ISTANBUL TECHNICAL UNIVERSITY ⋆ INSTITUTE OF SCIENCE & TECHNOLOGY

KINEMATIC MODELING AND PATH PLANNING WITH COLLISION AVOIDANCE FOR MULTIPLE CARTESIAN ARMS

MASTER OF SCIENCE

Nihan Yeşiloğlu

Department: INTERDISCIPLINARY PROGRAM
Program: MECHATRONICS ENGINEERING

Advisor: Assoc. Prof. Dr. Hakan Temeltaş

January 2006

ISTANBUL TECHNICAL UNIVERSITY ⋆ INSTITUTE OF SCIENCE & TECHNOLOGY

KINEMATIC MODELING AND PATH PLANNING WITH COLLISION AVOIDANCE FOR MULTIPLE CARTESIAN ARMS

MASTER OF SCIENCE

Nihan Yeşiloğlu

Department: INTERDISCIPLINARY PROGRAM
Program: MECHATRONICS ENGINEERING

January 2006

ACKNOWLEDGEMENT

During my effort to succeed the master of science degree, first of all I am really grateful to my husband Murat Yeşiloğlu for his support to get rid of problems and for his patience, and my advisor Assoc. Prof. Dr. Hakan Temeltaş for his support and help. Finally, I am also grateful to my family for their support in my education life.

January 2006 Nihan Yeşiloğlu

CONTENTS

LIST OF TABLES

Tables

LIST OF FIGURES

Figures

LIST OF SYMBOLS

I	:	Identity Matrix
θ_k	:	angle of joint k
$\Delta\theta$	:	change of angle of the joints
$\widehat{k}$	:	: skew symetric of axis of rotation of joints
V_t	:	tip point's velocity
t_0	:	initial time
t_f	:	final time
Δt	:	control sampling period for the manipulator
$h_i(t)$	:	trajectory function for segment i
t	:	normalized time variable
τ	:	real time in seconds
τ_i	:	real time at the end of the ith trajectory segment
t_i	:	real time required to travel through the ith trajectory segment
θ_0	:	initial position
$\theta_1 : \theta(t_1)$	:	lift-off position
$\theta_2 : \theta(t_2)$	:	set-down position
$\theta_f : \theta(t_f)$	:	final position
V_0	:	initial velocity
a_0	:	initial acceleration
V_f	:	final velocity
a_f	:	final acceleration
x	:	unknown coefficients matrix of polynomial trajectories
Q	:	Quaternion
Q^*	:	Conjugate of quaternion
r	:	minimum distance between the robot links

ÖZET

Kartezyen robotlar, endüstride geniş kullanım alanı bulmaktadır. Birden fazla kartezyen robotun ortak bir iş yapmasına gerek duyulan durumlar vardır. Bu tezde yapılan çalışmanın temeli, aynı çalışma uzayındaki kartezyen robotların çarpışma olmaksızın yörünge planlamasıdır. Bu çalışmanın amacı, aynı çalışma uzayındaki kartezyen robotların konumlandırılması için gerekli algoritmaları bulmak veya türetmektir.Çarpışma sakınımlı yörünge planlaması algoritmalarını kullanarak istenen işın başarılması uzaysal işlem cebriyle kinematik olarak modellenmış robotık sısteme dayanır. Yörünge planlaması metodları kartezyen robotlara uygulanarak çarpışma olmayan yörüngenin bulunması için algoritmalar geliştirilir.

SUMMARY

Cartesian robots are already being widely used in industry and their use will substantially increase as the developing technology brings the prices down of high payload and high precision linear motors. There are applications where more than one cartesian robots are required to perform a common task. The focus of the research presented in this thesis is the collision free path planning for multiple cartesian robots sharing the same task space. The objective of this research is to obtain or derive necessary algorithms to coordinate multiple cartesian robots sharing the same workspace. Using path planning algorithms with collision avoidance, the desired task is achieved based on the kinematic model of the complete robotic system which is rooted in the spatial operator algebra. Path planning methods are applied to the cartesian robots and the algorithms to find collision-free path for the cartesian robots are developed.

CHAPTER 1

Introduction

The field of robotics has been rapidly growing ever since its first conceptual introduction in 1920. Advances in electric machinery and materials made a big impact on the hardware of the robots today as the payload capacity to weight ratio substantially increased. As conventional motors provide rotary motions, revolute joints became more popular than the prismatic ones. However, recent technology gives rise to high precision linear actuators. Therefore, it is reasonable to expect in near future that prismatic joints will claim the same popularity as the revolute ones have. Cartesian robots are already being widely used in industry, and their use will substantially increase as the developing technology brings the prices down of high payload and high precision linear motors. There are applications where more than one cartesian robots are required to perform a common task. The focus of the research presented in this thesis is the collision free path planning for multiple cartesian robots sharing the same task space.

1.1 Background and Motivation

Gantry robots or cartesian robots in general have a large application area such as "pick and place" type of use. When there are multiple cartesian robots needed to perform a task, the coordination among them has to be provided. This coordination also needs to cover collision avoidance problem. This research concentrates on the very same issue.

1.2 Objective of the Research

The objective of this research is to obtain or derive necessary algorithms to coordinate multiple cartesian robots sharing the same workspace. Using path planning algorithms with collision avoidance, the desired task is achieved based on the kinematic model of the complete robotic system which is rooted in the spatial operator algebra. This objective is carried through by the following two stages:

- Path planning algorithms on the kinematic model of a single cartesian robot using spatial operator algebra.
- Collision free path planning algorithms for a robotic system consisting of multiple cartesian robots.

1.3 Work Completed

Two 6 DOF Cartesian robots were kinematically modeled using Spatial Operator Algebra. Joint velocities were propagated from base to the tip point yielding frame independent linear and angular velocity vectors of the tip point. Path planning, on the other hand, was done to make sure the desired position and orientation were achieved at the desired time while the motion is smooth. Smoothness here implies that the accelerations (second derivatives) are continues. The developed paths were then applied to the multiple robot system and were corrected to avoid collision, based on the collision-avoidance algorithm. The planned path which became free of collision was checked last for joint acceleration limits. To do that, joint velocities were calculated using inverse kinematics, yielding the joint accelerations by taking time derivative of calculated joint velocities.

1.4 Remaining Work

The work described in this thesis will be applied to a real system to be manufactured. This will be possible by the grant under a project with TUBITAK.

1.5 Literature Review

Spatial operator algebra is used for kinematic and dynamic modeling of rigid multibody systems and allows a systematic formulation of the dynamical equations of motion of multibody systems and the development of efficient computational algorithms. Featherstone [1], presents a work on the computation of robot dynamics using articulated body inertias. After that, the works on spatial operator algebra continues by Rodriguez. Rodriguez[2], states in its paper that the inverse and forward dynamic of problems for multi-link serial manipulators are solved by using recursive techniques from linear filtering and smoothing theory. The dynamics of multibody systems is covered in the book of Kane and Levinson with the formulation and applications[3]. Jain[4], states in its paper that a unified formulation about serial rigid multibody systems can be developed. Rodriguez *et. al.*, propose usage of spatial operator algebra in manipulator dynamics, mass matrix computation and the application of the method to perform high-level ma-

nipulation in [5]. Other works of Rodriguez et al. on spatial operator algebra are mentioned in [6],[7],[8], [9], [10],[11], [12], [13], [14].

There are a number of methods for path planning of robot manipulators, which are classified into two major approaches: the joint interpolated approach and cartesian space approach. The joint interpolated approaches plans polynomial trajectories that yield smooth trajectories. The methods for polynomial trajectory generation is mentioned in [15],[16]. In addition to these, the polynomial curve fitting methods' comparison by simulation is explained in [17],[18] and [19]. On the other hand, there are sampling- based path planning methods and combinatorial path planning methods which are discussed in various works in literature [20], [21],[22],[23],[24]. The Cartesian space approaches cover Homogeneous transformation matrix approach, planning straight line trajectories using quaternion representation and bounded deviation joint path [25],[26].

In the case that, more than one robot operate simultaneously in a common workspace, the problem of avoiding potential collisions between the robots should be considered carefully. For collision avoidance problem, zone blocking methods can be used to avoid collision problem. Besides zone-blocking methods, some other collision avoidance methods are proposed in [20], [27], [28],[29] and [30]. Lee et al [27], presented several time adjusting methods for two robots by collision map. In the paper, robot arm is modeled by a sphere. Chang [31] , proposes an effective collision avoidance method for two robot manipulators which are approximated by polyhedra as the extend of Lee et al.This paper determines minimum time delay needed for avoiding collisions between the robots by using distance functions.In a similar method, in which the complex 3D problems are changed to simple 2D ones, Wu *et. al.* [32] proposed that links of robots in 3D can be simplified to a 2D Space/Time graph. Robots can move with the proper velocity to avoid potential collisions with obstacles or with other robots by constructing an optimal path on the Space/Time graph [32].

1.6 Organization of the Thesis

In Chapter 2, the derivation of kinematical model of multiple robots sharing the same workspace is given. How the path planning is done is the topic of Chapter 3. Collision avoidance is being covered in Chapter 4. Chapter 5 is for the simulation results. Finally, Chapter 6 concludes the thesis.

CHAPTER 2

Kinematic Modeling of Multiple Cartesian Arms with Common Task Space

2.1 Spatial Operator Algebra

Spatial operator algebra is a recursive method that uses coordinate-free vectorial notation. It is useful for both kinematic and dynamic modeling of manipulators. Dynamical modeling, however, is not in the scope of this thesis. The main advantage of spatial operator algebra, as far as this research is concerned, is the fact that it brings a systematic way while keeping the physical insight to the model.

Let $\vec{h}_k$ be the axis of rotation vector of the link k. O_k is a point located on $\vec{h}_k$. The link vector from O_k to O_{k+1} is given by $\vec{\ell}_{k,k+1}$. Angular velocity vector of link k is $\vec{\omega}_k$, and the linear velocity of link k at point O_k is $\vec{v}_k$.

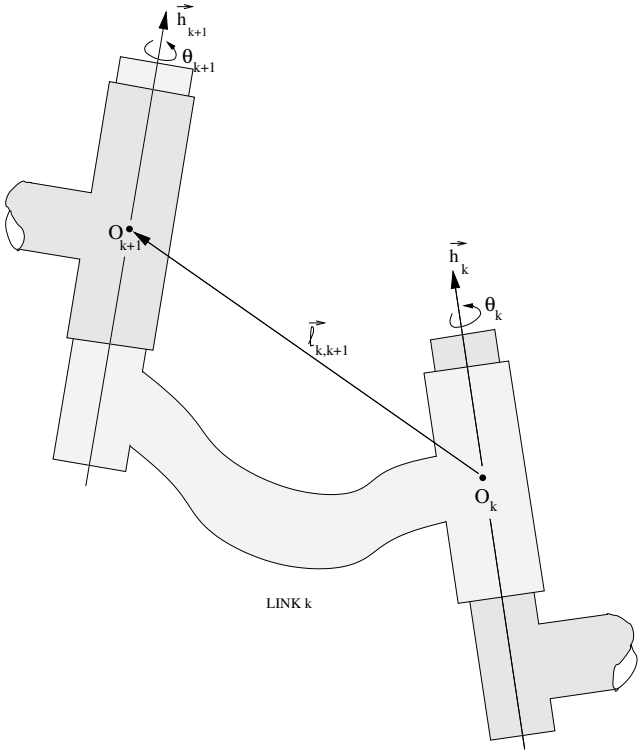

Figure 2.1: Link k

A rigid link is depicted in figure 2.1. The link velocities are propagated from base to tip.

$$\vec{\omega}_k = \vec{\omega}_{k-1} + \vec{h}_k \dot{\theta}_k \tag{2.1}$$

$$\vec{v}_k = \vec{v}_{k-1} - \widehat{\ell}_{k-1,k} \times \vec{\omega}_{k-1} \tag{2.2}$$

When we put the equations (2.1) and (2.2) into the matrix form, we get

$$V_k = \phi_{k,k-1} H_k \dot{\theta}_k \tag{2.3}$$

where

$$V_k = \begin{bmatrix} \vec{\omega}_k \\ \vec{v}_k \end{bmatrix} \tag{2.4}$$

$$\phi_{k,k-1} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{k-1,k} & I \end{bmatrix} \tag{2.5}$$

$$H_k = \begin{bmatrix} \vec{h}_k \\ 0 \end{bmatrix} \tag{2.6}$$

Equation (2.6) is used for revolute joints. If the joint is prismatic then H_k is defined as equation (2.7).

$$H_k = \begin{bmatrix} 0 \\ \vec{h}_k \end{bmatrix} \tag{2.7}$$

The rigid recursion operator ϕ is defined as;

$$\phi_{k-1,k} = \begin{bmatrix} I & & & & \\ \phi_{2,1} & I & & & \\ \phi_{3,1} & \phi_{3,2} & I & & \\ \cdot & \cdot & \cdot & \cdot & \\ \phi_{n,1} & \cdot & \cdot & \phi_{n,n-1} & I \end{bmatrix} \tag{2.8}$$

where k is the k^{th} link and $k-1$ is k-1. previous link (k is starting from zero to the number of joints) and $\phi_{k-1,k}$ can be used to compute spatial recursions starting from the base to tip. The operator H, converts the scalar rotational rates along the joint axes into 6-dimensional spatial velocities across the joints. H is a diagonal matrix and is expressed as; $H = diag[H_1, H_2, ...H_n]$. [33].

2.2 Kinematic Modeling of 6 DOF Cartesian Manipulator using spatial vectors

In the cartesian robot, there are four joints; first three of them are prismatic and the last one is spherical joint. H_k is a diagonal matrix and formed by the spatial vectors of the joints according to the base frame.

$$\begin{bmatrix} \vec{\omega}_k \\ \vec{v}_k \end{bmatrix} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{k-1,k} & I \end{bmatrix} \begin{bmatrix} \vec{\omega}_{k-1} \\ \vec{v}_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \vec{h}_k \end{bmatrix} \dot{\theta}_k \tag{2.9}$$

In the spatial operator algebra, the equation 2.9 is the velocity expression for prismatic joint. First matrix V_k is the velocity matrix which is produced by the angular velocity ω_k and linear velocity v_k. The second matrix is known as the propagation matrix,$\phi_{k,k-1}$, which propagates from the link k to $k-1$. The last matrix in the equation 2.9 is formed by spatial vectors whose first row is zero for prismatic joint, since there is no angular rotation.

The $\vec{H}_k$ is diagonal of $\vec{H}_1$, $\vec{H}_2$, $\vec{H}_3$ and $\vec{H}_4$. In equation 2.10, spatial vectors of the four joints is expressed.

$$\vec{H}_k = \begin{bmatrix} \vec{H}_1 & & & 0 \\ & \vec{H}_2 & & \\ & & \vec{H}_3 & \\ 0 & & & \vec{H}_4 \end{bmatrix} \tag{2.10}$$

The $\vec{H}_1$, $\vec{H}_2$ and $\vec{H}_3$ are shown in equation 2.11 and since, first three joints are prismatic joint, the first row of the H matrixes of the joints are zero.

$$\vec{H}_1 = \begin{bmatrix} 0 \\ \vec{h}_1 \end{bmatrix} \ , \ \vec{H}_2 = \begin{bmatrix} 0 \\ \vec{h}_2 \end{bmatrix} \ , \ \vec{H}_3 = \begin{bmatrix} 0 \\ \vec{h}_3 \end{bmatrix} \tag{2.11}$$

The fourth joint is 3 DOF spherical joint and its H vector is modeled as the $\vec{H}_4$ shown in 2.12. In here, $\vec{h}_x$, $\vec{h}_y$ and $\vec{h}_z$ are unit spatial vectors which are chosen according to the frame assignments and the vectors expresses the rotation in x, y and z axes. It is one 3 DOF joint composed of three revolute joints.

$$\vec{H}_4 = \begin{bmatrix} \vec{h}_x & \vec{h}_y & \vec{h}_z \\ 0 & 0 & 0 \end{bmatrix} \tag{2.12}$$

Since $h_x = [1;\ 0;\ 0]$, $h_y = [0;\ 1;\ 0]$, and $h_z = [0;\ 0;\ 1]$; the first three row of the $\vec{H}_4$ will be equal to the identity matrix.

$$\vec{H}_4 = \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{2.13}$$

The identity in in 2.13 is 3×3 and then the $\vec{H}_4$ is 6×3. Since, the size of $\vec{H}_1$, $\vec{H}_2$ and $\vec{H}_3$ are each 6×1, the size of $\vec{H}_k$ is 24×6.

2.3 Frame Assignment and Kinematic Model for Cartesian Robot

The frame assignment for cartesian robot 1 is shown in equations and spatial vectors are chosen in the initial configuration of the robots are shown in appendix.

$$\vec{h}_1 = \vec{y}_1 \ , \ \vec{h}_2 = \vec{z}_1 \ , \ \vec{h}_3 = \vec{x}_1 \tag{2.14}$$

$$\vec{l}_{1,2} = 2\vec{y}_1 \ , \ \vec{l}_{2,3} = -\vec{z}_1 \ , \ \vec{l}_{3,4} = -0.5\vec{z}_2 \ , \ \vec{l}_{4,t} = 0.5 = \vec{z}_3 \tag{2.15}$$

The frame assignment and unit vectors for cartesian robot 2 are also expressed in the equations above.

$$\vec{h}_1 = \vec{z}_1 \ , \ \vec{h}_2 = \vec{x}_1 \ , \ \vec{h}_3 = \vec{y}_1 \tag{2.16}$$

$$\vec{l}_{1,2} = 1\vec{z}_1 \ , \ \vec{l}_{2,3} = 2\vec{y}_1 \ , \ \vec{l}_{3,4} = 0.5\vec{y}_2 \ , \ \vec{l}_{4,t} = 0.5\vec{z}_2 \tag{2.17}$$

A ϕ matrix, known as propagation matrix, is generally $\Phi_{k-1,k}$ and shown in 2.9 is expressed in equation 2.18 and 2.19 for the four joints of cartesian robot.

$$\phi_{2,1} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{1,2} & I \end{bmatrix} \phi_{3,1} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{1,3} & I \end{bmatrix} \phi_{3,2} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{2,3} & I \end{bmatrix} \tag{2.18}$$

$$\phi_{4,1} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{1,4} & I \end{bmatrix} \phi_{4,2} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{2,4} & I \end{bmatrix} \phi_{4,3} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{3,4} & I \end{bmatrix} \tag{2.19}$$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} I & & & 0 \\ \phi_{2,1} & I & & \\ \phi_{3,1} & \phi_{3,2} & I & \\ \phi_{4,1} & \phi_{4,2} & \phi_{4,3} & I \end{bmatrix} \begin{bmatrix} H_1 & & & 0 \\ & H_2 & & \\ & & H_3 & \\ 0 & & & H_4 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} \tag{2.20}$$

Since equation 2.20 represents $\phi \cdot H \cdot \theta$, the obtained matrix is the joint's velocity matrix of the cartesian robots.[5]. The size of the propagation matrix ϕ is 24×24, the size of $\vec{H}$ is 24×6 and the size of $\dot{\theta}$ matrix is 6×1, then the size of joint's velocity matrix is 24×1. Therefore, $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$ and $\vec{v}_4$ have both angular and linear velocity components, which both have size 3×1 and then the size of the velocity matrix for one joint is 6×1.

2.4 Jacobian Computation for Cartesian Arms

The Jacobian of the Cartesian robot can be found by the help of propagation matrix ϕ, spatial vector matrix H and σ_t.[8]

$$J = \sigma_t \quad \phi \quad H \tag{2.21}$$

From the equation 2.21, jacobian of the robot can be computed directly. σ_t matrix shown in 2.22, is composed of $\phi_{t,4}$ matrix which is the propagation from fourth joint to the tip and the zero matrix in 2.23. When the $\vec{\ell}_{t,4}$, which means the distance between the tip point and joint 4 is zero, the propagation matrix, $\phi_{t,4}$, becomes an identity matrix whose size is 6×6. The number of zeros in the σ_t matrix represents the total number of joints except the last joint.

$$\sigma_t = \begin{bmatrix} 0 & 0 & 0 & \phi_{t,4} \end{bmatrix} \tag{2.22}$$

The ϕ matrix in 2.23, is 6×6 matrix and σ_t matrix in 2.22 is 6×24 matrix. So, the jacobian matrix for the cartesian arm will be 6×6 matrix.

$$\phi_{t,4} = \begin{bmatrix} I & 0 \\ -\widehat{\ell}_{t,4} & I \end{bmatrix} \tag{2.23}$$

2.5 Rotation of Frames, Rotation Matrix and Velocity Computation for Cartesian Arms

The Propagation matrix ϕ, the H matrix are written for the initial frame assignment in previous sections. With the given θ the propagation matrix and the H matrix is changed by the help of the rotation matrix R.

$$R = I + \sin(\Delta\theta)\widehat{k} + (1 - \cos(\Delta\theta))\widehat{k}^2 \tag{2.24}$$

For calculating the rotation matrix R, Rodriguez's Formula as seen in the equation 2.24 is used [34]. $\widehat{k}$ in the equation is the skew symmetric matrix of calculated $\vec{H}_k$ vectors.

$$\theta_k = \theta_{k-1} + dt \times \dot{\theta}_{k-1} \tag{2.25}$$

According to the equation 2.25, the angle of the link k is equal to sum of the angle of the link k-1 which is the previous link, and the change in the angle which is $\dot{\theta}_{k-1} \times dt$. Thus, the $\theta_k - \theta_{k-1}$ is the change of the angle of link k-1 to link k which is $\Delta\theta$.

The Rotation matrix is changed by the given θ and $\vec{x}; \vec{y}; \vec{z}$ spatial vectors change the ϕ and H matrix.

The $\dot{\theta}$ is given to the system as input and $\Delta\theta$ and θ is calculated in given conditions. Then, the tip point's velocity is calculated from 2.26 using Jacobian and $\dot{\theta}$ and can be given as;

$$V_t = J \cdot \dot{\theta} \tag{2.26}$$

This V_t, have size of 6×1 and is composed of angular and linear velocity components both have size of 3×1. First three of the components is angular velocity vectors and the last three components are linear velocity components.

CHAPTER 3

Path Planning

3.1 Path Planning Methods

Path Planning is often formulated by transforming the workspace volume occupied by the robot into a single vector or point in the robot configuration space. (C-space). The workspace obstacles are transformed into the forbidden regions of the C-space. As a result, a collision-free robot path is a curve, which circumvents the forbidden regions in the C-space.

Before moving the robot arm, it has to be known whether there are any obstacles present in the path that the robot arm has to traverse (obstacle constraint) and whether the arm's end effector needs to traverse along a specified path(path constraint). The control problem of a manipulator can be divided into two subproblems: the trajectory planning subproblem and the motion control subproblem [25].

The goal of trajectory planning is to generate the reference inputs to the motion control system which ensures that the manipulator executes the planned trajectories. Planning consists of generating a time sequence of the values attained by a polynomial function interpolating the desired trajectory. There are two ways of trajectory planning which is in joint variable space and in the cartesian space [35].For Cartesian space planning,the time history of the end effector's position, velocity and acceleration are planned and the corresponding joint positions, velocities and accelerations are found from the tip point's information [25].Trajectory planning in cartesian space, allows accounting for the presence of path constraints, these are due to the regions of workspace, which are forbidden to the arm, eg. due to the interference with the obstacles. If it is desired to plan a trajectory in the joint space, the values of joints have to be determined from the end effector's orientation and position information [35]. Planning in the joint variable space has three advantages:

- the trajectory can be planned directly during motion,
- the trajectory planning can be done in near real time,

- the joint trajectories are easier to plan [25].

In general; the basic algorithm for generating joint trajectory set points is quite simple. When $t = t_0$, there is a loop: wait for the next control interval. $t = t+\Delta t$; h(t) is found which is the manipulator joint position should be at time t. If $t = t_f$, then exit and go to loop. In here; Δt is the control sampling period for the manipulator. From the algorithm; we see that computation consists of a trajectory function h(t) that should be updated in every control interval.

Thus, there are four constraints that are effective on the planned trajectory. These are;

- the trajectory set points must be readily calculable in a non-iterative manner,
- intermediate positions should be specified deterministically,
- the joint position and its first and second derivatives must be continuous
- the planned path is smooth and finally undesirable motions, such as "wandering" must be minimized [25].

3.2 Joint Space Path Planning

The planning algorithm generates a function $h(t)$ interpolating the given vectors of joint variables at each point, due to the constraints. In general, this algorithm is required to specify the following features:

- The generated trajectories must not be demanding from a computational view.

- Joint positions, velocities and accelerations must be continuous of time.

- Undesirable effects such as non-smooth trajectories on the path must be minimized [35].

The basic principles in joint space path planning is summarized below.

1) When picking up an object, the motion of the hand should be directed away from an object. Otherwise the hand may crash to the object.

2) If we specify a departure position (lift-off) along the normal vector to the surface out from the initial position and we require the end-effector to pass through this position, then admissible departure motion is occurred. If we specify the time required for this motion, we should control the speed at which the object to be lifted.

3) The same set of lift-off requirements for the arm motion are also true for the set-down point of the final position motion.

4) There are four positions for each arm motion : initial, lift-off, set-down and final.(Figure 3.1).

5) Position constraints are initial position, lift-off position, set-down position and final position. At initial position velocity and acceleration is usually given as zero, at lift-off position the motion is continuous for the intermediate points. Set-down position is same as the lift-off position. At the final position, velocity and accelerations are normally zero.

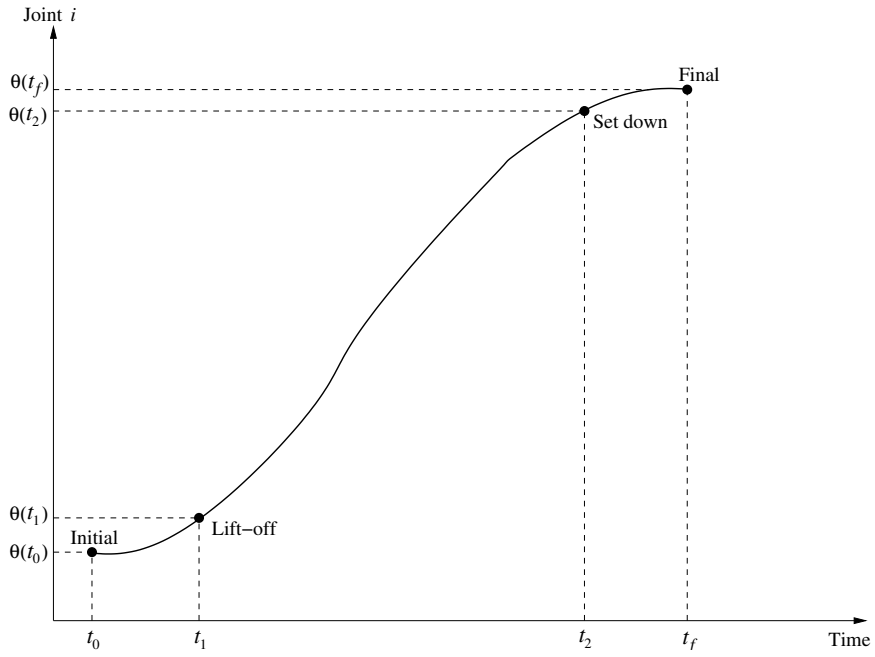

Figure 3.1: Position conditions for a joint trajectory

In addition, the extrema of all joint trajectories must be within the physical and geometric limits of each joint.[25].

According to these constraints, we select a class of polynomial functions of degree n or less such that the required joint position, velocity, and acceleration at the knot points,(initial, lift-off, set-down and final positions) are satisfied and the joint position, velocity and acceleration are continuous on the time interval $[t_0, t_f]$. (Figure 3.1).

One method is to specify a seventh degree of polynomial for each joint i,

$$h_i(t) = a_7t^7 + a_6t^6 + a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \tag{3.1}$$

where the unknown coefficients a_j can be determined from the known positions and continuity conditions. However, it is difficult to find the extrema of the high degree polynomial in equation (3.1), and it also has undesirable motion. An alternative method to this, to split the trajectory into several trajectory segments. So that, lower degree polynomials can be used to interpolate in each trajectory segments.

There are different ways for splitting a joint trajectory and the most common methods are *4-3-4 Trajectory, 3-5-3 Trajectory* and *5-Cubic Trajectory*. In 4-3-4 Trajectory, each joint has three segments. the first segment is a fourth degree polynomial from initial position to the lift-off position. The second segment is the third-degree polynomial from the lift-off position to the set-down position. The last segment is the fourth-degree polynomial from the set-down to the final position. In 3-5-3 Trajectory, each joint has three segments. the first segment is a thirth- degree polynomial from initial position to the lift-off position. The second segment is the fifth-degree polynomial from the lift-off position to the set-down position. The last segment is the third-degree polynomial. In 5-Cubic Trajectory, cubic spline polynomials of third-degree are used with five trajectory segments.

3.2.1 4-3-4 Polynomial Trajectory

When time varying from $t = 0$(initial time) to $t = 1$(final time), we determine N joint trajectories in each segment and a normalized time variable, $t \in [0, 1]$ which allows us to treat equations of each trajectory segments for each joint angle in the same way. The variables are defined as,

t = normalized time variable, $t \in [0, 1]$.
τ = real time in seconds.
τ_i = real time at the end of the ith trajectory segment.
t_i = real time required to travel through the ith segment.

$$t_i = \tau_i - \tau_{i-1} \tag{3.2}$$

$$t = \frac{\tau - \tau_{i-1}}{\tau_i - \tau_{i-1}} \tag{3.3}$$

The 4-3-4 trajectory consists of polynomial segment $h_i(t)$ which forms the joint trajectory for joint i.The polynomial equations for each segments shown in equations (3.4), (3.5), (3.6). $h_1(t)$ is the first segment, $h_2(t)$ is the second segment and $h_n(t)$ indicates the last trajectory segment.

$$h_1(t) = a_{14}t^4 + a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} \tag{3.4}$$

$$h_2(t) = a_{23}t^3 + a_{22}t^2 + a_{21}t + a_{20} \tag{3.5}$$
$$h_n(t) = a_{n4}t^4 + a_{n3}t^3 + a_{n2}t^2 + a_{n1}t + a_{n0} \tag{3.6}$$

The boundary conditions of the polynomials are shown below;

1) Initial position is $\theta_0 = \theta(t_0)$.

2) Magnitude of the initial velocity is v_0.

3) Lift-off position is $\theta_1 = \theta(t_1)$.

4) Continuity in position, velocity and acceleration at t_1.

5) Set down position is $\theta_2 = \theta(t_2)$.

6) Continuity in position, velocity and acceleration at t_2

7) Final position is $\theta_f = \theta(t_f)$, magnitudes of final velocity and acceleration are v_f and a_f.

The first and second derivatives of these polynomial equations can be written as in (3.7) and (3.8);

$$v_i(t) = \frac{dh_i(t)}{d\tau} = \frac{dh_i(t)}{dt}\frac{dt}{d\tau} = \frac{1}{\tau_i - \tau_{i-1}}\frac{dh_i(t)}{dt} = \frac{1}{t_i}\dot{h}_i(t) \tag{3.7}$$

$$a_i(t) = \frac{d^2h_i(t)}{d\tau^2} = \frac{1}{(\tau_i - \tau_{i-1})^2}\frac{d^2h_i(t)}{dt^2} = \frac{1}{t_i^2}\ddot{h}_i(t) \tag{3.8}$$

For the first polynomial shown in equation (3.4),

$$v_1(t) = \frac{\dot{h}_1(t)}{t_1} = \frac{a_{14} + a_{13} + a_{12} + a_{11} + a_{10}}{t_1} \tag{3.9}$$
$$a_1(t) = \frac{\ddot{h}_1(t)}{t_1^2} = \frac{12a_{14}t^2 + 6a_{13}t + 2a_{12}}{t_1^2} \tag{3.10}$$

For $t = 0$;

$$a_{10} = h_1(0) = \theta_0 \tag{3.11}$$
$$v_0 = \frac{\dot{h}_1(0)}{t_1} = \frac{a_{11}}{t_1} \quad which \quad gives \quad a_{11} = v_0 t_1 \tag{3.12}$$
$$a_0 = \frac{\ddot{h}_1(0)}{t_1^2} = \frac{2a_{12}}{t_1^2} \quad which \quad gives \quad a_{12} = \frac{a_0 t_1^2}{2} \tag{3.13}$$

In the second polynomial shown in equation (3.5), for $t = 0$;

$$a_{20} = h_2(0) = \theta_2(0) \tag{3.14}$$

$$v_1 = \frac{\dot{h}_2(0)}{t_2} = \frac{a_{21}}{t_2} \quad , \quad which \quad gives \quad a_{21} = v_1 t_2 \tag{3.15}$$

$$a_1 = \frac{\ddot{h}_2(0)}{t_2^2} = \frac{2a_{22}}{t_2^2} \quad which \quad gives \quad a_{22} = \frac{a_1 t_2^2}{2} \tag{3.16}$$

Since the velocity and acceleration at this point should be continuous with end of the first segment's velocity and acceleration;

$$\frac{\dot{h}_2(0)}{t_2} = \frac{\dot{h}_1(1)}{t_1} \tag{3.17}$$

$$\frac{\ddot{h}_2(0)}{t_2^2} = \frac{\ddot{h}_1(1)}{t_1^2} \tag{3.18}$$

For the last fourth degree polynomial, which is the last trajectory segment in (3.6); when the boundary conditions are applied;

$$a_{n0} = h_n(0) = \theta_f \tag{3.19}$$

$$v_f = \frac{\dot{h}_n(0)}{t_n} = \frac{a_{n1}}{t_n} which \quad gives \quad a_{n1} = v_f t_n \tag{3.20}$$

$$a_f = \frac{\ddot{h}_n(0)}{t_n^2} = \frac{2a_{n2}}{t_n^2} \quad which \quad gives \quad a_{n2} = \frac{a_f t_n^2}{2} \tag{3.21}$$

For $t = -1$, which is the beginning position of the segment, from the velocity and acceleration continuity condition;

$$\frac{\dot{h}_2(1)}{t_2} = \frac{\dot{h}_n(-1)}{t_n} \quad and \quad \frac{\ddot{h}_2(1)}{t_2^2} = \frac{\ddot{h}_n(-1)}{t_n^2} \tag{3.22}$$

The change of the angles between the trajectory segments can be represented as,

$$\delta_1 = \theta_1 - \theta_0 = h_1(1) - h_1(0) = a_{14} + a_{13} + a_{12} + a_{11} \tag{3.23}$$

$$\delta_2 = \theta_2 - \theta_1 = h_2(1) - h_2(0) = a_{23} + a_{22} + a_{21} \tag{3.24}$$

$$\delta_n = \theta_f - \theta_2 = h_n(0) - h_n(-1) = -a_{n4} + a_{n3} - a_{n2} + a_{n1} \tag{3.25}$$

When we rewrite all the equations in the matrix form,

$$y = \begin{bmatrix} \delta_1 - a_{12} - a_{11} \\ -a_0 t_1 - v_0 \\ a_0 \\ \delta_2 \\ -a_f t_n + v_f \\ a_f \\ \delta_n + a_{n2} - a_{n1} \end{bmatrix} \qquad x = \begin{bmatrix} a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{n3} \\ a_{n4} \end{bmatrix} \tag{3.26}$$

$$C = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3/t_1 & 4/t_1 & -1/t_2 & 0 & 0 & 0 & 0 \\ 6/t_1^2 & 12/t_1^2 & 0 & -2/t_1^2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1/t_2 & 2/t_2 & 3/t_2 & -3/t_n & 4/t_n \\ 0 & 0 & 0 & 2/t_2^2 & 6/t_2^2 & 6/t_n^2 & -12/t_n^2 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{3.27}$$

From the matrixes of (3.26) and (3.27), accepting that $y = Cx$, the solution of the 4-3-4 path planning is found by $x = C^{-1}y$. The boundary conditions for the last trajectory segment was changed from [0,1] to [-1,0] and used in the matrix equations above, writing $t = \overline{t} + 1$ the obtained polynomial is;

$$\begin{aligned} &h_n(t) = a_{n4}t^4 + (-4a_{n4} + a_{n3})t^3 + (6a_{n4} - 3a_{n3} + a_{n2})t^2 \\ &+(-4a_{n4} + 3a_{n3} - 2a_{n2} + a_{n1})t + (a_{n4} - a_{n3} + a_{n2} - a_{n1} + a_{n0}) \end{aligned} \tag{3.28}$$

After finding the coefficients of $h_n(t)$, the coefficients are put in the equation (3.28) and the last trajectory segment's polynomial is found.

3.2.2 3-5-3 Polynomial Trajectory

In 3-5-3 Trajectory, each joint has three segments. the first segment is a thirth-degree polynomial from initial position to the lift-off position. The second segment is the fifth-degree polynomial from the lift-off position to the set-down position. The last segment is the third-degree polynomial.

$$h_1(t) = a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} \quad (first \quad segment) \tag{3.29}$$

$$h_2(t) = a_{52}t^5 + a_{42}t^4 + a_{32}t^3 + a_{22}t^2 + a_{12}t + a_{02} \quad (second \quad segment) \tag{3.30}$$

$$h_n(t) = a_{n3}t^3 + a_{n2}t^2 + a_{n1}t + a_{n0} \quad (third \quad segment) \tag{3.31}$$

The 3-5-3 trajectory consists of polynomial segment $h_i(t)$ which forms the joint trajectory for joint i.The polynomial equations for each segments shown in equations (3.29), (3.30), (3.31). $h_1(t)$ is the first segment, $h_2(t)$ is the second segment and $h_n(t)$ indicates the last trajectory segment same as in 4-3-4 trajectory.

The first and second derivatives of these polynomial equations can be written as in (3.32) and (3.33);

$$v_i(t) = \frac{1}{t_i}\dot{h}_i(t) \tag{3.32}$$

$$a_i(t) = \frac{1}{t_i^2}\ddot{h}_i(t) \tag{3.33}$$

For 3-5-3 Joint Trajectory, the initial velocity, v_0, and initial acceleration, a_0, are zero and also the final velocity, v_f, and the final acceleration, a_f, are zero. All three segments of the polynomial are in the normalized time variable $t \in [0, 1]$.

For the first segment,

$$h_1 \quad (0) \quad = a_{10} = \theta_0 \quad (initial \;\; position) \tag{3.34}$$

$$v_0 = \frac{\dot{h_1}(0)}{t_1} = \frac{a_{11}}{t_1}(initial \;\; velocity) \tag{3.35}$$

$$a_0 = \frac{\ddot{h_1}(0)}{t_1^2} = \left(\frac{6a_{31}t + 2a_{21}}{t_1^2}\right)_{t=0} = \frac{2a_{21}}{t_1^2}(initial \;\; acceleration) \tag{3.36}$$

Then;

$$a_{11} = v_0 t_1 \qquad , \qquad a_{21} = \frac{a_0 t_1^2}{2} \tag{3.37}$$

For the second segment;

$$h_2 \quad (0) = \quad a_{10} = \theta_1 \tag{3.38}$$

$$v_1 = \frac{\dot{h_2}(0)}{t_2} = \left(\frac{5a_{52}t^4 + 4a_{42}t^3 + 3a_{32}t^2 + a_{22}t + a_{12}}{t_2}\right)_{t=0} \tag{3.39}$$

$$v_1 = \frac{a12}{t_2} \tag{3.40}$$

$$a_1 = \frac{\ddot{h_2}(0)}{t_2^2} = \frac{2a_{22}}{t_2^2} \tag{3.41}$$

For the last segment;

$$h_n \quad (0) \quad = a_{n0} = \theta_f \quad (final \;\; position) \tag{3.42}$$

$$v_f = \frac{\dot{h_n}(1)}{t_n} = \frac{4a_{n4} + 3a_{n3} + 2a_{n2} + a_{n1}}{t_n} \tag{3.43}$$

$$a_f = \frac{\ddot{h_n}(1)}{t_n^2} = \frac{12a_{n4} + 6a_{n3} + 2a_{n2}}{t_n^2} \tag{3.44}$$

The first and second derivatives of the polynomials at one's initial and other's final point should be equal.

$$\frac{\dot{h_1}(1)}{t_1} = \frac{\dot{h_2}(0)}{t_2} \tag{3.45}$$

$$\frac{\ddot{h_1}(1)}{t_1^2} = \frac{\ddot{h_2}(0)}{t_2^2} \tag{3.46}$$

$$\frac{\dot{h_2}(1)}{t_2} = \frac{\dot{h_n}(0)}{t_n} \tag{3.47}$$

$$\frac{\ddot{h_2}(1)}{t_2^2} = \frac{\ddot{h_n}(0)}{t_n^2} \tag{3.48}$$

Since we know the values of polynomials at t=0 and t=1; we can write these equations above.

$$\theta_1 - \theta_0 = h_1(1) - h_1(0) = a_{31} + a_{21} + a_{11} \tag{3.49}$$

$$\theta_2 - \theta_1 = h_2(1) - h_2(0) = a_{52} + a_{42} + a_{32} + a_{22} \tag{3.50}$$

$$\theta_f - \theta_2 = h_n(1) - h_n(0) = a_{n4} + a_{n3} + a_{n2} \tag{3.51}$$

From all these equations,

$$C = \begin{bmatrix} 3/t_1 & -1/t_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/t_2 & 2/t_2 & 3/t_2 & 4/t_2 & 5/t_2 & 0 & 0 & -1/t_n \\ 0 & 0 & 2/t_2^2 & 6/t_2^2 & 12/t_2^2 & 20/t_2^2 & 0 & -2/t_n^2 & 0 \\ 6/t_2^2 & 0 & -2/t_2^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3/t_n & 2/t_n & 1/t_n \\ 0 & 0 & 0 & 0 & 0 & 0 & 6/t_n^2 & 2/t_n^2 & 0 \end{bmatrix} \tag{3.52}$$

$$y = \begin{bmatrix} -2a_{21}/t_2 - a_{11}/t_1 \\ 0 \\ 0 \\ -2a_{21}/(t_1)^2 \\ \theta_1 - \theta_0 - a_{21} - a_{11} \\ \theta_2 - \theta_1 \\ \theta_f - \theta_2 \\ 0 \\ 0 \end{bmatrix} \qquad x = \begin{bmatrix} a_{31} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{42} \\ a_{52} \\ a_{n4} \\ a_{n3} \\ a_{n2} \end{bmatrix} \tag{3.53}$$

From the matrixes of C and y, accepting that $y = Cx$, the solution of the 3-5-3 path planning is found by $x = C^{-1}y$.

3.2.3 Cubic Spline Trajectory (Five Cubics)

A spline curve is a polynomial of degree k with derivative of order k-1 at the interpolation points. The reason to use cubic spline functions is that preserving continuity in first and second derivative at the interpolation points. Also, the degree of approximation and smoothness is achieved. In Cubic spline case, the first derivative represents continuity in velocity and the second derivative represents continuity in acceleration. The advantage of the cubic spline is that it is the lowest degree polynomial that represents continuity in velocity and acceleration. Thus, the effort for computation and the possibility of numerical calculations instability is reduced.

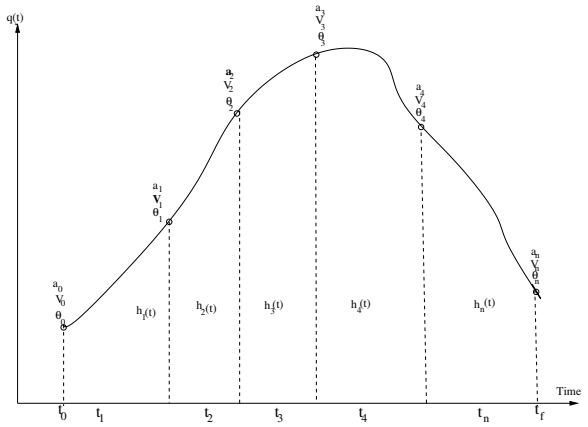

Figure 3.2: Boundary conditions for a 5-cubic joint trajectory

$$h_j(t) = a_{j3}t^3 + a_{j2}t^2 + a_{j1}t + a_{j0} \quad j = 1, 2, 3, ...n \tag{3.54}$$

The general equation of a five cubic polynomial for jth joint trajectory segment is seen in (3.54). Here, a_{ji} represents the ith coefficient of joint j and n represents the last trajectory segment.In the five-cubic interpolation, six interpolation points and five trajectory segments is used. Since we have four interpolation points(initial, lift-off, set-down and final), we have to select extra two interpolation points to provide boundary conditions for solving the unknown coefficients of the polynomial function.

The first and second derivatives of these polynomial equation with respect to time are;

$$v_j(t) = \frac{\dot{h}_j(t)}{t_j} = \frac{3a_{j3}t^2 + 2a_{j2} + a_{j1}}{t_j} \tag{3.55}$$

$$a_i(t) = \frac{\ddot{h}_j(t)}{t_j^2} = \frac{6a_{j3}t + 2a_{j2}}{t_j^2} \tag{3.56}$$

3.3 Planning of Cartesian Path Trajectories

When it is desired that the end-effector's motion in a robot manipulator follows a geometrically specified path in the cartesian space, it is necessary to plan this trajectory in the same space [35]. Although manipulators' joint variables represents the position and orientation of the end-effector, they are not convenient for specifying a path because of non-orthogonality of joint coordinates and inseparability of position and orientation [25]. Due to these reasons, other kinematic trajectory planning approaches are used for cartesian path plan such as *Quaternion Approach* and *Cubic Polynomial Joint Trajectories with Torque Constraint*.

3.3.1 Planning Straight Line Using Quaternion

Quaternions are an interesting mathematical concept with a deep relationship with the foundations of algebra and number theory. They are invented by W.R.Hamilton in 1843. In practice, they are most useful to us as a means of representing orientations [36]. Quaternions can be used to represent the orientation of a manipulator for planning a straight-line trajectory [25].

Quaternions consist of complex numbers and are used to represent rotations in the same way as complex numbers n the unit circle can represent planar rotations. Unlike Euler Angles, quaternions give all prametrization of special ortogonal cartesian space, by using four numbers instead of three for representing rotations [34].

A Quaternion vector is represented as;

$$Q = q_0 + q_1 i + q_2 j + q_3 k \quad , \quad q_i \in R \quad , \quad i = 0, 1, 2, 3 \tag{3.57}$$

where q_0 is the scalar component of Q and $\vec{q} = (q_1, q_2, q_3)$ is the vectoral component. Shortly, it can be expressed as; $Q = (q_0, \vec{q})$ when $q_0 \in R$ and $\vec{q} \in R^3$. The conjugate of a Quaternion $Q = (q_0, \vec{q})$ is given by $Q^* = (q_0, -\vec{q})$ and the magnitude of a quaternion represented as ;

$$\|Q\|^2 = Q \cdot Q^* = q_0^2 + q_1^2 + q_2^2 + q_3^2 \tag{3.58}$$

The inverse of quaternion is $Q^{-1} = Q^*/\|Q\|^2$. The unit quaternion has the magnitude $\|Q\| = 1$, meaning that $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ [34].

The multiplication of two quaternions, $Q = (q_0, \vec{q})$ and $P = (p_0, \vec{p})$, can be represented as;

$$Q \cdot P = (q_0 p_0 - \vec{q} \cdot \vec{p}\,,\ q_0 \vec{p} + p_0 \vec{q} + \vec{q} \times \vec{p}) \tag{3.59}$$

The rotation about axis n by an angle θ is represented by quternion;

$$Q(\theta, n) = \cos(\theta/2) + \sin(\theta/2) n \tag{3.60}$$

Our aim is to move the end-effector of the manipulator along a straight line path between two knot points, specified by F_0 and F_1 in time T, where F_0 and F_1 are homogeneous transformations represented as in equation (3.61);

$$F_i = \begin{bmatrix} R_i & p_i \\ 0 & 1 \end{bmatrix} \tag{3.61}$$

The motion along the path in which the origin of tool frame is translated from p_0 to p_1 coupled with the rotation of tool frame orientation part from R_0 to R_1. If $\lambda(t)$ is the remaining fraction of motion, it is expressed for the uniform motion as;

$$\lambda(t) = \frac{T - t}{T} \tag{3.62}$$

where T is the total time for the motion and t is the start time of the motion. The tool frame's position and orientation is given as;

$$p(t) = p_1 - \lambda(t)(p_1 - p_0) \tag{3.63}$$

$$R(t) = R_1\, Q(-\theta\lambda(t), n) = R_0^{-1} R_1 \tag{3.64}$$

If the end-effector of the manipulator moves from one segment to another with constant acceleration, it must accelerate or decelerate. In order to achieve this, the transition should start at time τ before the arm reaches the intersection of two segments and after the intersection complete its motion to the new segment at time τ.

The Boundary conditions are;

$$p(T_1 - \tau) = p_1 - \frac{\tau \Delta p_1}{T_1} \tag{3.65}$$

$$p(T_1 + \tau) = p_1 + \frac{\tau \Delta p_2}{T_2} \tag{3.66}$$

$$\frac{dp(t)}{dt}|_{t=T_1-\tau} = \frac{\Delta p_1}{T_1} \tag{3.67}$$

$$\frac{dp(t)}{dt}|_{t=T_1+\tau} = \frac{\Delta p_2}{T_2} \tag{3.68}$$

$$\Delta P_1 = p_1 - p_2 \quad , \quad \Delta P_2 = p_2 - p_1 \tag{3.69}$$

For constant acceleration;

$$\frac{d^2p(t)}{dt^2} = a_p \tag{3.70}$$

Integrating the equation (3.70) twice and boundary conditions are applied;

$$p(t') = p_1 - \frac{(\tau - t')^2}{4\tau T_1}\Delta p_1 + \frac{(\tau + t')^2}{4\tau T_2}\Delta p_2 \tag{3.71}$$

where $t' = T_1 - t$ is the time from the intersection of two segments.
The orientation can be found as;

$$R(t') = R_1\ Q[\frac{-(\tau - t')^2}{4\tau T_1}\theta_1, n]\ Q[\frac{(\tau + t')^2}{4\tau T_2}\theta_2, n] \tag{3.72}$$

CHAPTER 4

Collision Avoidance Problem

Depending upon how efficient robots are utilized in a factory environment, there may be a great deal of improvement in productivity, over-all cost reduction and quality of the products. Cartesian robots are mostly used for simple repetitive jobs, such as pick-and-place, machine loading and unloading, spray painting, and spot welding. Only one robot in a workspace may limit the type of tasks that can be performed. Two or more robots in a common workspace may be required to perform a common task or just to improve the performance. For example, multiple robots are needed to transport an object beyond the payload capability of a single robot. The underlined idea here is to provide a practical methodology that can make several robots operate safely in a common workspace. In the case that more than one robot operate simultaneously in a common workspace, Hence, the problem of avoiding potential collisions among the links of the robots should be carefully considered.

To solve the collision avoidance problem, zone-blocking methods have been proposed by several researchers such as Chang [31]. In such methods, only one robot is assumed to operate at a time. So, this semaphore mechanism is not efficient because of not providing the parallel tasking feature. Besides the zone-blocking methods, there are other collision avoidance methods proposed for multiple robots. These methods can be divided into two categories:

(1) time adjusting methods while maintaining the given geometric path

(2) trajectory modification methods which modifies given geometric path

The former adjusts the time evolution representing the moving speed of robots while the geometrical paths of the robots are fixed. The robot path, which guarantees a robot not to collide with stationary obstacles, can be obtained using some existing methods. One of the major features of time adjusting approaches is that the number of variables to be considered for collision avoidance does not

exceed the number of robots because one variable, usually the time, is enough to express the moving speed for each robot. For instance, in the case of two robots, at most two variables are needed for solving the collision avoidance problem. This fact suggests that a collision avoidance problem in multiple robots can be easily solved comparing with a collision avoidance problem for a single robot and stationary.

4.1 Collision Types

Analyzing the collision conditions, five collision types are found for any two straight-line moving objects when collision occurs in general. The possible collision types are described below:

Let a be the angle measured from the intersection of any two straight-line paths:

(1) *Acute collision*: collision of two objects with $0^o < a < 90^o$;

(2) *Obtuse collision*: collision of two objects with $90^o < a < 180^o$;

(3) *Perpendicular collision*: collision of two objects with $a = 90^o$;

(4) *San-Diego collision*: collision of two objects when they are moving along the same path and in the same direction, i.e., $a = 0^o$, and the speed of the object behind is higher than the speed of the object ahead; and

(5) *Head-On collision*: collision of two objects when they are moving along the same path but in opposite directions, i.e., $a = 180^o$.

Collision types (1), (2) and (3) can be avoided by simply changing the velocities of the objects. Collision types (4) and (5) result when two moving objects have parts of paths which coincide. We cannot avoid this collision by only adjusting their speeds; the assigned path must be changed [37].

4.2 Collision Avoidance Methods

Many methods have been proposed in recent years to solve the problems of collision-avoidance. Chang and his colleagues have proposed a simple time delay method avoid collisions between two robot arms In their work, links of robots

were approximated geometrically using polyhedra. The danger of collisions between two robots is expressed by a distance function associated with the robots in a working space. The collision map scheme in the form of a 2D Traveling Length v.s. Sampling Time (TLVST) graph can describe collisions between two 3D robots effectively [31].In a similar method in which the complex 3D problems are changed to simple 2D ones, Wu et. al. proposed that links of robots in 3D can be simplified to a 2D Space/Time graph. Robots can move with the proper velocity to avoid potential collisions with obstacles or with other robots by constructing an optimal path on the Space/Time graph [32].

4.3 Collision Detection

4.3.1 Geometric Modeling of Robot Links

A robot can be modeled by a proper superquadric equation if there is a sufficient number of given points on its links.Since an ellipse can be generalized to the superquadric form (n-ellipse), a robot link can be modeled as an ellipse by applying the fitting technique of superquadric modeling. In this way, the representation of a link can be described in a simple mathematical equation. [37]. The general equation of an ellipse is;

$$F(x,y) = \frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1 \tag{4.1}$$

To examine whether a point, (xo, yo), resides in an ellipse or not, F(x, y) should be calculated at that point. If $F(xo, yo) > 1$, the point is located outside the ellipse; otherwise the point is inside the boundary. The axes of the ellipse are the controllable parameters when an ellipse is fitted to a link. The best fitting function can be written as;

$$min \sum_{i=1}^{N} [R(xi, yi, r_x, ry)]^2 \qquad R(xi, yi, r_x, ry) = \sqrt{r_x r_y}(1 - F) \tag{4.2}$$

where xi, yi, $i = 1, 2..N$ are the points on the modeled object; r_x and r_y are the axes of the modeling ellipse. To solve the optimization equation, the parameters rx and ry are searched in the limited range.

4.3.2 Distance Measure

To detect the collision between two robots, the distance between the links must be computed and compared every instance of time. For the success of this process, the robots should be modeled properly. Here, as the robots are 6 DOF cartesian

robots, the distance measure computation between two robots are easier than a serial manipulator. Because, there is no collision between the first two links and it is enough to measure the distance between their third links with the tip point which are perpendicular to each other. Thus, we only detect the distance between the third links of the robot's position to avoid collision [38].

4.3.3 Minimum Distance Functions

For collision avoidance of the 6 DOF Cartesian Robot system, the distance between the links should be calculated every instance of time and the minimum distance between the links should be obtained. Minimum distance is calculated by minimum distance functions. [28].

As d is the minimum distance between the robot links, it can be formulated as;

$$r = min\|p_a - p_b\| \tag{4.3}$$

In equation 4.3, $p_a - p_b$ is the Euclidean distance between the two points p_a and p_b. Since the third links of the cartesian robots are perpendicular to each other, the distance between the links can be calculated easily. [16]. According to the hardware made by Matlab, the distance between the links is calculated simultaneously during the given trajectory and when the distance is smaller than the value we obtain as the minimum distance, one of the robot's links change its trajectory.

CHAPTER 5

Simulation Studies

5.1 System Description

Real life application of the research presented in this thesis is considered for two 6 DOF cartesian robotic arms sharing the same work space which is a pool less than half of which is full of sand. One of the arms will be carrying the sender antenna while the other arm will be carrying the receiver antenna so that the collected data via electromagnetic waves can be used to construct 3 dimensional image of the object burried under the sand forming an invisible object for a naked eye.

As said before, the robots are 6DOF cartesian robot and have four joints; three of them are prismatic joints and one of them is a spherical joint.

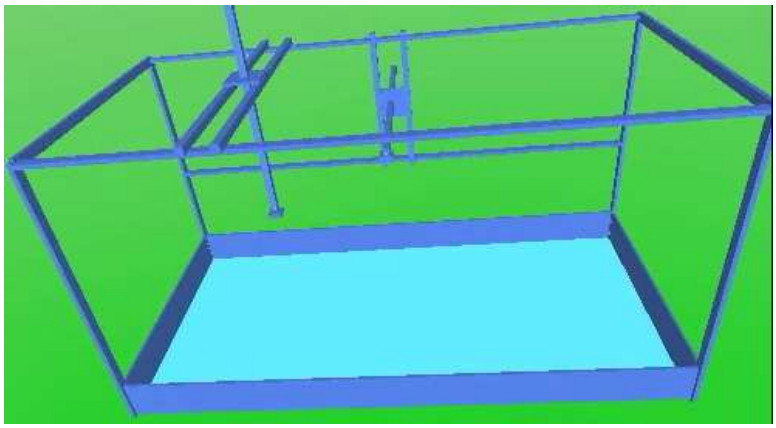

Figure 5.1: Virtual Reality Model of Cartesian Robots

The system is used for the determination of orientation of the bodies buried in the sand which is in the base, also the electromagnetic waves sent to the sand can help to predict the shape of the sand surface. The buried objects which are in rough surfaces reflects the electromagnetic waves so that the orientation of the objects can be screened by various methods. The goal of the project is to apply these methods for determining the position of the buried objects. In robotic

view, the cartesian robots in the common taskspace should move in the desired trajectories without collision. Kinematic modeling, path planning and collision avoidance are the works to be done.

5.2 Kinematic Model

H matrix is found as;

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9996 & 0.0161 & 0.0237 \\ 0 & 0 & 0 & -0.0168 & 0.9994 & 0.0318 \\ 0 & 0 & 0 & -0.0232 & -0.0322 & 0.9992 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

According to the frame assignments and from the equation $J = \sigma_t \phi H$, jacobian of the cartesian robot 2 is calculated as in the following. The jacobian in (5.2) is full rank, its determinant is not zero and it is not singular. So it has an inverse.

$$J = \begin{bmatrix} 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 1.0000 & 0 & 0 & -0.5000 & 0 \\ 0 & 0 & 1.0000 & 0.5000 & 0 & 0 \\ 1.0000 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.2}$$

When we give $\dot{\theta}$ as sinusoidal input, the tip velocity changes like the figure 5.2.In the figure 5.2, the first three are angular velocities ω_x, ω_y and ω_z and other three are linear velocities of v_x, v_y and v_z.

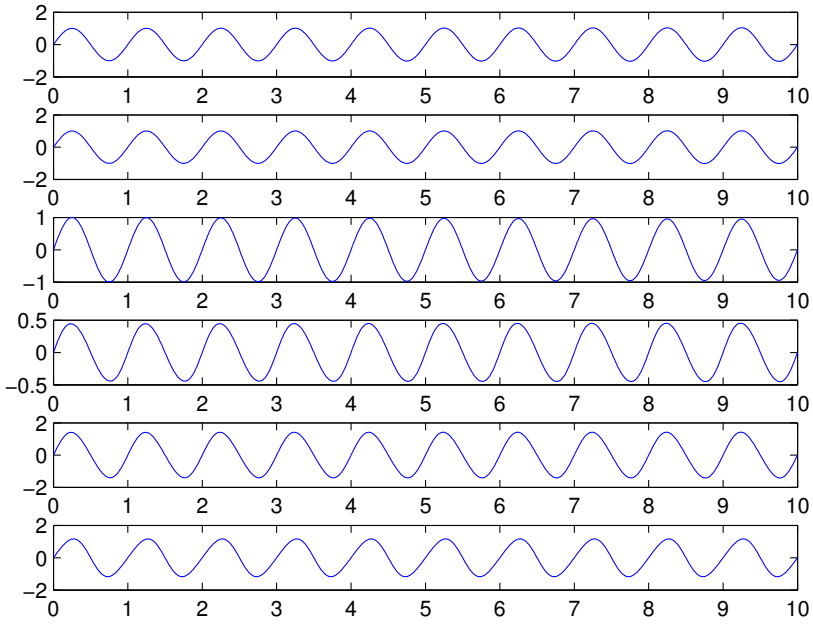

Figure 5.2: The Tip Point's Velocity

The joint velocities are found from $V = \phi H\dot{\theta}$, and for each joint there are three angular velocity, $(\vec{\omega}_x, \vec{\omega}_y, \vec{\omega}_z)$ and three linear velocity components($\vec{v}_x$, $\vec{v}_y$, $\vec{v}_z$) .

In the figure 5.3, joint 1 has only linear velocity in z direction(v_z) and all of other components are zero. The angular and linear velocity components of the joint 2 are all zero as it is a prismatic joint and has linear velocity in x and z direction as shown in figure 5.4. $\vec{v}_z$ is linear velocity component coming from joint 1 and $\vec{v}_x$ is linear velocity component of joint 2.

As seen in the first figure in 5.5, joint 3's angular velocity components are zero and there are three linear velocity components of joint three coming from joint 1($\vec{v}_x$), coming from joint 2($\vec{v}_z$) and linear velocity component of the joint 3($\vec{v}_y$).

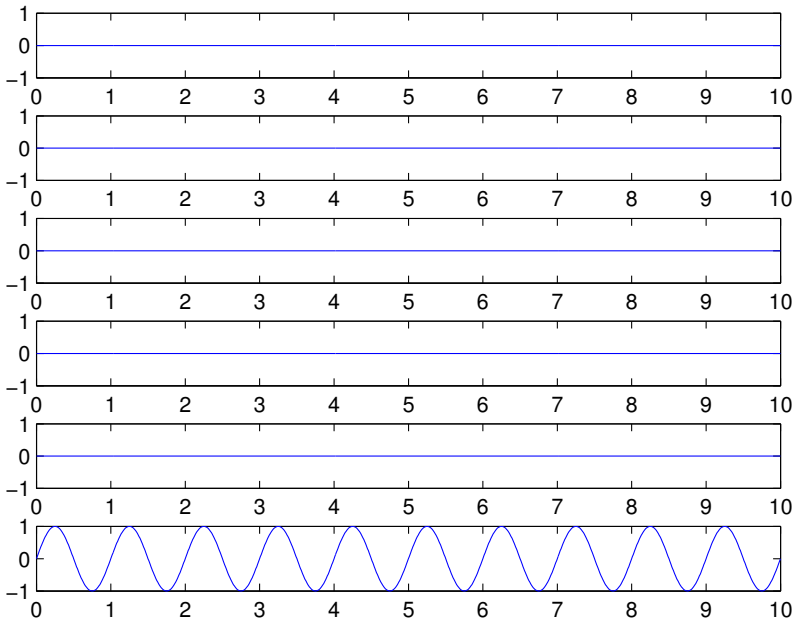

Figure 5.3: 6 axis velocity of Joint 1

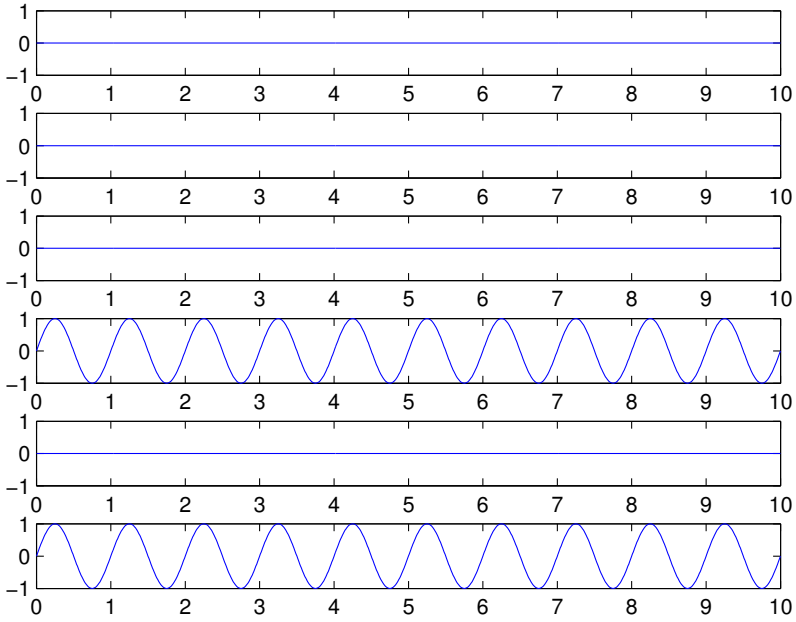

Figure 5.4: 6 axis velocity of Joint 2

In the same figure right 5.6, joint 4 has both linear velocity and angular velocity components in both directions because it is a spherical joint.

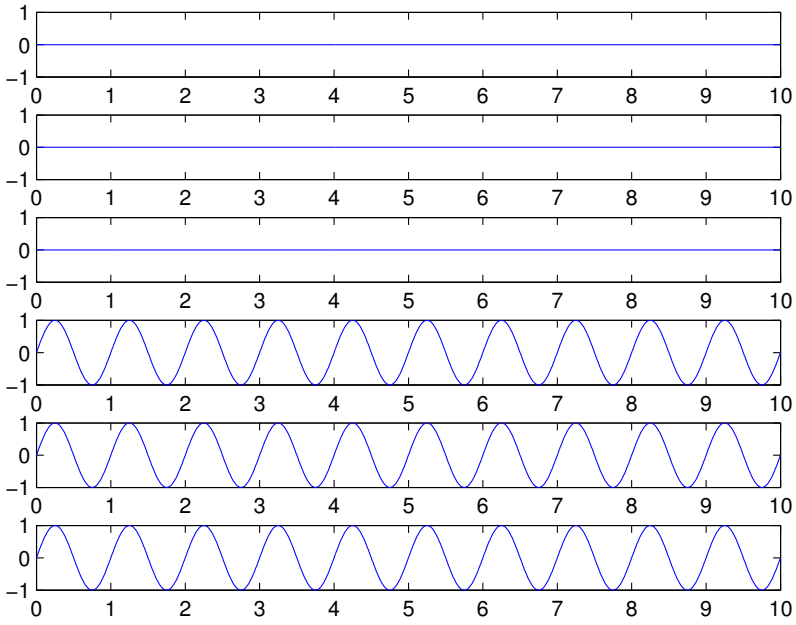

Figure 5.5: 6 axis velocity of Joint 3

5.3 Path Planning

In the path planning of the cartesian robot, two path planning methods 4-3-4 and 3-5-3 path planning methods are used. As comparing this methods, a trajectory is applied to the tip point for x, y and z direction. Then, the first and second derivative of this trajectory gives us the velocity and acceleration of the tip point. From the equation (5.3), multiplication of jacobian inverse with the derivative of the trajectory(tip point's velocity) gives us the joint's velocity.

$$\dot{\theta} = J^{-1} V_{tip} \tag{5.3}$$

5.3.1 4-3-4 Polynomial Trajectory

For 4-3-4 trajectory, the change of velocity and position according to the given position (4-3-4 trajectory) is seen in figures 5.7,5.10. These trajectories were made by fitting polynomials from one point to another point. The position change in x axes is from 0 to 10 ,the position change in y axes is from 0 to 4 and the position change in z axes is from 0 to 6. When we apply linear velocity to the tip point, we can find the joint's velocities from the equation (5.3).

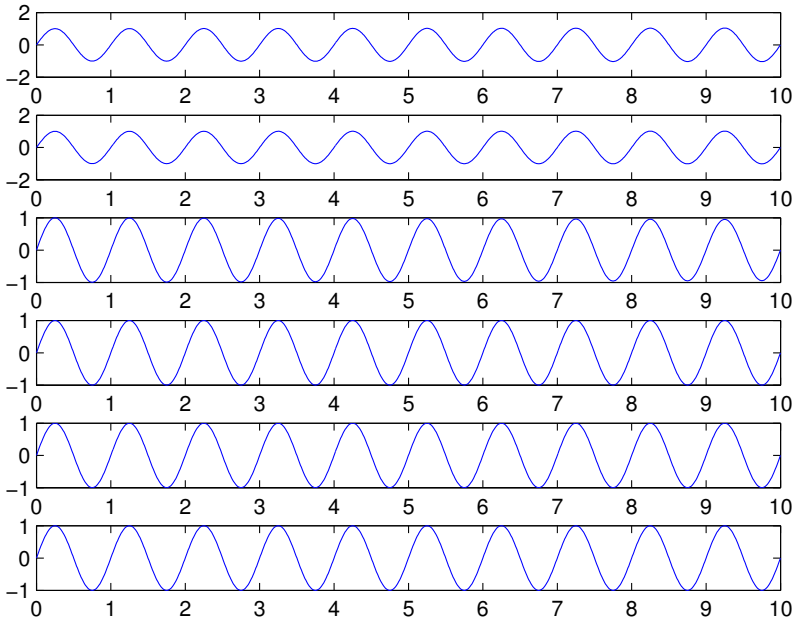

Figure 5.6: 6 axis velocity of Joint 4

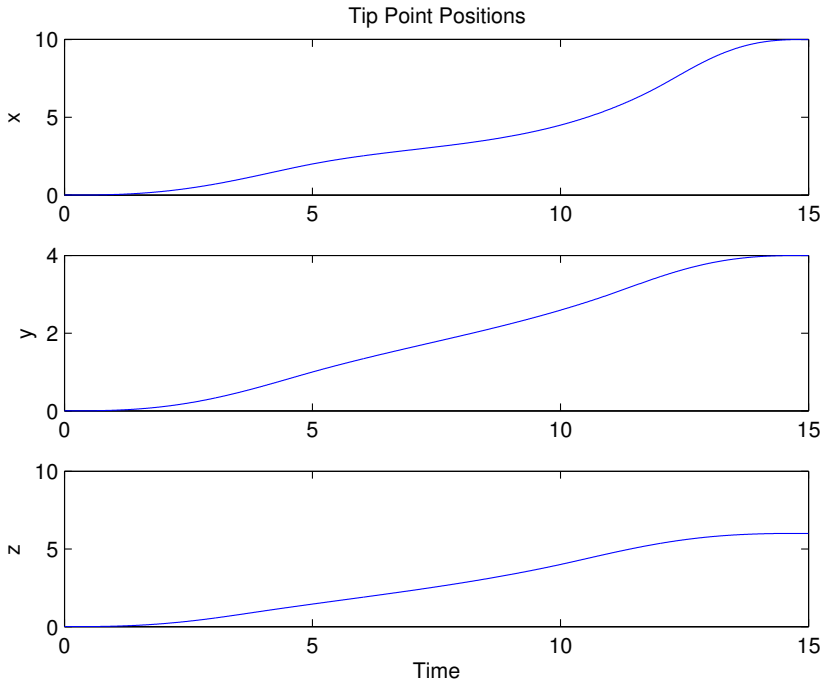

Figure 5.7: Position of Tip point for 4-3-4 trajectory

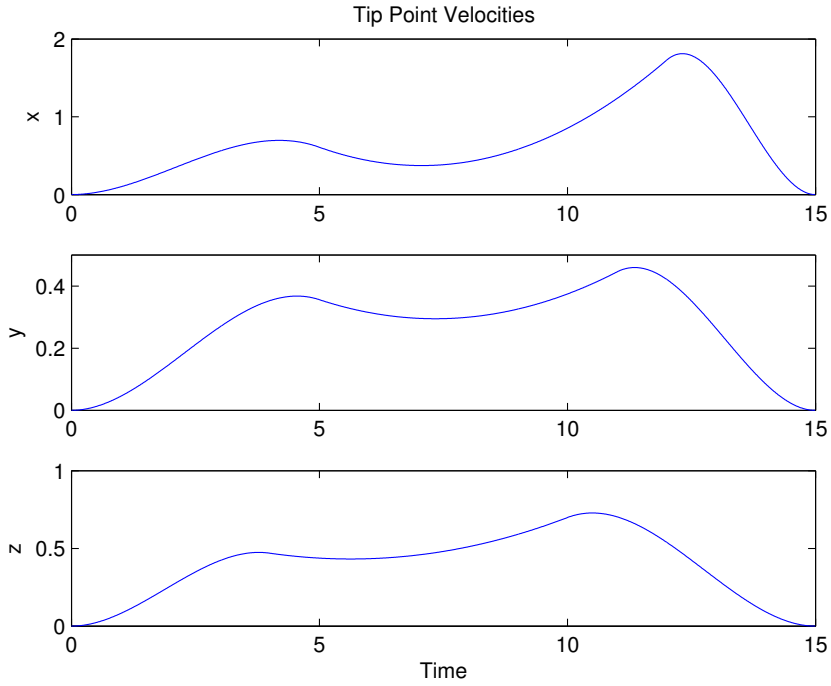

Figure 5.8: Velocity of Tip point for 4-3-4 trajectory

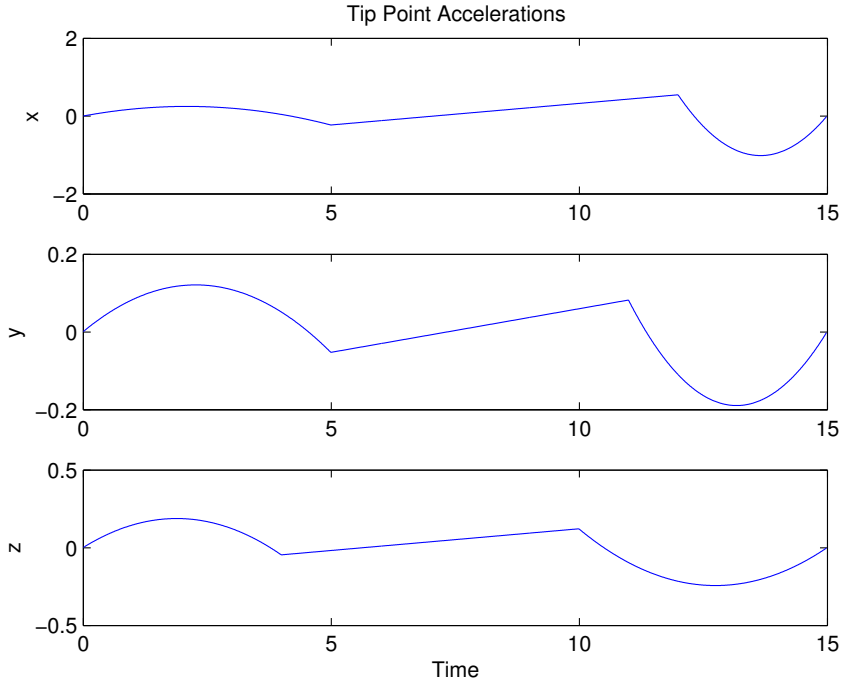

Figure 5.9: Acceleration of Tip point for 4-3-4 trajectory

The figure 5.7,5.8 and 5.9 shows the position,velocity and acceleration of the tip point for 4-3-4 polynomial trajectory. When we take the derivative of this trajectory, velocity of the tip point is found. Second derivative of the polynomial trajectory is the acceleration of the tip point.

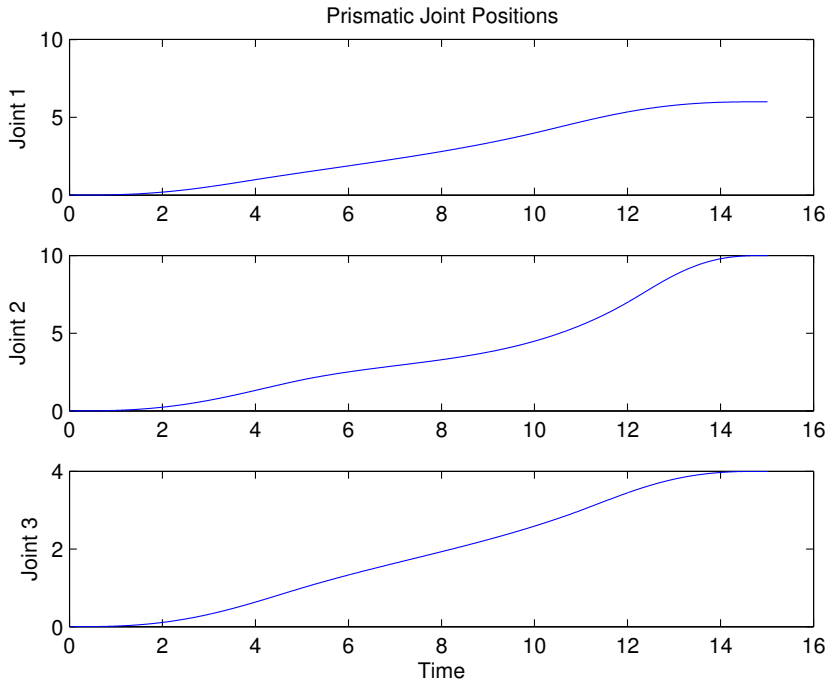

Figure 5.10: Position of Joints for 4-3-4 trajectory

In the figures 5.10, 5.11 and 5.12, the velocity and acceleration is continuous and smooth. Initial and final values of the velocity and acceleration is zero. The derivative and integral of the joint angle can be calculated from (5.4) and (5.5).

$$\theta(k+1) = \theta(k) + dt \cdot \dot{\theta}(k) \tag{5.4}$$

$$\ddot{\theta}(k-1) = (\dot{\theta}(k-1) - \dot{\theta}(k))/dt \tag{5.5}$$

For joint angles, we can integrate the velocity of the joint as seen in (5.4) and for the acceleration of joints, we derivate the $\dot{\theta}$ using equation (5.5). Since we apply linear velocity, the acceleration and velocity we found are prismatic joints' velocity and acceleration seen in the figures 5.10, 5.11 and 5.12.

5.3.2 3-5-3 Polynomial Trajectory

For 3-5-3 trajectory, the change of velocity and position according to the given position (3-5-3 trajectory) is seen in figures 5.13, 5.14 and 5.15 . Joint velocities

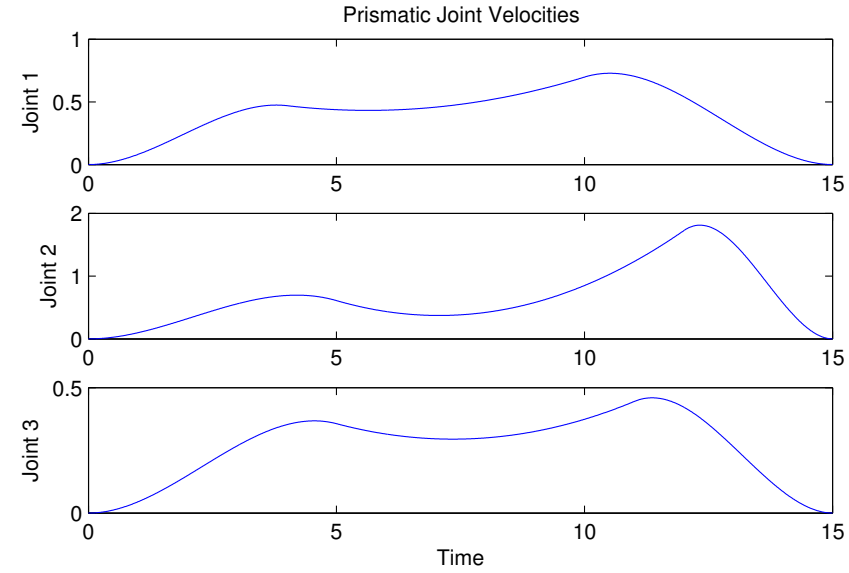

Figure 5.11: Velocity of Joints for 4-3-4 trajectory

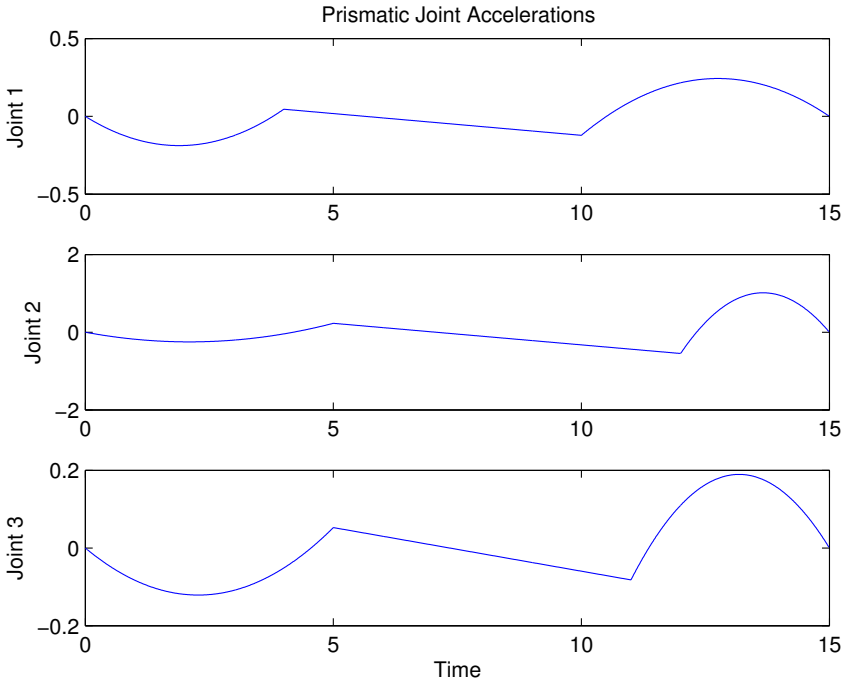

Figure 5.12: Acceleration of Joints for 4-3-4 trajectory

can be found from (5.3), and velocities, acceleration and the position of the joints are shown in figure 5.16.

In the figures 5.16, the velocity and acceleration is continuous and smooth. Initial and Final values of the velocity and acceleration is zero. The derivative and integral of the joint angle can be calculated from (5.4) and (5.5).

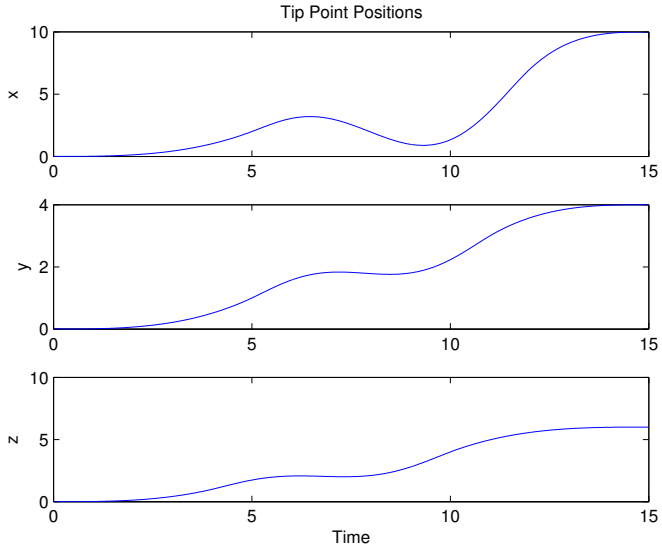

Figure 5.13: Position of Tip point for 3-5-3 trajectory

For joint angles, we can integrate the velocity of the joint as seen in (5.4) and for the acceleration of joints, we derivate the $\dot{\theta}$ using equation (5.5). Since we apply linear velocity, the acceleration and velocity we found are prismatic joints' velocity and acceleration seen in 5.16.

5.4 Collision Avoidance

To apply the method of collision avoidance, cartesian robots were modeled by Matlab's Virtual Reality Toolbox and method's simulations were made by Matlab's software and the results were visualized by Virtual Reality. The applied method used for the robots is minimum distance functions. The trajectories given to the robots are 4-3-4 polynomial trajectories.

The program written in Matlab measures the distance between the cartesian arm's third links. If the distance between the robot links is zero, the collision occurs. As seen in the figure 5.18, the collision occurs in the given trajectory when the distance between the links is zero.

For collision avoidance, we obtain a minimum distance between the links and when the robots are near to each other as near as the minimum distance Robot 2

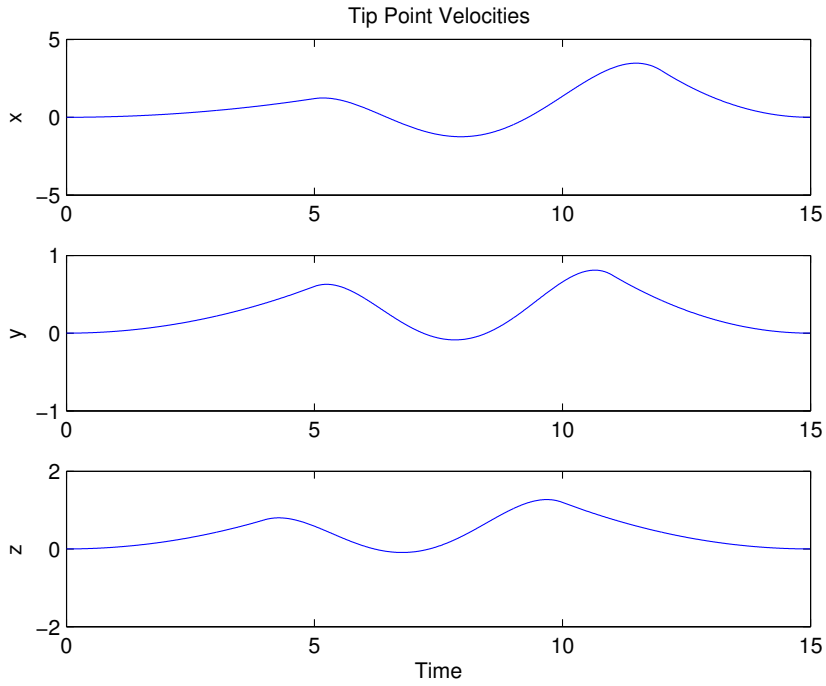

Figure 5.14: Velocity of Tip point for 3-5-3 trajectory

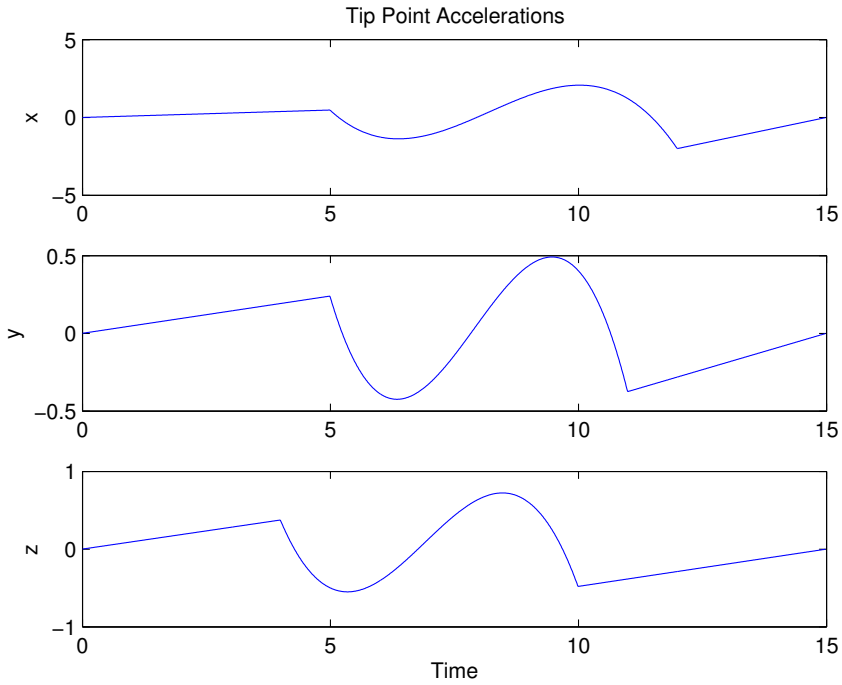

Figure 5.15: Acceleration of Tip point for 3-5-3 trajectory

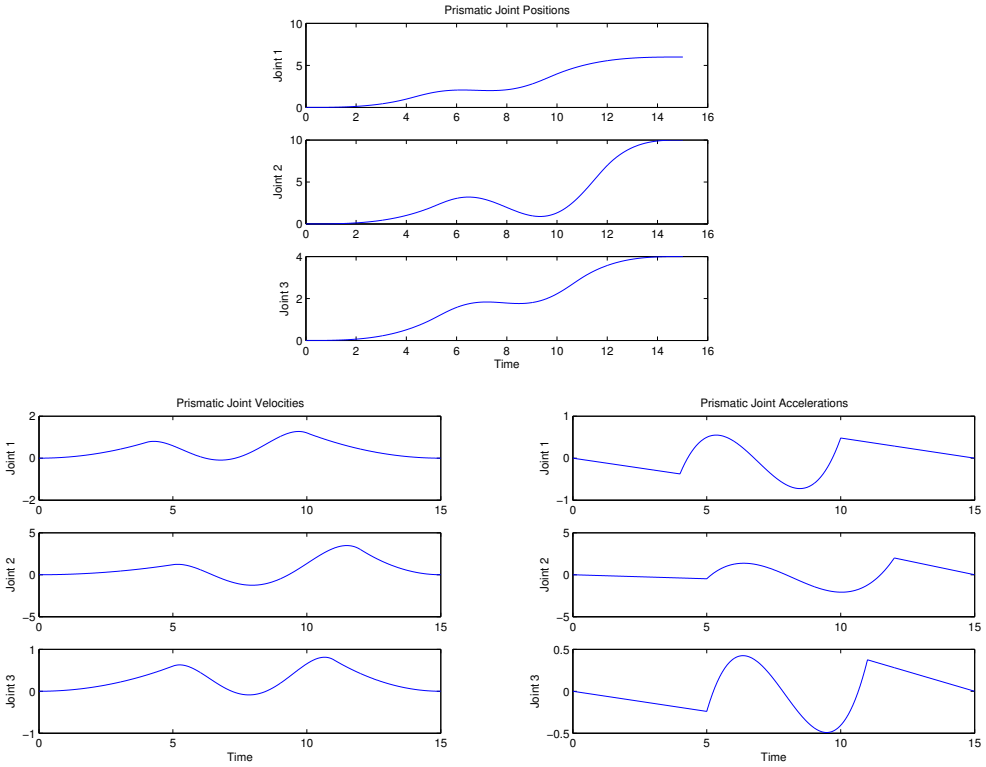

Figure 5.16: Position, Velocity and Acceleration of Joints for 3-5-3 trajectory

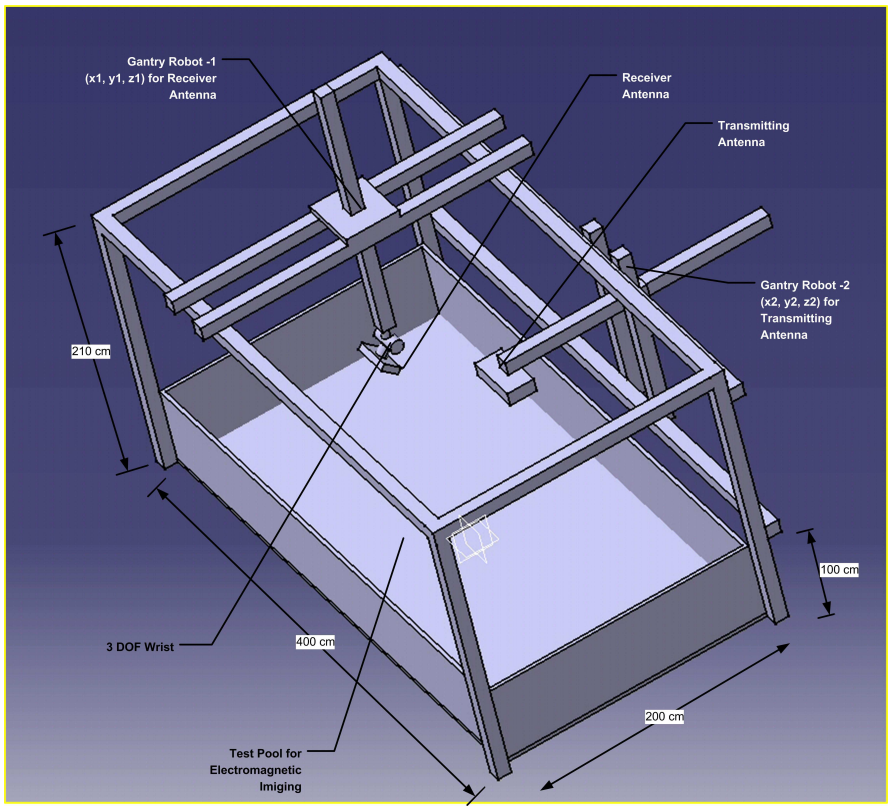

Figure 5.17: Catia Model of the Cartesian Robots

changes its position and then continues to its old trajectory. Thus, the collision avoidance is successfully applied to the system. The robots position change is related with the minimum distance to make the arm far enough from the other arm. It can be understood in figure 5.19, that the collision is prevented using this method of minimum distance functions.

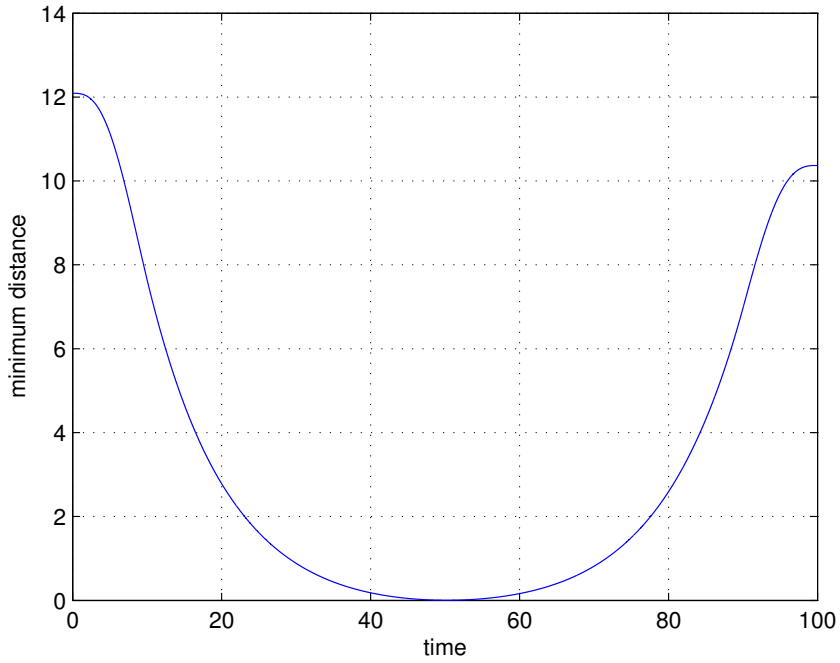

Figure 5.18: Collision occurs in the given trajectory

When two different trajectories which we know collision occurs are given to the system and collision avoidance technique are applied to the system, the collision avoidance is observed seen in the figures 5.20.

It is seen in example virtual reality simulation in Appendix C.1 and C.2 that two robots are going towards them and become closer at time t=t_0 and t=t_1. In the other scene, at time t=t_2 one of the robots changes its link orientation not to collide with the other one. At time t=t_f, the robot which avoided collision returns to its original path and continues to its motion.

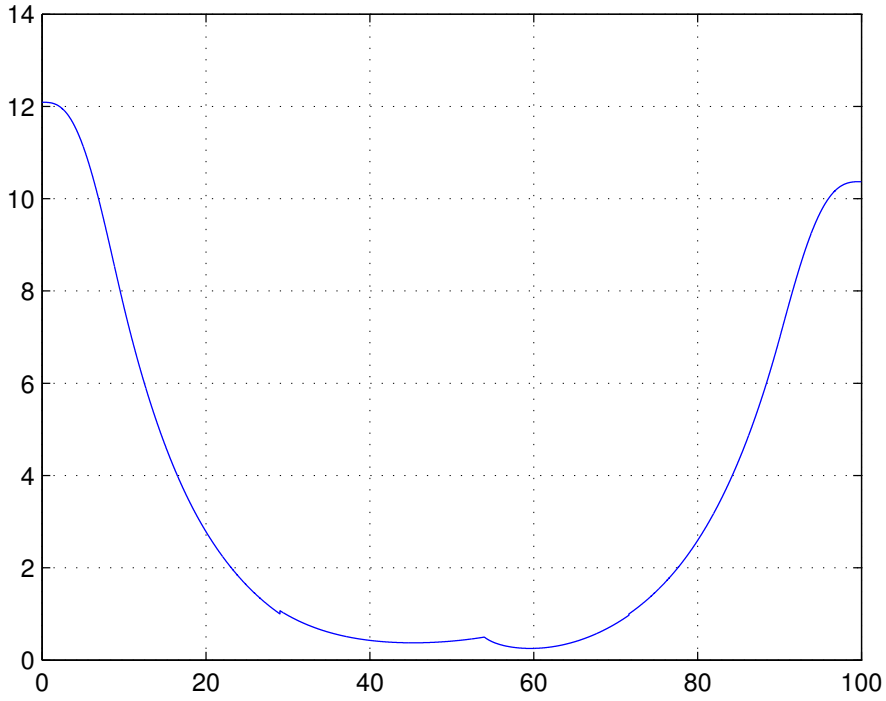

Figure 5.19: Collision Avoidance Method using Minimum Distance Functions

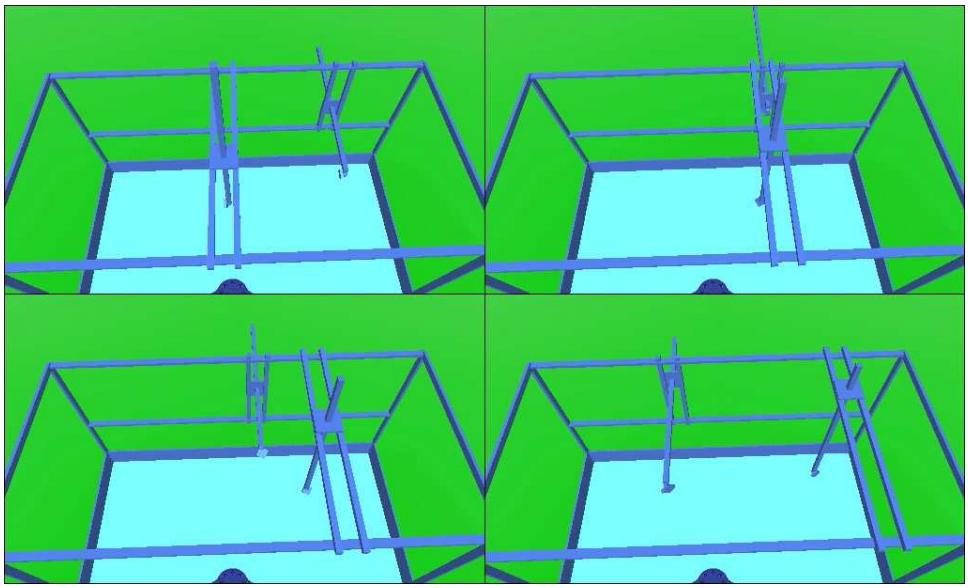

Figure 5.20: Collision Avoidance for given trajectories

CHAPTER 6

Conclusion

Two 6 DOF cartesian robotic arms sharing the same work space were modeled using spatial operator algebra. Collision free path planning algorithms were developed using polynomial based paths. Time derivative of the obtained trajectory for the tip point position and orientation was taken. Under a reasonable assumption that neither of the arms is at a singular configuration, unique inverse kinematic solution was achieved. This yielded the joint velocities. Taking the time derivative of that gave us joint accelerations which was made sure to be bounded so that while following the desired trajectory, the joints would not be over loaded. The results were found to be satisfactory and displayed in Chapter 5. Visualization of the system was done using the "Virtual Reality Toolbox" of MATLAB software.

The extend of this work is to apply these methodologies to a real system to be manufactured. This will be possible by the grant under a project with TUBITAK.

REFERENCES

[1] **R. Featherstone**. The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, 2(1):13–30, Spring 1983.

[2] **G. Rodriguez**. Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics. *IEEE Journal of Robotics and Automation*, 3(6), December 1987.

[3] **T. R. Kane**, and **D. A. Levinson**. *Dynamics: Theory and Applications.* McGraw-Hill, New York, 1985.

[4] **A. Jain**. Unified formulation of dynamics for serial rigid multibody systems. *Journal of Guidance*, 14(3):531–542, May-June 1991.

[5] **G. Rodriguez**, **K. Kreutz-Delgado**, and **A. Jain**. Spatial operator algebra for manipulator modeling and control. *International Journal of Robotics Research*, 10:371–381, August 1991.

[6] **G. Rodriguez**, **D.R. Meldrum**, and **F.G. Franklin**. An order(n) recursive inversion of the jacobian for an n-link serial manipulator. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1175–1180, San Francisco, CA, April 1991.

[7] **G. Rodriguez**, **A. Jain**, and **K. Kreutz**. Spatial operator algebra for multibody systems dynamics. *The Journal of the Astronautical Sciences*, 40(1):27–50, January-March 1992.

[8] **K. Kreutz-Delgado**, **A. Jain**, and **G. Rodriguez**. Recursive formulation of operational space control. *The Journal of the Astronautical Sciences*, 40(1):27–50, January-March 1992.

[9] **A. Jain**, and **G. Rodriguez**. Recursive flexible multibody system dynamics using spatial operators. *Journal of Guidance, Control and Dynamics*, 15:1453–1466, November 1992.

[10] **K. Kreutz-Delgado**, and **G. Rodriguez**. Spatial operator factorization and inversion of the manipulator mass matrix. *IEEE, Transactions on Robotics and Automation*, 8(4):65–76, February 1992.

[11] **A. Jain**, and **G. Rodriguez**. An analysis of the kinematics and dynamics of under-actuated manipulators. *IEEE Transactions on Robotics and Automation*, 9(4):411–422, August 1993.

[12] **A. Jain**, and **G. Rodriguez**. Linearization of manipulator dynamics using spatial operators. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):239–248, January-February 1993.

[13] **A. Jain**, **N. Vaidehi**, and **G. Rodriguez**. A fast recursive algorithm for molecular dynamics simulation. *Journal of Computational Physics*, 106:258–268, June 1993.

[14] **A. Jain**, and **G. Rodriguez**. Recursive dynamics algorithm for multibody systems with prescribed motion. *Journal of Guidance, Control and Dynamics*, 16(5):830–837, September-October 1993.

[15] **A. Visioli**. Trajectory planning of robot manipulators by using algebraic and trigonometric splines. *Robotica*, 18:611–631, 2000.

[16] **K.M. Tse**, and **C. Wang**. Evolutionary optimization of cubic polynomial joint trajectories for industrial robots. *IEEE*, 4:3272–3276, 1998.

[17] **W. Yongji**. Nonholonomic motion planning: A polynomial fitting approach. *Proceedings of the IEEE Conference on Robotics and Automation*, 4:2956–2961, 1996.

[18] **W.L. Xu**, **B.L. Ma**, and **S.K. Tso**. Curve fitting approach to motion planning of nonholonomic chained systems. *Proceedings of the International Conference on Robotics and Automation*, 1:811–816, 1999.

[19] **Z.S. Tumeh**. Cartesian-based time delay compensation in leader follower coordination of two manipulators using polynomial time functions. *Proceedings of IEEE Conference on Systems, Man and Cybernetics*, 2:979–984, 1998.

[20] **Hourtash A.**, and **M. Tarokh**. Manipulator path planning by decomposition: Algorithm and analysis. *IEEE Transactions on Robotics and Automation*, 7(1):42–70, 2001.

[21] **D. Hsu**, **J.C. Latombe**, and **R. Motvani**. Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications*, 9(4):495–512, 1999.

[22] **Lavalle S.M.**. *Planning Algorithms*. Cambridge University Press, 2005.

[23] **J.Y. Hwang**, **J.S. Kim**, **S.S. Lim**, and **Park K.H.**. A fast path planning by path graph optimization. *IEEE Transactions on Systems,Man and Cybernetics,Part A:Systems and Humans*, 33:121–128, 2003.

[24] **J. Hwang**, and **Ahuja N.**. Path planning using a potential field approach. *IEEE*, 1:569–575, 1989.

[25] **K.S. Fu**, **Gonzalez R.C.**, and **Lee C.S.G.**. *Robotics Control, Sensing, Vision, and Intelligence*. Mcgraw-Hill Book Company, 1987.

[26] **B.K. Choi**, and **D.W. Kim**. Bounded deviation joint path algorithm for piecewise cubic poynomial trajectories. *IEEE Transactions on Systems, Man and Cybernetics*, 20:725–733, 1990.

[27] **S.G. Lee**, and **B.H. Lee**. Collision-free motion planning for two robots. *IEEE Transactions on Systems, Man and Cybernetics*, 17(1):21–32, January-February 1987.

[28] **C Chang**, **M. J. Chung**, and **Z. Bien**. Collision-free motion planning for two articulated robot arms using minimum distance functions. *Robotica*, 8:137–144, 1990.

[29] **B.H. Lee**. Constraint identification in time-varying obstacle avoidance for mechanical manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):140–143, January-February 1989.

[30] **Z. Bien**, and **I. Lee**. A minimum-time trajectory planning method for two robots. *IEEE Transactions on Robotics & Automation*, 8:414–418, June 1992.

[31] **C. Chang**, **J. Chung**, and **B. H. Lee**. Collision avoidance of two general robot manipulators by minimum delay time. *IEEE Transactions on Systems,Man and Cybernetics*, 24(3):517–522, 1994.

[32] **Wu C.H.**, **Lee D.T.**, **Freter K.**, and **Kao-Shing Hwang**. An automated collision avoidance motion planner among moving objects or machines. In *Proceedings of IEEE Conference on Decision and Control*, pages 367–372, December 1991.

[33] **A. Jain**, and **Rodriguez G.**. Computational robot dynamics using spatial operators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000.

[34] **Richard M. Murray**, **Li Zexiang**, and **S. Shankar Sastry**. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[35] **B. Siciliano**, and **Sciavicco L.**. *Modeling and Control of Robot Manipulators(2nd Edition)*. Springer, 2003.

[36] **Rotenberg Steve**. Orientations and quaternions. Technical Report CSE 169, University of California, December 2005.

[37] **K. Hwang**, and **M. Tsai**. Online collision avoidance trajectory plannning of two planar robots based on geometric modeling. *Journal of Information Science and Engineering*, 15:131–152, 1999.

[38] **R. Zurawski**, and **Phang S.**. Path planning for robot arms operating in a common workspace. *Industrial Electronics, Control, Instrumentation, and Automation*, 2:618–623, November 1992.

APPENDIX A

Initial Configuration of The Cartesian Robot 1

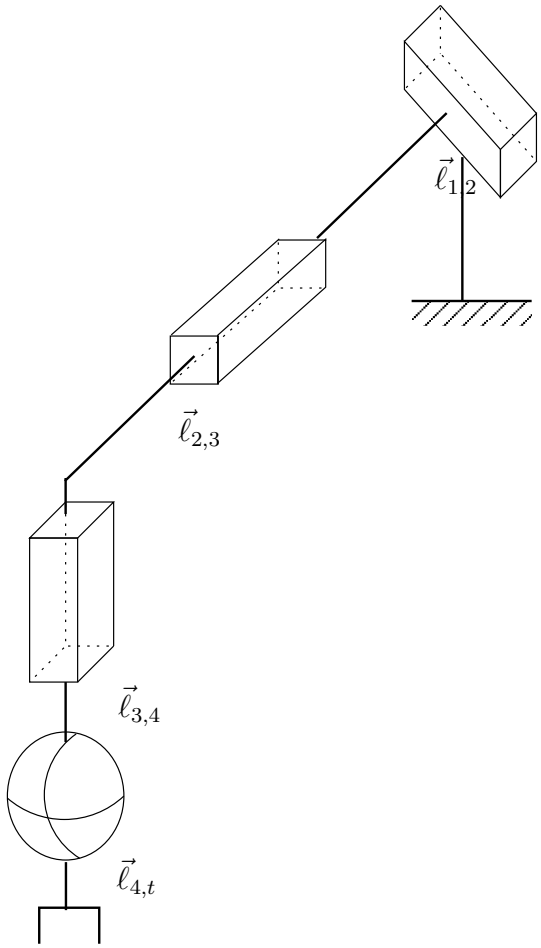

Figure A.1: Cartesian robot 1

APPENDIX B

Initial Configuration of The Cartesian Robot 2

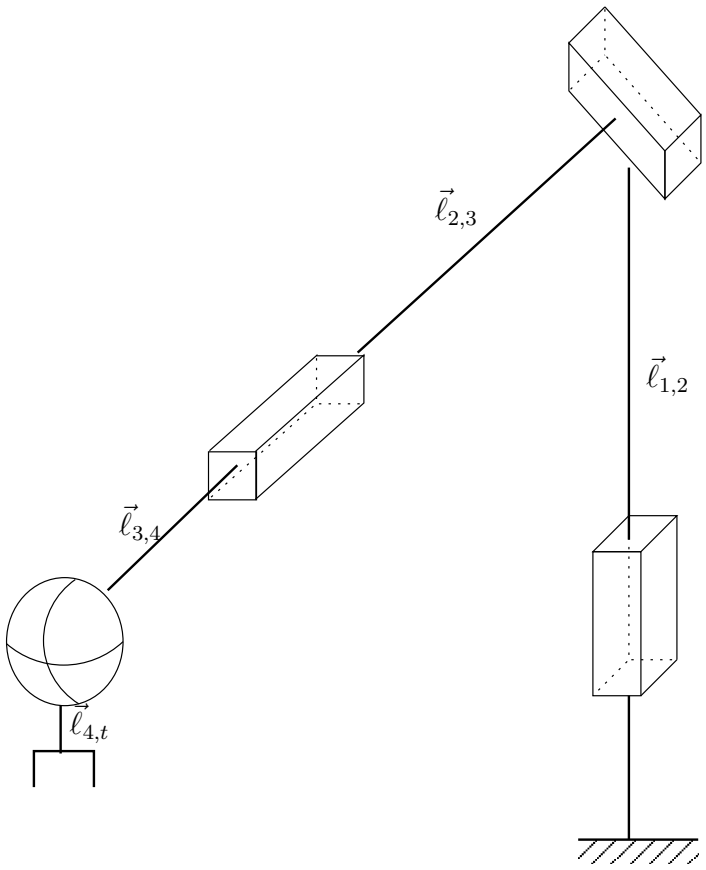

Figure B.1: Cartesian robot 2

APPENDIX C

Collision Avoidance in Virtual Reality Toolbox

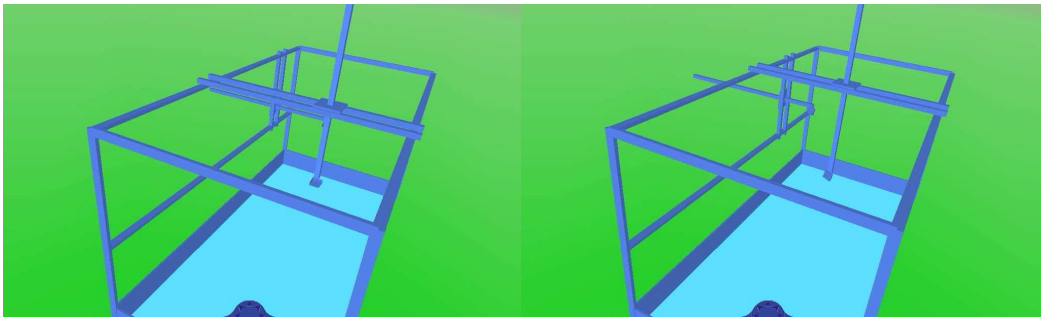

Figure C.1: Collision Avoidance for given trajectories at time=0 and time=t_1

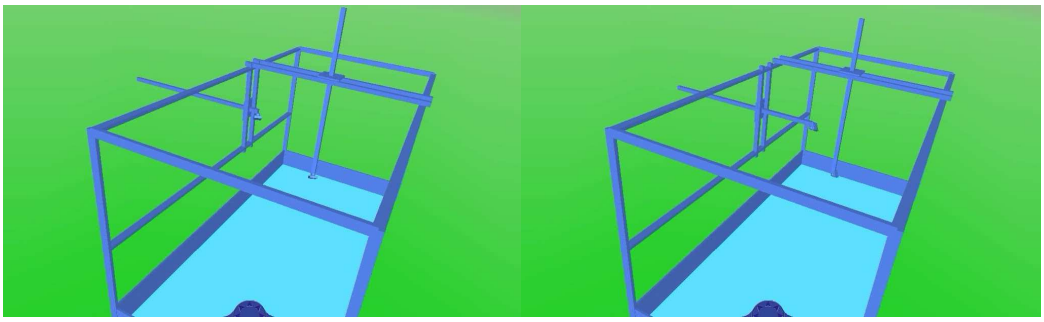

Figure C.2: Collision Avoidance for given trajectories at time=t_2 and $time = t_f$

BIOGRAPHY

Nihan Yeşiloğlu was born in İstanbul in 1981. She graduated *Summa Cum Laude* from Junior High School and *Cum Laude* from High School both in Kdz. Ereğli in 1996 and 1999, respectively. She graduated from Electrical Engineering Department of İstanbul Technical University in 2003.

She is currently pursuing her master's degree in Mechatronics Engineering at the Institute of Science and Technology of ITU, since 2003. She got married with Murat Yeşiloğlu on 21 January 2006. She will continue her education on Control and Automation Engineering doctora program at the Institute of Science and Technology of ITU.