}
\frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{q}_1} \right) = & m_1 l_1^2 \ddot{q}_1 + I_1 \ddot{q}_1 + m_2 l_1^2 \ddot{q}_1 + m_2 l_{c_2}^2 \ddot{q}_1 + m_2 l_{c_2}^2 \ddot{q}_2 + \\
& 2m_2 l_1 l_{c_2} \cos(q_2) \ddot{q}_1 - 2m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_1 \dot{q}_2 + m_2 l_1 l_{c_2} \cos(q_2) \ddot{q}_2 - \\
& m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_2^2 + I_2 \ddot{q}_1 + I_2 \ddot{q}_2 + m_3 l_1^2 \ddot{q}_1 + m_3 l_2^2 \ddot{q}_1 + m_3 l_2^2 \ddot{q}_2 + m_3 l_{c_3}^2 \ddot{q}_1 + \\
& m_3 l_{c_3}^2 \ddot{q}_2 + m_3 l_{c_3}^2 \ddot{q}_3 + 2m_3 l_1 l_2 \cos(q_2) \ddot{q}_1 - 2m_3 l_1 l_2 \sin(q_2) \dot{q}_1 \dot{q}_2 + m_3 l_1 l_2 \cos(q_2) \ddot{q}_2 - \\
& m_3 l_1 l_2 \sin(q_2) \dot{q}_2^2 + 2m_3 l_1 l_{c_3} \cos(q_2 + q_3) \ddot{q}_1 - 2m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_1 (\dot{q}_2 + \dot{q}_3) + \\
& m_3 l_1 l_{c_3} \cos(q_2 + q_3) \ddot{q}_2 - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_2 (\dot{q}_2 + \dot{q}_3) + m_3 l_1 l_{c_3} \cos(q_2 + q_3) \ddot{q}_3 - \\
& m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_3 (\dot{q}_2 + \dot{q}_3) + 2m_3 l_2 l_{c_3} \cos(q_3) \ddot{q}_1 - 2m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_1 \dot{q}_3 + \\
& 2m_3 l_2 l_{c_3} \cos(q_3) \ddot{q}_2 - 2m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_2 \dot{q}_3 + m_3 l_2 l_{c_3} \cos(q_3) \ddot{q}_3 - m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_3^2 + \\
& I_3 \ddot{q}_1 + I_3 \ddot{q}_2 + I_3 \ddot{q}_3
\end{aligned}
\tag{A-22}$$

$$\begin{aligned}
\frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{q}_2} \right) = & m_2 l_{c_2}^2 \ddot{q}_1 + m_2 l_{c_2}^2 \ddot{q}_2 + m_2 l_1 l_{c_2} \cos(q_2) \ddot{q}_1 - m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_1 \dot{q}_2 + I_2 \ddot{q}_1 + I_2 \ddot{q}_2 + \\
& m_3 l_2^2 \ddot{q}_1 + m_3 l_2^2 \ddot{q}_2 + m_3 l_{c_3}^2 \ddot{q}_1 + m_3 l_{c_3}^2 \ddot{q}_2 + m_3 l_{c_3}^2 \ddot{q}_3 + m_3 l_1 l_2 \cos(q_2) \ddot{q}_1 - \\
& m_3 l_1 l_2 \sin(q_2) \dot{q}_1 \dot{q}_2 + m_3 l_1 l_{c_3} \cos(q_2 + q_3) \ddot{q}_1 - m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_1 (\dot{q}_2 + \dot{q}_3) + \\
& 2m_3 l_2 l_{c_3} \cos(q_3) \ddot{q}_1 - 2m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_1 \dot{q}_3 + 2m_3 l_2 l_{c_3} \cos(q_3) \ddot{q}_2 - \\
& 2m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_2 \dot{q}_3 + m_3 l_2 l_{c_3} \cos(q_3) \ddot{q}_3 - m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_3^2 + \\
& I_3 \ddot{q}_1 + I_3 \ddot{q}_2 + I_3 \ddot{q}_3
\end{aligned}
\tag{A-23}$$

$$\begin{aligned}
\frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{q}_3} \right) = & m_3 l_{c_3}^2 \ddot{q}_1 + m_3 l_{c_3}^2 \ddot{q}_2 + m_3 l_{c_3}^2 \ddot{q}_3 + m_3 l_1 l_{c_3} \cos(q_2 + q_3) \ddot{q}_1 - \\
& m_3 l_1 l_{c_3} \sin(q_2 + q_3) \dot{q}_1 (\dot{q}_2 + \dot{q}_3) + m_3 l_2 l_{c_3} \cos(q_3) \ddot{q}_1 - \\
& m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_1 \dot{q}_3 + m_3 l_2 l_{c_3} \cos(q_3) \ddot{q}_2 - m_3 l_2 l_{c_3} \sin(q_3) \dot{q}_2 \dot{q}_3 + \\
& I_3 \ddot{q}_1 + I_3 \ddot{q}_2 + I_3 \ddot{q}_3
\end{aligned}
\tag{A-24}$$

$$\begin{aligned}
\frac{\partial L}{\partial q_1} = & \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_1} - m_1 l_{c_1} g \sin(q_1) - m_2 l_1 g \sin(q_1) - m_2 l_{c_2} g \sin(q_1 + q_2) - \\
& m_3 l_1 g \sin(q_1) - m_3 l_2 g \sin(q_1 + q_2) - m_3 l_{c_3} g \sin(q_1 + q_2 + q_3)
\end{aligned}
\tag{A-25}$$

$$\begin{aligned}
\frac{\partial L}{\partial q_2} = & \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_2} - m_2 l_1 l_{c_2} \dot{q}_1 (\dot{q}_1 + \dot{q}_2) \sin(q_2) - m_3 l_1 l_2 \dot{q}_1 (\dot{q}_1 + \dot{q}_2) \sin(q_2) - \\
& m_3 l_1 l_{c_3} \dot{q}_1 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \sin(q_2 + q_3) - m_2 l_{c_2} g \sin(q_1 + q_2) - \\
& m_3 l_2 g \sin(q_1 + q_2) - m_3 l_{c_3} g \sin(q_1 + q_2 + q_3) \\
& \dots\dots\dots(A-26)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L}{\partial q_3} = & \frac{1}{2} \dot{q}_o^2 \frac{\partial I_{ass}}{\partial q_3} - m_3 l_1 l_{c_3} \dot{q}_1 (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \sin(q_2 + q_3) - \\
& m_3 l_2 l_{c_3} (\dot{q}_1 + \dot{q}_2) (\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \sin(q_3) - \\
& m_3 l_{c_3} g \sin(q_1 + q_2 + q_3) \\
& \dots\dots\dots(A-27)
\end{aligned}$$

B

Appendix

Best Eigenvector Line Fitting Method (BEF)

BEF is a method used for fitting a straight line to a set of point. In this method the definition of the “best fit” differs from the classic Minimum Squared Error (MSE). A line is considered a best fit to a set of points if it minimizes the sum of the squares of the perpendicular distances from the points to the line. Figure (B-1) illustrates the difference in the two methods. The MSE method selects the line minimizing the sum of the squares of the lengths of the vertical solid lines whereas the BEF minimizes the sum of the squares of the lengths of the dotted lines.

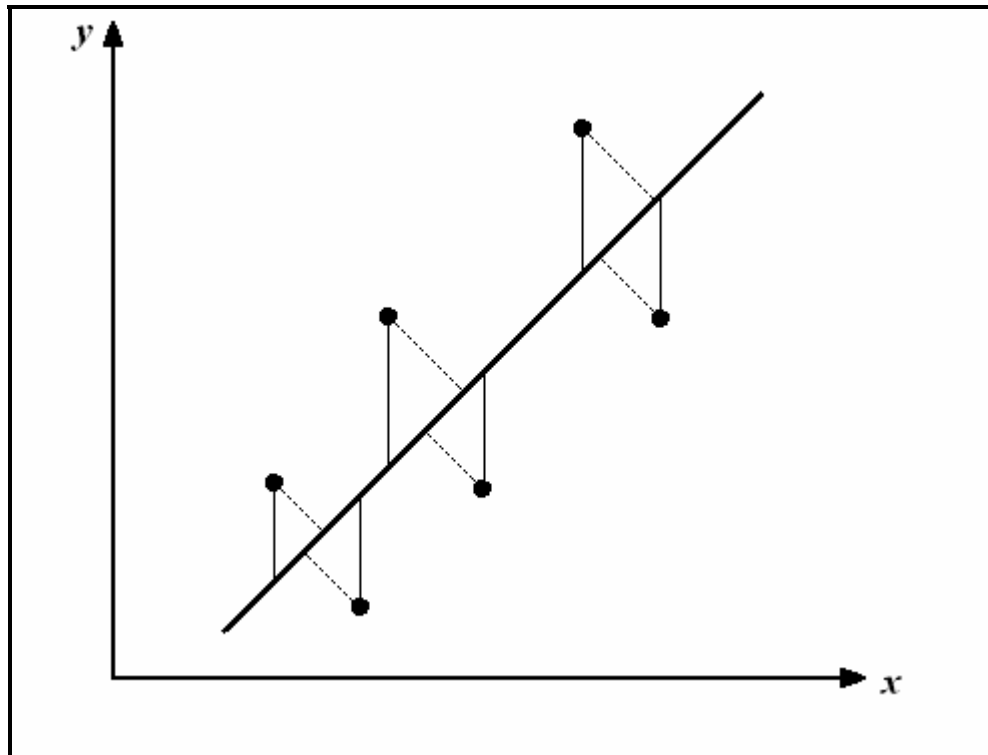


Figure (B-1) a comparison between the MSE and the BEF methods.

Suppose a set of (n) points $\{(x_i, y_i)\}$, $i=1 \dots n$, to which a line is to be fitted. Denote the i^{th} point (x_i, y_i) as the vector (v_i) and let (d_i) be the perpendicular

distance from (v_i) to the line. So, the task is to minimize $d^2 = \sum_{i=1}^n d_i^2$, see figure (B-1).

If the line is characterized by its unit normal vector (N) as shown in figure(B-2), then:

$$d_i^2(N) = (N \cdot v_i)^2 = (N^T v_i)^2$$

$$d^2(N) = \sum_{i=1}^n d_i^2(N)$$

$$d^2(N) = \sum_{i=1}^n (N^T v_i)^2$$

$$d^2(N) = \sum_{i=1}^n (N^T v_i)(N^T v_i)$$

$$d^2(N) = N^T \left(\sum_{i=1}^n v_i v_i^T \right) N$$

The symmetric matrix $S = \sum_{i=1}^n v_i v_i^T$ is called the scatter matrix of (n) given points. The best fitting line is in the direction of the principal eigenvector of (S) which is the eigenvector associated with the largest eigenvalue.

Figure (B-3) shows the difference between the two methods. For the given set of points, the BEF results the correct fitting line while the MSE fails.

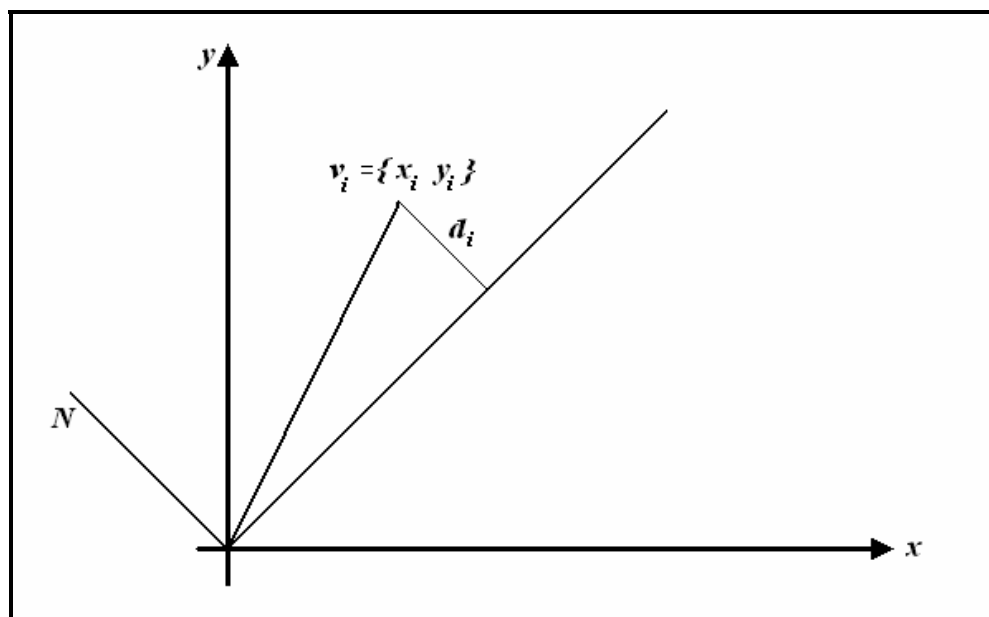


Figure (B-2) line geometry

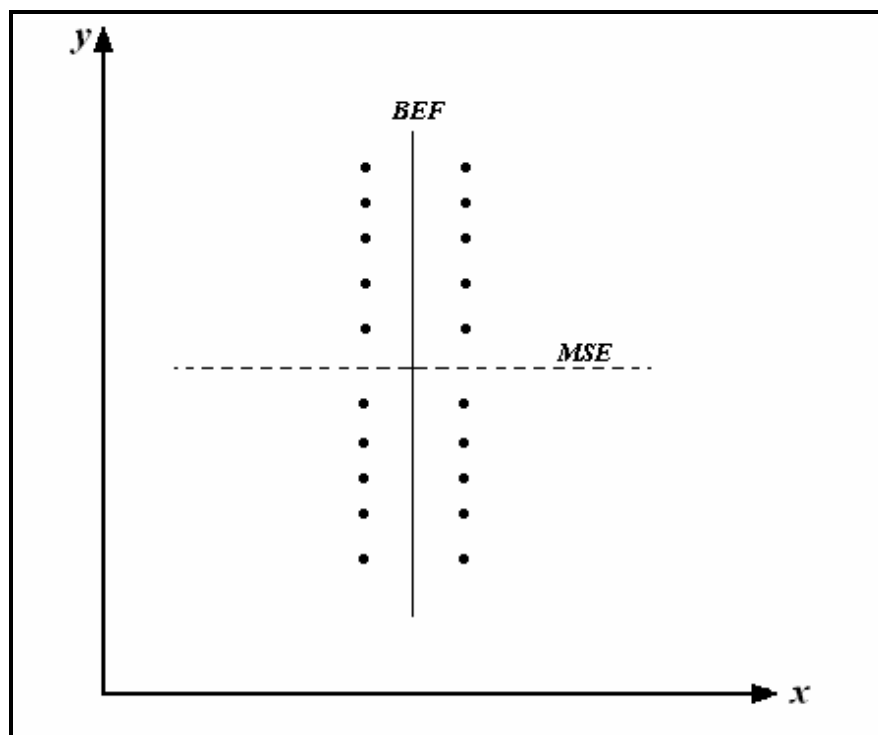


Figure (B-3) two different fits to the same set of points.

الخلاصة

الإنسان الآلي (الروبوت) هو جهاز الكتروميكانيكي يستخدم من قبل الانسان لأنجاز عمليات متكررة بدقة عالية لأن الانسان لا يستطيع أن يكرر نفس العملية لعدة مرات و بنفس الكفاءة. تتحدد حركة الانسان الآلي بواسطة مسيطر يقوم بأرسال اشارات السيطرة للمحركات في مفاصل الانسان الآلي وذلك حسب اشارات تغذية مآخوذة من متحسسات مصممة لهذا الغرض. في حالة العمليات المتكررة، فان الانسان الآلي يستلم اشارات السيطرة بواسطة برنامج حاسوبي معد لهذا الغرض مما يعني ان الانسان الآلي يعمل بصورة مستمرة بدون الأخذ بنظر الاعتبار المتغيرات المحتملة والاطاء العرضية التي قد تحدث في البيئة المحيطة بالروبوت.

يحصل الروبوت على مرونة أكثر في التعامل مع هذه الاخطاء عند تزويده بأمكانية التفاعل مع محيطه بصريا. تعطي انظمة السيطرة المعتمدة على الأشارات الصورية للروبوت القدرة على رؤية المحيط و الاستجابة للتغيرات فيه.

في هذا العمل، تمت محاكاة منظومة سيطرة بصرية للسيطرة على حركة ذراع آلي ذي اربعة درجات حرية بأستخدام كاميرا صغيرة توضع عند نهاية الذراع. كذلك، تم نمذجة الموديل الديناميكي لهذا النوع من الأذرع الآلية. أدخلت على الموديل مجموعة من المسارات تتدرج في تعقيدها وتم حساب القوى و العزوم المؤثرة على كل مفصل من مفاصل الذراع. أستخدمت طريقة جديدة لتحديد قيم الزوايا للمفاصل في الذراع اثناء قيامه بأتباع أي من تلك المسارات. أثبتت هذه الطريقة كفاءتها من خلال سرعتها ودقتها في استخراج النتائج.

تم استخدام المعالجة الصورية لتتبع حركة جسم معين امام الكاميرا عن طريق تحديد شكل الجسم وموقعه الانى ومن ثم توليد اشارات السيطرة اللازمة لتحريك الذراع لتعقب ذلك الجسم. تم استخدام طريقة (Hough Transform) و طريقة (Template Matching) في عملية تحديد حركة الجسم.

تم بناء برنامج فرعي يقوم مقام المسيطر (PID Controller) لتوليد اشارات السيطرة للمحركات حسب حركة الجسم المعني.

استخدمت لغة (Visual Basic 6.0) في بناء جميع اجزاء برنامج المحاكاة في هذا البحث.



جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية الهندسة

أستخدام المعلومات الصورية للسيطرة على حركة ذراع ألي

رسالة

مقدمة إلى كلية الهندسة في جامعة بابل وهي جزء من متطلبات نيل
درجة ماجستير علوم في الهندسة الميكانيكية
(مكانيك تطبيقي)

أعدت من قبل

فراس صبحي حميد احمد

بكالوريوس علوم في الهندسة الميكانيكية