OPTIMIZATION OF THE TOOL PATH IN A ROBOTIC ENVIRONMENT

by

MUKUND VENKATACHARI NARASIMHAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2006

DEDICATION

This dissertation is dedicated to Lord Sri Ram & Krishna, and to their most humble devotee Hanuman, to my parents and my brother for their support and blessings.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and gratitude to Dr. Wang for all the help, guidance, motivation and technical ideas. He was always ready to look into the results, analyze and show me the direction to proceed in the right direction. I would like to thank Dr Lawrence for all the help and financial support in the form of his teaching assistant. This helped me take care of things and to concentrate on my dissertation and research. I would also want to thank Dr Dragan for all the help. If not for his combinatorial optimization and non linear programming courses, I wouldn't have thought in those lines and developed these algorithms.

I sincerely thank my other committee members Dr. Chan and Dr. Nomura for serving on my committee. They have been a continuous source of inspiration throughout my study at The University of Texas at Arlington. I also sincerely thank Dr. Yih for teaching me the basics of robotics and explaining me clearly about the symbolic notation and transformations. If not for him, I couldn't have refined the CB notation and developed this new notation. I am fortunate for having come in contact with all of them.

I want to thank my parents and family members for their never-ending support and motivation. I would also like to thank all my friends for their encouragement and support during my graduate study.

November 20, 2006

ABSTRACT

OPTIMIZATION OF THE TOOL PATH

IN A ROBOTIC ENVIRONMENT

Publication No. ______

Mukund Venkatachari Narasimhan, PhD.

The University of Texas at Arlington, 2006

Supervising Professor: Bo Ping Wang

The rising costs and demand coupled with shrinking energy generating resources has triggered the need for optimum use of energy resources, minimizing the overhead costs and maximizing the profits. Robots have been long touted as a fit replacement of human labor force in manufacturing sectors. Slowly there is increased presence of robots in automobile and electronic industries. Typically they are used for welding, painting, ironing, assembly, pick and place, inspection, and testing. Mass production, fulfilling the ever increasing demand, can be accomplished by robots because of their precision, speed of operation and high endurance capabilities.

This dissertation comprises of two parts, the first part concentrates on optimizing the tool path. The path could be either a closed loop where in the robotic manipulator would get back to the home position after completing the task or could be an open segment where the manipulator would start and end at different locations after the completion of the task. Optimization of the tool path is similar to solving a Traveling Salesman Problem. A technique of insertion and reordering is applied to obtain an optimized tool path.

The second part deals with the direct and inverse kinematics of the given robot configuration. A new notation, which takes into account all the six parameters necessary to define a rigid body in space, is developed to analyze kinematic analysis of industrial robots. Using the same notation, an inverse kinematics solver is used to compute the joint parameters necessary for the robot manipulator to reach the target point in space. This solver uses an iterative procedure to solve complex non-linear inverse kinematics problems. This solver tested on robots with revolute or prismatic or a combination of both yielded satisfactory results. The same solver can be used to analyze cylindrical, helical, spherical, combinations of all joints, planar and spatial mechanisms.

The combination of optimizing the tool path and use of robots in the mass production of high demand products would go a long way in minimizing the costs, maximizing the profits and as well as delivering supply on time.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Objective

Robots are a tailor made fit for the electronics industry. In both military and consumer electronics, production would be next to impossible without robotics for many reasons, ranging from quality and quantity to ergonomics and economics [1]. Robots are used for a variety of tasks like assembling circuit boards, installing the circuit boards onto the chassis of electronics, inspection, testing, pick and place, welding, soldering, spray painting and the list continues. Robots are the probable solution to keep tabs on high labor costs, high production and efficiency. In the future as electronics get smaller and more sophisticated, robotics will be increasingly called upon to assemble, pack, inspect and test these micro electronic products.

The rising costs and demand coupled with shrinking energy generating resources has triggered the need for optimum use of energy resources, minimizing the overhead costs and maximizing the profits. This is where optimization comes into picture. So the objectives of this dissertation are developing a new algorithm to optimize the tool path of the robotic manipulator, developing a better joint transformation technique using a new notation terminology, developing a new inverse kinematics solver using this notation.

The robotic tool has to traverse to various locations for machining, assembly and for pick or place operations. These locations or co-ordinate points are similar to the nodes/cities in Traveling Salesman Problem. Therefore the optimization of the tool path is similar to obtaining a solution to a symmetric TSP. Insertion and Re-ordering Algorithm is used to obtain the solution to TSP, the same algorithm could also be used to solve open segment path with fixed start and end points.

A new notation, which takes into account all the six parameters necessary to define a rigid body in space, is developed to analyze industrial robots. Transformation matrices, thus generated from these parameters are used to study direct kinematics and for generating workspace or work envelop of the given robotic configuration. For the robotic manipulator to reach the intended target point, one must figure out the joint parameter values. Inverse Kinematics deals with determining these parameter values, but solving this is a bit hard as the equations involved are highly nonlinear. A new iterative algorithm is used to generate accurate solutions by determining the joint parameter values. This algorithm produced satisfactory results when tested on robots which had revolute, prismatic and an arbitrary combination of revolute and prismatic joints. The algorithm can also be extended to analyze other types of joints like cylindrical, helical, spherical joints and as well as planar & spatial mechanisms. All these algorithms are integrated into a software tool MyRobot. This CAD/CAE tool, built using java and java3d, has a good user friendly interface. The main reason for choosing this programming language is because of its platform independent characteristic. This makes it easy for the application to run on any operating systems.

Any kind of robot with a combination of revolute and prismatic joint pairs could be built without any hassle. One can study the direct kinematics, workspace generation, inverse kinematics, optimized path planning capabilities of the modeled robot.

1.2 Background

The Traveling Salesman Problem (TSP) is a deceptively simple combinatorial optimization problem. The problem is about a salesman who wants to visit a number of cities cyclically. The conditions being, he has to visit each city once and needs to get back to the start city at the end of the tour. The hard part is due to the fact that if one considers n cities, then one has to probably determine them systematically and finally find the minimum path length. This requires atleast (n-1)! steps, n>3. So if one considers a problem with 10 cities then its equal to figuring out the minimum of 9! equal to 362880 different paths. This will increase rather exponentially as n increases and it will take ages to verify each route and figure out the minimum length.

Dorigo and Gambardella [2] developed a distributed algorithm that was applied to get a solution for the Traveling Salesman Problem (TSP). Real ants are capable of finding the shortest path from a food source to their nest without using visual cues by exploiting pheromone information. The distributed algorithm used a set of cooperating agents called ants to find good solutions to TSP. These Ant agents were shown to cooperate using an indirect form of communication mediated by a pheromone that get deposited on the edges of the TSP graph while building solutions. An evolutionary algorithm for solving traveling salesman problem was proposed by Tsai et al [3]. They enhanced the ability of exploration and exploitation by incorporating global and local searches and developed a neighbor-join operator. A novel meta-heuristic approach called Ant Colony Optimization with Multiple Ant Clans was developed by Cheng-Fa Tsai et al [4]. This was based on parallel genetic algorithm that searches the solution space to obtain Global Minimum. This technique was used to solve TSP. For solving large TSP, two other techniques Multiple Nearest Neighbor and Dual Nearest Neighbor were used.

A hybrid algorithm for solving vehicle route planning problems was proposed by Lee [5]. This algorithm combined both ant colony optimization (ACO) and genetic algorithm (GA) and used the prominent features of respective techniques. This way this technique avoided premature convergence and was able to fetch feasible solutions. A combination of local optimization heuristics and phenotype genetic operators was used by Pullan [6] to solve TSP. The local optimization heuristics reduced the search domain, while the phenotype genetic operators eliminated the creation of invalid tours and assisted the generation of sub-optimal schema. Xuan and Li [7] introduced a new Local Evolutionary Algorithm (LEA) to solve the TSP. The algorithm used fast local search methods in neighborhood search and utilized the robustness of evolutionary methods, in global search in order to obtain global optimum. Cuiru Wang [8] et al used Modified Particle Swarm Optimization technique to solve the typical combinatorial optimization problem, the Traveling Salesman Problem. Particle Swarm Optimization algorithm was modified based on the concepts of Adjustment Operator and Adjustment Sequence.

The word Robot is derived from a Czech word "robota", which means tedious labor, serf or one in servient labor. Robot Institute of America [9] in 1979 defined as "A reprogrammable, multifunctional manipulator designed to move materials, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks". Merriam-Webster Online Dictionary defines it as "a machine or device that looks like a human being and automatically performs complicated and repetitive tasks". The term "robot" was first used in a play called "R.U.R." or "Rossum's Universal Robots" (1921) by the Czech writer Karel Capek. From this began the concept of robot and ever since they have undergone a lot of developments. One can find robots employed in a variety of tasks ranging from simple pick and place to handling hazardous radio active materials, arc welding, spray painting, assembly and inspections. Now a days, they assist medical doctors to perform precise non invasive medical surgeries. The increasing needs for high productivity, desired quality and lower costs are pushing the industries to look towards computer based automated systems. The robots bring many benefits to workers and to company owners by taking care of dirty, difficult and dangerous jobs and by being cost effective [10]. The integration of computer to control these automated systems will increase productivity. Some of the fields where computers are extensively used include CAD/CAM, computer simulations & modeling and analysis.

Since 1950, many developments of computer software packages have taken place in the field of mechanism, which include robots. Among them are DRAM (Dynamic Response of Articulated Machinery) by Chace and Smith, IMP (Integrated

Mechanisms Program) by Sheth and Uicker, ADAMS (Automated Dynamic Analysis of Mechanical Systems) by Orlandea and Chace, DADS (Dynamic Analysis and Design System) by Wehage and Haug, LINCAGES (LINkages Computer Analysis and Graphically Enhanced Synthesis) by Minnesota Technology Transfer, IGRIP from Delmia corporation, DYMES (Dynamics of Mechanical Systems), COSMOS/Motion by Solid Works Corporation, Pro/Mechanica Motion simulation package by PTC, RobotAssist by New River Kinematics, Roboworks from Newtonium, ROBOTICA, SDS, WORKSPACE, DynaMech and others.

Kinematics is the science of motion, which treats motion without regard to the forces that causes it [11]. Kinematics is one of the most important fields in robotic engineering since robot manipulation can only be achieved by the control of manipulator or the end effector and its associated parts, tools, and objects in Euclidean three dimensional space. The main objective in robot manipulation is the ability to position the end effector at a specified location with specified orientation. It can be reached by solving the direct and inverse kinematic specifications for a particular robot manipulator. The direct kinematics is used to determine the joint displacements, velocities, accelerations, and all higher order derivatives of the position variables (with respect to time or any other variable(s)). The relative segmental displacement, velocity and acceleration are determined from the prescribed geometric specifications of the robot. On the other hand, the inverse kinematic problems for robot manipulators deals with calculating a possible sets of joint parameters which could be used to attain the given position and orientation of the manipulator.

Mechanisms can be divided into planar mechanisms and spatial mechanisms, according to the relative motion of the rigid bodies. In planar mechanisms, all of the relative motions of the rigid bodies are in one plane or in parallel planes. If there is any relative motion that is not in the same plane or in parallel planes, the mechanism is called the spatial mechanism. In other words, planar mechanisms are essentially two dimensional while spatial mechanisms are three dimensional.

An open chain mechanism, commonly known as a robot or manipulator is an interconnection of rigid links, connected at the joints that allow relative motion of the adjacent links. The relationships and behavior of these links and joints can be modeled by using symbolic notation and deriving transformations from it.

Denavit and Hartenberg, in 1955, modeled and analyzed mechanical systems by developing the first symbolic notation [12]. This notation, D-H notation employs only four parameters (a, α, β, s) to describe the shape and joint characteristics of each link. This notation is used extensively in the analysis of robotic manipulators and mechanisms. But some complications arise while analyzing certain mechanisms as D-H notation only defines relative joint positions, due to its joint definition. It also fails when adjacent joint axes are not parallel or normal to each other. A modified notation was developed by Sheth and Uicker in 1972 [13], whose purpose was to improve and extend the use of D-H notation to determine the exact joint position in space. S-U notation introduced six constant joint parameters (a, b, c, α, β, γ), plus a variable part that contains the same number of parameters as the degrees of freedom of the joint. The Cylindrical-Bryant angles notation or C-B notation in short, was developed by Yih in

1991 to model and analyze spatial robots [14]. This uses cylindrical and bryant angles transformation matrices to describe the shape and joint characteristics of each link. Unlike D-H notation, C-B notation permits determination of exact joint positions in space. This exact joint position is the actual location of the physical joint center in space. This notation uses five parameters (θ, h, r, α, β) and in some cases six parameters (θ, h, r, α, β, γ) to describe the link shape and joint behavior. Even this notation has a problem as there are no variables to represent the transformation in the y-direction.

Roth [15] determined that the reachable workspace of a general manipulator was bounded by the surface of a generalized torus. Gupta and Roth [16] extended this approach by considering *n* degree-of-freedom manipulators. Kumar and Waldron [17] introduced the concept of dexterous workspace. With regard to workspace determination of serially linked manipulators, important work was done by Tsai and Soni [18]. They solved the accessible region of planar two link robot arms in terms of equivalent area, and developed an algorithm to observe the workspace for n-R robots. Ceccarelli [19-20], developed a formulation for the workspace boundary of general N-revolute manipulators as well as proposed in the form of Feasible Workspace Regions to determine regions within which a toroidal workspace can be prescribed for a feasible dimensional design of two-revolute manipulators. An optimization approach to computing the boundaries of the workspaces of planar manipulators was developed by J.A. Snyman et al. [21]. A.B. Kyatkin and G.S. Chirikjian [22] synthesized binary manipulators using Fourier transform on the Euclidean group. Yunfeng Wang and Gregory S. Chirikjian [23] developed a diffusion-based algorithm for workspace generation of highly articulated manipulators. The diffusion-based algorithm makes the workspace generation problem as simple as solving a diffusion-type equation which has an explicit solution. The equation is a partial differential equation defined on the motion group and describes the evolution of the workspace density function depending on the manipulator length and kinematic properties. Analytical methods for identifying the boundary to the workspace of serial mechanical manipulators and the boundary to voids in the workspace as well as a complete solution to the problem of determining singular behavior in the workspace of open-chain mechanisms were developed by Abdel-Malek et al. [24-25]. Lai and Menq [26] conducted a theoretical study on dexterous workspace of robotic manipulators and developed a method based on the concept that the boundary of the robot's dexterous workspace governed by the boundary of $W_1(4)$, where W_1 (4) is the reachable space of joint 4 when joints 1-3 were allowed to rotate freely. Zhang and Sikka [27] derived a complete description of the workspace of a two degrees of freedom planar robot manipulator. This derivation took into account the effects of kinematic parameters on the work space. This derivation was based on a method using polynomial discriminants of the kinematic equations and expressed the workspace as a set of boundary curves, so that they could be used conveniently in a number of applications. Ceccarelli [28] introduced the concept of feasible workspace regions and used it to develop two revolute manipulators for a workspace prescribed through arbitrarily given workspace points.

Krzysztof Tcho´n and Robert Muszy´nski [29] developed a new method of solving the singular inverse kinematic problem for non-redundant robotic manipulators.

An evolutionary symbolic regression algorithm in order to give models approximating the inverse kinematic model (IKM) of any general 6R manipulator, by program-functions with variable forms and sizes was implemented by F. Chapelle and P. Bidaud [30]. Yang et al. developed an inverse kinematics solution for manipulators based on fuzzy logic. The algorithm was implemented on a simple two-DOF robot manipulator [31]. Juan Manuel Ahuactzin and Kamal K. Gupta [32] proposed a novel and global approach to solving the point-to-point inverse kinematics problem for highly redundant manipulators. Farbod Fahimi et al. [33] developed a new and efficient kinematic position and velocity solution scheme for spatial hyper-redundant manipulators. The Backbone curve concepts and a modal approach were used to resolve the manipulator's redundancy. A. Ramdane-Cherif et al. [34] developed an efficient algorithm to solve inverse kinematic problem of redundant robots subjected to a set of criterion and constraints. The method used is based on formulating a simple optimization problem using neural network. Alberto Borboni [35] developed a new general fuzzy iterative algorithm based classical algorithms for the solution of the inverse kinematic problems for serial manipulators. Luya Li et al. [36] developed an effective method for determining the inverse kinematics of redundant robots and implementing real-time, kinematic control with joint velocity constraints. A new algorithm proposed by Chan and Lawrence [37], gave accurate solutions near & at singular points and as well as for targets which were out of the manipulator's reach. The algorithm used the error damped pseudo-inverse to obtain stable joint corrections near singularities. Kun Ji and Yih [38] used a generalized approach for solving the inverse kinematics problem. The method was applicable to non-redundancy control of most of industrial robots and also to the redundancy control of Light Duty Utility Arm. The joint variables in multi variable, non linear kinematic equations were separated and restructured into a system of linear jacobian equations, these equations were re-written and solved in matrix form. A closed-form inverse kinematics solver for non-redundant reconfigurable robots, based on the Product-of-Exponentials (POE) formula was proposed by Chen and Gao [39]. POE reduction techniques and sub-problems were used to obtain inverse kinematic solutions to a large number of possible configurations. A technique, by Antonelli et al [40], based on a new second-order inverse kinematics algorithm was used to enable handling of velocity and acceleration constraints while tracking the desired end-effector path. This ensured that the joint variables were within their limits.

Fernando Durate et al. [41] developed a new method that optimized the manipulability through a least square rational function approximation, to determine the joints position and also the chaos revealed by kinematic trajectory planning scheme while controlling redundant and hyper redundant manipulators. Lianfang Tian et al. [42] developed a novel genetic algorithm using a floating-point representation to search for optimal end effector trajectory for a redundant manipulator. Reza Fotouhi et al. [43] developed a two-phase trajectory planning for a two-link rigid manipulator. A new approach based on interval analysis was developed by Aurelio Piazzi and Antonio Visioli [44] to find the global minimum-jerk (MJ) trajectory of a robot manipulator within a joint space scheme using cubic splines. Galicki M and Friedrich-Schiller [45] developed a solution to the problem of tracking a prescribed geometric path by the end effector of a kinematically redundant manipulator at the control loop level. A novel collision avoidance algorithm based on a generalized potential field model of 3-D workspace was developed by Lin and Chuang to solve path planning problem of a high degree of freedom robot manipulators in 3D workspace [46].

1.3 New Notation Review

Symbolic notations make modeling of robotic configuration easier. The notation consists of essential parameters for a complete description of a mechanical system. With the help of this generalized symbolic notation, transformation matrices are generated, this in turn is used to model link shapes and to incorporate the constraint motions of joints. There are many notations like D-H, S-U and C-B notations that can be used for modeling robots, but they come with their own pros and cons. D-H notation require just four parameters to represent joint transformations, but complications arise while analyzing certain mechanisms. The reason is due to the fact that it uses relative joint motions due to its joint definition and also it fails when adjacent joint axes are not parallel or normal to each other. S-U on the other hand uses six constant joint parameters plus a variable part that contains same number of parameters as the degrees of freedom of the joint. C-B notation to some extend can be used to analyze most of the robotic configurations because it permits the determination of exact joint positions in space. It uses five parameters to define the transformation from one joint to other adjacent joint. But this fails in some cases as there is no parameter to take care of the y-axis translation. So the new notation removes this constraint by using up an extra variable to represent a translation parameter in the y-direction. This way it satisfies the requirement of the need of six degree of freedom, three rotational components and three translation components, required to define a body or point with respect to a reference, in space. This notation can be called as a modified or refined form of CB notation developed by Yih.

1.3.1 Advantages of this notation over CB Notation

Consider a robot configuration as shown in Figure 1.1, analysis of this configuration using CB notation fails as there is no additional parameter that defines translation along Y axis. This situation could be avoided using this new notation.

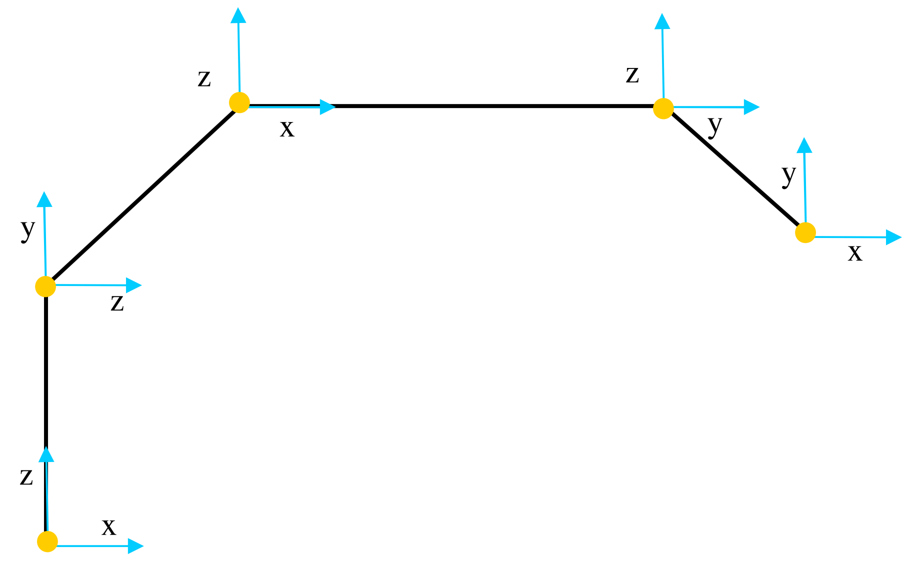


Figure 1.1 Shows an Arbitrary Configuration Where CB Notation Fails.

1.3.2 Notation and Transformation Matrix

This notation utilizes six parameters to define an adjacent joint, these are

θ - represents the rotational parameter around the principal Z-axis.

h - represents the translation along the principal Z-axis.

r - represents the translation along the X-axis.

b - represents the translation along the principal Y-axis.

α - represents the rotation around the principal X-axis.

β - represents the rotation around the principal Y-axis.

.

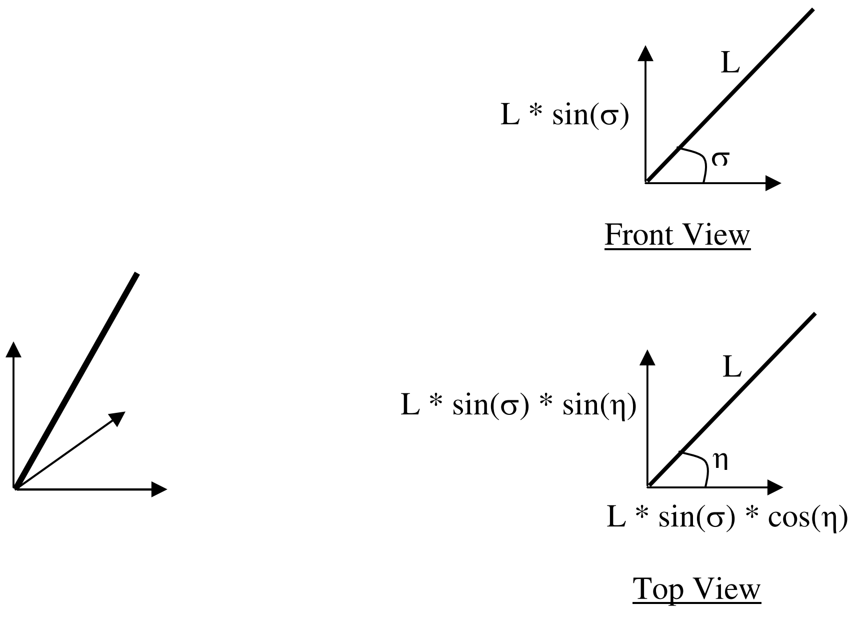


Figure 1.2 Representation of h, r and b Parameters.

Two parameters σ and η describe the orientation of the link in a 3D co-ordinate system. σ depicts the in-plane angle between the link and the coordinate system, while η depicts the out-of-plane angle made by the link with respect to the coordinate axes.

The basic homogeneous transformation matrix that is essential to determine the exact joint position in space is infact the product of individual transformation matrices containing the above defined parameters. Therefore

$$T(\theta, h, r, b, \alpha, \beta) = T(Z, \theta) * T(Z, h) * T(X, r) * T(Y, b) * T(X, \alpha) * T(Y, \beta)$$

where,

$$T(Z, \theta) = \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.1}$$

$$T(Z, h) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

$$T(X, r) = \begin{bmatrix} 1 & 0 & 0 & r \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

$$T(Y, b) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

$$T(X, \alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha & -s\alpha & 0 \\ 0 & s\alpha & c\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

$$T(Y, \beta) = \begin{bmatrix} c\beta & 0 & s\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s\beta & 0 & c\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.6}$$

$T(\theta, h, r, b, \alpha, \beta) =$

$$\begin{bmatrix} (c\theta c\beta - s\theta s\alpha s\beta) & -(s\theta c\alpha) & (c\theta s\beta + s\theta s\alpha c\beta) & (-bs\theta + rc\theta) \\ (s\theta c\beta + c\theta s\alpha s\beta) & (c\theta c\alpha) & (s\theta s\beta - c\theta s\alpha c\beta) & (bc\theta + rs\theta) \\ -(c\alpha s\beta) & s\alpha & (c\alpha c\beta) & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.7)$$

This homogeneous transformation matrix that incorporates three rotational and three translation components is called the shape matrix. This transformation matrix describes the link shape and joint behavior. Hence this transformation matrix can be written in another form depicting the orientation part represented by the direction cosine matrix D_i and position vector represented by a vector P_i.

$$T_i(\theta_i, h_i, r_i, b_i, \alpha_i, \beta_i) = \begin{bmatrix} & & & P_x \\ & D_i & & P_y \\ & & & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.8)$$

1.3.3 Characteristic or Shape Matrices of Kinetic Pairs

A characteristic or shape matrix describes both the shape of the link and the behavior constrained by kinematic pairs. The five basic kinematic pairs are Revolute, Prismatic, Cylindrical, Helical, and Spherical joints.

1.3.3.1 Revolute Pair

A revolute pair, Figure 1.3, has an axial rotation θ_i along the axis. Its characteristic matrix is given in equation 1.9.

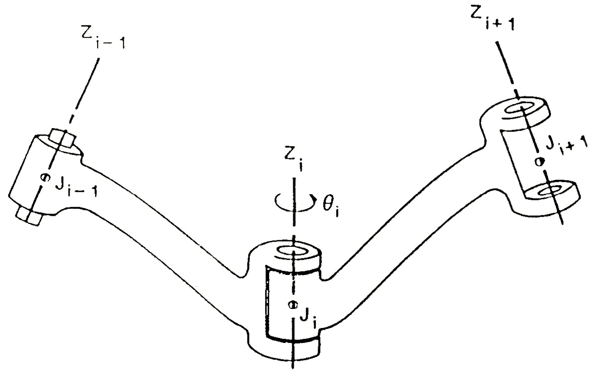


Figure 1.3 The Revolute Pair (R).

$$T_i(\theta_i, h_i, r_i, b_i, \alpha_i, \beta_i) = \begin{bmatrix} \multicolumn{3}{c}{D(\theta)_i} & P(\theta)_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.9)$$

where θ_i is the only design variable and varies within the range of joint constraint.

1.3.3.2 Prismatic Pair

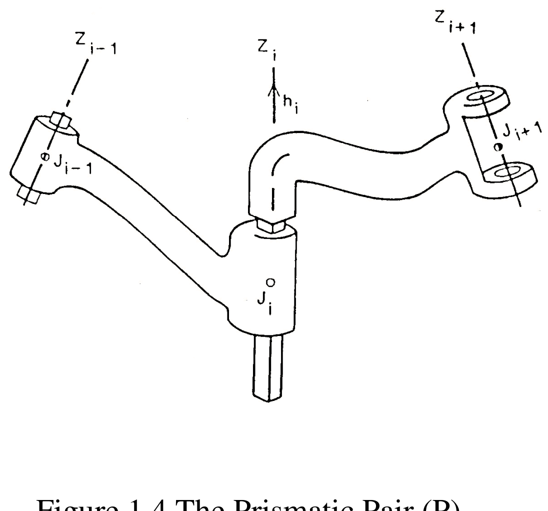


Figure 1.4 The Prismatic Pair (P).

A prismatic joint, Figure 1.4, allows translation along its generatix only. The characteristic matrix is as given as

$$T_i(\theta_i, h_i, r_i, b_i, \alpha_i, \beta_i) = \begin{bmatrix} & D_i & & P(h)_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.10}$$

where, h_i is the only design variable and this varies within the range of joint constraint and θ_i is constant.

1.3.3.3 Cylindrical Pair

This type of joint is characterized by a rotation θ_i about the cylinder axis, and a translation h_i along the axis. So this has two degrees of freedom (θ_i, h_i). The cylindrical joint is illustrated in Fig 1.5.

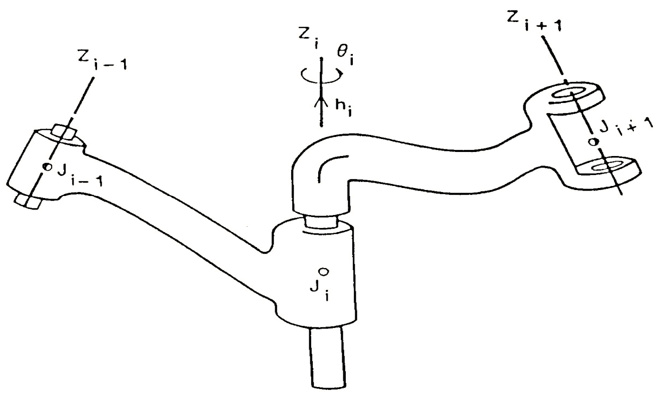


Figure 1.5 The Cylindrical Pair (C).

The characteristic matrix is given by

$$T_i(\theta_i, h_i, r_i, b_i, \alpha_i, \beta_i) = \begin{bmatrix} & D(\theta)_i & & P(\theta \cap h)_i \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.11)$$

1.3.3.4 Helical Pair

The helical joint produces a helical movement, and the two joint variables, θ_i and h_i, can be correlated by its lead of the screw L_i, with θ_i in radians. The helical joint is shown in Figure 1.6.

$$\theta_i = f(h_i) = h_i / L_i \text{ or } h_i = f(\theta_i) = \theta_i * L_i \quad (1.12)$$

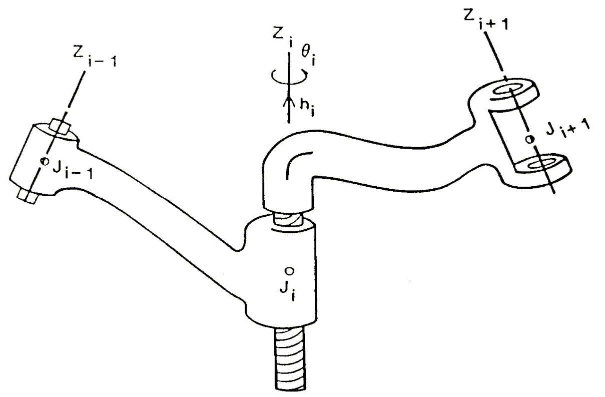


Figure 1.6 The Helical Pair (H).

The characteristic matrix for a helical joint is thus given as

$$T_i(\theta_i, h_i, r_i, b_i, \alpha_i, \beta_i) = \begin{bmatrix} & D(\theta)_i & & P(\theta \cup h)_i \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.13)$$

Here the number of design variable is reduced to 1.

1.3.3.5 Spherical Pair

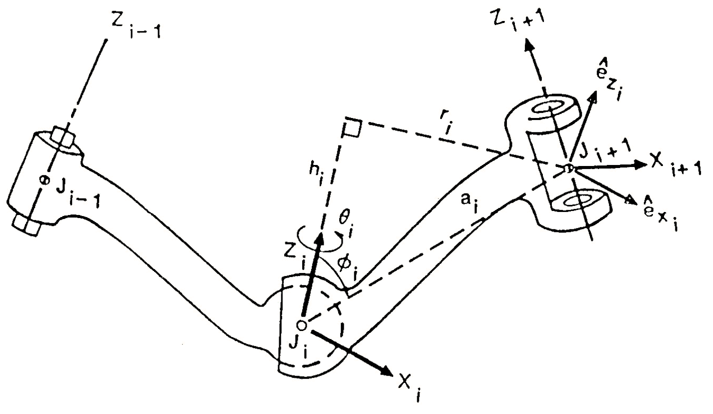


Figure 1.7 The Spherical Pair (S).

The definition for joint axis Z_i can be infinite because the spherical joint is free to rotate in all the directions. In this case α_{i-1} and β_{i-1} are set equal to zero. From Figure 1.8, the relative link length, a_i, between J_i and J_{i+1} remains constant all times. However angle ϕ_i is measured from Z_i to a_i but the parameters h_i and r_i vary with angle ϕ_i. $h_i = a_i c\phi_i$ and $r_i = a_i s\phi_i$. A spherical joint may have three degrees of freedom and each rotates about the Cartesian coordinate axis indicated by $(\psi_{xi}, \psi_{yi}, \psi_{zi})$. By definition, ψ_{zi} is equivalent to θ_i. Irrespective of the order of rotation about the X_i and Y_i axes,

$$T_i(\theta_i, h_i, r_i, b_i, \alpha_i, \beta_i) = \begin{bmatrix} & & & a_i s\phi_i c\theta_i \\ & D(\theta)_i & & a_i s\phi_i s\theta_i \\ & & & a_i c\phi_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.14)$$

where element $c\psi_{xi}\, c\psi_{yi}$ is the direction cosine between Z_i and a_i. Since the angle between Z_i and a_i was previously defined as ϕ_i, then $\phi_i = c^{-1}(c\psi_{xi}\, c\psi_{yi})$.

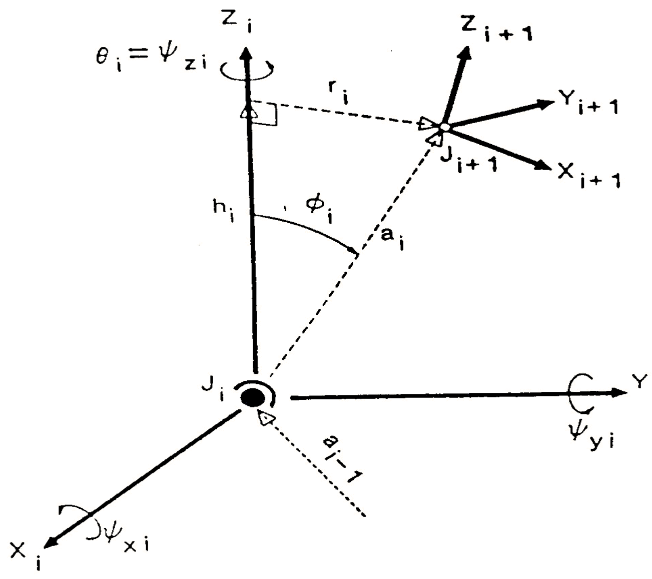


Figure 1.8 Joint Parameters of a Spherical Pair (S).

The characteristic matrix is as given below.

$$T_i(\theta_i, h_i, r_i, b_i, \alpha_i, \beta_i) = \begin{bmatrix} & D(\theta)_i & & P(\theta \cap \phi)_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.15}$$

1.3.4 Examples Describing the Generation of Transformation Matrices

Consider three revolute joints i, i+1 and i+2, the parameters of i^{th} joint are determined in the following way. The transformation is generated by multiplying the sequence of individual transformation component matrices in the right order.

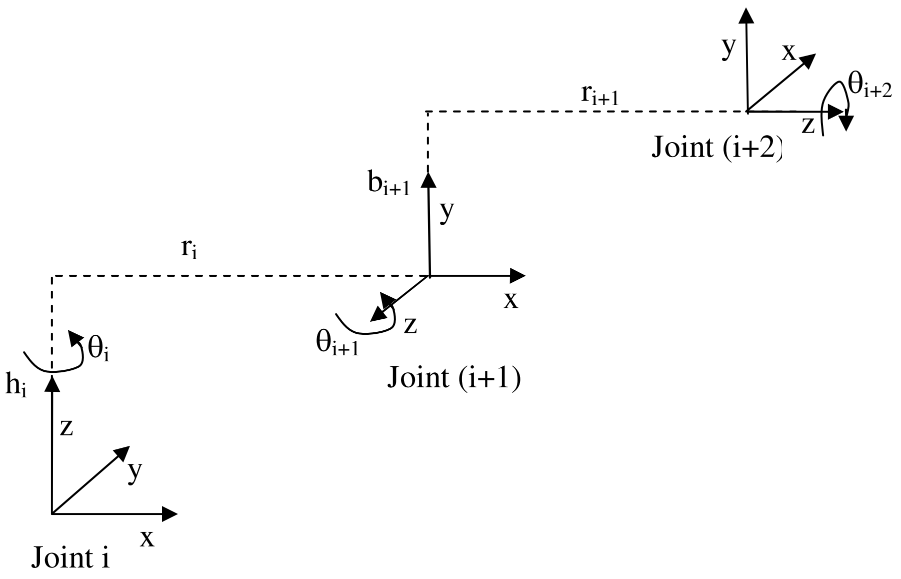


Figure 1.9 An Example Describing the Generation of Transformation Matrix.

$\theta = \theta_i$ is the principle parameter when a revolute joint is considered.

$h = h_i$ is the translation along the Z direction.

$r = r_i$ is the translation along the X direction.

$b = 0$, as there is no need for any translation along Y direction.

$\alpha = 90°$, compare the local co-ordinate axes of joints i and i+1. For these two axes to match, a rotation of +90° around ith joint's X axis in counter clockwise direction is required. Hence the value of α is equal to +90°.

$\beta = 0°$, no need of any rotation around Y axis.

After this sequence of transformations, both i and i+1 joints match well. Now consider joints i+1 and i+2, the parameter values are as follows,

$\theta = \theta_{i+1}$ is a parameter when a revolute joint is considered.

$h = 0$, as there is no need for translation along the Z direction.

$r = r_{i+1}$ is the translation along the X direction.

$b = b_{i+1}$, as there is no need for any translation along Y direction.

$\alpha = 0°$, no need of any rotation around X axis.

$\beta = 90°$, to match the local co-ordinates of i+1 and i+2 joints, a rotation of +90° is necessary around Y axis. After this sequence of transformations, both i and i+1 joints match well.

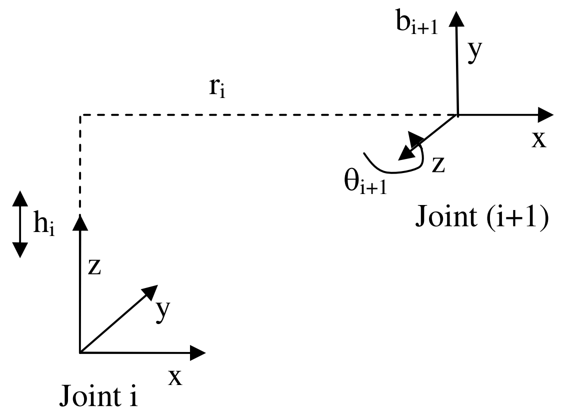


Figure 1.10 Example 2, Describing the Generation of Transformation Matrix.

Here joint i is a prismatic joint and i+1 is a revolute joint. For a prismatic joint, h is a principle parameter, as this defines the limit of the joint. The values of other parameters are as follows:

$\theta = 0$, as it is a prismatic joint.

$h = h_i$ is the principle parameter that describes the translation along the Z direction.

$r = r_i$ is the translation along the X direction.

b = 0, as there is no need for any translation along Y direction.

$\alpha = 90°$, compare the local co-ordinate axes of joints i and i+1. For these two axes to match, a rotation of +90° around ith joint's X axis in counter clockwise direction is required. Hence the value of α is equal to +90°.

$\beta = 0°$, no need of any rotation around Y axis.

In this dissertation only revolute and prismatic joints are considered for validating the results.

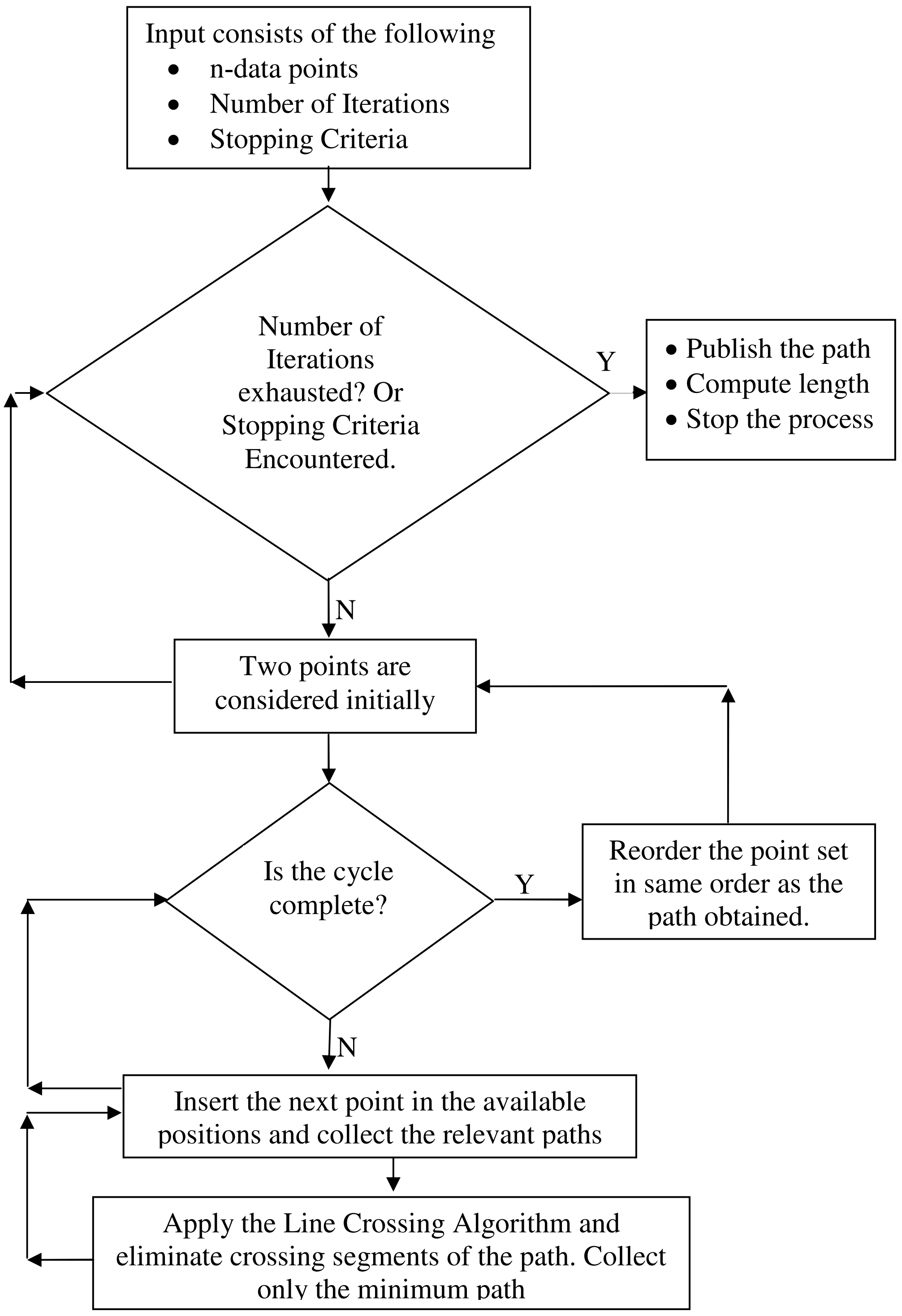
CHAPTER 2

TOOL PATH OPTIMIZATION

2.1 Introduction

Optimization of the tool path traversed by the robotic manipulator is required to achieve cost effectiveness, increased productivity and efficiency. This is similar to a symmetric traveling salesman problem. In general, The Traveling Salesman Problem (TSP) is a deceptively simple combinatorial optimization problem. The problem is about a salesman who wants to visit a number of cities cyclically. The conditions being, he has to visit each city once and needs to get back to the start city at the end of the tour. The hard part is due to the fact that if one considers n cities, then one has to probably determine them systematically and finally find the minimum path length. The process of finding the minimum path length can be accomplished easily if the points are grouped in a definite order. The key to finding the solution lies in this process. To compute the optimized path, a new algorithm called Insertion & Reordering Algorithm is used. In order to expedite the process of finding the optimized path, this algorithm is coupled with piece-wise optimizer, where in the initial path is generated by Insertion & Reordering technique and later, each of the segment is checked to see if its optimized or further more reordering is necessary. The process is iterated till the minimum value stabilizes.

2.2 Insertion and Reordering Algorithm



Input consists of the following
• n-data points
• Number of Iterations
• Stopping Criteria
Number of Iterations exhausted? Or Stopping Criteria Encountered.
Y
• Publish the path
• Compute length
• Stop the process
N
Two points are considered initially
Is the cycle complete?
Y
Reorder the point set in same order as the path obtained.
N
Insert the next point in the available positions and collect the relevant paths
Apply the Line Crossing Algorithm and eliminate crossing segments of the path. Collect only the minimum path

Figure 2.1 Flow-chart Portraying Insertion & Reordering Algorithm.

This algorithm consists of two parts, the insertion part which is mainly to find the best non-intersecting path and the reordering part of stacking the points in the same order as the best path obtained from the insertion step.

Step 1: Two points out of the n available points is considered. The shortest path between two points will be the straight line connecting them.

Step 2: The third point is considered and could be inserted in any of the available three positions, 3-1-2, 1-3-2, 1-2-3.

Step 3: The line crossing algorithm is applied to the paths in the set to check if any segment of the path intersect. All such paths whose segments intersect are rejected.

Step 4: Among the available paths in the result set, the one with the least length is selected.

Step 5: The steps 2 to 4 is repeated till all the n points are exhausted. This completes one cycle.

Then the entire process is repeated for number of iterations till the value of the path length is stabilized.

2.3 Example Showing the Steps

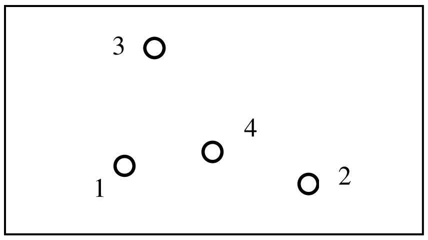


Figure 2.2 Initial Data-set.

Consider a simple example with four nodes as shown in figure 2.2. The objective is to find the minimum path length connecting these four points. Let us consider just two points initially and continue the process till an optimized solution is obtained.

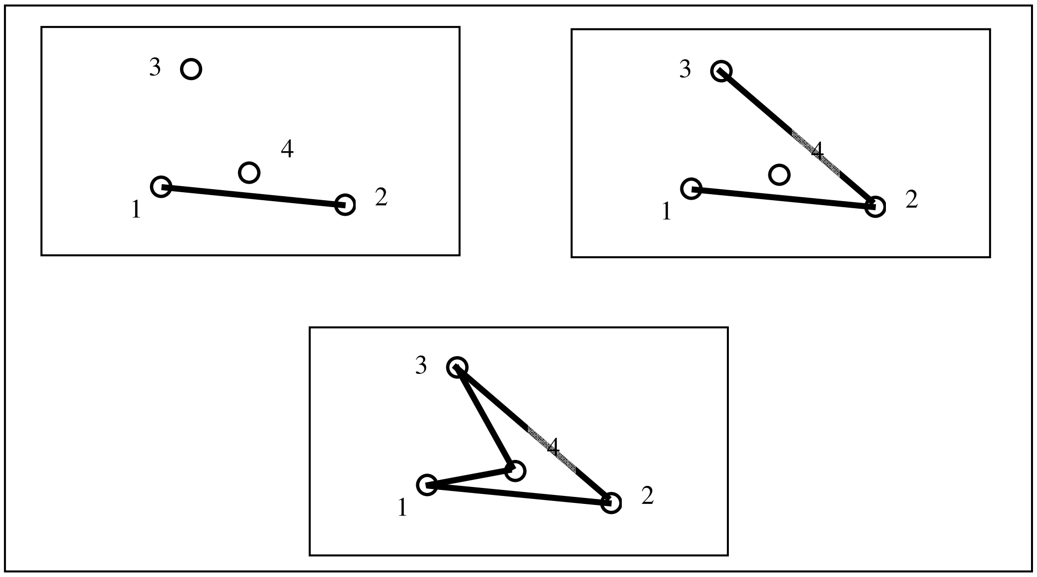


Figure 2.3 Subsequent Steps.

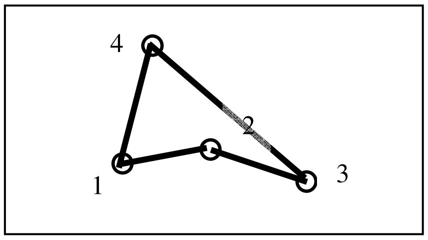


Figure 2.4 Reordering and Insertion Step.

This algorithm may take a longer time if the number of nodes increases and hence using the algorithm for number of iterations makes it a hopeless candidate as it would eat up the time required to complete the process. Hence a variant of this algorithm called Piece-wise Insertion & Reordering Algorithm is developed.

2.4 Piece-wise Insertion & Reordering Algorithm

This is similar to the parent algorithm with a slight difference. Consider two points, the least distance between the two is a line connecting them. Now the third point can be inserted in just one position available between the first and second. Basically the start and end points are defined and rest of the points is fitted between them. The non-intersecting path with the least length is generated at the end of one cycle. The points are reordered based on this path and the entire process is iterated till a stable solution is obtained. Let us see an example of how this works,

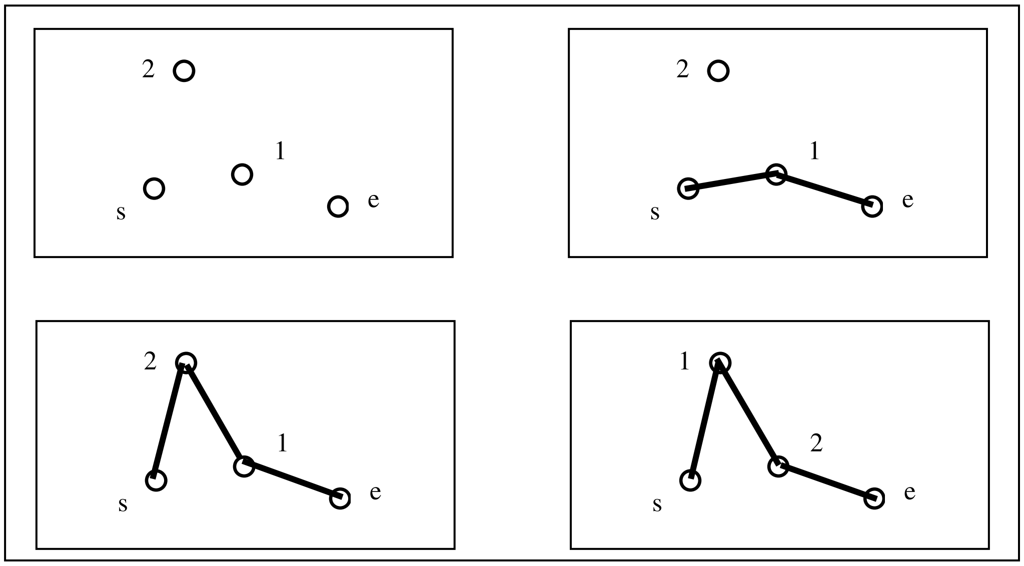


Figure 2.5 Step by Step Explanation of Piece-wise Insertion & Reordering Algorithm.

A software tool was designed and developed in Java. Java takes care of the number crunching and optimization evaluation, while Java3D is used for display purpose. This algorithm tested on standard data obtained from "The TSPLIB Symmetric Traveling Salesman Problem Instances" [75] produced satisfactory results. Results are displayed in subsequent pages.

2.5 Test Results

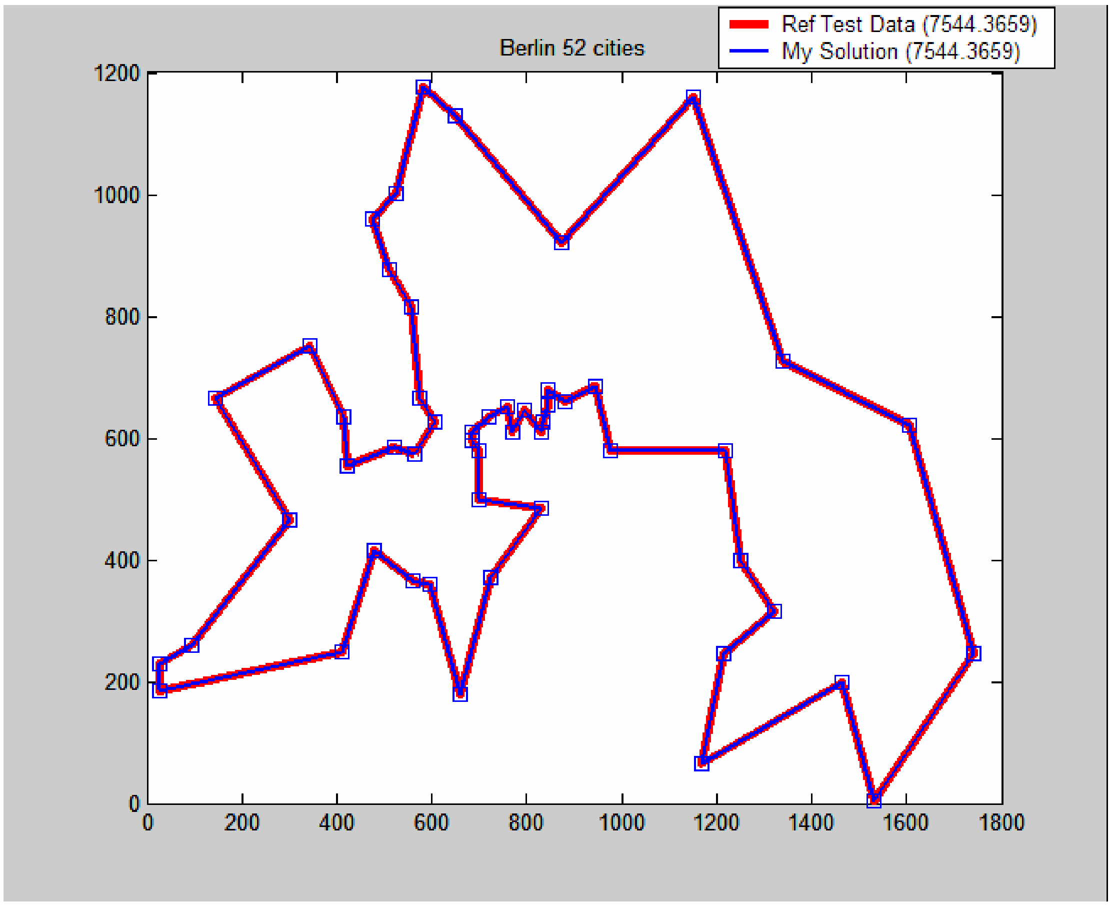


Figure 2.6 Test Case - Berlin with 52 Cities.

Matlab was used in this case to super-impose the results. For this test case, both the reference result and the result computed by the proposed algorithm were the same. The next two test cases, Djibouti & Western Sahara cities, were taken from Georgia Tech TSP repository.

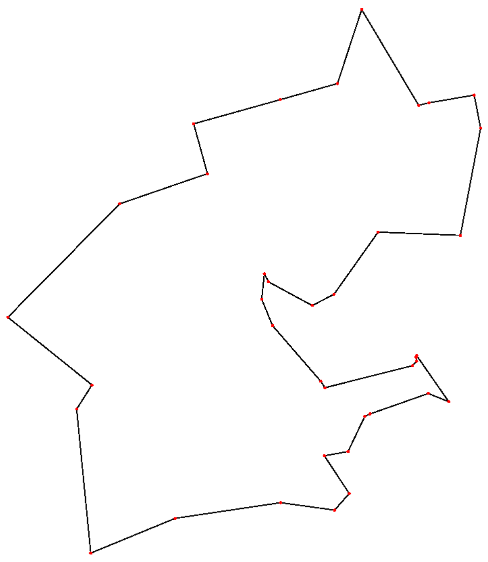
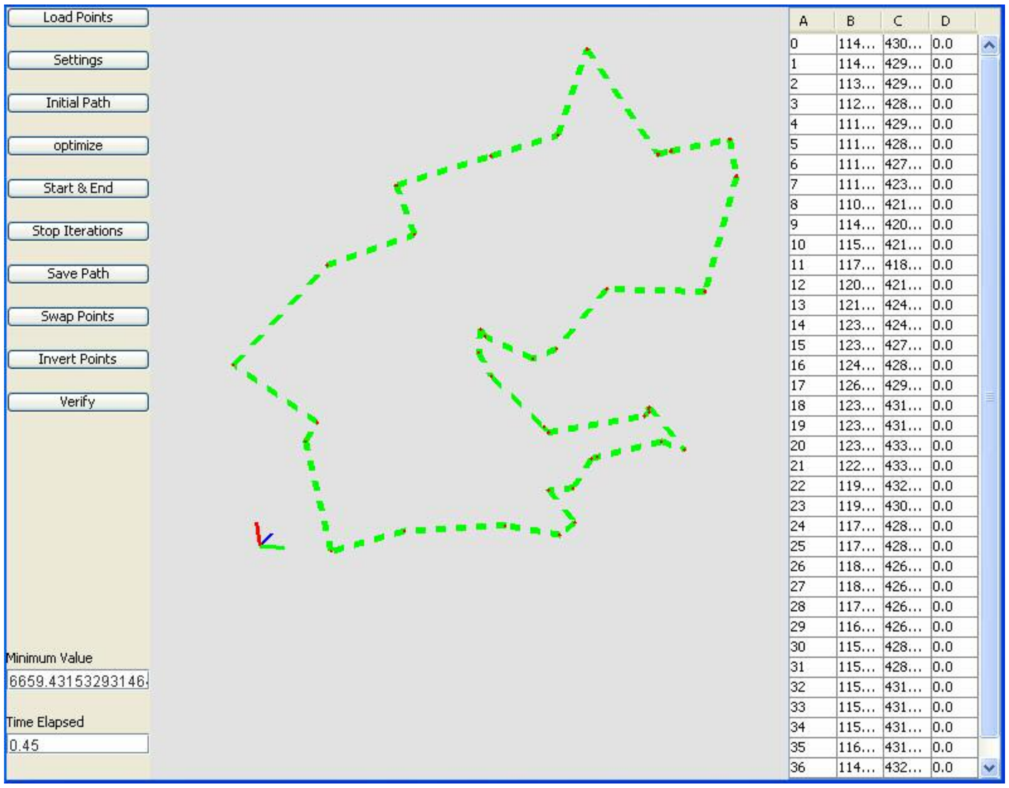


Figure 2.7 Reference Test Data for Djibouti Cities.



Minimum Value

6659.43153293146

Time Elapsed

0.45

A	B	C	D
0	114...	430...	0.0
1	114...	429...	0.0
2	113...	429...	0.0
3	112...	428...	0.0
4	111...	429...	0.0
5	111...	428...	0.0
6	111...	427...	0.0
7	111...	423...	0.0
8	110...	421...	0.0
9	114...	420...	0.0
10	115...	421...	0.0
11	117...	418...	0.0
12	120...	421...	0.0
13	121...	424...	0.0
14	123...	424...	0.0
15	123...	427...	0.0
16	124...	428...	0.0
17	126...	429...	0.0
18	123...	431...	0.0
19	123...	431...	0.0
20	123...	433...	0.0
21	122...	433...	0.0
22	119...	432...	0.0
23	119...	430...	0.0
24	117...	428...	0.0
25	117...	428...	0.0
26	118...	426...	0.0
27	118...	426...	0.0
28	117...	426...	0.0
29	116...	426...	0.0
30	115...	428...	0.0
31	115...	428...	0.0
32	115...	431...	0.0
33	115...	431...	0.0
34	115...	431...	0.0
35	116...	431...	0.0
36	114...	432...	0.0

Figure 2.8 Solution Generated by the Proposed Algorithm.

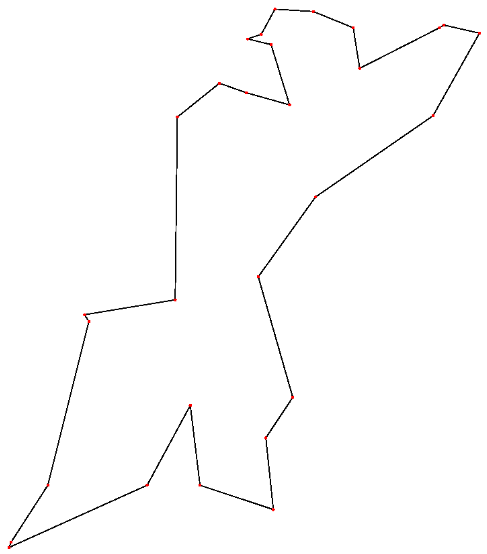
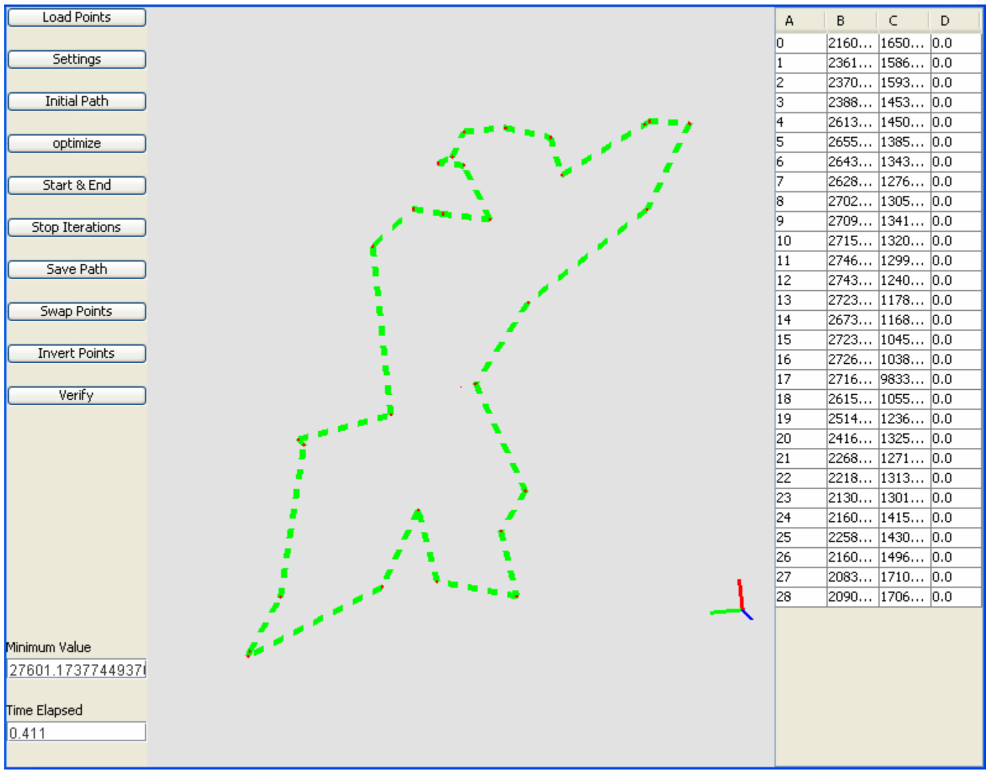


Figure 2.9 Reference Test Data for Western Sahara Cities.



Minimum Value

27601.1737744937

Time Elapsed

0.411

A	B	C	D
0	2160...	1650...	0.0
1	2361...	1586...	0.0
2	2370...	1593...	0.0
3	2388...	1453...	0.0
4	2613...	1450...	0.0
5	2655...	1385...	0.0
6	2643...	1343...	0.0
7	2628...	1276...	0.0
8	2702...	1305...	0.0
9	2709...	1341...	0.0
10	2715...	1320...	0.0
11	2746...	1299...	0.0
12	2743...	1240...	0.0
13	2723...	1178...	0.0
14	2673...	1168...	0.0
15	2723...	1045...	0.0
16	2726...	1038...	0.0
17	2716...	9833...	0.0
18	2615...	1055...	0.0
19	2514...	1236...	0.0
20	2416...	1325...	0.0
21	2268...	1271...	0.0
22	2218...	1313...	0.0
23	2130...	1301...	0.0
24	2160...	1415...	0.0
25	2258...	1430...	0.0
26	2160...	1496...	0.0
27	2083...	1710...	0.0
28	2090...	1706...	0.0

Figure 2.10 Solution Generated by the Proposed Algorithm.

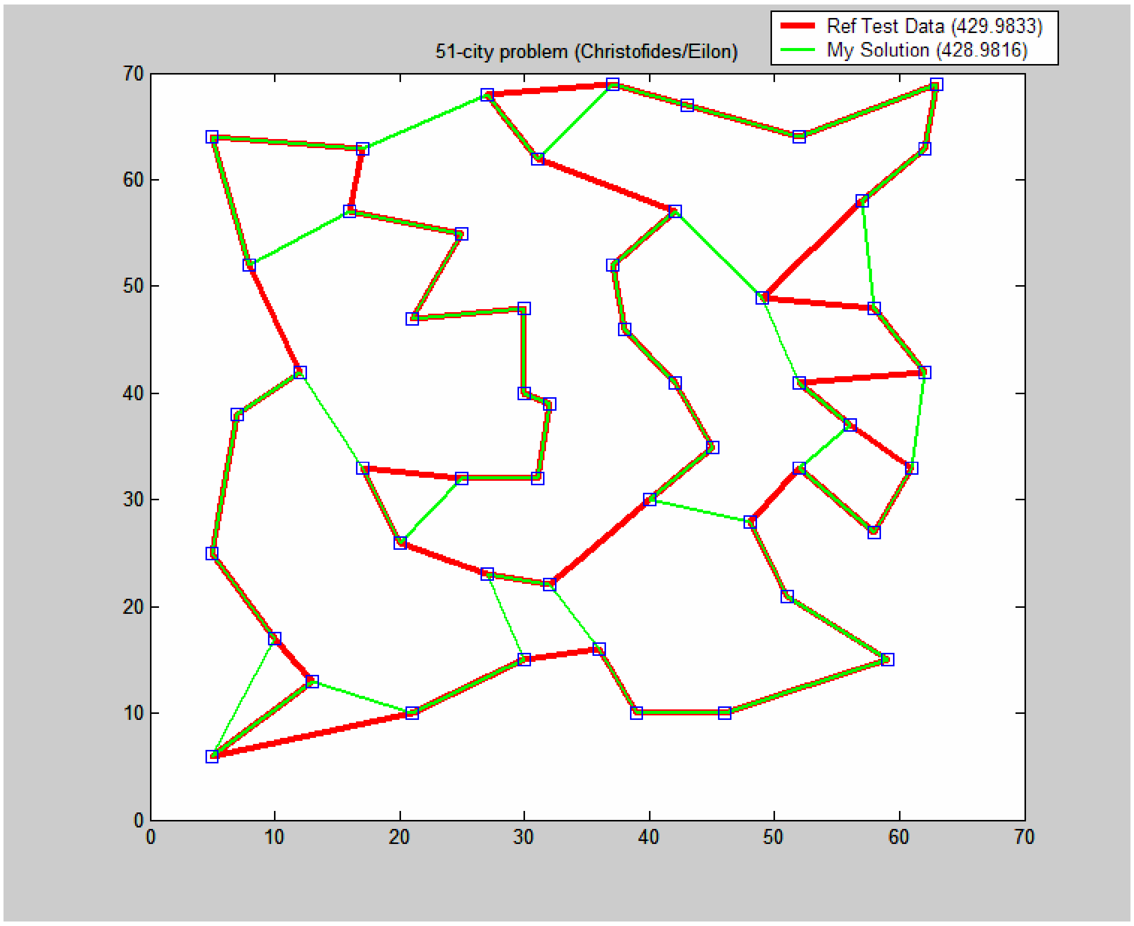


Figure 2.11 Test Case - 51 Cities.

The result shown was 429.9833 and is depicted by thick red line.

The proposed algorithm generated a path slightly different and the optimized path length was 428.9816 and is depicted by thin green line.

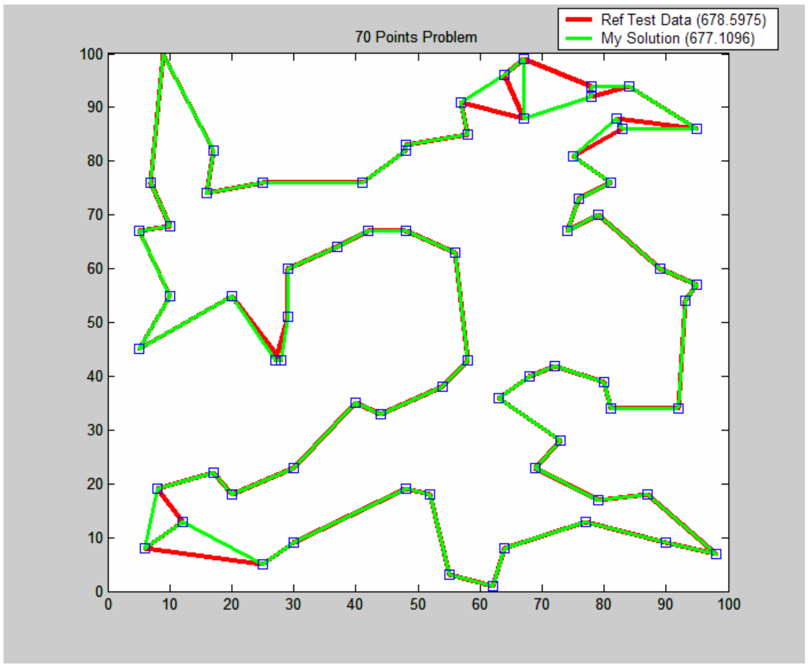


Figure 2.12 Test Case – 70 Point Problem.

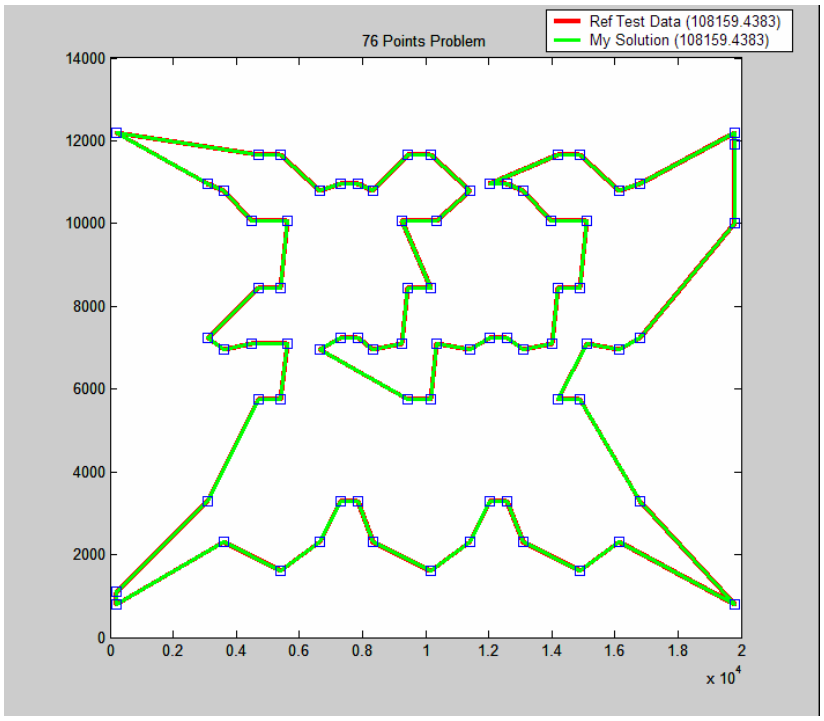


Figure 2.13 Test Case – 76 Point Problem.

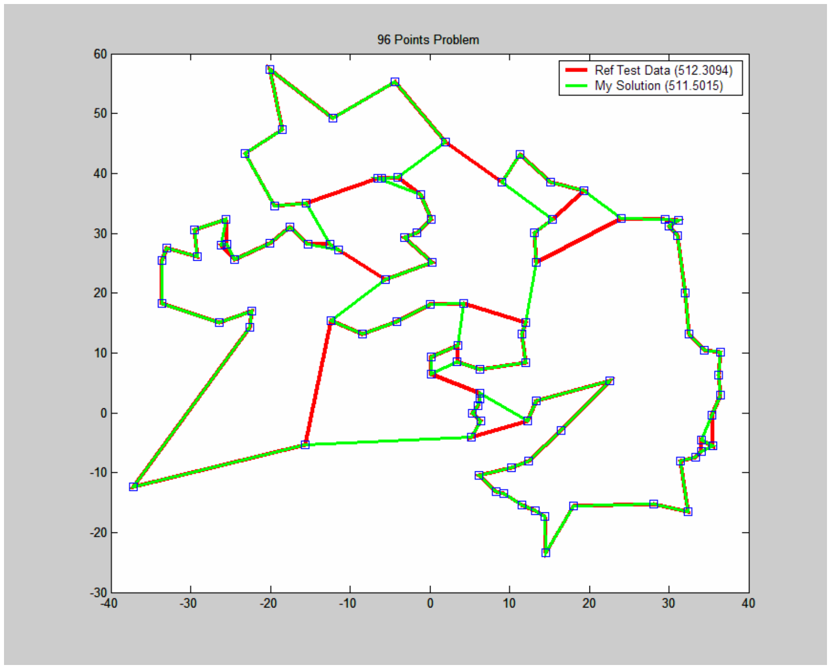


Figure 2.14 Test Case – 96 Point Problem.

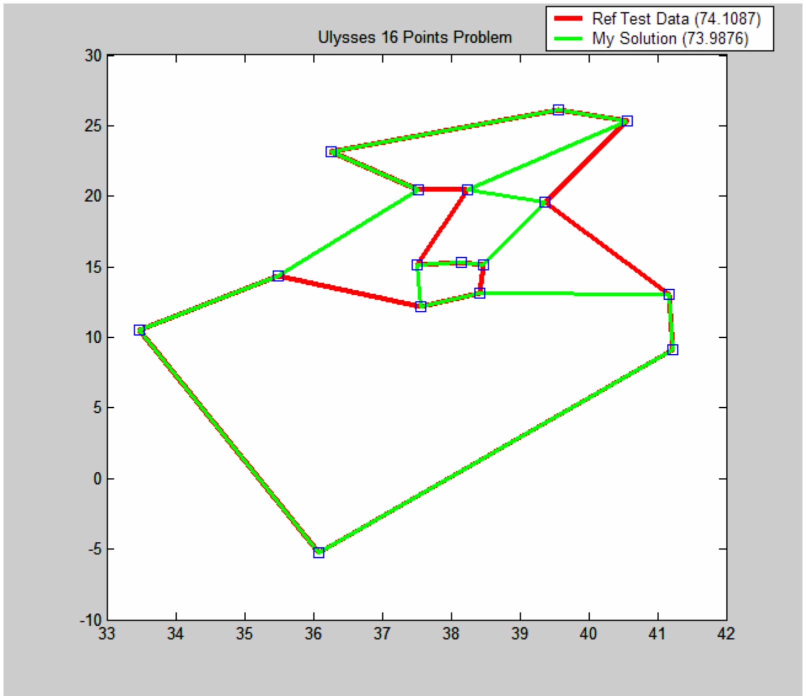


Figure 2.15 Test Case – Ulysses 16 Point Problem.

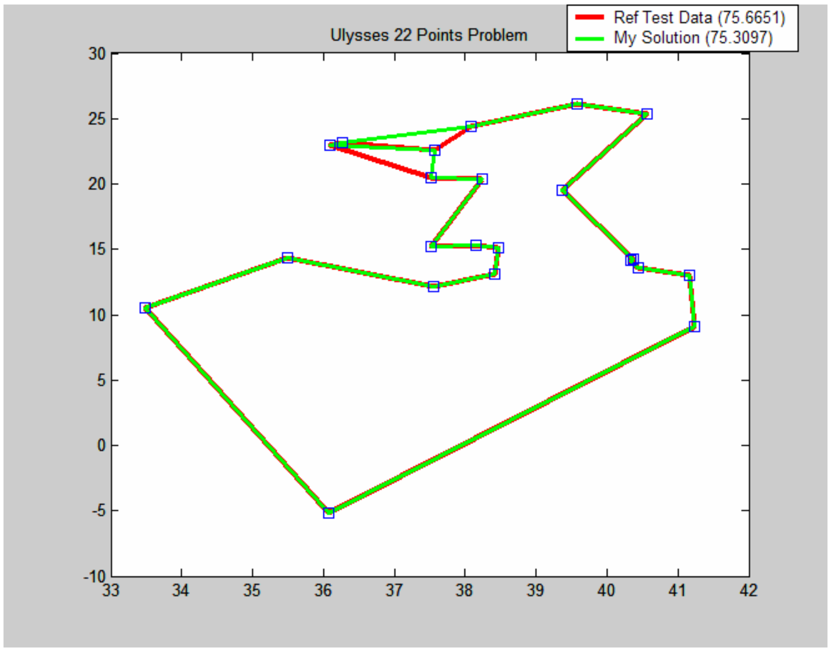


Figure 2.16 Test Case – Ulysses 22 Point Problem.

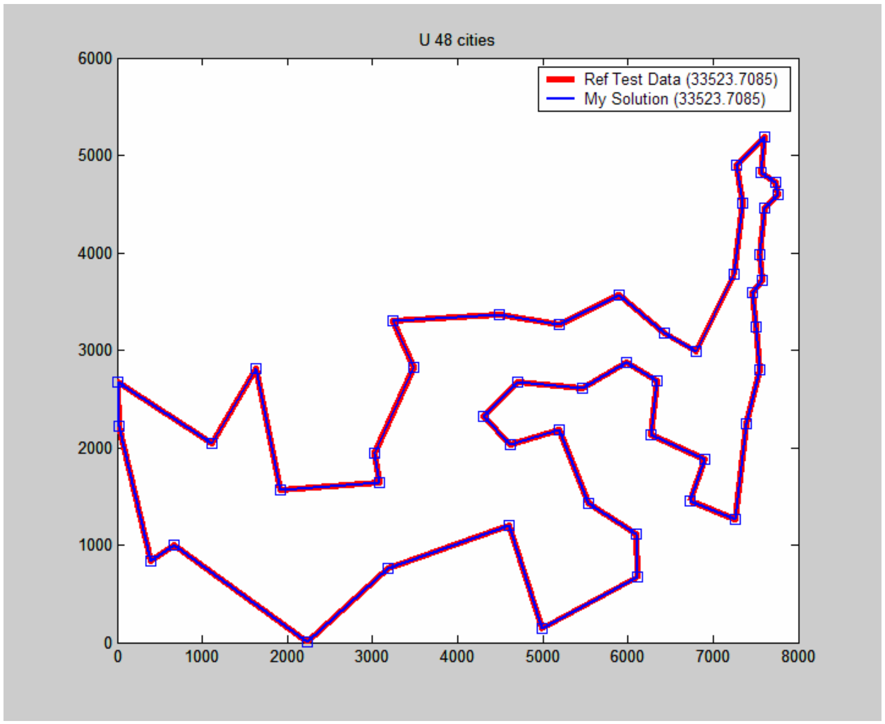
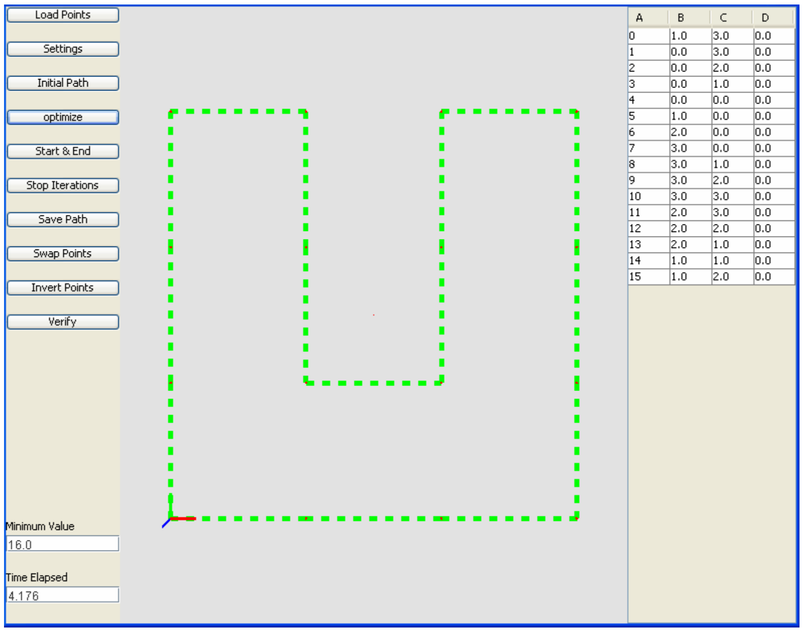


Figure 2.17 Test Case – US 48 Cities Problem.

2.6 Other Test Cases

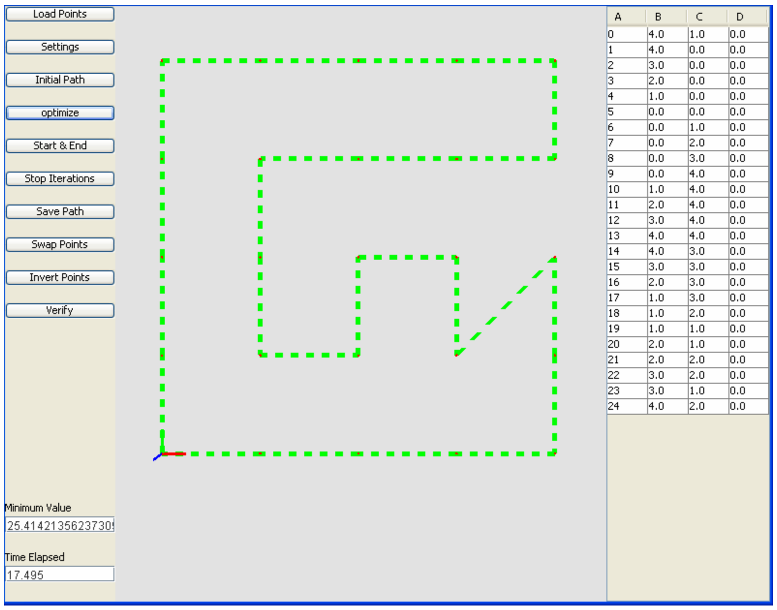


Minimum Value
16.0

Time Elapsed
4.176

A	B	C	D
0	1.0	3.0	0.0
1	0.0	3.0	0.0
2	0.0	2.0	0.0
3	0.0	1.0	0.0
4	0.0	0.0	0.0
5	1.0	0.0	0.0
6	2.0	0.0	0.0
7	3.0	0.0	0.0
8	3.0	1.0	0.0
9	3.0	2.0	0.0
10	3.0	3.0	0.0
11	2.0	3.0	0.0
12	2.0	2.0	0.0
13	2.0	1.0	0.0
14	1.0	1.0	0.0
15	1.0	2.0	0.0

Figure 2.18 Test Case – 4 X 4 Grid Problem.



Minimum Value
25.4142135623730

Time Elapsed
17.495

A	B	C	D
0	4.0	1.0	0.0
1	4.0	0.0	0.0
2	3.0	0.0	0.0
3	2.0	0.0	0.0
4	1.0	0.0	0.0
5	0.0	0.0	0.0
6	0.0	1.0	0.0
7	0.0	2.0	0.0
8	0.0	3.0	0.0
9	0.0	4.0	0.0
10	1.0	4.0	0.0
11	2.0	4.0	0.0
12	3.0	4.0	0.0
13	4.0	4.0	0.0
14	4.0	3.0	0.0
15	3.0	3.0	0.0
16	2.0	3.0	0.0
17	1.0	3.0	0.0
18	1.0	2.0	0.0
19	1.0	1.0	0.0
20	2.0	1.0	0.0
21	2.0	2.0	0.0
22	3.0	2.0	0.0
23	3.0	1.0	0.0
24	4.0	2.0	0.0

Figure 2.19 Test Case – 5 X 5 Grid Problem.

2.7 Optimizer - Software Tool

The algorithm was incorporated into a software application, Optimizer. Java is used for number crunching and for computing the path, while Java3D is used for display purpose. The functionalities and steps required to compute an optimized path is as follows.

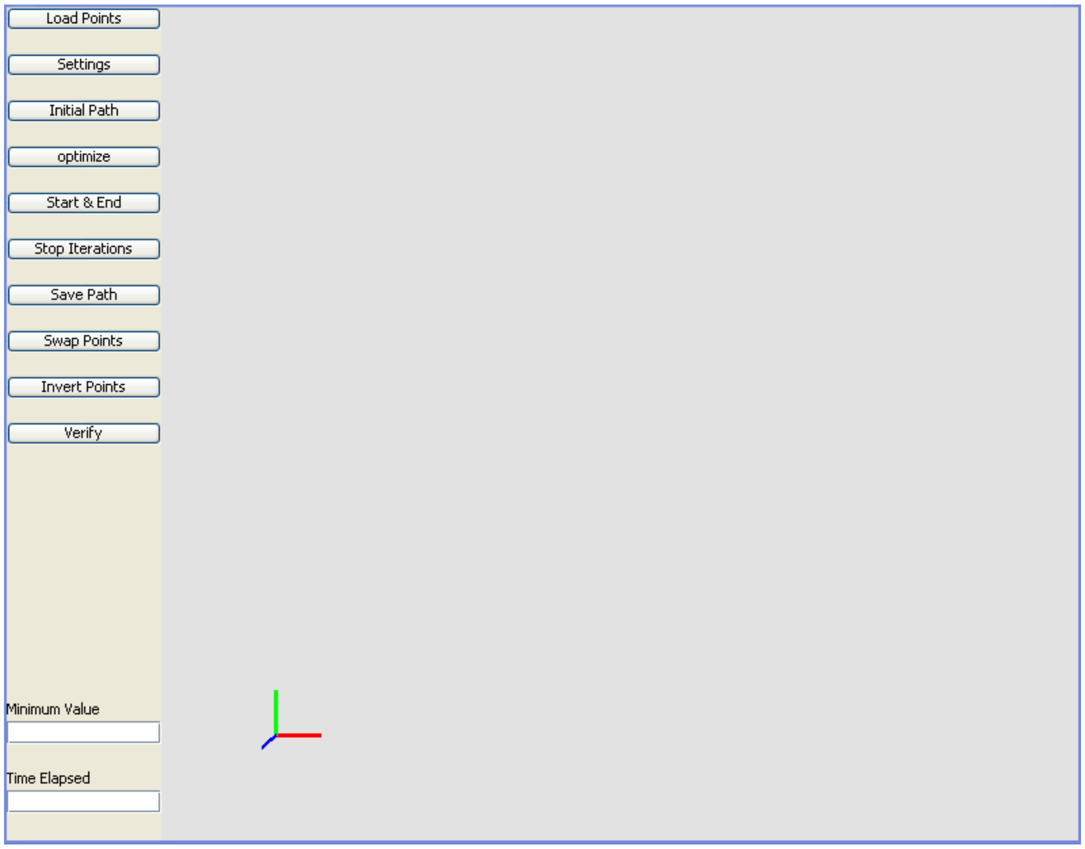


Figure 2.20 The Main Screen.

The left part of the application displays the various functionalities incorporated in the Optimizer. The data set can be loaded into the application using the “Load Points” feature. The “Settings” feature allows the user to set various parameters like number of iterations required, the number of points to be grouped while using the piece wise algorithm and the number of points to be reordered. The “Initial Path” feature uses the regular Insertion and Reordering Algorithm, this will ensure that the initial path or initial reordering has non-intersecting individual segments. “Optimize” features runs the Piece-wise algorithm and checks for the points grouping to have an optimized length. The algorithm will reorder the points and optimize if the segment considered isn’t having the minimum length for that part of the path. “Start & End” feature manually allows the user to fit the best path between a start and end point. The iterations can be stopped any moment by using the “Stop Iteration” feature. The generated optimized path could be saved for later use by using the “Save Path” feature. Sometimes it is essential that the points are interchanged so that the chances of arriving at the solution are brighter as there will be randomized arrangement of the points. “Invert Points” can be used to reverse the arrangement of the path, so that one can test the values of reverse and forward paths. The “Verify” feature can be used when one needs to manually check if the path generated between two points displayed is indeed the minimum path.

CHAPTER 3

DIRECT KINEMATICS

Kinematic analysis deals with position or displacement, velocity, and acceleration. Among these three, the most important of them being the position analysis. Roth first defined the performance evaluation of the manipulators by workspace. Roth and Shimano (1977) were the first to describe the reachable space or volume, which they called the workspace. Kumar & Waldron (1981) and Tsai & Soni (1983) defined working volume of a robot as the sum total of all locations that can be reached by the robot end effector [74].

The workspace of a robotic manipulator is defined as the set of all 3-dimensional points that the manipulator or end effector can be reached. The outer surface of the working volume is called the working envelope. There are two definitions of workspace, dexterous workspace which is the volume of space the robot end effector can reach with all orientations. At each point in this type of workspace, the end effector can be arbitrarily oriented at any point. Second, the reachable workspace which is the volume of space the robot can reach in at least one orientation. The study of robotic workspace is important in knowing before hand if the location to be machined or inspected is reachable, it's also useful in arranging the associated flexible manufacturing cell of a robot and assessing its efficiency in a manufacturing line.

The working volume should be extensive enough to perform the required work and must be devoid of voids. Voids are places inside the working envelope that are not part of the working volume or in short, those places in the workspace which cannot be reached by robot end effector. Velocity and acceleration spaces are obtained by determining velocities and accelerations over the entire workspace. Their applications are essential for the velocity and acceleration control of a robot following a specific contour.

3.1 Displacement Analysis

The general homogeneous characteristic matrix, T_i, for different kinematic lower pairs are given by

$$T(\theta, h, r, b, \alpha, \beta) =$$

$$\begin{bmatrix} (c\theta c\beta - s\theta s\alpha s\beta) & -(s\theta c\alpha) & (c\theta s\beta + s\theta s\alpha c\beta) & (-bs\theta + rc\theta) \\ (s\theta c\beta + c\theta s\alpha s\beta) & (c\theta c\alpha) & (s\theta s\beta - c\theta s\alpha c\beta) & (bc\theta + rs\theta) \\ -(c\alpha s\beta) & s\alpha & (c\alpha c\beta) & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The relative orientation and position of the n-th reference frame, with respect to the universal coordinates, are obtained by a sequence of multiplications.

$$H = \prod_{i=1}^{n} T_i = \begin{bmatrix} & D_i & & P_i \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

where T_i is the characteristic matrix of the ith joint. The resultant homogeneous matrix H contains the direction cosine matrix D and the position vector P to specify the orientation and position respectively. These two components of H are critical for the control of the manipulator. Also H is the resultant system matrix for the position analysis of a specific reference point of the robot manipulator. The reference point may be the end effector or the manipulator, or any other designated point in the system.

3.2 Velocity and Acceleration Analysis

Velocity analysis, where in the values of the velocity components can be computed by equation 3.2.

$$V = \frac{d}{dp}(H) = \frac{d}{dp}\left(\prod_{i=1}^{n} T_i\right) = \frac{d}{dp}\left(\begin{bmatrix} & D_i & & P_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\right) \qquad (3.3)$$

where, p equals to θ_i if the joint under consideration is a revolute joint. In revolute joints, θ is a variable parameter. If a prismatic joint is considered then p equals h_i, as h is a variable parameter.

Similarly the acceleration analysis can be accomplished by taking the differentiation of V with respect to p. This p equals θ_i, if the joint under consideration is a revolute joint. If a prismatic joint is considered then p equals h_i.

$$A = \frac{d}{dp}(V) \qquad (3.4)$$

3.3 Algorithm for Workspace Generation

Once the joint local axes and the joint types are assigned, the algorithm generates individual characteristics matrix based on the different type of joint. Four curves are generated initially. The first curve is generated by moving the last joint n from its minimum position to maximum joint position, keeping all other joints at the minimum position. Second curve is generated by fixing joint n-1 at its maximum joint range and again n^{th} joint is moved from minimum to maximum position. Third and fourth curves are generated by fixing the n^{th} joint at minimum and maximum positions and moving joint n-1 around its range. At the end of this step, four curves are generated. The subsequent curves are generated using the same procedure till all the joints, except the base one, are expended. These collections of curves are then swept about the base to generate three dimensional work volume. If all the joints move in the same plane as in scara type robots, then only a surface is generated.

However this has to be noted that the workspace generated is based on the maximum reach of the links on the sagittal plane. While generating the workspace, all the joints that have motion in a plane other than the sagittal plane are fixed at any convenient position and not allowed to move. Thus in these cases, the workspace generated do not represent an actual true workspace.

The Workspace generating is incorporated into a java application MyRobot. The user needs to assign just the principal joint axis, i.e. the Z axis for each of the joints, internally the application assigns the other two axes. The axes are assigned in such a way that either α or β is set to zero. This way the sequence of matrix operation reduces to five individual transformation component matrices.

3.4 Results

Various configurations with all revolute joints, all prismatic joints and a combination of both were tested using MyRobot tool. The examples include Cincinnati

Milacron T3 Robot, Bendix AA/CNC Industrial Robot, KR 60 P/2 Robot from KUKA Roboter GmbH, UPJ Robot from Motoman Corporation, Denso's HS-E Series Robot, and Cartesian Robot.

3.4.1 Cincinnati Milacron T3 Robot

This robot is composed of six revolute joints, Figure 3.1, three for the base, shoulder, and elbow, and three for the wrist and hand. This robot was designed to do simple tasks like machine tending and medium duty material handling.

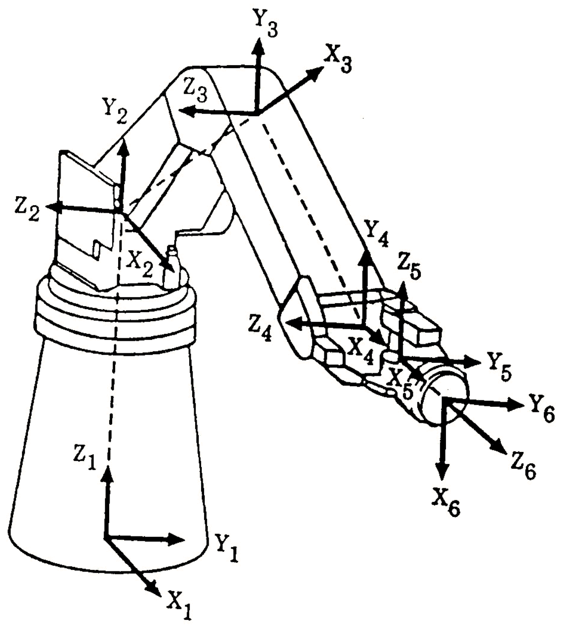
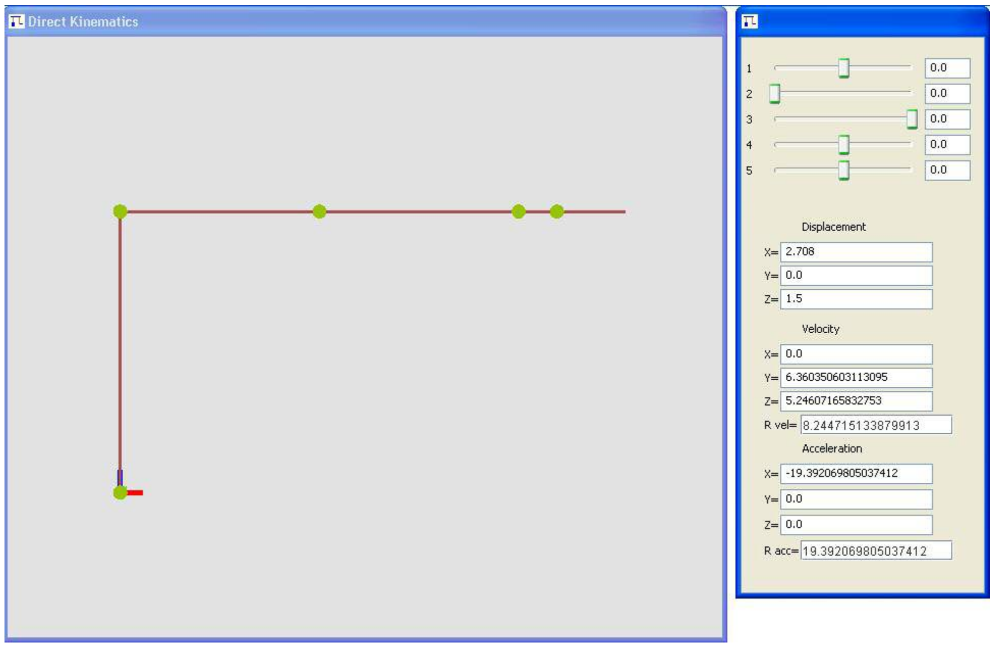


Figure 3.1 Cincinnati Milacron T3 Robot (6 Revolute Pairs).

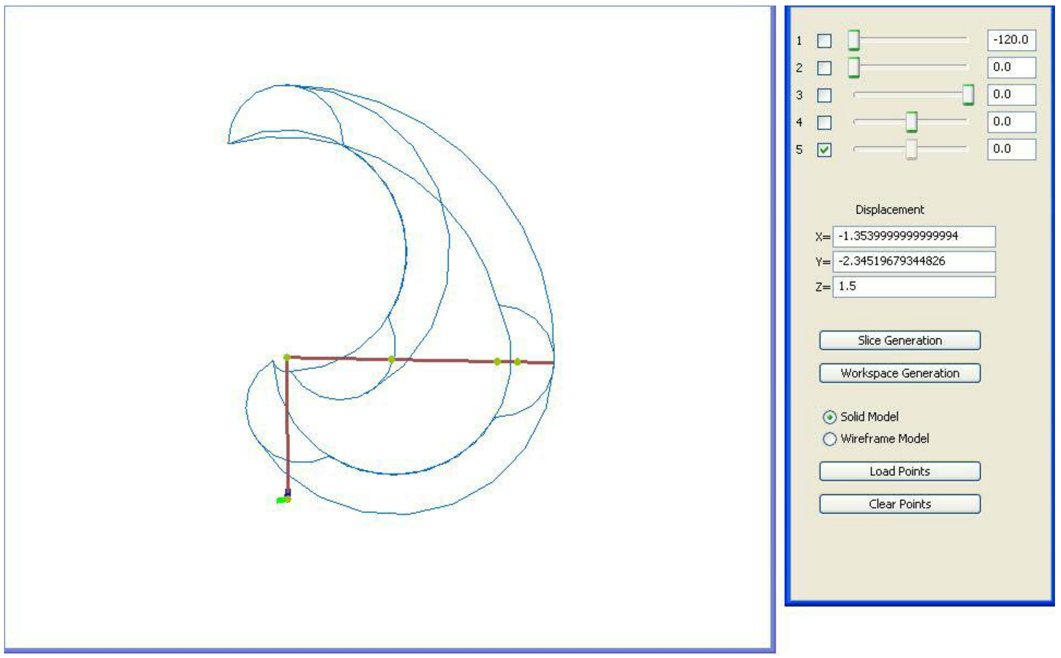
Table 3.1 Notation Table of Cincinnati Milacron T3 Robot

Joint	θ (deg)	h (m)	r (m)	b (m)	α (deg)	β (deg)	Variable
1 (R)	-120,120	1.5	0	0	90	0	θ_1
2 (R)	0,90	0	1.067	0	0	0	θ_2
3 (R)	-150,0	0	1.067	0	0	0	θ_3
4 (R)	-90,90	0	0.205	0	-90	0	θ_4
5 (R)	-90,90	0	0.369	0	0	90	θ_5
6 (R)	-135,135	0	0	0	0	0	θ_6



```
1  0.0
2  0.0
3  0.0
4  0.0
5  0.0

Displacement
X= 2.708
Y= 0.0
Z= 1.5

Velocity
X= 0.0
Y= 6.360350603113095
Z= 5.24607165832753
R vel= 8.244715133879913

Acceleration
X= -19.392069805037412
Y= 0.0
Z= 0.0
R acc= 19.392069805037412
```

Figure 3.2 Home Position and Direct Kinematic Parameters of Milacron T3 Robot



1 ☐ -120.0
2 ☐ 0.0
3 ☐ 0.0
4 ☐ 0.0
5 ✓ 0.0

Displacement
X= -1.3539999999999994
Y= -2.34519679344826
Z= 1.5

[● Solid Model]
○ Wireframe Model

Figure 3.3 Slice of Milacron T3 Robot Workspace taken at Sagittal Plane.

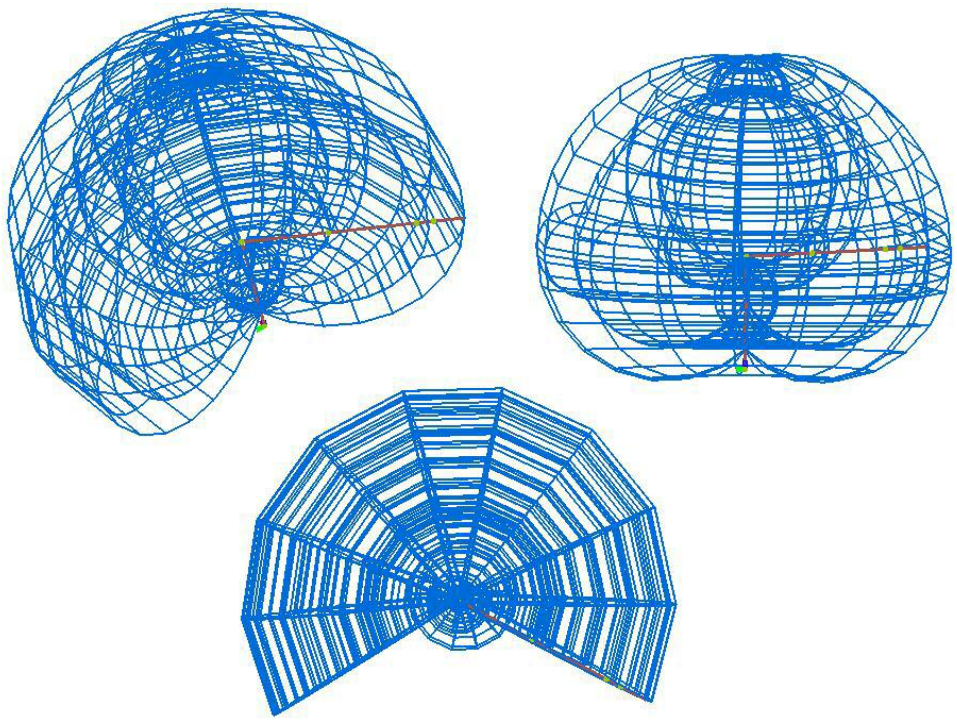


Figure 3.4 Workspace Generated by MyRobot.

3.4.2 KR 60 P/2 from KUKA Roboter GmbH

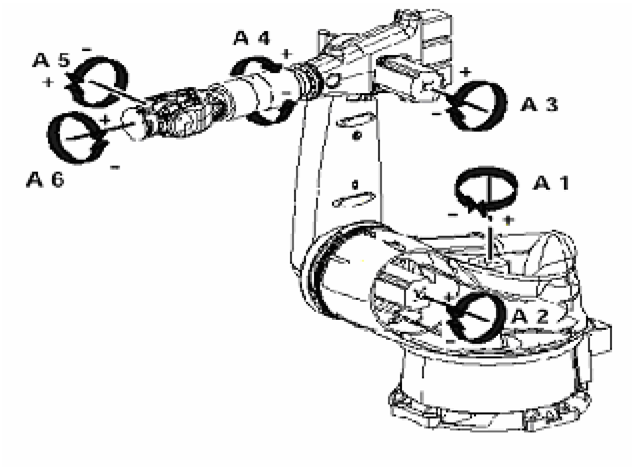
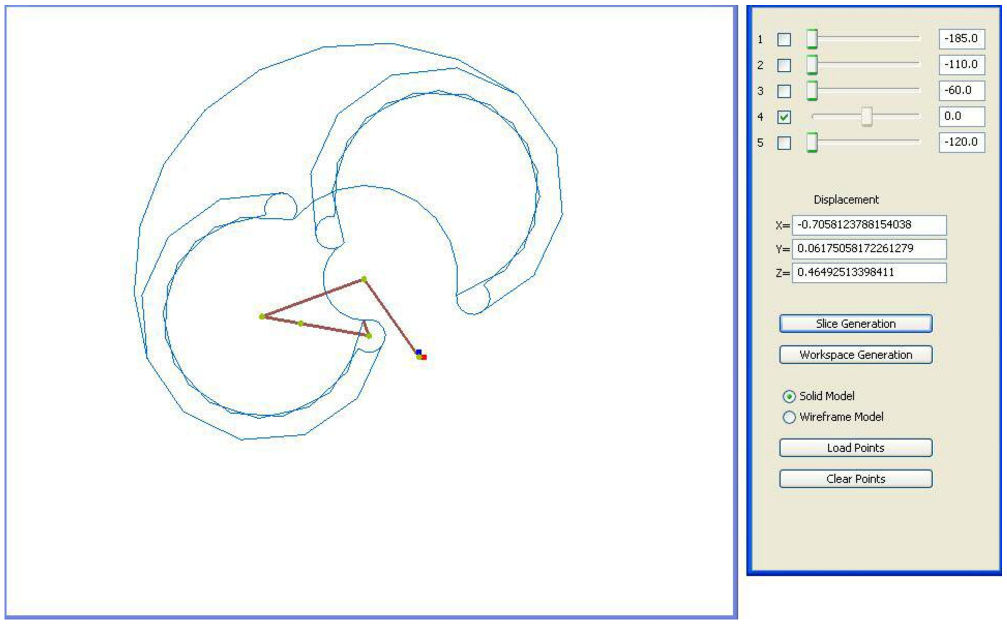


Figure 3.5 KR 60 P/2 Robot from KUKA Roboter GmbH.

The KR 60 P/2 from KUKA Roboter GmbH, Figure 3.4, is a six-axis industrial robot with articulated kinematics for all point-to-point and continuous path controlled tasks. Their main areas of application are press-to-press transfer, machine linking, handling, and palletizing.

Table 3.2 Notation Table of KR 60 P/2 Robot

Joint	θ (deg)	h (m)	r (m)	b (m)	α (deg)	β (deg)	Variable
1 (R)	-185,185	0.99	0.7	0	90	0	θ_1
2 (R)	-110,40	0	0	1.4	0	0	θ_2
3 (R)	-60,210	0	0.507	0	0	90	θ_3
4 (R)	-350,350	0.893	0	0	0	-90	θ_4
5 (R)	-120,120	0	0.21	0	0	90	θ_5
6 (R)	-350,350	0	0	0	0	0	θ_6



1	☐	-185.0
2	☐	-110.0
3	☐	-60.0
4	✓	0.0
5	☐	-120.0

Displacement

X= -0.7058123788154038

Y= 0.06175058172261279

Z= 0.46492513398411

[✓ Solid Model] Wireframe Model

Figure 3.6 KR 60 P/2 Workspace Slice at the Sagittal Plane.

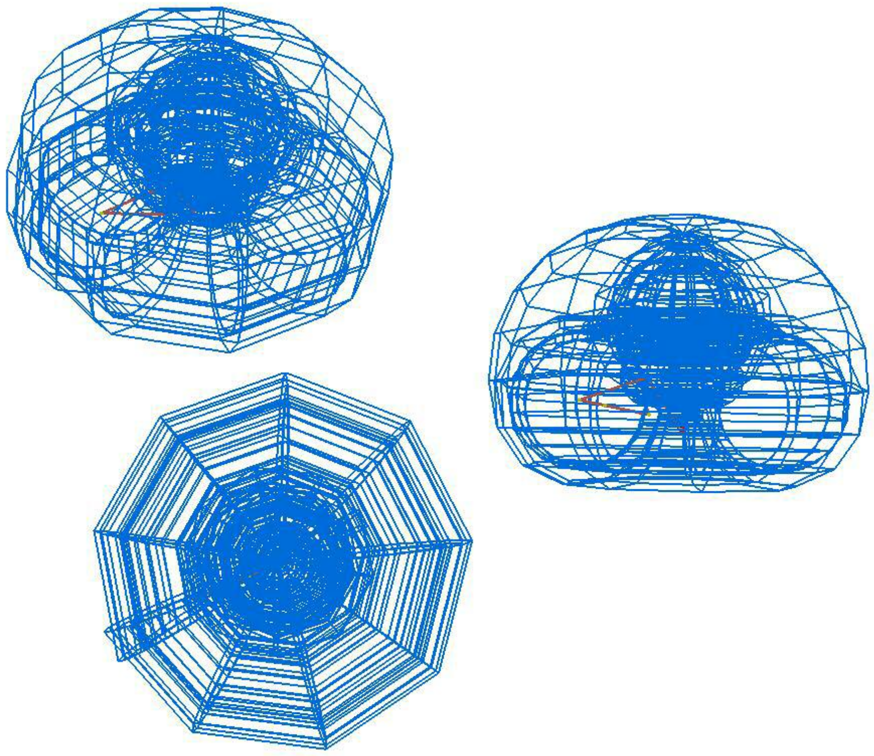


Figure 3.7 Workspace of KR 60 P/2.

3.4.3 Motoman UPJ Robot

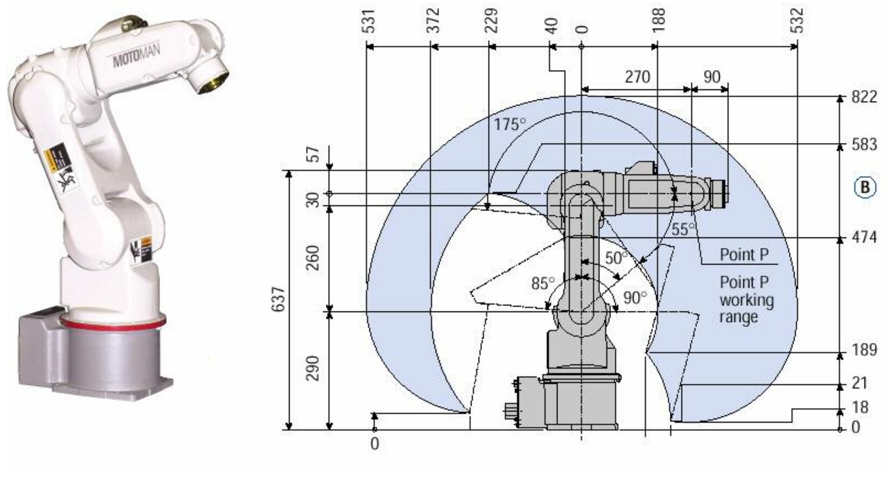
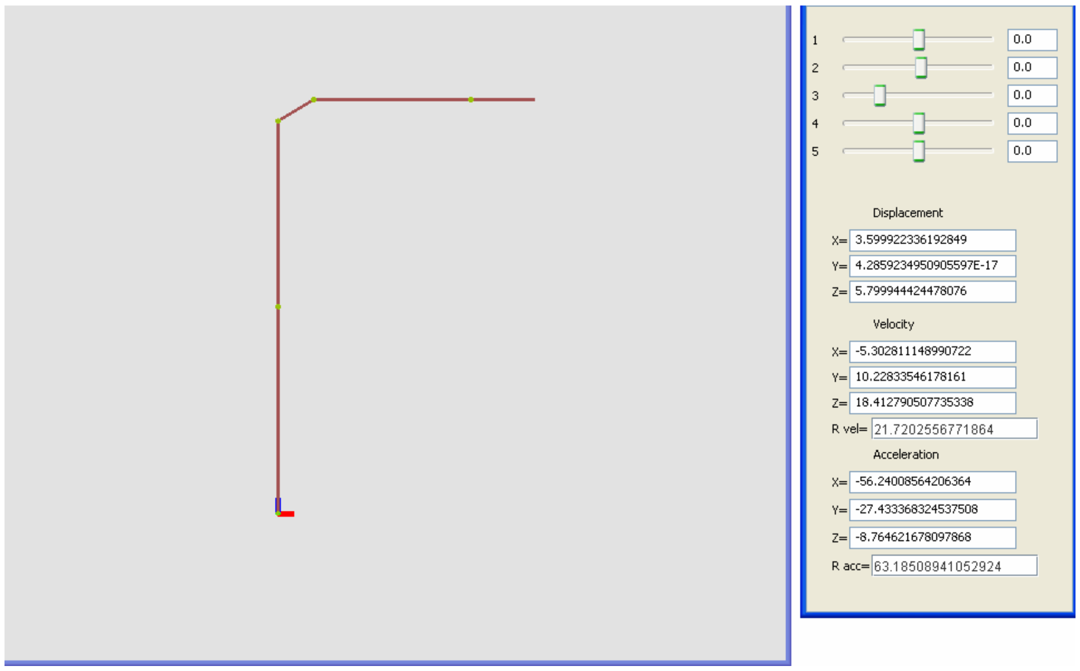


Figure 3.8 Motoman's UPJ Robot's Specifications.

The Motoman UPJ is a compact, high-speed robot that requires minimal installation space. The UPJ robot offers superior performance in small part handling, dispensing and assembly. It is also ideal for lab automation, inspection/testing, education, and research applications.

Table 3.3 Notation Table of UPJ Robot

Joint	θ (deg)	h (m)	r (m)	b (m)	α (deg)	β (deg)	Variable
1 (R)	-160,160	2.9	0	0	90	0	θ_1
2 (R)	-90,85	0	0	2.6	0	0	θ_2
3 (R)	-55,175	0	0.497	0.298	0	90	θ_3
4 (R)	-170,170	2.2	0	0	0	-90	θ_4
5 (R)	-120,120	0	0.9	0	0	90	θ_5
6 (R)	-360,360	0	0	0	0	0	θ_6



```
1  0.0
2  0.0
3  0.0
4  0.0
5  0.0

Displacement
X= 3.599922336192849
Y= 4.2859234950905597E-17
Z= 5.799944424478076

Velocity
X= -5.302811148990722
Y= 10.22833546178161
Z= 18.412790507735338
R vel= 21.7202556771864

Acceleration
X= -56.24008564206364
Y= -27.433368324537508
Z= -8.764621678097868
R acc= 63.18508941052924
```

Figure 3.9 UPJ Robot at Home Position.

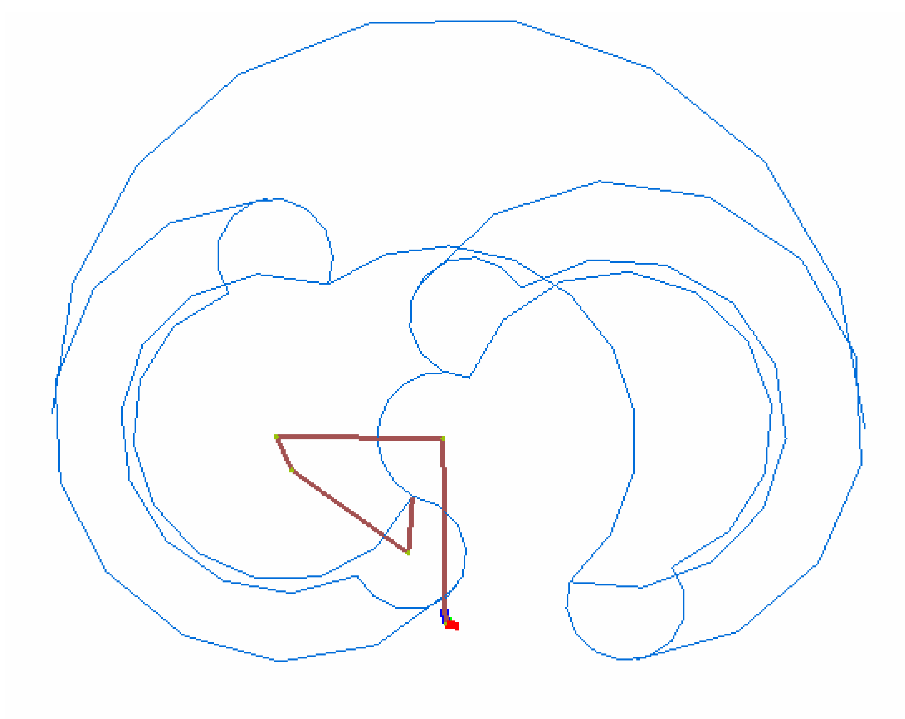


Figure 3.10 UPJ Robot's Workspace Slice.

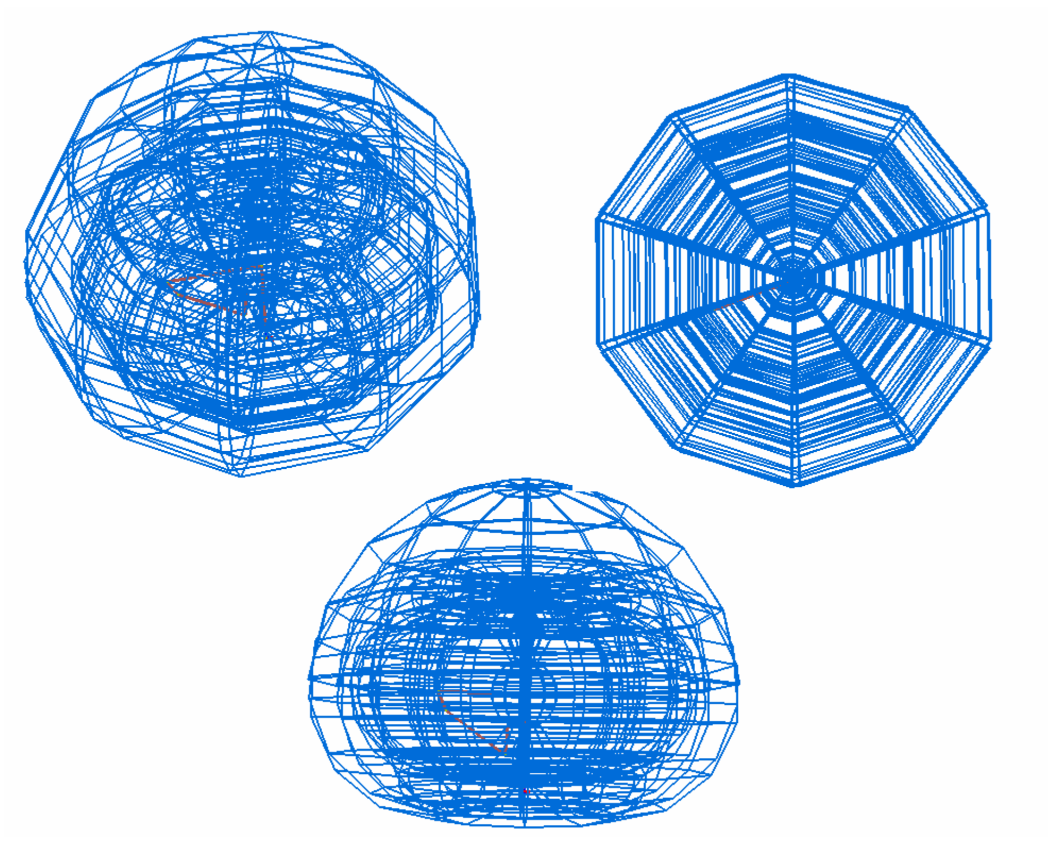


Figure 3.11 Workspace of Motoman's UPJ Robot.

3.4.4 Bendix AA/CNC Industrial Robot

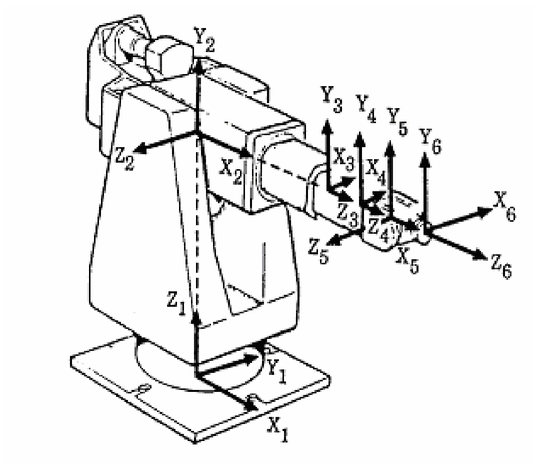
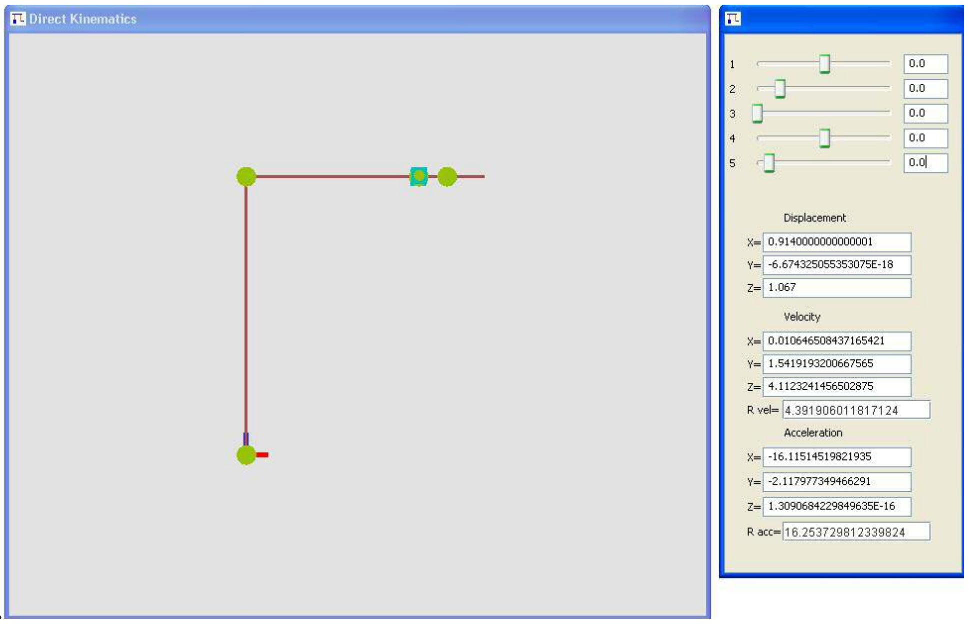


Figure 3.12 Bendix AA/CNC Industrial Robot.

The Bendix robot, Figure 3.11, is composed of six joints, revolute-revolute-prismatic joints (R-R-P) for the base, shoulder, and elbow, respectively; and three revolute joints for the wrist and hand.

Table 3.4 Notation Table of Bendix Robot

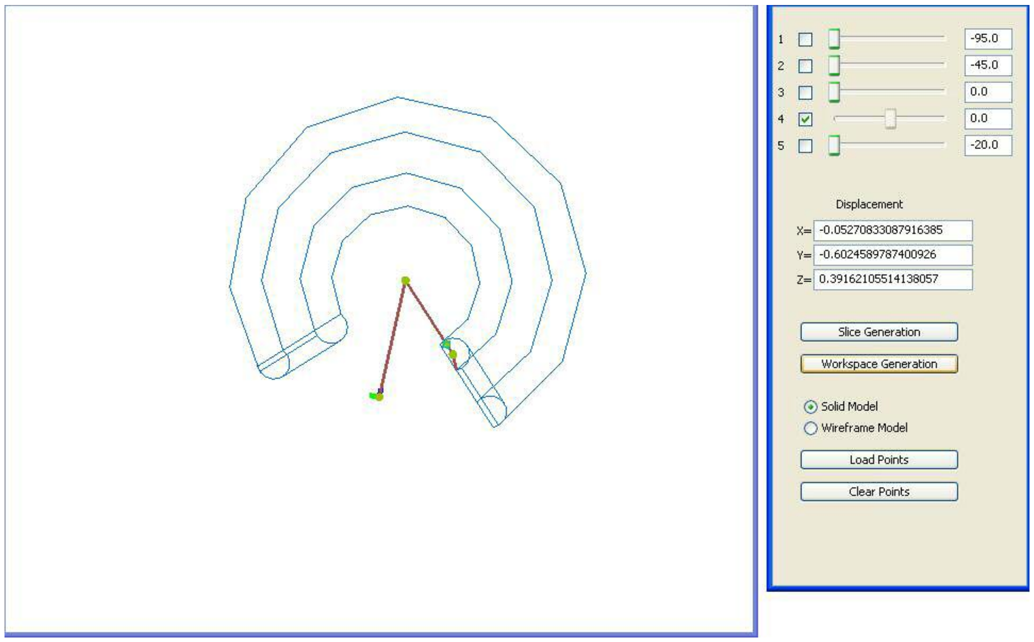
Joint	θ (deg)	h (m)	r (m)	b (m)	α (deg)	β (deg)	Variable
1 (R)	-95,95	1.067	0	0	90	0	θ_1
2 (R)	-45,225	0	0.659	0	0	90	θ_2
3 (P)	0	0,0.61	0	0	0	0	h_3
4 (R)	-95,95	0.109	0	0	0	-90	θ_4
5 (R)	-20,200	0	0.146	0	0	90	θ_5
6 (R)	-360,360	0	0	0	0	0	θ_6



```
1  0.0
2  0.0
3  0.0
4  0.0
5  0.0

Displacement
X= 0.9140000000000001
Y= -6.674325055353075E-18
Z= 1.067

Velocity
X= 0.010646508437165421
Y= 1.5419193200667565
Z= 4.1123241456502875
R vel= 4.391906011817124

Acceleration
X= -16.11514519821935
Y= -2.117977349466291
Z= 1.3090684229849635E-16
R acc= 16.253729812339824
```

.

Figure 3.13 Portrays Bendix AA/CNC Industrial Robot at Home Position.



1 ☐ -95.0
2 ☐ -45.0
3 ☐ 0.0
4 ☑ 0.0
5 ☐ -20.0

Displacement
X= -0.05270833087916385
Y= -0.6024589787400926
Z= 0.39162105514138057

[✓ Solid Model] Wireframe Model

Figure 3.14 Slice of Bendix Workspace at the Sagittal Plane.

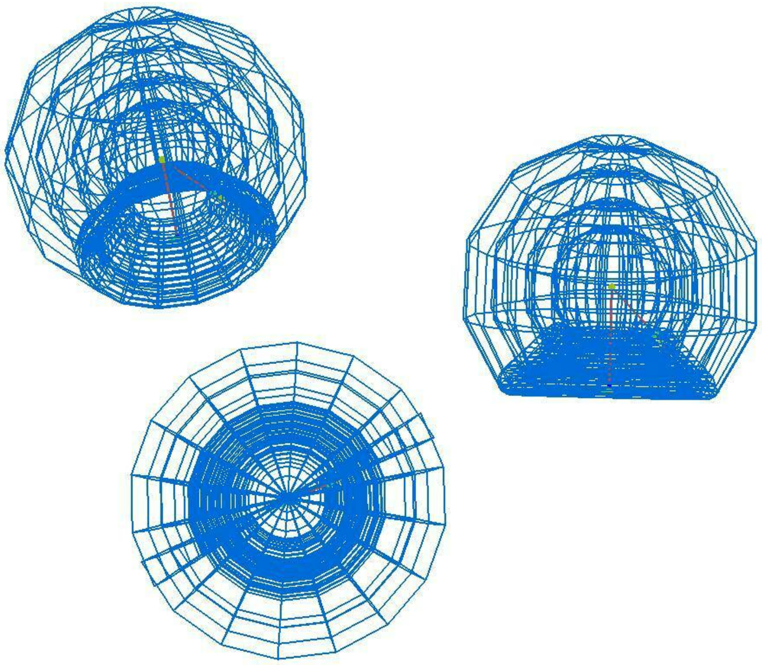


Figure 3.15 Bendix Workspace Shown from Different Viewpoints.

3.4.5 Denso HS-E Series Horizontal Articulated Robot

This is a four joint scara type robot. These robots are typically used in clean rooms as pick and place robots.

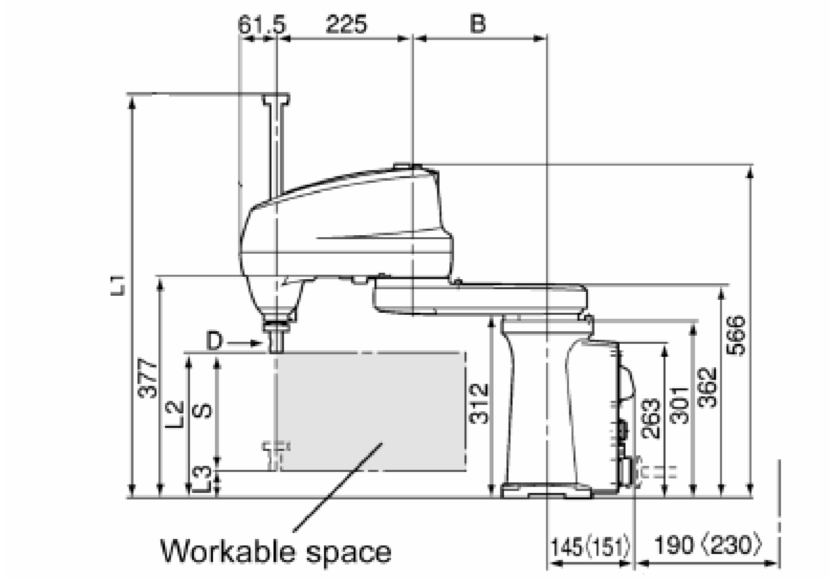
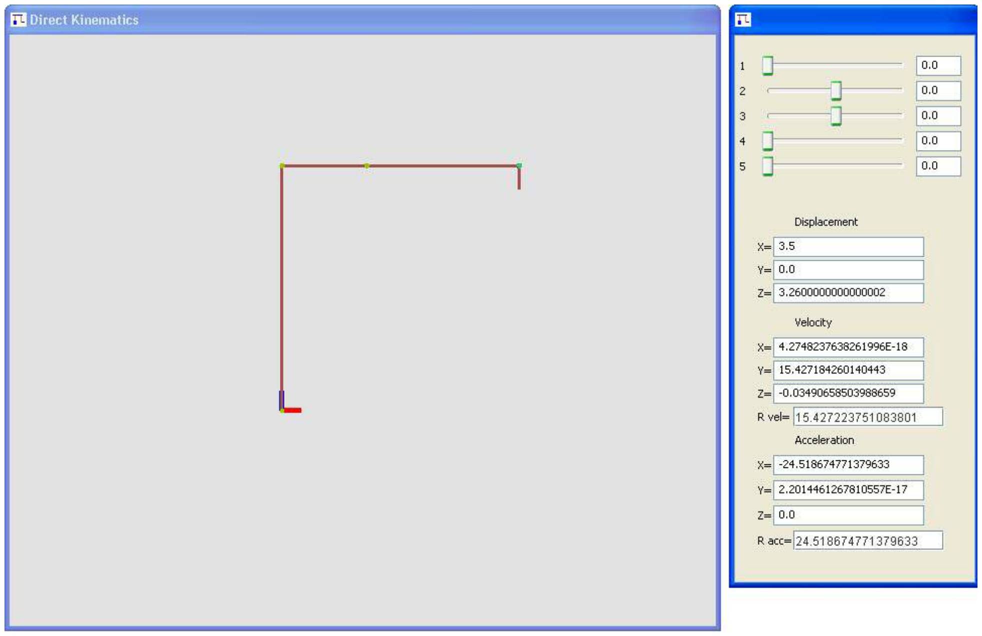


Figure 3.16 HS-E Series Horizontal Articulated Robot

Table 3.5 Denso HS-E Series Notation Table

Joint	θ (deg)	h (m)	r (m)	b (m)	α (deg)	β (deg)	Variable
1 (R)	0,0	3.62	0	0	0	0	θ_1
2 (R)	-155,155	0	1.25	0	0	0	θ_2
3 (R)	-145,145	0	2.25	0	0	180	θ_3
4 (P)	0	0,2	0	0	0	0	h_4
5 (R)	0,360	0.36	0	0	0	0	θ_5



```
1  0.0
2  0.0
3  0.0
4  0.0
5  0.0

Displacement
X= 3.5
Y= 0.0
Z= 3.2600000000000002
Velocity
X= 4.2748237638261996E-18
Y= 15.427184260140443
Z= -0.03490658503988659
R vel= 15.427223751083801
Acceleration
X= -24.518674771379633
Y= 2.2014461267810557E-17
Z= 0.0
R acc= 24.518674771379633
```

Figure 3.17 HS-E Series Robot at Home Position.

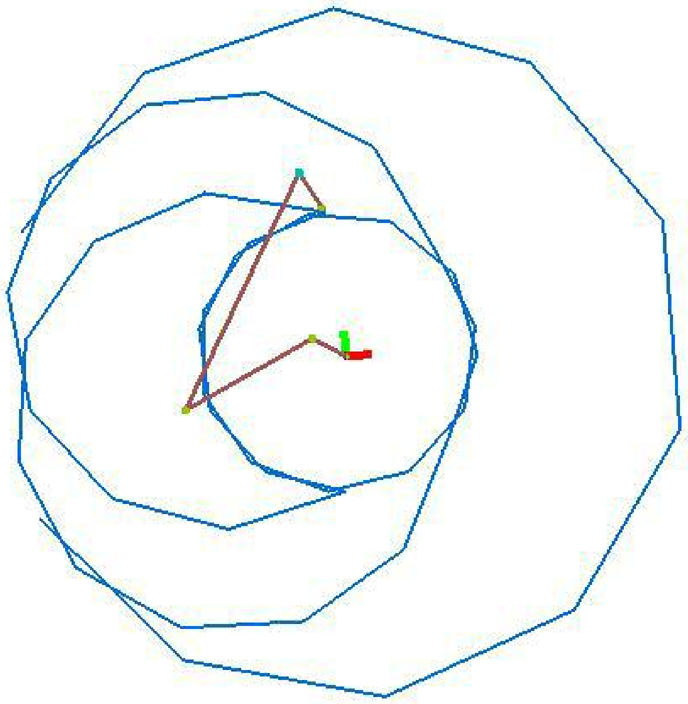


Figure 3.18 Workspace of Denso’s HS-E Series Robot.

3.4.6 Cartesian Robot

A cartesian coordinate robot is an industrial robot whose three principal axes of control are linear (i.e. they move in a straight line rather than rotate) and are at right angles to each other.

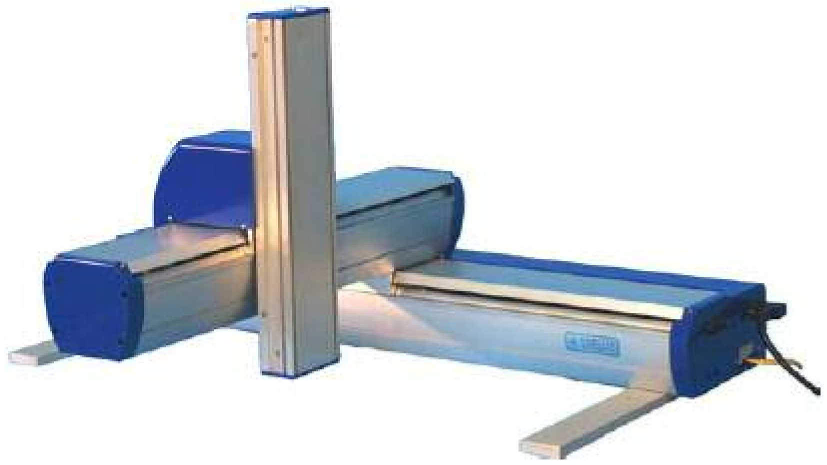
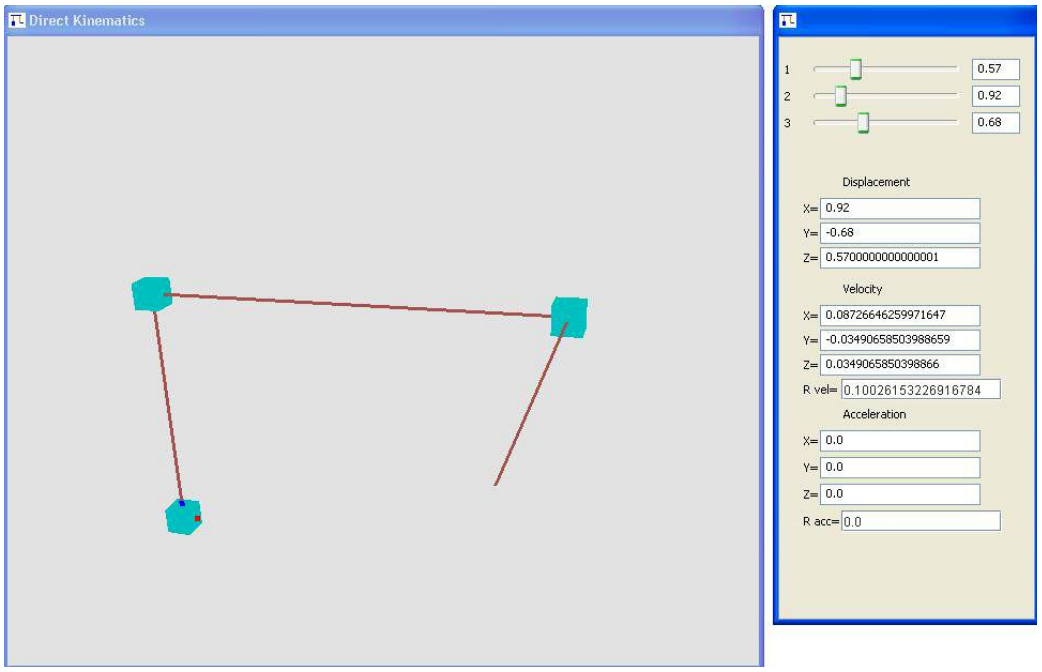


Figure 3.19 Cartesian Robot.

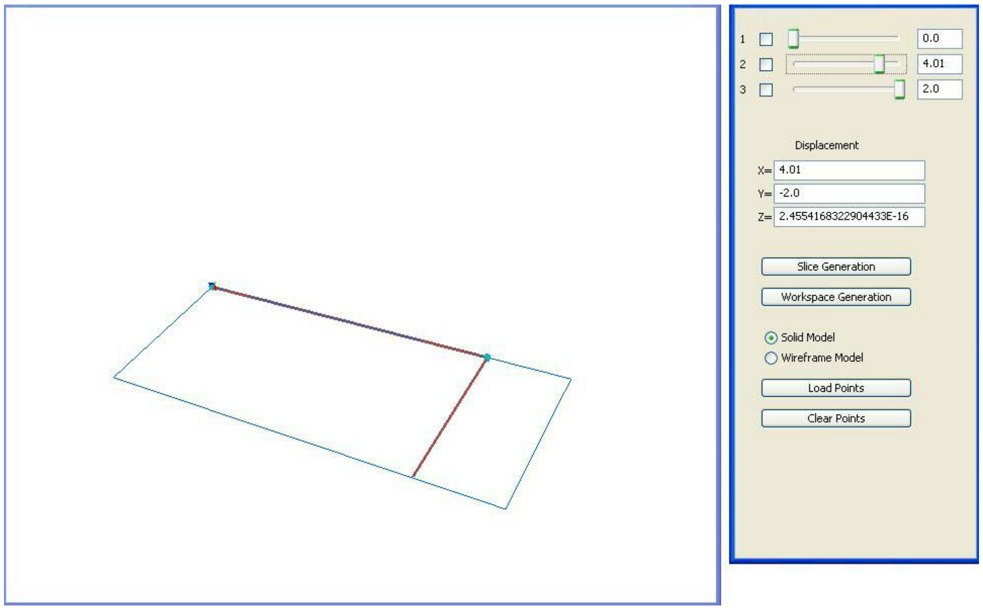
Table 3.6 Cartesian Robot Notation Table

Joint	θ (deg)	h (m)	r (m)	b (m)	α (deg)	β (deg)	Variable
1 (P)	0	0,2	0	0	0	90	h_1
2 (P)	0	0,5	0	0	90	0	h_2
3 (P)	0	0,2	0	0	0	0	h_3



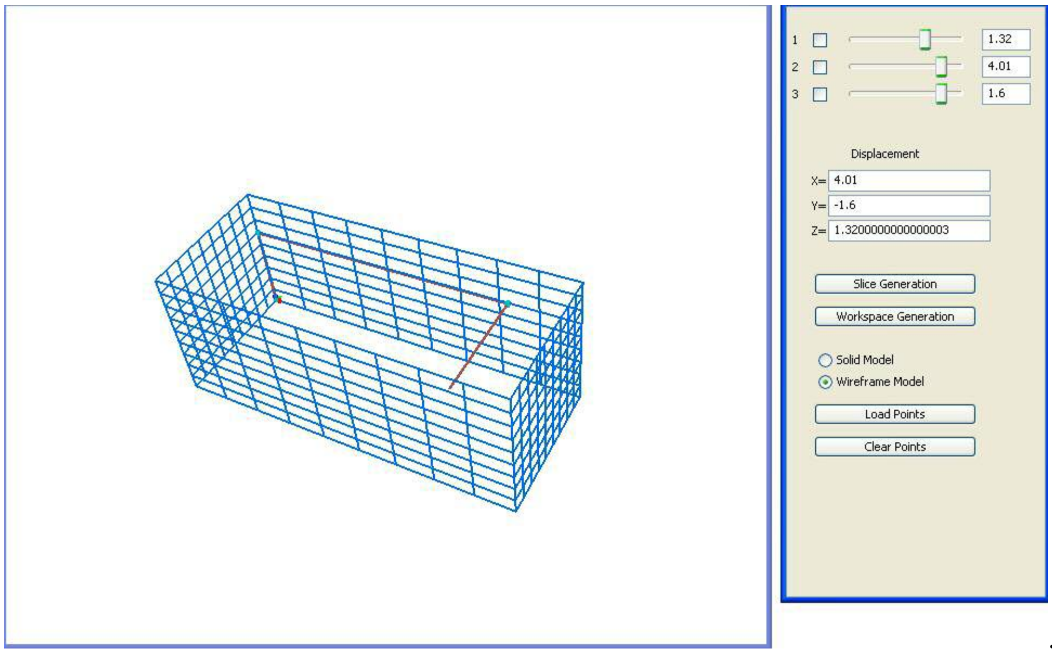
```
1  0.57
2  0.92
3  0.68

Displacement
X= 0.92
Y= -0.68
Z= 0.5700000000000001

Velocity
X= 0.08726646259971647
Y= -0.03490658503988659
Z= 0.0349065850398866
R vel= 0.10026153226916784

Acceleration
X= 0.0
Y= 0.0
Z= 0.0
R acc= 0.0
```

Figure 3.20 Cartesian Robot at Home Position.



```
1  0.0
2  4.01
3  2.0

Displacement
X= 4.01
Y= -2.0
Z= 2.4554168322904433E-16
```

Figure 3.21 A Slice of the Cartesian Robot's Workspace.



1 ☐ 1.32
2 ☐ 4.01
3 ☐ 1.6

Displacement
X= 4.01
Y= -1.6
Z= 1.3200000000000003

Solid Model [✓ Wireframe Model]

Figure 3.22 Workspace of the Cartesian Robot.

CHAPTER 4

INVERSE KINEMATICS

Kinematics is a branch of mechanics that deals with describing the motions of objects without considering the factors that cause or affect the motion. This branch can be further divided into direct kinematics and inverse kinematics. Direct kinematics is the process of calculating the position, velocity and acceleration in space of the end effector or any designated point of the manipulator. In inverse kinematics a target point is given and the values of the joint parameters have to be determined to reach that specific point. So it can be said that inverse kinematics does the reverse of direct kinematics. The inverse kinematic equations of a manipulator are nonlinear and hence difficult to solve. The difficulties faced in solving this kind of problem include

- Multiple to infinite solutions.
- No solutions because of divergence.
- No closed-form (analytical or approximate) solution.

To solve these type problems, one can use Analytical Methods or Iterative (Differential) Methods like Geometric and Analytical Jacobian, Jacobian Transpose Method, Pseudo-Inverse, Pseudo-Inverse with Optimization, Extended Jacobian Method.

4.1 Proposed Inverse Kinematics Solver

A Numerical Iterative technique similar to Gauss Siedel method is used to generate joint parameter values. Consider a robot with all revolute joints and hence the parameter is θ for all the joints. Input initial or guess values for the joint parameters, then use forward kinematics procedure to find the end position of the robotic manipulator. This is given by

$$D = \prod_{i=1}^{n} T_i = \begin{bmatrix} & C_i & & P`_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.1)$$

The vector P_i is set equal to the intended target coordinate values. Check if P_i is equal to $P`_i$ Now each of the transformation matrices T_i is function of θ_i, and hence the individual θ_i have to be separated in order to get the orientations of the links. Hence the following procedure is performed to obtain the orientation values.

$$M = T_i = \left[\prod_{j=1}^{i-1} T_j\right]^{-1} D \left[\prod_{j=i+1}^{n} T_j\right]^{-1} \qquad (4.2)$$

M is purely a function of θ_i and since it represents rotation around the Z axes. The value of θ_i can be obtained by the following process,

$$T(\theta, h, r, b, \alpha, \beta) =$$

$$\begin{bmatrix} (c\theta c\beta - s\theta s\alpha s\beta) & -(s\theta c\alpha) & (c\theta s\beta + s\theta s\alpha c\beta) & (-bs\theta + rc\theta) \\ (s\theta c\beta + c\theta s\alpha s\beta) & (c\theta c\alpha) & (s\theta s\beta - c\theta s\alpha c\beta) & (bc\theta + rs\theta) \\ -(c\alpha s\beta) & s\alpha & (c\alpha c\beta) & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.3)$$

Now consider equation 4.3 and compare it with right hand side matrix of equation 4.2. It is clear that Y component equals b*cθ + r*sθ and X component is equal to -b*sθ + r*cθ.

$$\frac{b\cos(\theta)+r\sin(\theta)}{-b\sin(\theta)+r\cos(\theta)}=\frac{Row2,Col4}{Row1,Col4}=\frac{Y}{X} \tag{4.4}$$

Divide the numerator and denominator of equation 4.4 by cos(θ)

$$\frac{b+r\tan(\theta)}{-b\tan(\theta)+r}=\frac{Y}{X} \tag{4.5}$$

$$bX+rX\tan(\theta)=-bY\tan(\theta)+rY \tag{4.6}$$

Reordering equation 4.6,

$$\tan(\theta)=\frac{rY-bX}{rX+bY} \tag{4.7}$$

$$\theta=\tan^{-1}\left(\frac{rY-bX}{rX+bY}\right) \tag{4.8}$$

Each of θ_i are computed at the end equation 4.8, then these new values are substituted back into equation 4.2 and next set of θ_i are obtained. At end of each cycle, the new set of θ_i are substituted in direct kinematics equation 4.1. The vector P` obtained at the end of this step is compared with vector P, which is essentially the coordinate values of the intended target point. If the required precision is achieved then the iterations are stopped and the values of these joint parameters are displayed. If not, the iteration process, equation 4.1, 4.2 and 4.8, is continued till the stopping criterion is encountered. If prismatic joined are considered then h_i is the variable and only the entries in row three and column four of matrix on right hand side is equated to h_i.

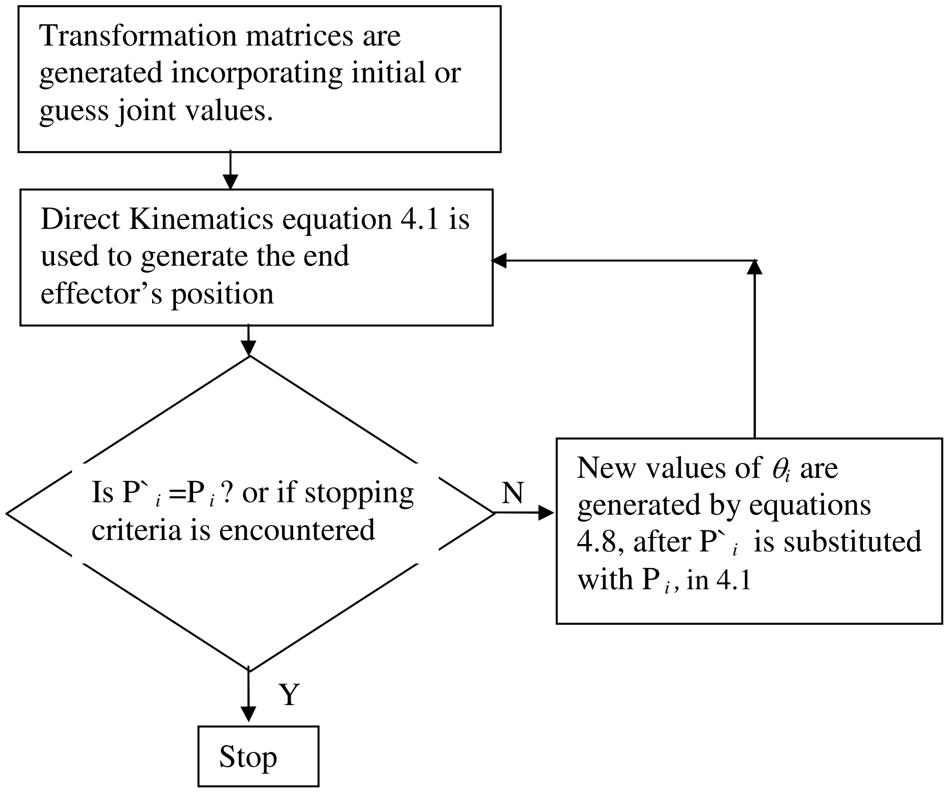

Transformation matrices are generated incorporating initial or guess joint values.
Direct Kinematics equation 4.1 is used to generate the end effector's position
Is P`$_i$ =P$_i$? or if stopping criteria is encountered
N
New values of θ_i are generated by equations 4.8, after P`$_i$ is substituted with P$_i$, in 4.1
Y
Stop

Figure 4.1 Flowchart of Inverse Kinematics Solver.

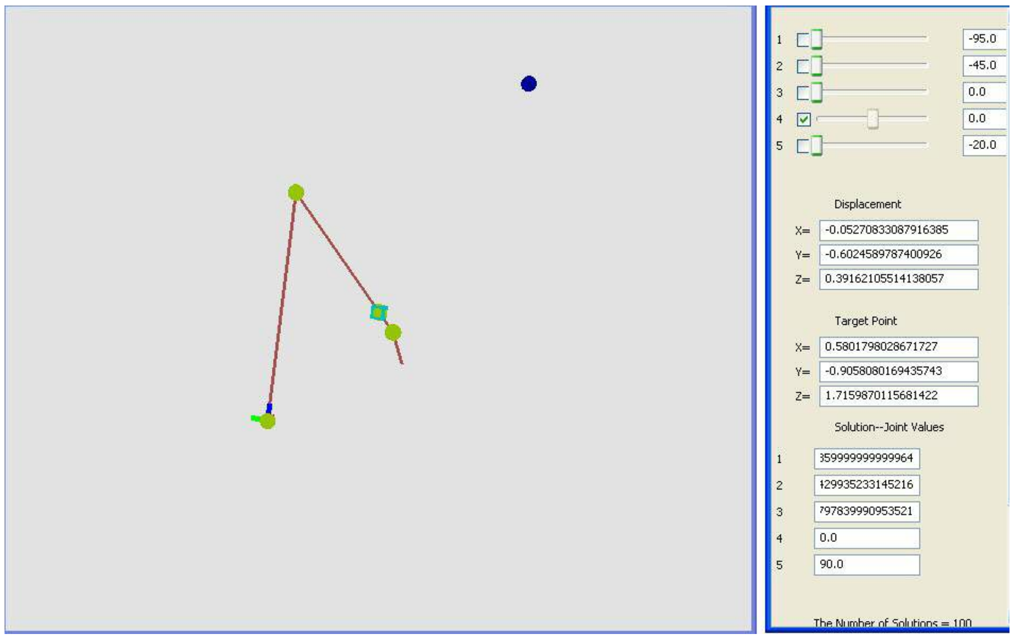
4.2 Example Describing the Inverse Kinematics Solver

Consider an example of a robot with three revolute joints. Let the target point to be reached by a vector P=(x, y, z). The forward kinematics is given by $D = T_1 * T_2 * T_3$. D is a homogeneous matrix of dimension 4 x 4. If the entries of matrix (1, 4), (2, 4) and (3, 4) equals the component of the vector x, y and z, then no further iterations needed. If they aren't equal then these steps are performed,

Replace the entries of (1, 4), (2, 4) and (3, 4) with x, y and z. Then operation $T_3 = T_2^{-1} T_1^{-1} D$ is performed. Now compute the value θ_3, update this new value and re-check if the position vector components of the matrix $D = T_1 * T_2 * T_3$, equals those of x, y and z. If the criteria is satisfied then iterations are stopped, else $T_2 = T_1^{-1} D T_3^{-1}$ is computed and from this value of θ_2 is computed, again update the new value of θ_2 in the matrix T_2. Now compute D and re-check again. The iterations are stopped if the stopping criteria is encountered, if not then $T_1 = D T_3^{-1} T_2^{-1}$ is performed. New value of θ_1 is updated and matrix D is computed. So at the end of one cycle, the coordinate values of the end effector are verified with those of the intended target point. Further iterations are continued until the end effector reaches the target point.

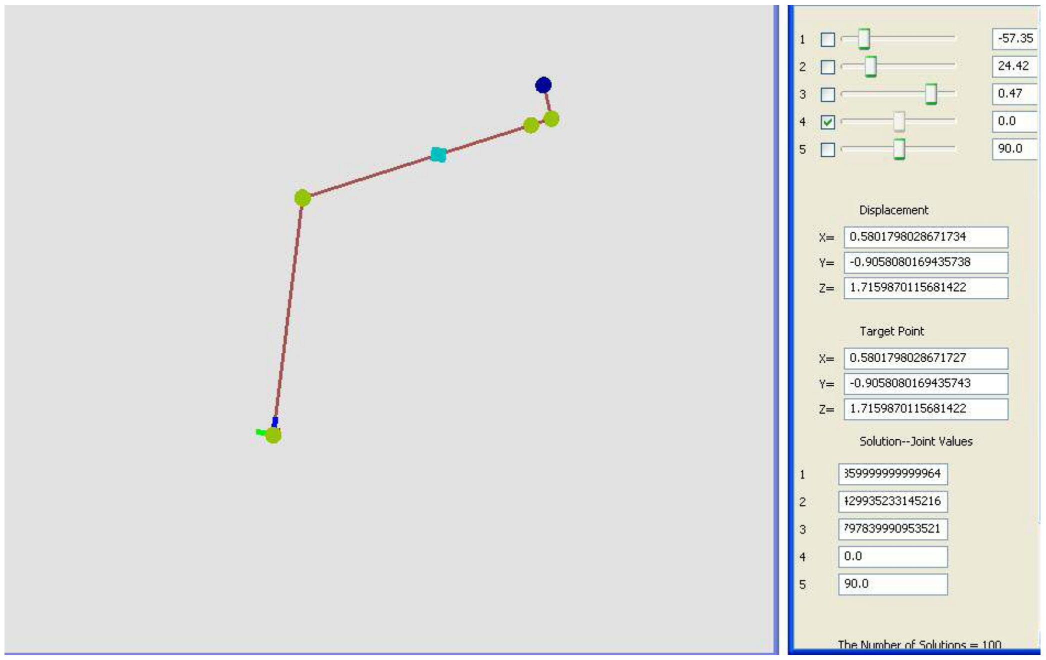
This method of finding the joint values was tested on many robot configurations. The results obtained were satisfactory as the robot manipulator was able to reach the intended target points within the maximum number of iterations and within the prescribed precision.

4.2.1 Inverse Kinematics Results for Bendix Robot



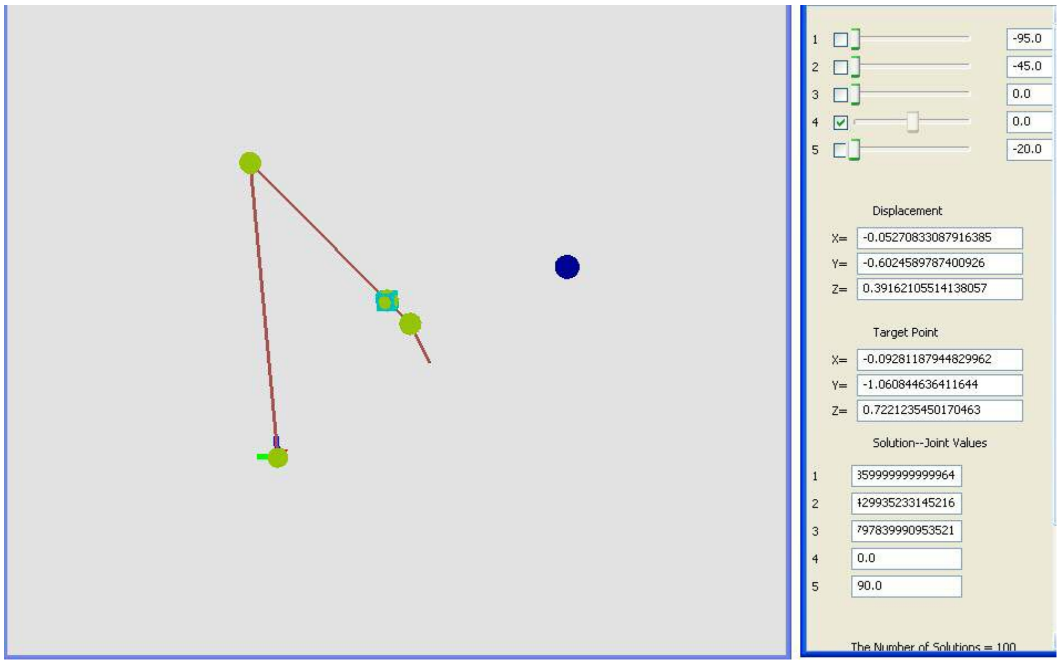
```
1 ☐ -95.0
2 ☐ -45.0
3 ☐ 0.0
4 ☑ 0.0
5 ☐ -20.0

Displacement
X= -0.05270833087916385
Y= -0.6024589787400926
Z= 0.39162105514138057

Target Point
X= 0.5801798028671727
Y= -0.9058080169435743
Z= 1.7159870115681422

Solution--Joint Values
1 359999999999964
2 429935233145216
3 797839990953521
4 0.0
5 90.0

The Number of Solutions = 100
```

Figure 4.2 Bendix Robot at Home Position.



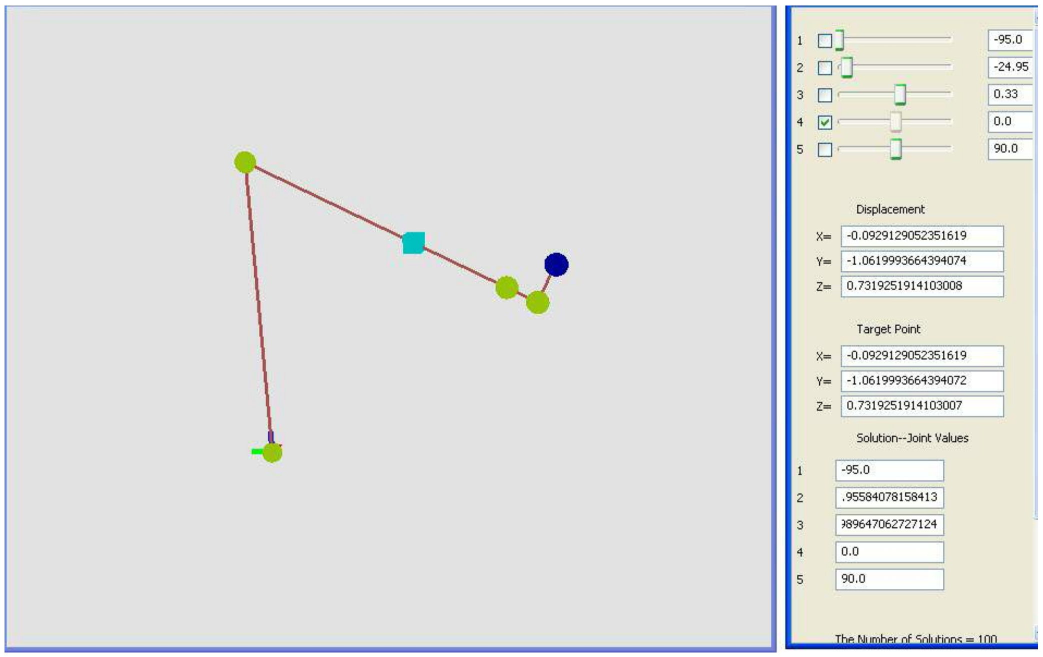
```
1 ☐ -57.35
2 ☐ 24.42
3 ☐ 0.47
4 ☑ 0.0
5 ☐ 90.0

Displacement
X= 0.5801798028671734
Y= -0.9058080169435738
Z= 1.7159870115681422

Target Point
X= 0.5801798028671727
Y= -0.9058080169435743
Z= 1.7159870115681422

Solution--Joint Values
1 359999999999964
2 429935233145216
3 797839990953521
4 0.0
5 90.0

The Number of Solutions = 100
```

Figure 4.3 Displays Bendix Robot Manipulator Reaching the Target Point.



```
1 ☐ -95.0
2 ☐ -45.0
3 ☐ 0.0
4 ☑ 0.0
5 ☐ -20.0

Displacement
X= -0.05270833087916385
Y= -0.6024589787400926
Z= 0.39162105514138057

Target Point
X= -0.09281187944829962
Y= -1.060844636411644
Z= 0.7221235450170463

Solution--Joint Values
1 359999999999964
2 429935233145216
3 797839990953521
4 0.0
5 90.0

The Number of Solutions = 100
```

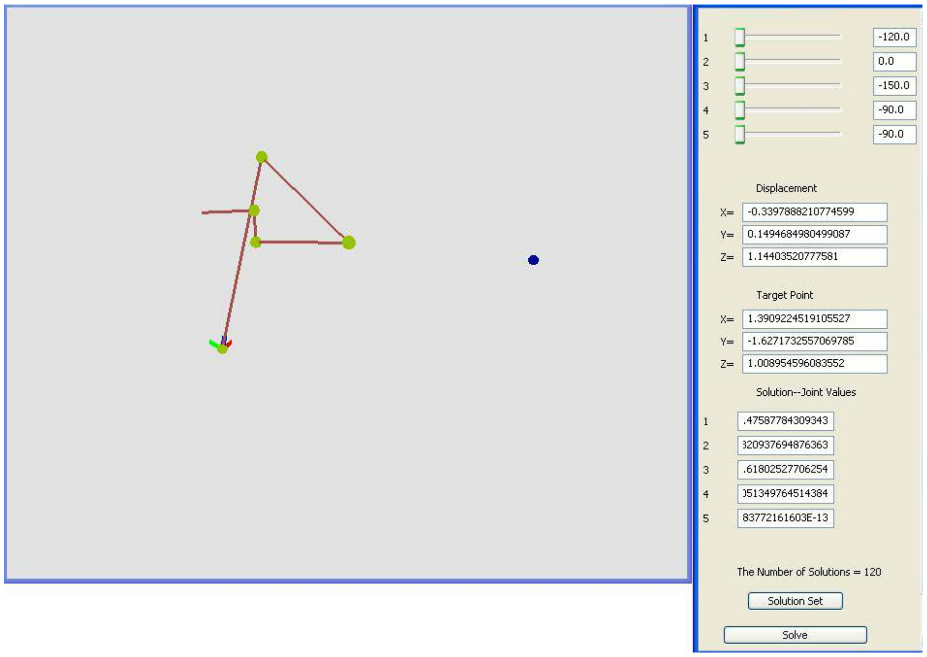
Figure 4.4 Another Example of Bendix Robot.



```
1 ☐ -95.0
2 ☐ -24.95
3 ☐ 0.33
4 ☑ 0.0
5 ☐ 90.0

Displacement
X= -0.0929129052351619
Y= -1.0619993664394074
Z= 0.7319251914103008

Target Point
X= -0.0929129052351619
Y= -1.0619993664394072
Z= 0.7319251914103007

Solution--Joint Values
1 -95.0
2 .95584078158413
3 389647062727124
4 0.0
5 90.0

The Number of Solutions = 100
```

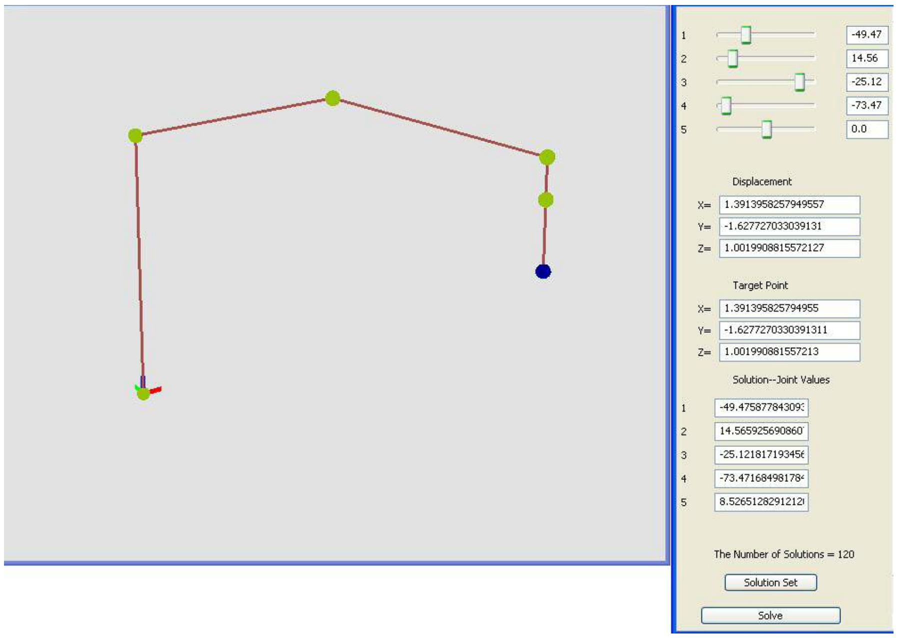
Figure 4.5 Bendix Robot Reaches Intended Target Point.

4.2.2 Inverse Kinematics Results for Cincinati Robot



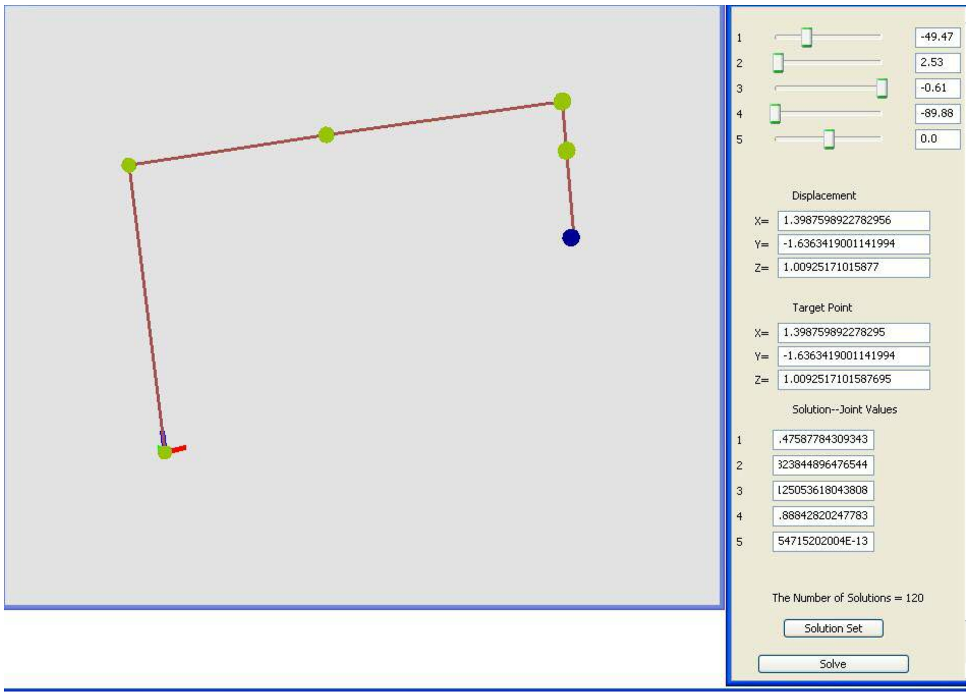
```
1  -120.0
2  0.0
3  -150.0
4  -90.0
5  -90.0

Displacement
X= -0.3397888210774599
Y= 0.1494684980499087
Z= 1.14403520777581

Target Point
X= 1.3909224519105527
Y= -1.6271732557069785
Z= 1.008954596083552

Solution--Joint Values
1  .47587784309343
2  320937694876363
3  .61802527706254
4  51349764514384
5  83772161603E-13

The Number of Solutions = 120
```

Figure 4.6 Milacron Robot at Home Position.



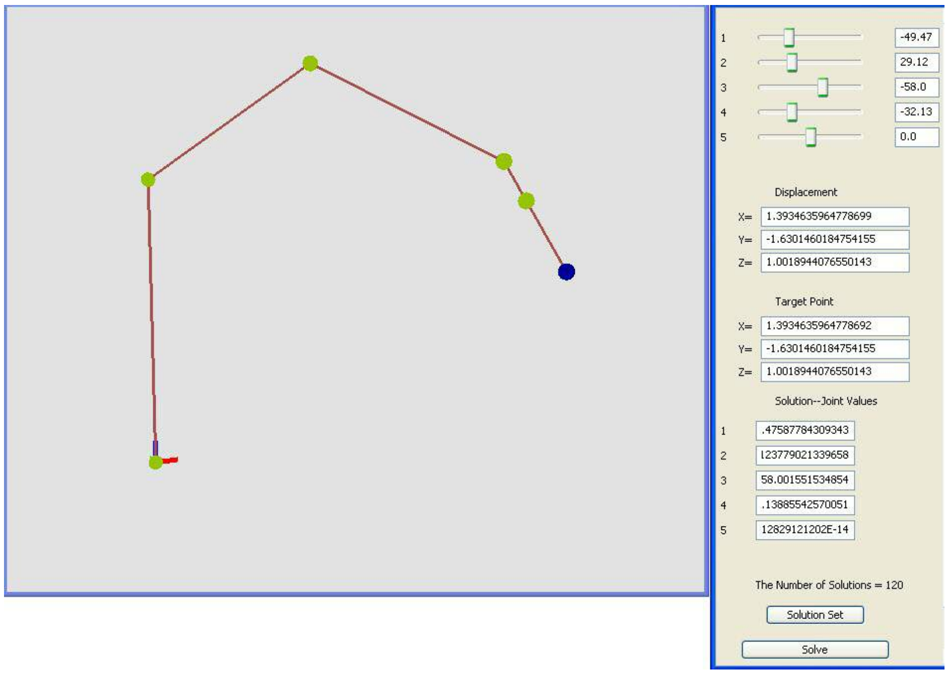
```
1  -49.47
2  14.56
3  -25.12
4  -73.47
5  0.0

Displacement
X= 1.3913958257949557
Y= -1.627727033039131
Z= 1.0019908815572127

Target Point
X= 1.391395825794955
Y= -1.6277270330391311
Z= 1.001990881557213

Solution--Joint Values
1  -49.47587784309
2  14.565925690860
3  -25.12181719345
4  -73.47168498178
5  8.5265128291212

The Number of Solutions = 120
```

Figure 4.7 Milacron Robot's End Effector at Target Point.



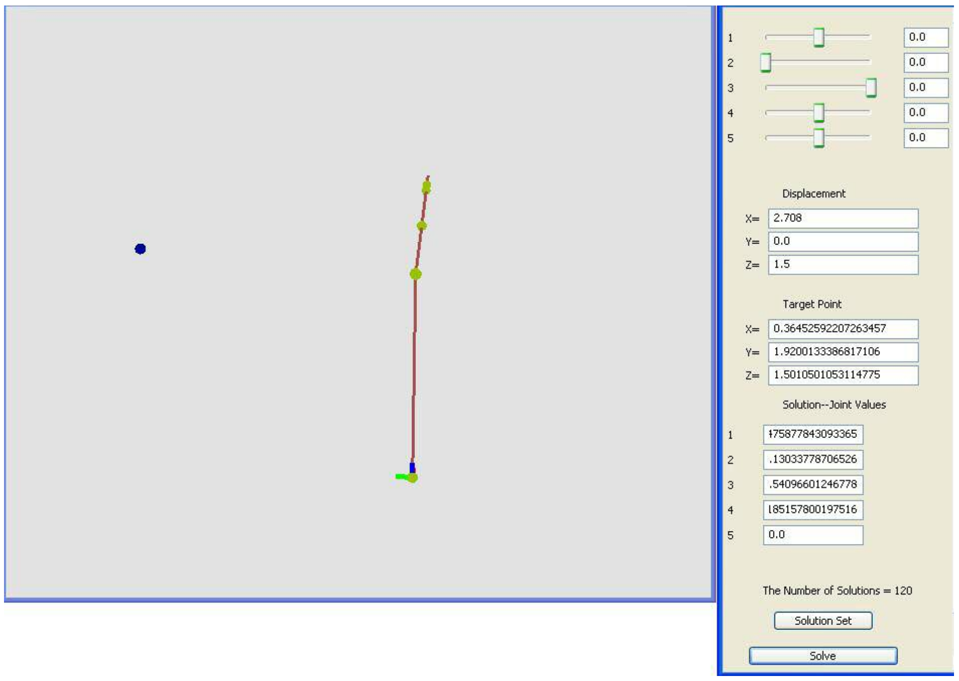
```
1    -49.47
2    2.53
3    -0.61
4    -89.88
5    0.0

Displacement
X= 1.3987598922782956
Y= -1.6363419001141994
Z= 1.00925171015877

Target Point
X= 1.398759892278295
Y= -1.6363419001141994
Z= 1.0092517101587695

Solution--Joint Values
1  .47587784309343
2  323844896476544
3  125053618043808
4  .88842820247783
5  54715202004E-13

The Number of Solutions = 120
```

Figure 4.8 One of the Multiple Solutions to Inverse Kinematics Problem.



```
1    -49.47
2    29.12
3    -58.0
4    -32.13
5    0.0

Displacement
X= 1.3934635964778699
Y= -1.6301460184754155
Z= 1.0018944076550143

Target Point
X= 1.3934635964778692
Y= -1.6301460184754155
Z= 1.0018944076550143

Solution--Joint Values
1  .47587784309343
2  123779021339658
3  58.001551534854
4  .13885542570051
5  12829121202E-14

The Number of Solutions = 120
```

Figure 4.9 One More Solution to same Target Point.



1	0.0
2	0.0
3	0.0
4	0.0
5	0.0

Displacement

X= 2.708
Y= 0.0
Z= 1.5

Target Point

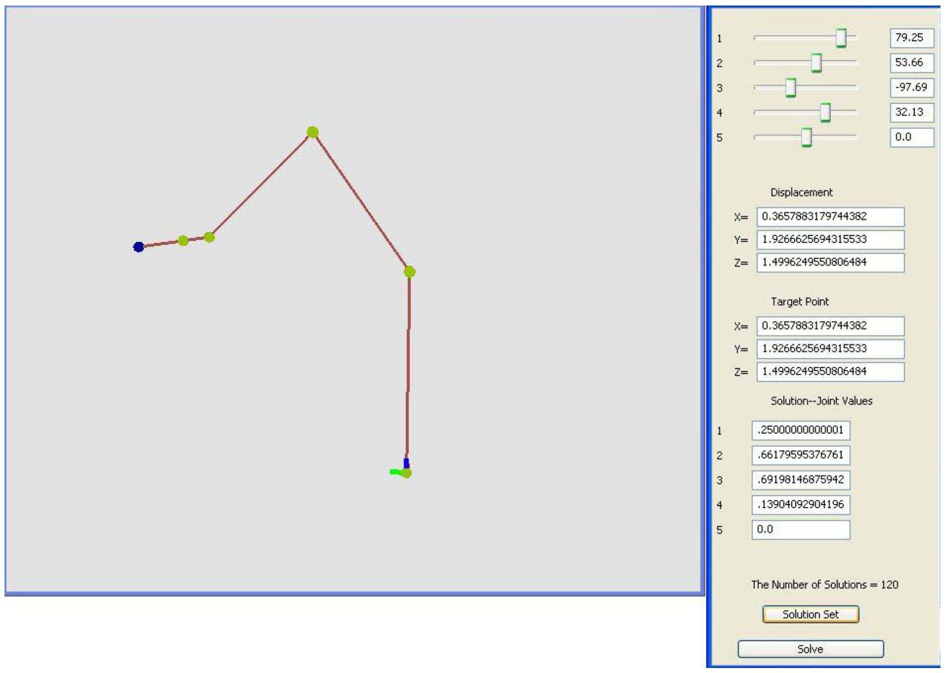
X= 0.36452592207263457
Y= 1.9200133386817106
Z= 1.5010501053114775

Solution--Joint Values

1	ɫ75877843093365
2	.13033778706526
3	.54096601246778
4	⁞85157800197516
5	0.0

The Number of Solutions = 120

Figure 4.10 Another Example with a Different Target Point.



1	79.25
2	53.66
3	-97.69
4	32.13
5	0.0

Displacement

X= 0.3657883179744382
Y= 1.9266625694315533
Z= 1.4996249550806484

Target Point

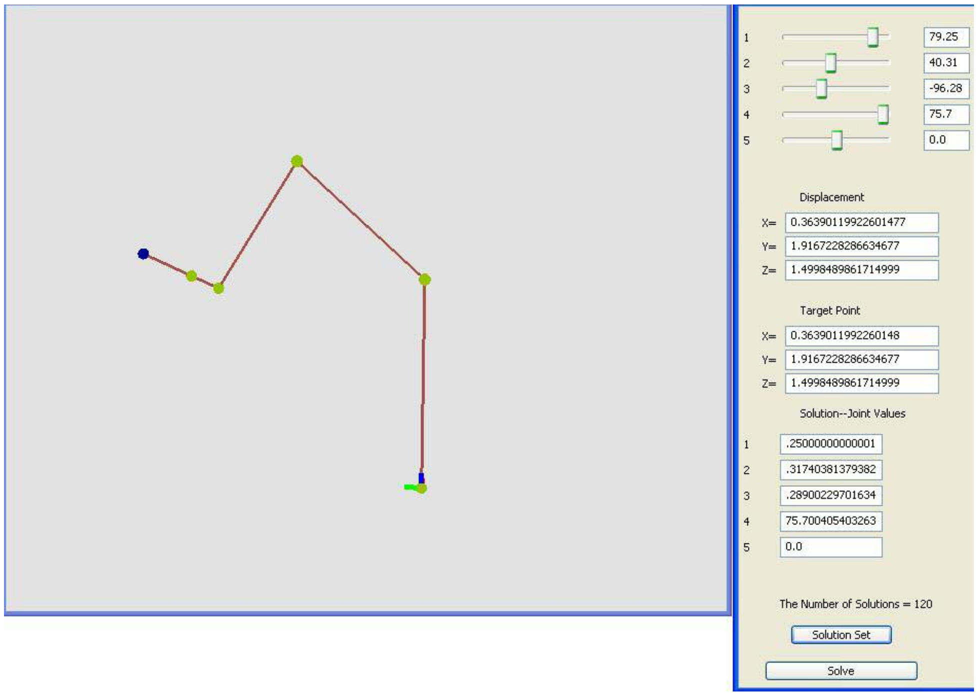
X= 0.3657883179744382
Y= 1.9266625694315533
Z= 1.4996249550806484

Solution--Joint Values

1	.25000000000001
2	.66179595376761
3	.69198146875942
4	.13904092904196
5	0.0

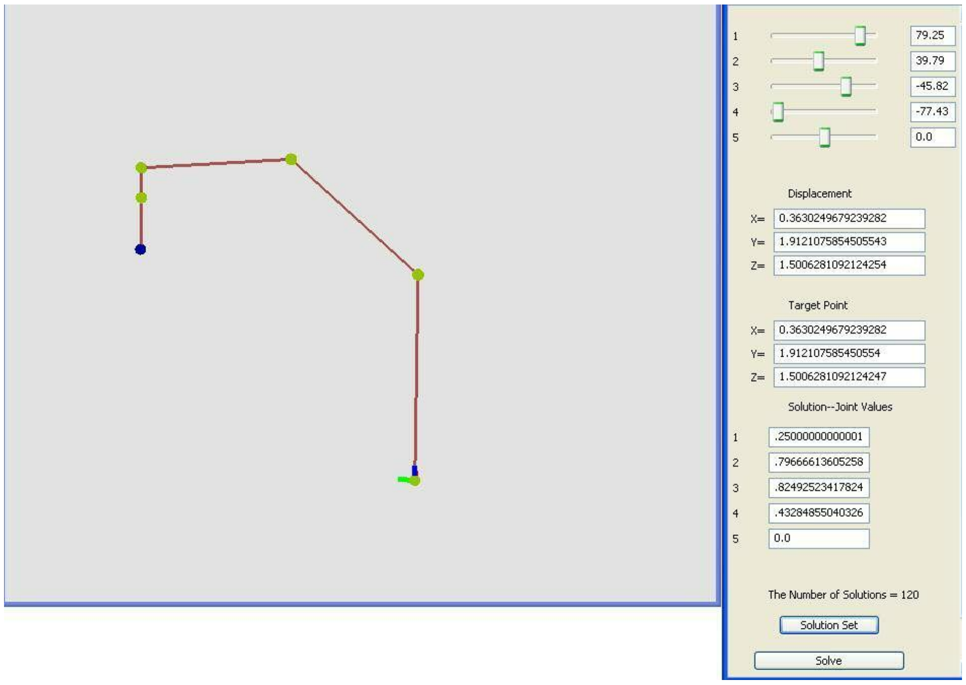
The Number of Solutions = 120

Figure 4.11 First Solution.



```
1  79.25
2  40.31
3  -96.28
4  75.7
5  0.0

Displacement
X= 0.36390119922601477
Y= 1.9167228286634677
Z= 1.4998489861714999

Target Point
X= 0.3639011992260148
Y= 1.9167228286634677
Z= 1.4998489861714999

Solution--Joint Values
1  .25000000000001
2  .31740381379382
3  .28900229701634
4  75.700405403263
5  0.0

The Number of Solutions = 120
```

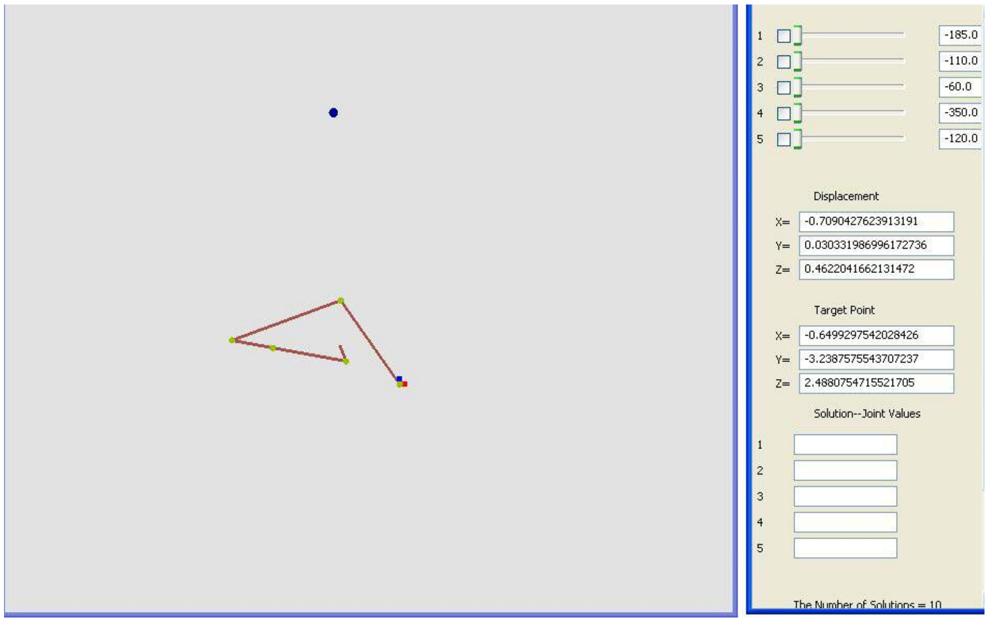
Figure 4.12 Second Solution.



```
1  79.25
2  39.79
3  -45.82
4  -77.43
5  0.0

Displacement
X= 0.3630249679239282
Y= 1.9121075854505543
Z= 1.5006281092124254

Target Point
X= 0.3630249679239282
Y= 1.912107585450554
Z= 1.5006281092124247

Solution--Joint Values
1  .25000000000001
2  .79666613605258
3  .82492523417824
4  .43284855040326
5  0.0

The Number of Solutions = 120
```

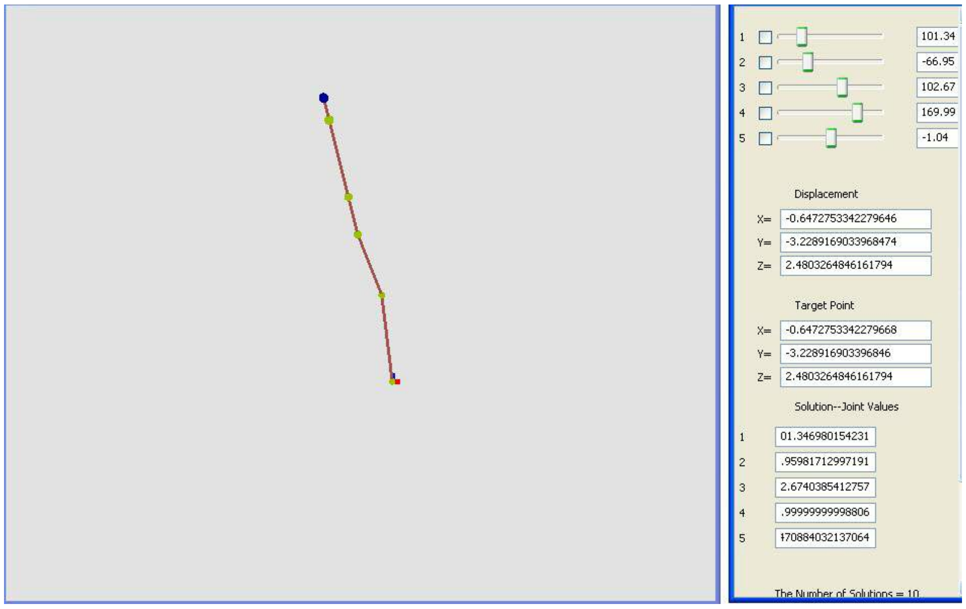
Figure 4.13 Third Solution.

4.2.3 Inverse Kinematics Results for Kuka Robot



```
1  -185.0
2  -110.0
3  -60.0
4  -350.0
5  -120.0

Displacement
X= -0.7090427623913191
Y= 0.030331986996172736
Z= 0.4622041662131472

Target Point
X= -0.6499297542028426
Y= -3.2387575543707237
Z= 2.4880754715521705

Solution--Joint Values
1
2
3
4
5

The Number of Solutions = 10
```

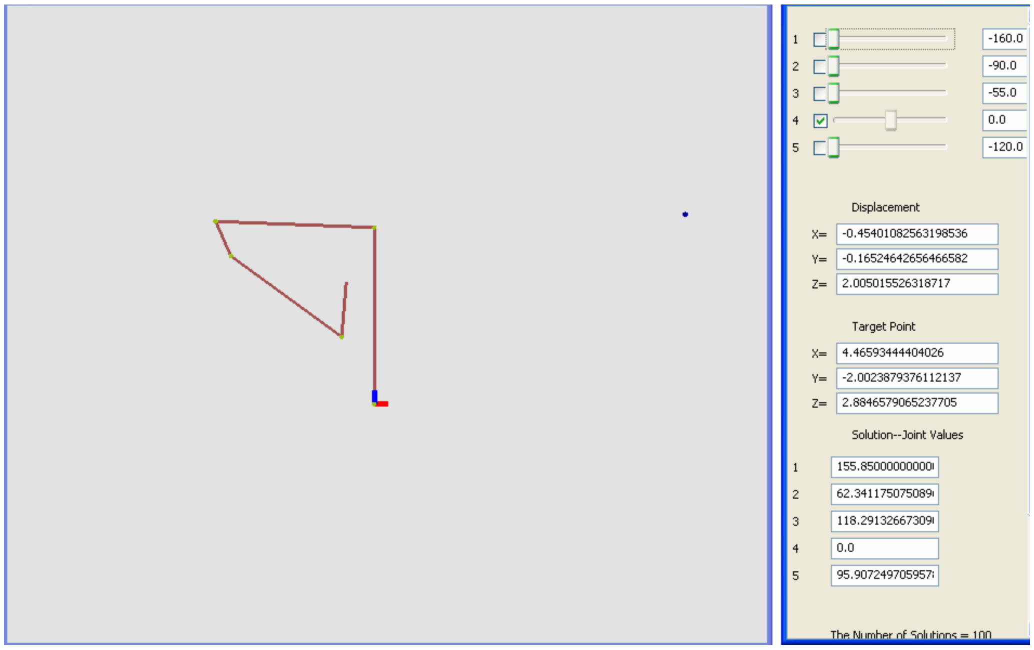
Figure 4.14 Kuka Robot at Home Position.



```
1  101.34
2  -66.95
3  102.67
4  169.99
5  -1.04

Displacement
X= -0.6472753342279646
Y= -3.2289169033968474
Z= 2.4803264846161794

Target Point
X= -0.6472753342279668
Y= -3.228916903396846
Z= 2.4803264846161794

Solution--Joint Values
1  01.346980154231
2  .95981712997191
3  2.6740385412757
4  .99999999998806
5  ł70884032137064

The Number of Solutions = 10
```

Figure 4.15 Kuka Robot at Target Position.

4.2.4 Inverse Kinematics Results for Motoman UPJ Robot



1 -160.0
2 -90.0
3 -55.0
4 ✓ 0.0
5 -120.0

Displacement

X= -0.45401082563198536
Y= -0.16524642656466582
Z= 2.005015526318717

Target Point

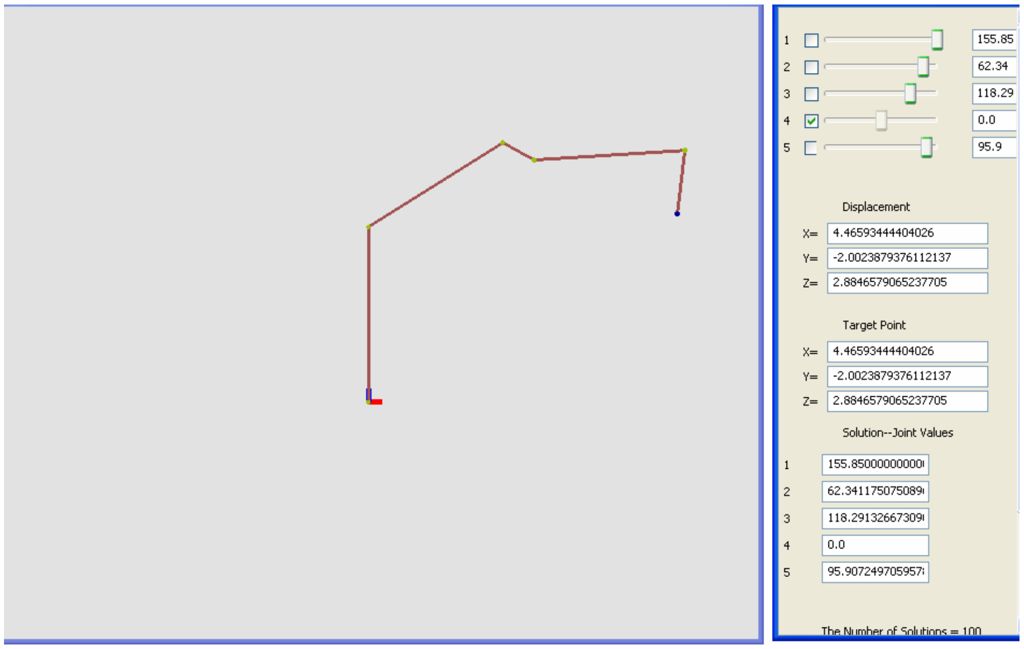
X= 4.46593444404026
Y= -2.0023879376112137
Z= 2.8846579065237705

Solution--Joint Values

1 155.85000000000
2 62.341175075089
3 118.29132667309
4 0.0
5 95.907249705957

The Number of Solutions = 100

Figure 4.16 UPJ Robot at Home Position.



1 155.85
2 62.34
3 118.29
4 ✓ 0.0
5 95.9

Displacement

X= 4.46593444404026
Y= -2.0023879376112137
Z= 2.8846579065237705

Target Point

X= 4.46593444404026
Y= -2.0023879376112137
Z= 2.8846579065237705

Solution--Joint Values

1 155.85000000000
2 62.341175075089
3 118.29132667309
4 0.0
5 95.907249705957

The Number of Solutions = 100

Figure 4.17 UPJ Robot at Target Position.

4.2.5 Inverse Kinematics Results for Scara Type Robot

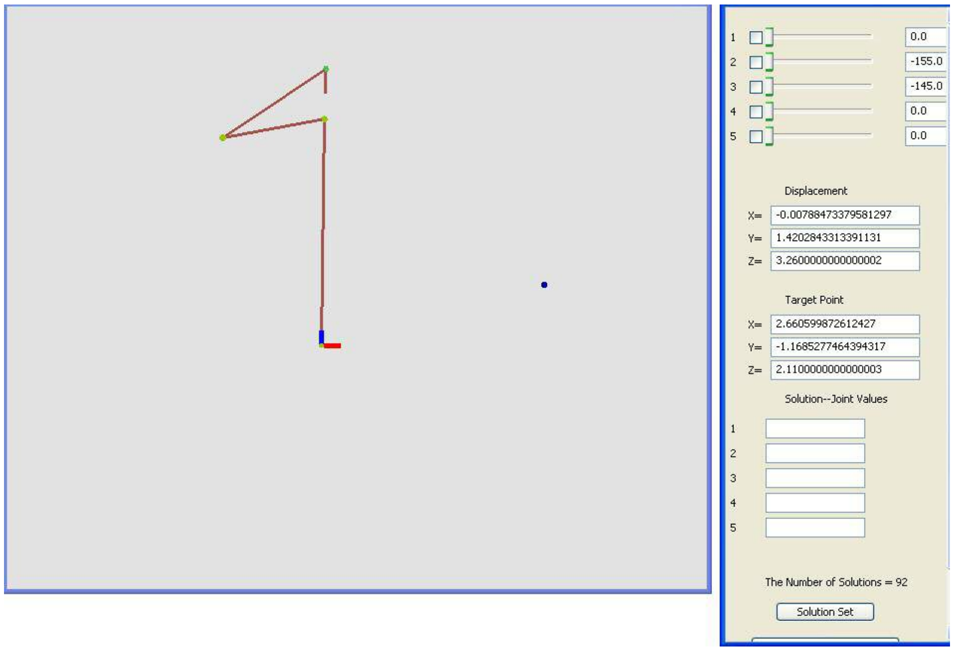

```
1  0.0
2  -155.0
3  -145.0
4  0.0
5  0.0

Displacement
X= -0.00788473379581297
Y= 1.4202843313391131
Z= 3.2600000000000002

Target Point
X= 2.660599872612427
Y= -1.1685277464394317
Z= 2.1100000000000003

Solution--Joint Values
1
2
3
4
5

The Number of Solutions = 92
```

Figure 4.18 Scara Robot at Home Position.

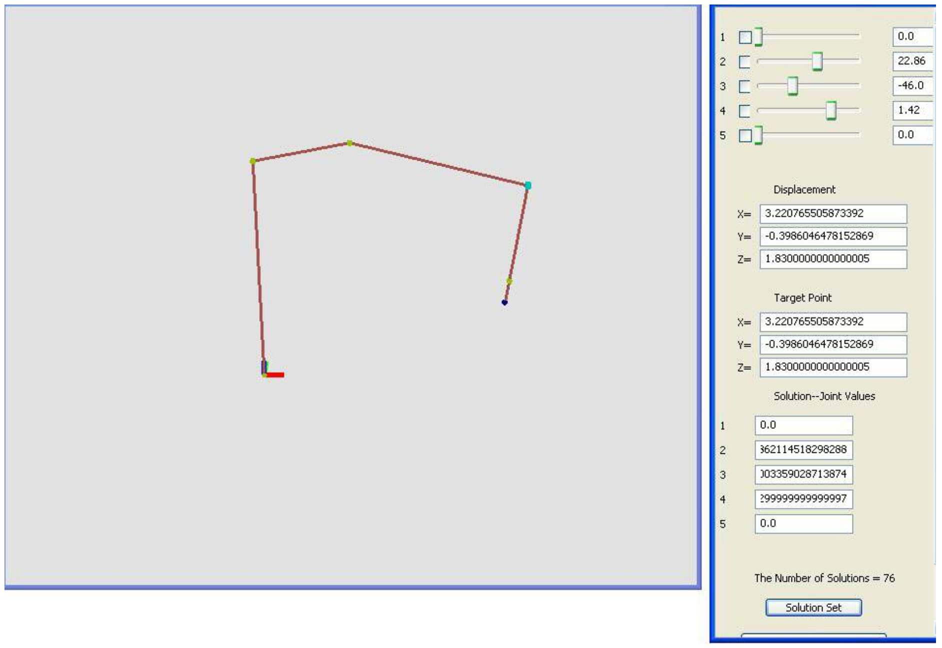
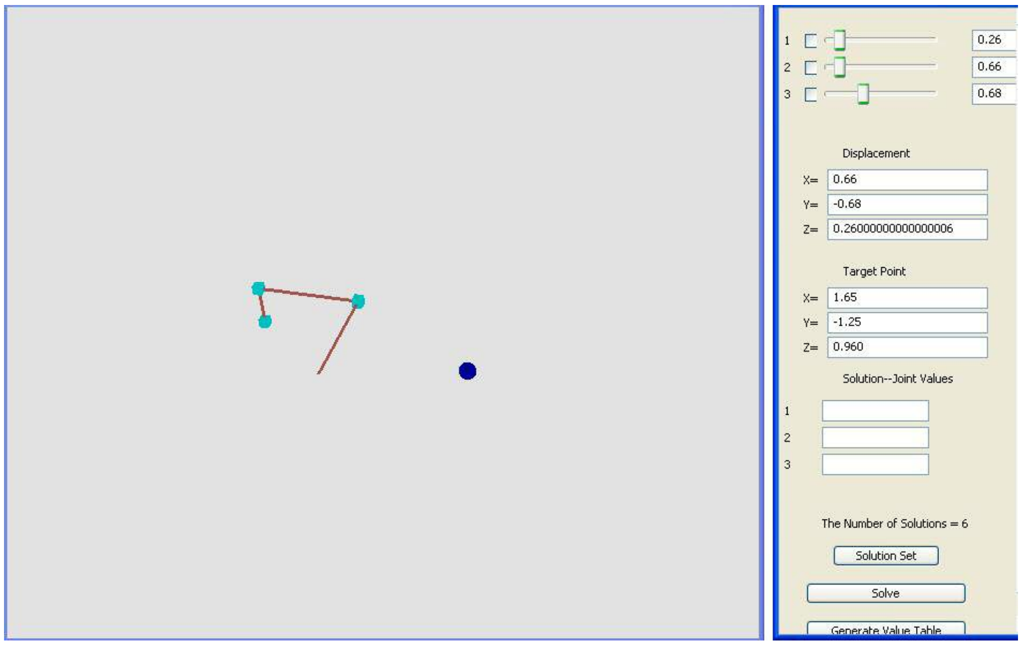

```
1  0.0
2  22.86
3  -46.0
4  1.42
5  0.0

Displacement
X= 3.220765505873392
Y= -0.3986046478152869
Z= 1.8300000000000005

Target Point
X= 3.220765505873392
Y= -0.3986046478152869
Z= 1.8300000000000005

Solution--Joint Values
1  0.0
2  362114518298288
3  )03359028713874
4  2999999999999997
5  0.0

The Number of Solutions = 76
```

Figure 4.19 Scara Robot at Target Position.

4.2.6 Inverse Kinematics Results for Cartesian Robot



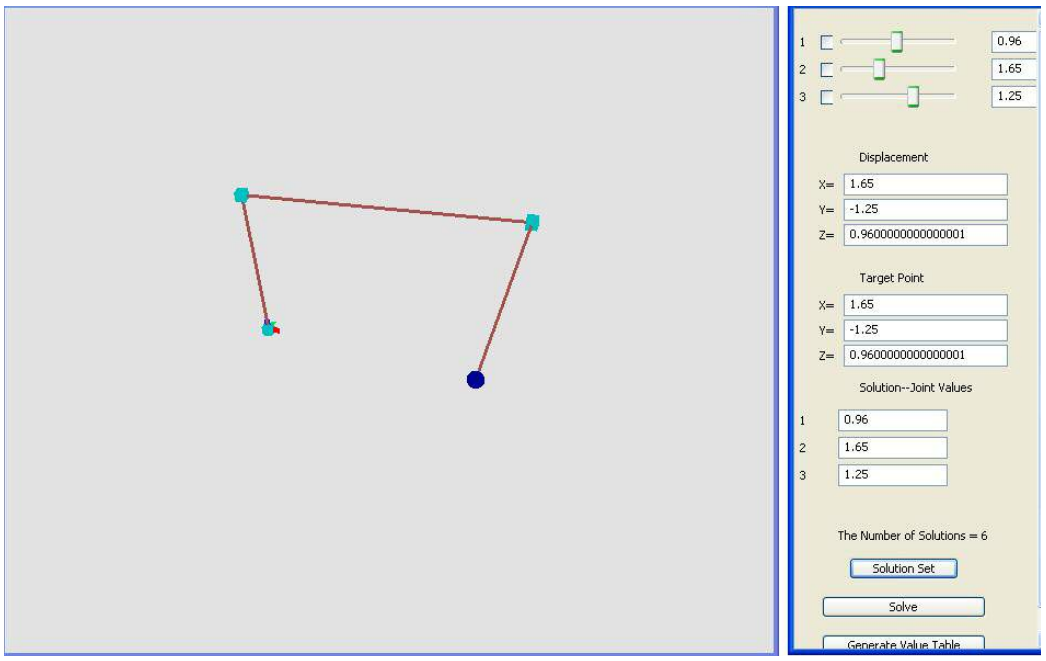
0.26
0.66
0.68

X= 0.66
Y= -0.68
Z= 0.26000000000000006

X= 1.65
Y= -1.25
Z= 0.960

The Number of Solutions = 6

Figure 4.20 Cartesian Robot at Home Position.



0.96
1.65
1.25

X= 1.65
Y= -1.25
Z= 0.9600000000000001

X= 1.65
Y= -1.25
Z= 0.9600000000000001

1 0.96
2 1.65
3 1.25

The Number of Solutions = 6

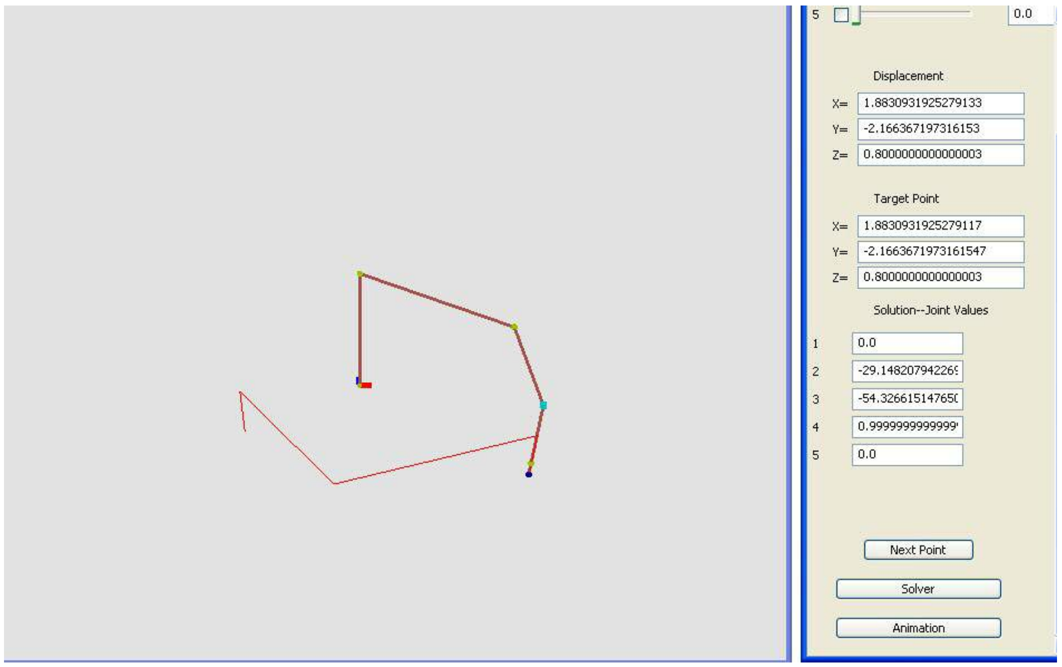
Figure 4.21 Cartesian Robot at Target Position.

CHAPTER 5

APPLICATIONS

5.1 Pick and Place Operation

Robots are used in clean rooms for simple pick and place operations. This is to ensure that a cleaner environment is maintained and also while handling minute things, so that no dirt particles or moisture affect the quality of the product or process. They are also used in material handling areas. An example to demonstrate this operation is as follows,



5 0.0

Displacement

X= 1.8830931925279133

Y= -2.166367197316153

Z= 0.8000000000000003

Target Point

X= 1.8830931925279117

Y= -2.1663671973161547

Z= 0.8000000000000003

Solution--Joint Values

1 0.0

2 -29.14820794226

3 -54.32661514765

4 0.9999999999999

5 0.0

Figure 5.1 Depicts the Path to be Followed for Pick & Place Operation.

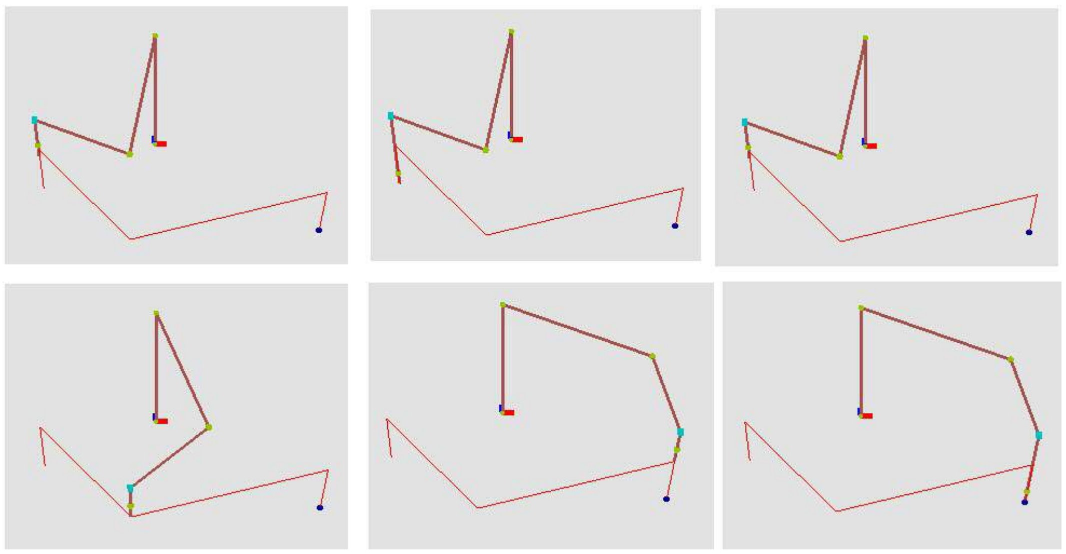
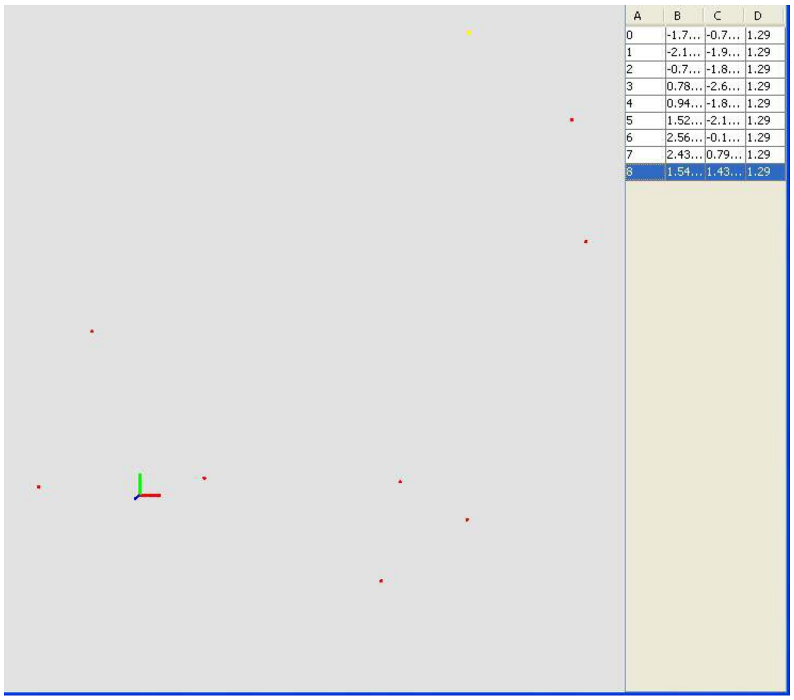


Figure 5.2 Pick & Place Operation.

5.2 Integration of Optimization & Inverse Kinematics

In this section, a robot in an electronic assembly set up is considered. Usually, for things in micro-scale, precision machines like robots are employed to perform micro-assembly. Robots suit well for such kinds of dexterous and repetitive tasks. The first step is to identify the locations on the electronic board where the robot would put in components. These locations can be retrieved from any computer aided drafting software. After the points are retrieved, these are passed onto the Optimizer part. The optimization algorithm computes the minimum distance required by the robot manipulator to complete the assembly. The minimum path is thus generated and the points or nodes that make up this path are then passed onto the inverse kinematics solver to generate the joint parameter values. This way the orientation and joint inputs necessary to position the end effector onto the intended target position is accomplished.



A	B	C	D
0	-1.7...	-0.7...	1.29
1	-2.1...	-1.9...	1.29
2	-0.7...	-1.8...	1.29
3	0.78...	-2.6...	1.29
4	0.94...	-1.8...	1.29
5	1.52...	-2.1...	1.29
6	2.56...	-0.1...	1.29
7	2.43...	0.79...	1.29
8	1.54...	1.43...	1.29

Figure 5.3 Positions Depicting the Assembly Locations.

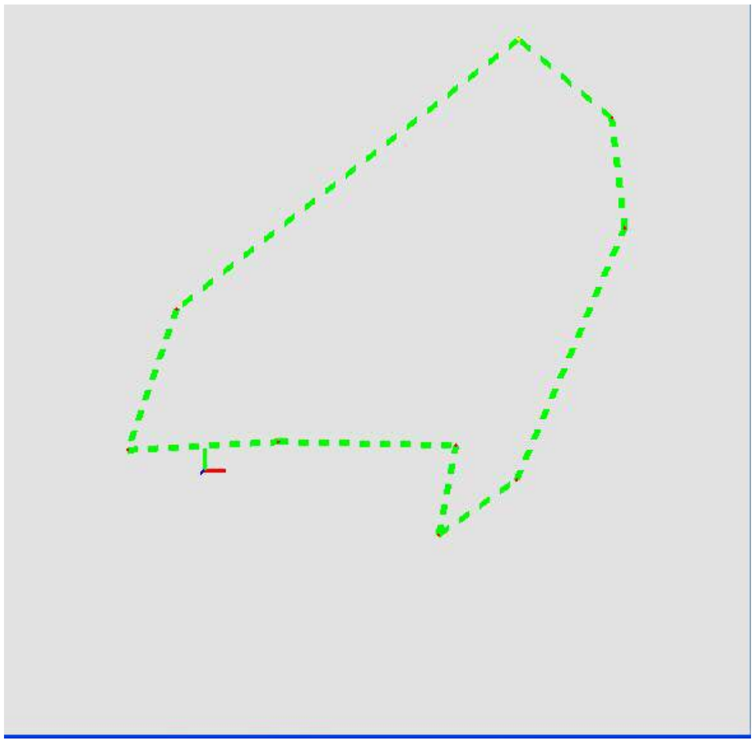


Figure 5.4 Display of the Optimized Path.

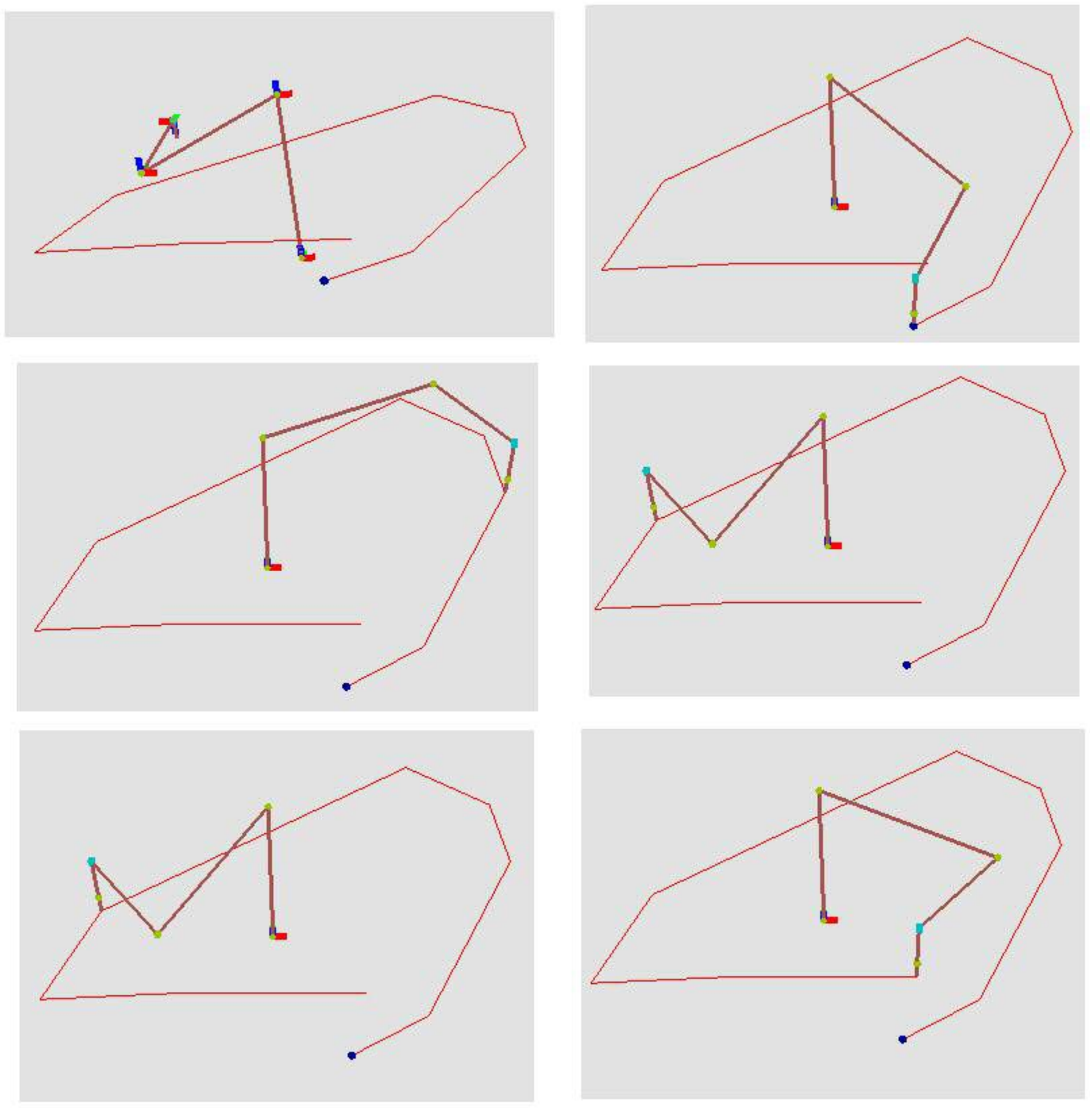


Figure 5.5 Step by Step Depiction of Assembly Process.

5.3 Application of Inverse Kinematics Solver

The inverse kinematics solver can be extended to analyze closed chain mechanisms. The difference between open chain mechanism or robot and closed chain mechanism is that there is end effector also has a constraint. Here is an example to explain the process of kinematic analysis of a four bar mechanism.

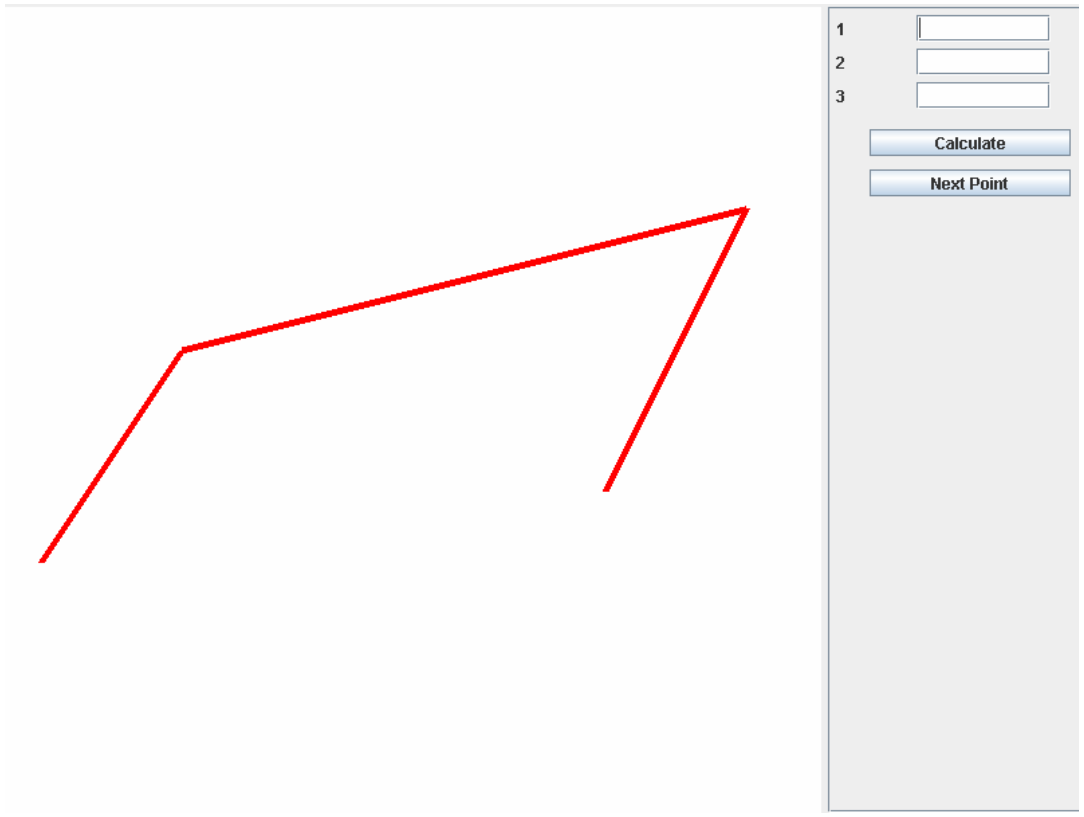


Figure 5.6 Four Bar Mechanism at the Initial Position.

Both the end points of this mechanism are fixed to the ground by using a pin joint. An input angle is given to one of the links. In this case, an input angle of 10 degrees (in counter clockwise direction) is given to first link. The orientation and the position of the other joints are determined. The same inverse kinematics solver is applied to analyze the displacement positions of the joints.

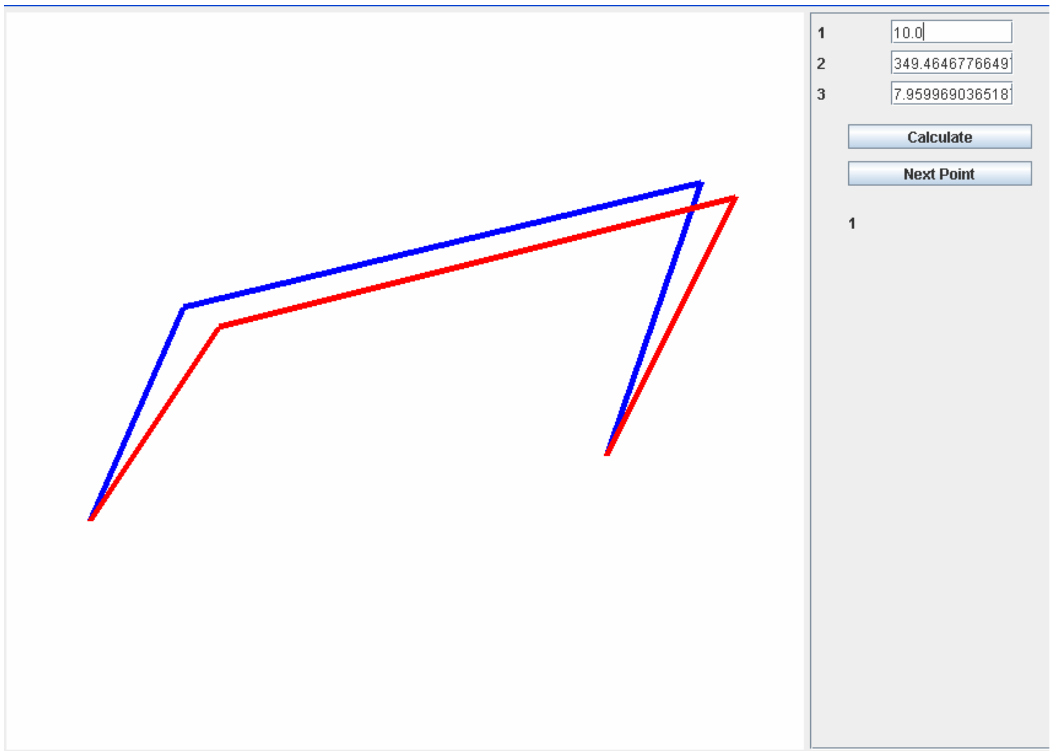


Figure 5.7 Final Position and Orientation of the Links.

Consider a five bar mechanism as shown in figure, the objective is to make the end point of the link protruding from the third link to reach the intended target point. If the target point is within the space in which the mechanism moves, then it is possible to use the same inverse kinematics solver to find a solution, the solution in the form of finding the individual link orientations and joint angles required to reach the target. In this example, the target point considered is 4.48824, 4.74996, figure shows both the initial position and the position where the mechanism end point reaches the target.

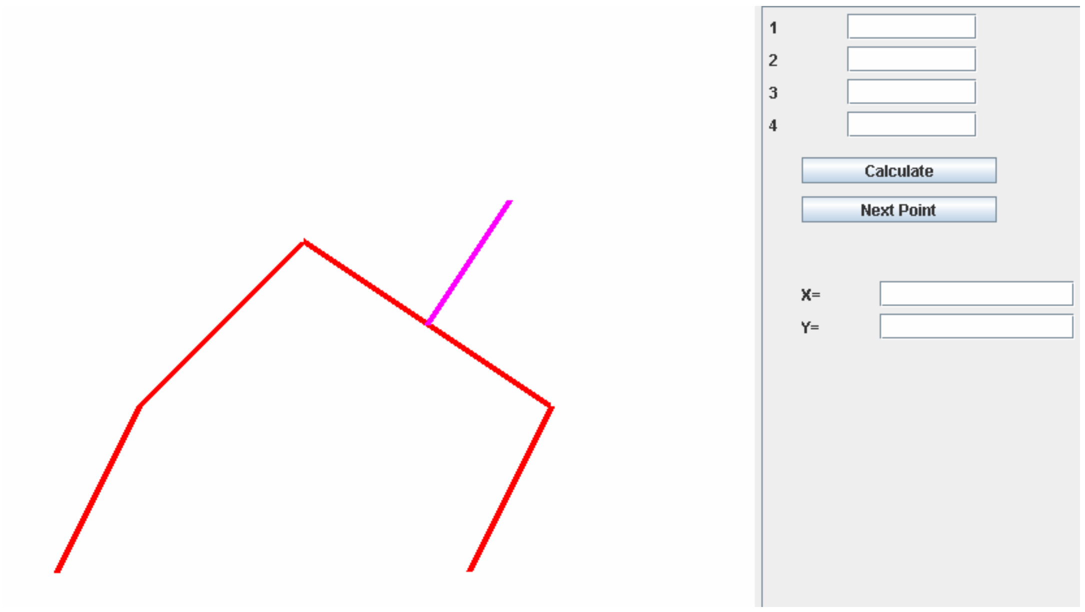
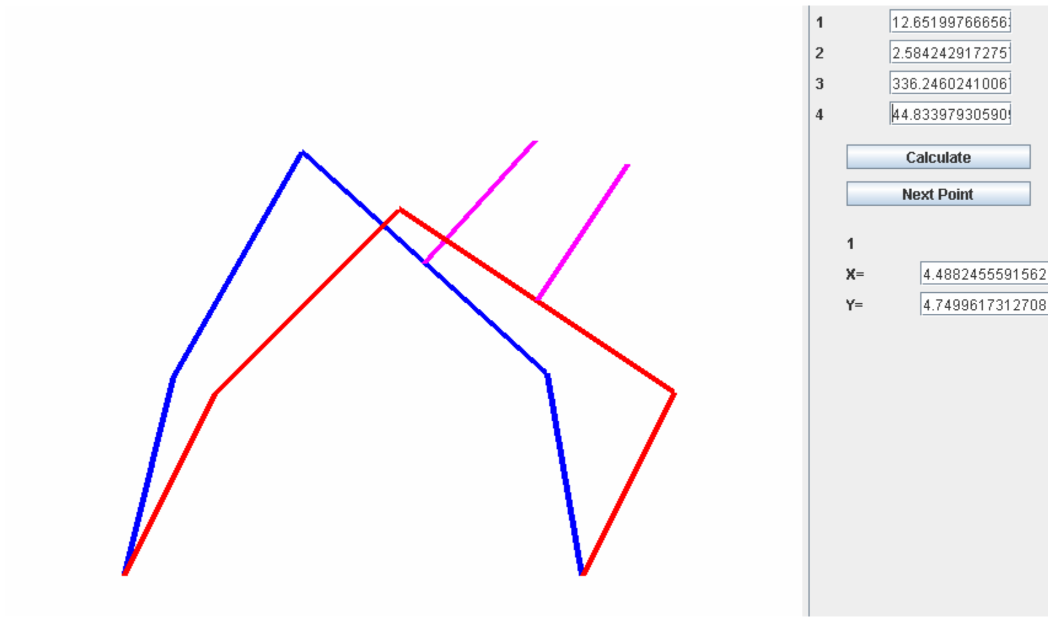


Figure 5.8 Initial Position of the Links.



1 12.65199766656

2 2.584242917275

3 336.2460241006

4 44.83397930590

1

X= 4.4882455591562

Y= 4.7499617312708

Figure 5.9 Final Position of the Links.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Summary and Conclusion

In this dissertation, a new notation was developed to analyze industrial robots. Since it takes into account all the six parameters required to describe a point or body in space, a right way of defining the configuration of the robot can be accomplished. The defects in CB notation arising due to the fact that there is no variable taking care of the translation component, is rectified by this new notation. The user needs to just assign the principal joint direction, i.e. the Z axis direction, and its easier to fix other two axes, thus an opportunity to collapse six parameters to five and hence shrinking the computations necessary to generate transformation matrices. From these matrices, workspace can be generated.

Inverse Kinematic equations are highly non-linear and hence difficult to solve, but by using an iterative numerical technique, inverse kinematics of certain robot configurations can be solved using this technique. The requirement of this technique is that all the joints except the base joint move in the same plane. The same technique can be extended to analyze closed chain mechanisms. The Insertion and Reordering algorithm along with piece-wise algorithm makes sure that a feasible path can be obtained in a very quick time.

The integration of optimization and inverse kinematics in robotics helps minimize the costs, maximize profits and ensure that the supply demand equations are stable.

6.2 Future Work

In the future, it is to be seen if the new notation can be tested on parallel robots and on other complex systems. Also the possibility of extending the inverse kinematics solver to obtain solutions to spatial mechanisms will result in the development of new breed of mechanisms. Synthesis of mechanisms would be a best field to try out the inverse kinematics solver. The link lengths could be generated depending on the points to be traversed by the mechanism. As of now MyRobot tool can be used to analyze revolute, prismatic or a combination of these two, but in the near future, additional joints could be added to analyze a much wider variety of industrial robots.

REFERENCES

[1] http://www.roboticsonline.com/public/articles/archivedetails.cfm?id=1694

[2] M. Dorigo, and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem", IEEE Transactions on Evolutionary Computation, Vol. 1, p. 53-66 (1997).

[3] Huai-Kuang Tsai, Jinn-Moon Yang, and Cheng-Yan Kao, "Solving traveling salesman problems by combining global and local search mechanisms", CEC '02. Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 2, p. 1290-1295 (May 12-17 2002).

[4] Cheng-Fa Tsai, Chun-Wei Tsai, and Ching-Chang Tseng, "A new approach for solving large traveling salesman problem", Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 2, p. 1636-1641 (May 12-17 2002).

[5] Zne-Jung Lee, "A Hybrid Algorithm Applied to Traveling Salesman Problem", Proceedings of the 2004 IEEE International Conference on Networking, Sensing & Control, p. 237-240 (2004).

[6] W. Pullan, "Adapting the genetic algorithm to the travelling salesman problem", The 2003 Congress on Evolutionary Computation, Vol. 2, p. 1029-1035 (Dec 8-12 2003).

[7] Wang Xuan, and Yuanxiang Li, "Solving traveling salesman problem by using a local evolutionary algorithm", IEEE International Conference on Granular Computing, Vol. 1, p. 318-321 (July 25-27 2005).

[8] Cuiru Wang, Jiangwei Zhang, Jing Yang; Chaoju Hu, and Jun Liu, "A Modified Particle Swarm Optimization Algorithm and its Application For Solving Traveling Salesman Problem", International Conference on Neural Networks and Brain, Vol. 2, p. 689-694 (Oct 13-15 2005).

[9] http://cache.ucr.edu/~currie/roboadam.htm

[10] R. L. Hoekstra, “Robotics and Automated Systems”, South-Western Publishing Co, Cincinnati (1986).

[11] Jack Kimbrell, “Kinematics analysis and synthesis”, McGraw-Hill, NY (1991).

[12] J. Denavit, and R. S. Hartenberg, “A Kinematic notation for lower-pair mechanisms based on matrices”, ASME Trans. J. Appl. Mech. 77, p. 215-221 (1955).

[13] J. J. Uicker, Jr., J. Denavit, and R. S. Hartenberg, “An iterative method for the displacement analysis of spatial mechanisms”, ASME Trans. J. Appl. Mech., 86, p. 309-314 (1964).

[14] T. C. Yih, “A new Method for the Geometric Modeling of Lower Pairs and Its Application to the Kinematic Spaces of Spatial Robots”, Journal of Robotic Systems, Vol. 8(4), p. 415-442 (1991).

[15] B. Roth, “Performance Evaluation of Manipulators from a Kinematic Viewpoint’’, in Performance Evaluation of Programmable Robots and Manipulators, NBS Special Publication No. 459, p. 39–61 (Oct 1976).

[16] K. C. Gupta, and B. Roth, "Design Considerations for Manipulator Workspace", ASME J. Mech. Des., Vol. 104, p. 704–711 (1982).

[17] A. Kumar, and K. Waldron, "The Workspaces of a Mechanical Manipulator”, ASME J. Mech. Des., Vol. 103, p. 665–672 (1981).

[18] A. H. Soni, and Y. C. Tsai, “An algorithm for the workspace of a general n-R robot”, ASME J. Mech. Trans. Automat. Des., Vol. 105, p. 52-57 (Mar 1983).

[19] Marco Ceccarelli, "A Formulation for the Workspace Boundary of General N-Revolute Manipulators", Mech. Mach. Theory, Vol. 31, p. 637–646 (1998).

[20] Marco Ceccarelli, "Designing Two-Revolute Manipulators for Prescribed Feasible Workspace Regions", Journal of Mechanical Design, Vol. 124, p. 427-434 (Sept 2002).

[21] J. A. Snyman, L. J. du Plessis, and Joseph Duffy, “An Optimization Approach to the Determination of the Boundaries of Manipulator Workspaces”, ASME J. Mech. Des., Vol. 122, p. 447-456 (Dec 2000).

[22] A. B. Kyatkin, and G. S. Chirikjian, "Synthesis of binary manipulators using Fourier transform on the Euclidean group", ASME J. Mech. Des., Vol. 121, p. 9-14 (Mar 1999).

[23] Yunfeng Wang, and Gregory S. Chirikjian, “A Diffusion-based Algorithm for Workspace generation of Highly Articulated Manipulators”, Robotics and

Automation, Proceedings. ICRA '02. IEEE International Conference, Vol. 2, p. 1525-1530 (May 2002).

[24] Abdel-Malek, K., Yeh, H J, and Othman, S. "Interior and Exterior Boundaries to the Workspace of Mechanical Manipulators", Robotics and Computer-Integrated Manufacturing, Vol. 16, No. 5, p. 365-376 (2000).

[25] Karim Abdel-Malek, and Harn-Jou Yeh, "Crossable Surfaces of Robotic Manipulators With Joint Limits", Transactions of the ASME, Vol. 122, p. 52-60 (Mar 2000).

[26] Z. C. Lia, and C. H. Menq, "The dexterous workspace of simple manipulators", IEEE Journal of Robotics and Automation, Vol. 4 (Issue 1), p. 99-103 (Feb 1988).

[27] H. Zhang, and P. Sikka, "Characterization of the workspace for planar robot manipulators", IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Vol. 1, p. 158-161 (May 9-10 1991).

[28] Marco Ceccarelli, "Designing Two-Revolute Manipulators for Prescribed Feasible Workspace Regions", ASME Journal of Mechanical Design, Vol 124, p. 427-434 (Sept 2002).

[29] Krzysztof Tcho´n, and Robert Muszy´nski, "Singular Inverse Kinematic Problem for Robotic Manipulators: A Normal Form Approach", IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, Vol. 14, No. 1, p. 93 -104 (Feb 1998).

[30] F. Chapelle, and P. Bidaud, "A Closed Form for Inverse Kinematics Approximation of General 6R manipulators using Genetic Programming",

Proceedings of 2001 IEEE, International Conference on Robotics & Automation, Korea, Vol. 4, p. 3364 -3369 (May 2001).

[31] Yang Ming, Lu Guizhang, and Li Jiangeng, "An Inverse Kinematics Solution for Manipulators Based on Fuzzy Logic", IEEE, Vol. 4, p. 400-404 (2001).

[32] Juan Manuel Ahuactzin and Kamal K. Gupta, "The Kinematic Roadmap: A Motion Planning Based Global Approach for Inverse Kinematics of Redundant Robots", IEEE TRANSACTIONS ON ROBOTICS & AUTOMATION, Vol. 15, No. 4, p. 653 -669 (Aug 1999).

[33] Farbod Fahimi, Hashem Ashrafiuon, and C. Nataraj, "An Improved Inverse Kinematic and Velocity Solution for Spatial Hyper-Redundant Robots", IEEE TRANSACTIONS ON ROBOTICS & AUTOMATION, Vol. 18, No. 1, p. 103-107 (Feb 2002).

[34] A. Ramdane-Cherif, B. Daachi, A.benallegue, and N. Levy, "Kinematic Inversion", Proceedings of 2002 IEEE, International Conference on Intelligent Robots & Systems, Switzerland, Vol. 2, p. 1904 -1909 (Oct 2002).

[35] Alberto Borboni, "Solution of the Inverse Kinematic problem of a serial manipulator by a Fuzzy Algorithm", IEEE International Fuzzy Systems Conference, Vol. 1, p. 336-339 (2001).

[36] Luya Li, William A. Gruver, Qixian Zhang, and Zongxu Yang, "Kinematic Control of Redundant Robots and the Motion Optimizability Measure", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS - part B CYBERNETICS, Vol. 31, No.1, p. 155-160 (Feb 2001).

[37] S. K. Chan, and P. D. Lawrence, "General inverse kinematics with the error damped pseudoinverse", IEEE International Conference on Robotics and Automation, Vol. 2, p. 834-839 (Apr 24-29 1988).

[38] Kun Ji, and T. C. Yih, "Inverse kinematic control of robots by applying the C-B notation", Proceedings of the 1995 IEEE IECON 21st International Conference on Industrial Electronics, Control, and Instrumentation, Vol. 1, p. 98-103 (Nov 6-10 1995).

[39] I-Ming Chen, and Yan Gao, "Closed-form inverse kinematics solver for reconfigurable robots", Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, Vol. 3, p. 2395-2400 (2001).

[40] G. Antonelli, S. Chiaverini, G. Fusco, "A new on-line algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits", IEEE Transactions on Robotics and Automation, Vol. 19, p. 162-167 (Feb 2003).

[41] Fernando Durate, J. A. Teneiro Machado, and Laszlo Horvath, "A Trajectory Planning Algorithm for Redundant Manipulators", IEEE ISIE'99, Slovenia, Vol. 3, p. 1002-1007 (1999).

[42] Lianfang Tian, C. Collins, and Renxin Chu, "Optimal trajectory planning of redundant manipulators in constrained workspace", IEEE Electronics Letters, Vol. 38, No. 14, p. 762 -764 (July 2002).

[43] Reza Fotouhi-C., Walerian Szyszkowski and Peter N. Nikiforuk, "Trajectory Planning and Speed Control for a Two-Link Rigid Manipulator", Journal of Mechanical Design, Vol. 124, p. 585-589 (Sept 2002).

[44] Aurelio Piazzi, and Antonio Visioli, "Global Minimum-Jerk Trajectory Planning of Robot Manipulators", IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, Vol. 47, No. 1, p. 140-149 (Feb 2000).

[45] Galicki M and Friedrich-Schiller, "Real-Time Trajectory Generation for Redundant Manipulators with Path Constraints", International Journal of Robotics Research, vol. 20, no. 8, p. 676-693(18) (Aug 2001).

[46] Chien-Chou Lin, and Jen-Hui Chuang, "Potential-based path planning for robot manipulators in 3-D workspace", Proceedings. ICRA '03. IEEE International Conference on Robotics and Automation, Vol. 3, p. 3353-3358 (Sept 14-19 2003).

[47] R. Costa-Castello, and L. Basanez, "Understanding workspace structure of multi-robot systems", Proceedings. ETFA '99. 1999 7th IEEE International Conference on Emerging Technologies and Factory Automation, Vol. 1, p. 129-135 (Oct 18-21 1999).

[48] T. Hamada, K. Kamejima, and I. Takeuchi, "Dynamic work space model matching for interactive robot operation", International Workshop on Industrial Applications of Machine Intelligence and Vision, p. 82-87 (Apr 10-12 1989).

[49] Guilin Yang, and I-Ming Chen, "Equivolumetric partition of solid spheres with applications to orientation workspace analysis of robot manipulators", IEEE Transactions on Robotics, Vol. 22, p. 869-879 (Oct 2006).

[50] Sukhan Lee, Daesik Jang, Eunyoung Kim, Suyeon Hong, and JungHyun Han, "A real-time 3D workspace modeling with stereo camera", IEEE/RSJ International Conference on Intelligent Robots and Systems, p. 2140-2147 (Aug 2-6 2005).

[51] S. Kucuk, and Z. Bingul, "The inverse kinematics solutions of industrial robot manipulators", Proceedings of the IEEE International Conference on Mechatronics, p. 274-279 (Jun 3-5 2004).

[52] S. Kucuk, and Z. Bingul, "The inverse kinematics solutions of fundamental robot manipulators with offset wrist", IEEE International Conference on Mechatronics, p. 197-202 (Jul 10-12 2005).

[53] John J. Craig, "Introduction to Robotics, Mechanics and Control", Addison-Wesley Longman, Inc (1999).

[54] API help files at this website www.java.sun.com.

[55] P. Naughton, and H. Schildt, "The Complete Reference-Java 2", Tata McGraw-Hill Publishing Company Limited (1999).

[56] Venkat Krovi, G. K. Ananthasuresh, and Vijay Kumar, "Kinematic Synthesis of Spatial R-R Dyads for Path Following With Applications to Coupled Serial Chain Mechanisms", Journal of Mechanical Design, Vol. 123, p. 359-366 (Sept 2001).

[57] Daniel C. H. Yang, and Jason W. Rauchfuss, "A Geometric Approach for Determining Exact Point Accessibility of Robotic Manipulations", Journal of Mechanical Design, Vol. 122, p. 287-293 (Sept 2000).

[58] Constantinos Mavroidis, Eric Lee, and Munshi Alam, "A New Polynomial Solution to the Geometric Design Problem of Spatial R-R Robot Manipulators

Using the Denavit and Hartenberg Parameters", Transactions of the ASME, Vol. 123, p. 58-67 (Mar 2001).

[59] Kenichi Hamamoto, and Toshiharu Sugie, "Iterative Learning Control for Robot Manipulators Using the Finite Dimensional Input Subspace", IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, Vol. 18, No. 4, p. 632-635 (Aug 2002).

[60] Belén Curto, Vidal Moreno, and Francisco J. Blanco, "A General Method for C-Space Evaluation and its Application to Articulated Robots", IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, Vol. 18, No. 1, p. 24-31 (Feb 2002).

[61] T. C. Yih, and B. Donoso, "On the 3-Dimensional Workspace of Robots by Applying C-B Notation", Proceedings of National Conference on Applied Mechanisms and Robotics, Cincinnati, Ohio, p. 20-25 (Nov 1989).

[62] Ali Faraz, and Shahram Payandeh, "On inverse Kinematics and Trajectory Planning for Tele-Laparoscopic Manipulation", Proceedings of the 1999 IEEE International conference on Robotics & Automation, Michigan, Vol. 3, p. 1734 – 1739 (May 1999).

[63] Joono Cheong, Wan Kyun Chung, and Youngil Youm, "Inverse kinematics of multilink flexible robots for high-speed applications", IEEE Transactions on Robotics and Automation, Vol. 20, p. 269-282 (Apr 2004).

[64] T. Kishimoto, and Y. Fujimoto, "Numerically stable inverse kinematics calculation of robot manipulators even in singular configuration", 30th Annual

Conference of IEEE on Industrial Electronics Society, Vol. 2, p. 1036-1040 (Nov 2-6 2004).

[65] Jian Xie, Shizuo Yan, and Wenyi Qiang, "A method for solving the inverse kinematics problem of 6-DOF space manipulator", 1st International Symposium on Systems and Control in Aerospace and Astronautics, p. 4 (Jan 19-21 2006).

[66] D. N. Vila-Rosado, and J. A. Dominguez-Lopez, "A Matlab toolbox for robotic manipulators", Sixth Mexican International Conference on Computer Science, p. 256-263 (Sept 26-30 2005).

[67] Z. Bingul, P. Koseeyaporn, and G. E. Cook, "Windows-based robot simulation tools", 7th International Conference on Control, Automation, Robotics and Vision, Vol. 2, p. 1037-1041 (Dec 2-5 2002).

[68] L. Guilamo, J. Kuffner, K. Nishiwaki, and S. Kagami, "Efficient prioritized inverse kinematic solutions for redundant manipulators", IEEE/RSJ International Conference on Intelligent Robots and Systems, p. 3921-3926 (Aug 2-6 2005).

[69] Yoon Young Kim, Gang Won Jang, and Sang Jun Nam, "Inverse kinematics of binary manipulators by the optimization method in a continuous variable space", IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 4, p. 3823-3828 (2004).

[70] A. Hourtash, and M. Tarokh, "Manipulator path planning by decomposition: algorithm and analysis", IEEE Transactions on Robotics and Automation, Vol. 17, p. 842-856 (Dec 2001).

[71] Guo Tong-ying, Qu Dao-kui, and Dong Zai-li, "Research of Path Planning for Polishing Robot Based on Improved Genetic Algorithm", IEEE International Conference on Robotics and Biomimetics, p. 334-338 (Aug 22-26 2004).

[72] D. G. Wilson, R. D. Robinett, and G. R. Eisler, "Discrete dynamic programming for optimized path planning of flexible robots", IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 3, p. 2918-2923 (2004).

[73] Wen Ye, Dengwu Ma, and Hongda Fan, "Path planning for space robot based on the self-adaptive ant colony algorithm", 1st International Symposium on Systems and Control in Aerospace and Astronautics, p. 4 (Jan 19-21 2006).

[74] Robert E. Parkin, "Applied Robotic Analysis", Prentice Hall (1991).

[75] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

BIOGRAPHICAL INFORMATION

Mukund Venkatachari Narasimhan has a Bachelor of Engineering degree in Mechanical Engineering from BMS College of Engineering, Bangalore, India and a Master of Science in Mechanical Engineering from University of Texas at Arlington. He is a member of Tau Beta Pi, a National Engineering Society. He has worked in the field of CAD CAM and has extensive experience in this field. His interests include robotics, kinematics, computer programming, mathematical modeling and optimization.