Path Planning and Trajectory Optimization of Delta Parallel Robot

zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)
der Fakultät für Maschinenbau
der Universität Paderborn

genehmigte
DISSERTATION

von
M.Sc. Zeeshan Shareef
aus Karachi, Pakistan

Tag des Kolloquiums:	7. Mai 2015
Referent:	Prof. Dr.-Ing. habil. Ansgar Trächtler
Korreferent:	Prof. Dr. Sina Ober-Blöbaum
Korreferent:	Prof. Dr.-Ing. habil. Walter Sextro

Erklärung:

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Paderborn, Mai 2015.

Foreword

This work summarizes my three years of research at the department of Control Engineering and Mechatronics, University of Paderborn, Germany.

First of all, I am very grateful to Prof. Dr.-Ing. habil. Ansgar Trächtler for showing his confidence in me and to give me a chance to join his research group. His valuable comments and suggestions guided me to a successful completion of my dissertation. I would also like to thank Prof. Dr. Michael Dellnitz and Prof. Dr. Wilhelm Schäfer as being my co-supervisors.

In addition, I sincerely acknowledge the International Graduate School (IGS), for financial and administrative support during the whole duration of my research. I am very grateful to Prof. Eckhard Steffen and Ms. Astrid Canisius for their assistance and to help at the initial stages of settlement in Paderborn.

I am very thankful to my all colleagues at the chair of Control Engineering and Mechatronics for the help that they provided me and specially for friendly working environment. I would like to thank specially Dr. Karl-Peter Jäker for his technical discussions, interesting suggestions and talking with me every morning to improve my German language skills. I was very lucky that from the beginning I got involved in a project supervised by Dr.-Ing. Viktor Just. His managerial skills, working attitude and the way to handle the problems always motivate me. Besides that, I appreciate the support of Dipl.-Ing. Martin Leibenger, Dipl.-Ing. Heinrich Teichrieb and Dipl.-Wirt.-Ing. Christopher Lankeit to help me in the practical implementation phase.

This work is a result of many lengthy discussions with M.Sc Salman Zaidi and M.Sc. Zubair Usman. They both helped me a lot in research publications, proofreading my documents and motivating me at each and every step. I would also like to thank them for the proofreading of my dissertation. Special thanks to Mr. Zubair Usman who always considered me as his elder brother. He always tried to help me in each and every possible way.

I would like to thank Dr. Abubakr for motivating me at each and every step. Although, I worked with him only for nine months, but in this short period of time I learned a lot from him. He showed me the proper direction for research. I would also like to thank my some very close friends Hassan Ali, Abid Hussan, Sehar Shahzad Farooqe, Faisal Farooque and Syed Atif Adnan. Their love, friendship and brotherhood means a lot to me.

I would like to thank my German Language teachers Ms. Jennifer Breithecker and Ms. Olha Rachynska for teaching me throughout my PhD studies. They tried their best to teach me. Besides that, I would also like to appreciate the encouragement to learn

German language from my colleagues M.Sc. Johannes Renninger and Dr. Karl-Peter Jacker. The special thanks goes to Ms. Farisoroosh Abrishamchian. We were sitting together since March 2011. Her friendly behavior relaxed the office environment. She was really helpful and specially she translated a lot of documents and emails for me from German Language to English Language.

My last thoughts go to my parents in Pakistan. I thank them for inculcating in me the pursuit of learning and scholarship. Most of all, I am eternally grateful to them for their loving affection and prayers. The integrity of my father and the meticulous hardworking nature of my mother have been a guiding light for me throughout my life. Off course, I would also like to thank my elder brothers and sisters because they are the ones who teach me basics of computer and some other programs. In the early stage of my education, they helped me a lot in making assignments and presentations. Although I am living alone from seven years because of my higher studies but still I can feel their love and affections.

At the end, I owe my deepest thankfulness to my wife, Anum, for her understanding, motivating words. It was her who bore the burden of my workload, busy weekdays, never-ending publication deadlines and living in a small studio apartment. I thank her for appreciating the value of my work, and for believing in my judgement to pursue this degree despite all the hardship. Words cannot express my appreciation for providing the comfort of a good home, wonderful food and great companionship.

Paderborn, May 2015 Zeeshan Shareef

to my parents and my wife

Abstract

In this dissertation, path planning and trajectory optimization are performed using newly developed methodologies and results are compared with existing state-of-the-art techniques. Different state-of-the-art path planning techniques are discussed and compared not only in terms of numerical accuracy but also in terms of other properties. Trajectory optimization of a predefined geometrical path is discussed using three different optimization techniques: Phase-Plane method, Dynamic Programming and the newly developed method Discrete Mechanics and Optimal Control (DMOC). A joint selection criterion and modification in the algorithm are presented to remove the flaws present in Dynamic Programming using Joint Space for trajectory optimization. Two numerical examples are considered to compare these optimization techniques. In this thesis, a novel idea is proposed to combine path planning and trajectory optimization in a single step. One of the biggest advantages of combining these two steps together is that all constraints, either on kinematics or on dynamic properties, are considered in one step. DMOC is used to obtain the optimal solution of simultaneous path planning and trajectory optimization. The Delta parallel robot is used to verify the proposed generalized methodologies.

Zusammenfassung

In dieser Dissertation werden Pfadplanung und Trajektorienoptimierung mit Hilfe neu entwickelter Methoden behandelt. Die Ergebnisse werden mit den etablierten Methoden verglichen. Verschiedene Verfahren zur Pfadplanung auf dem aktuellen Stand der Technik werden erörtert und nicht nur bezüglich ihrer numerischen Genauigkeit, sondern auch hinsichtlich anderer Eigenschaften verglichen. Für den vordefinierten geometrischen Pfad wird die Trajektorienoptimierung mittels drei unterschiedlicher Optimierungsmethoden untersucht: Phase-Plane Methoden, dynamische Programmierung und durch die neu entwickelte Methode Discrete Mechanics and Optimal Control (DMOC). Um Schwachstellen bei der Verwendung der dynamische Programmierung für Optimierung im Gelenkwinkelraum zubeheben wird ein Kriterium für die Gelenkauswahl und eine Modifikation der Algorithmen vorgestellt. Es werden zwei numerische Beispiele für den Vergleich dieser Optimierungstechniken verglichen. In dieser Arbeit wird eine neuartige Idee vorgestellt, um die Pfaplanung und Trajektorienoptimierung in einem einzigen Schritt zu vereinigen. Einer der größten Vorteile der Vereinigung dieser beiden Schritte ist, dass alle Randbedingungen - sowohl die kinematischen als auch die dynamischen Eigenschaften - in einem Schritt berücksichtigt werden. Um die optimale Lösung der simultanen Pfadplanung und Trajektorienoptimierung zu erhalten, wird die DMOC-Methode verwendet. Der Delta Parallel Roboter wird benutzt um die vorgeschlagenen generalisierten Methoden zu verifizieren.

Contents

Page

List of Abbreviations

DOF	Degree of Freedom
PRM	Probabilistic Roadmap
GA	Genetic Algorithm
DP	Dynamic Programming
DMOC	Discrete Mechanics and Optimal Control
BVP	Bounded Value Problem
QP	Quadratic Programming
SQP	Sequential Quadratic Programming
VLC	Velocity Limit Curve
SPPTO	Simultaneous Path Planning and Trajectory Optimization
EA	Evolutionary Algorithms
NN	Neural Network

List of Symbols

$\theta_1, \theta_2, \theta_3$	Joint angles
$\varphi_1, \varphi_2, \varphi_3$	Rotation angles of robotic joints
$\delta\vec{q}$	Variation
$\vec{\tau}$	Vector of torques acting on joint
$\vec{\tau}_{Gb}$	Torque due to the gravitational force
$\vec{\theta}$	Vector of joint angles, $\vec{\theta} = [\theta_1\ \theta_2\ \theta_3]$
$\vec{\dot{\theta}}$	Vector of joint angular velocities, $\vec{\dot{\theta}} = [\dot{\theta}_1\ \dot{\theta}_2\ \dot{\theta}_3]$
$\Phi\left(\dot{s}\left(j_k, k\right)\right)$	Performance index to move from point k to $k+1$ with a pseudo-velocity $\dot{s}[j_k]$.
$\Phi\left(\dot{q}_i^*\left(j_k, k\right)\right)$	Performance index to move from point k to $k+1$ with a velocity $\dot{q}_i^*[j_k]$.
C	Cost function in objective function
C_d	Discrete cost function
$\mathbf{C}$	Centrifugal and Coriolis Coefficients matrix
C^E	Curve in Euclidean space
$\vec{c}(s)$	Vector of dimension $n \times 1$ representing $\mathbf{C}$ in path parametric domain, $\vec{c}(s) = [c_1, c_2, \ldots, c_n]$
e	Length of equilateral triangle's each side formed at travelling plate of Delta parallel robot
$\mathcal{F}$	Force
f	Length of one side of the equilateral triangle that inscribed the circle formed by the three actuators points
F_1, F_2, F_3	Position of each actuator connected to fixed base of Delta parallel robot
$F_{i_x}, F_{i_y}, F_{i_z}$	x-, y- and z-coordinates of point F_i, $i \in \{1, 2, 3\}$
$\vec{f}_k^-, \vec{f}_k^+$	Left and right discrete forces
$\vec{F}_{in}$	Inertial force
$\mathbb{F}^{f-}L_d, \mathbb{F}^{f+}L_d$	Legendre transform

$\vec{G}$	Gravitational force vector
$\vec{F}_g$	Gravitational force
g	Gravitational constant (9.8m/sec^2)
$\vec{g}(s)$	Vector of dimension $n \times 1$ representing $\vec{G}$ in path parametric domain, $\vec{g}(s) = [g_1, g_2, \ldots, g_n]$
I_m	Inertia of motor
$\mathbf{I}_b$	Inertia matrix
I_{b_i}	Inertia of each arm, $i \in \{1, 2, 3\}$
i^*	Reference non-stationary joint
J	Objective function
J_d	Discretized objective function
$\mathbf{J}$	Jacobian matrix
J_1, J_2, J_3	Connection point between each arm and forearm of Delta parallel robot
$J_{i_x}, J_{i_y}, J_{i_z}$	x-, y- and z-coordinates of point J_i, $i \in \{1, 2, 3\}$
L	Lagrange function
L_d	Discrete Lagrange function
L_A	Length of arm of Delta parallel robot
L_B	Length of forearm of Delta parallel robot
$\mathcal{L}$	Maximum deceleration among all joint actuators deceleration
L_i	Deceleration of joint actuators, $i \in \{1, 2, 3\}$
$\mathbf{M}$	Mass matrix
$\vec{m}(s)$	Vector of dimension $n \times 1$ representing $\mathbf{M}$ in path parametric domain, $\vec{m}(s) = [m_1, m_2, \ldots, m_n]$
m_a	Mass of the arm
m_b	Mass of the forearm (single rod)
m_c	Mass of the travelling plate
m_{elbow}	Mass of the elbow
m_{nt}	Total mass
N	Number of discretized points

O	Origin of coordinates system of Delta parallel robot
$\vec{P}_0$	Position vector of TCP, $\vec{P}_0 = [P_{0_x}\ P_{0_y}\ P_{0_z}]$
$P_{0_x}, P_{0_y}, P_{0_z}$	Position of TCP in x-, y- and z-direction
$\vec{\dot{P}}_0$	Velocity vector of TCP, $\vec{\dot{P}}_0 = [\dot{P}_{0_x}\ \dot{P}_{0_y}\ \dot{P}_{0_z}]$
$\dot{P}_{0_x}, \dot{P}_{0_y}, \dot{P}_{0_z}$	Velocity of TCP in x-, y- and z-direction
P_1, P_2, P_3	Midpoint of equilateral triangle's each side formed by connecting forearms to travelling plate of Delta parallel robot
P'_1, P'_2, P'_3	Corners of equilateral triangle formed by connecting forearms to travelling plate of Delta parallel robot
p	Parameter vector
$\vec{q}^0$, $\vec{\dot{q}}^0$	Initial conditions
$\vec{q}^f$, $\vec{\dot{q}}^f$	Terminal conditions
Q	Configuration space
$\vec{q}$	Generalized configuration vector
$\vec{q}_d$	Discretized path
$\vec{\dot{q}}$	Velocity of generalized configuration vector
R_A	Distance from the center of base to the motor joint
R_B	Distance from the center of travelling plate to the joint
s	Path parameter
$\dot{s}$	Pseudo-velocity, $\dot{s} = ds/dt$
$\ddot{s}$	Pseudo-acceleration, $\ddot{s} = d\dot{s}/dt$
$s_0,\ s_f$	Path parameter at starting and final time, respectively
$\dot{s}_{min},\ \dot{s}_{max}$	Minimum and maximum pseudo-velocity
T	Kinetic energy
t_0	Start time
t_f	Final time
Δt	Time-step
$\mathcal{T}Q$	Tangent space
$\mathcal{T}^*_{q(t)}Q$	Cotangent space

U_i	Deceleration of joint actuators, $i \in \{1, 2, 3\}$
$\mathcal{U}$	Minimum acceleration among all joint actuators' acceleration
$\vec{u}$	Vector of control variable
$\vec{u}_d$	Vector of discretized control variable
$\vec{u}_F$	Force as vector of control variable
$\vec{u}_k$	Discretized point of control variable vector
$\vec{u}_{min}$	Minimum joint torque/force
$\vec{u}_{max}$	Maximum joint torque/force
V	Potential energy
$\mathcal{W}_R$	Space occupied by robotic manipulator
$\mathcal{W}_O$	Space occupied by Obstacle
$\mathcal{W}_W$	Complete work space of robotic manipulator

List of Figures

List of Tables

1 Introduction

Nowadays, no one can deny the importance of robotics in our daily life. It is a continuously expanding field because of its wide applications in agriculture, industry, home, surveillance, hospitals, etc. Robots can be roughly divided into two types based on their structure and usage, namely mobile robots and industrial robots. Mobile robots have the capability to move around in their environment and are not fixed to one physical location. In contrast, industrial robots usually consist of a jointed arm (multi-linked manipulator) and gripper assembly (or end effector) that is attached to a fixed surface. Mobile robots have a very diversified and broad area of application. These robots are used in navigation, surveillance, agriculture, cooperation, and in many other fields as it has a larger workspace compared to industrial robot.

One of the earliest mobile robot was developed by W. G. Walter [1], shown in Figure 1.1. It was an electromechanical system that moves toward the light source, avoiding the moving obstacles on its way [7, 8]. It has three wheels whereby only the front wheel was actuated by two motors. One motor was used for rotational motion of the front wheel and the other motor was used to steer the direction of the wheel. Walter's robot was designed to demonstrate and analyze biologically inspired behaviours and to demonstrate the interaction between two sensory systems: light-sensitive and touch-sensitive control mechanisms. This robot also had the ability to move towards a charging station to recharge its battery.

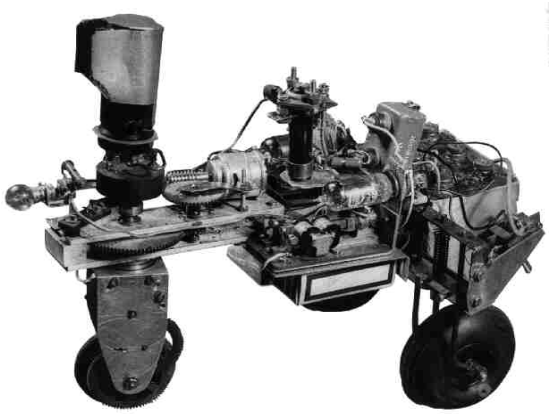


Figure 1.1: The first mobile robot built by W. G. Walter [1]

The successful implementation of the first mobile robot opened a new era for mobile robotics. According to the working environment in which they travel, mobile robots can be classified into three types, namely land or home robots referred to as Unmanned Ground Vehicles (UGVs), aerial robots referred to as Unmanned Aerial Vehicles (UAVs) and Underwater robots are usually called Autonomous Underwater

Vehicles (AUVs). The mobile robots are not only developed for educational purposes but also for many other applications. Typical mobile robots are shown in Figure 1.2.

(a) Ground Vehicle (Warrior)

(b) Aerial Vehicle (Parrot AR Drone)

(c) Underwater Vehicle (PRO 3 ROV)

Figure 1.2: Typical examples of mobile robots

Although mobile robots have a lot of applications and the advantage of mobility and large workspace, however they have some disadvantages of design constraints including light weight and short durability of batteries. Short durability of batteries is always a serious issue and sometimes the running operations must be stopped in order to charge the battery. On the other hand, industrial robots are normally fixed to a certain position and do not have such restrictions.

Unlikely mobile robots, there is no more issue of battery charging in industrial robots as they are normally operated using the electric supply. One of the major advantages of industrial robots is their load handling capabilities which give them a paramount importance in industrial applications. From the economic point of view, the ability of industrial robots to operate virtually non-stop makes them very useful for the manufacturing sector. Besides, industrial robots are very much capable of performing assigned tasks at high speed and with good precision. These properties of industrial robots led to their usefulness in the high-tech manufacturing, for example, in the automotive production.

The first industrial robot was developed by G. P. Taylor [2], shown in Figure 1.3. This industrial robot was a device like crane and powered by a single motor. Automation in five axes was achieved using punched paper tape. To perform the desired operation in this robot, first the motor's revolutions were plotted on graph paper and then this information is transformed to the paper tape. Almost after 15 years in 1955, George Devol and Joe Engelberger developed the first company, called *Unimation, Inc.*, based on the G. Devol's first robotics patent [9]. They used hydraulic actuators in the first industrial robot. This industrial robot was installed in 1961 at a manufacturing plant of General Motors [10].

The installation of industrial robots at manufacturing plants opened the doors for industrial robots in reconfigurable interactive manufacturing, shop floor logistics, ma-

nipulation, plant servicing and inspection. The use of robots in all types of industries is increasing day by day. According the current statistics, 168,000 units of industrial robots were sold in year 2013 that showed about 6% increase in its demand compared to year 2012 in which 159,346 industrial robots were sold out[1].

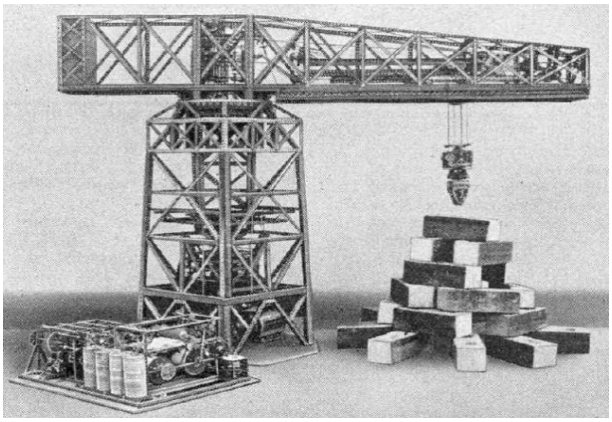


Figure 1.3: First industrial robot built by G. P. Taylor [2]

According to mechanics, industrial robots can be divided into two major categories, namely serial manipulators and parallel manipulators. Serial manipulators are the most common industrial robots and they are designed as a series of links that extend from a base to the end effector. Serial robots usually require six joints to place a manipulated object in an arbitrary position and orientation in the workspace of a robot. Unlikely, parallel manipulator uses several serial chains to support a single platform. In parallel manipulators *"parallel"* is used in the topological terms, not in the geometrical sense. These linkages act together, but it is not supposed that they are placed as parallel lines. Parallel manipulators have closed kinematic chains and each chain is usually short, simple and must still move within its own degree of freedom. It is this closed-loop stiffness that makes the overall parallel manipulator stiff relative to its components, unlike the serial chain that becomes progressively less rigid with more components. Typical serial and parallel manipulators are shown in Figure 1.4.

If we compare the serial manipulators with parallel manipulators, serial manipulators have the advantages of heavy load bearing capabilities and large workspace compared to the parallel manipulators. Despite the disadvantage of small workspa- ce, the parallel manipulators have the advantage of high speed and high precision. Unlike serial manipulators, parallel manipulators are stiff relative to its components and there is no unwanted flexibility and sloppiness in joints. As parallel manipulators consist of several independent serial chains, so the error in one serial chain does not accumulate and does not affect the other serial chains compared to the serial manipulators in which

[1] Source : International Federation of Robotics (www.ifr.org)

(a) Example of Parallel Manipulator (Courtesy of ABB Flexible Automation)

(b) Example of Serial Manipulator (KUKA Robotics Cooperation)

Figure 1.4: Typical examples of industrial robots

errors are accumulated and amplified from link to link. The nonlinear behaviour of parallel manipulator is the biggest disadvantage and this nonlinear behaviour is the main reason that parallel manipulators are used rarely in precision machining, despite their high-speed and high-precision.

Parallel manipulators can be further divided into four types [11]:

1) Symmetric parallel manipulator

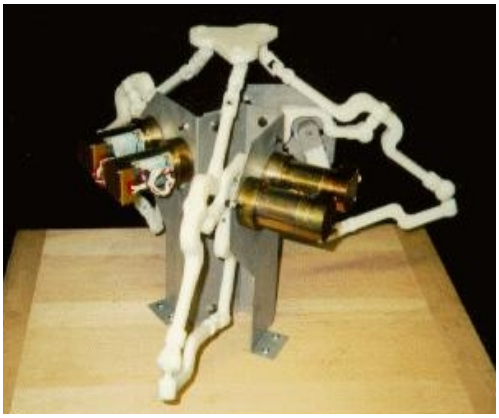
2) Planar parallel manipulator

3) Spherical parallel manipulator

4) Spatial parallel manipulator

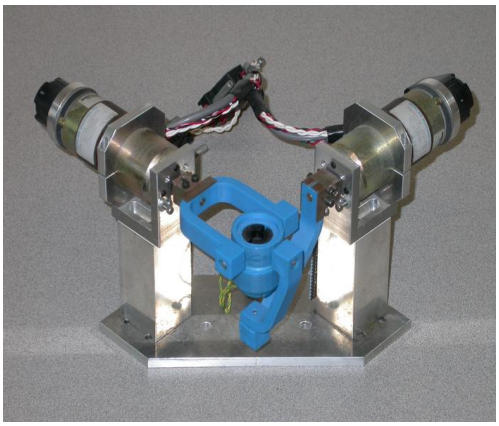
Symmetrical parallel manipulator, shown in Figure 1.5(a), have number of limbs equals to the number of degrees of freedom. A planar parallel manipulator, shown in Figure 1.5(b), can be built by connecting the rigid platform by two or more planar kinematic chains. The spherical parallel manipulators are only restricted for the spherical movement of the end effector. Figure 1.5(c) shows a spherical parallel manipulator used for the precise control of tracking camera. Spatial parallel manipulators are the most popular parallel manipulator type, shown in Figure 1.5(d), attracted the attention of researchers and industrialists.

Besides their industrial applications, parallel manipulators are also used in many other areas. One of the widely used parallel manipulator is Stewart Gough platform [12]. Stewart proposed this parallel manipulator for the purpose of flight simulator and different versions of this platform are still in use. Stewart platform is also used for some other applications, e.g. milling machines [13], pointing devices [14] and underground



(a) Symmetrical parallel manipulator

(b) Planar parallel manipulator



(c) Spherical parallel manipulator

(d) Spatial parallel manipulator

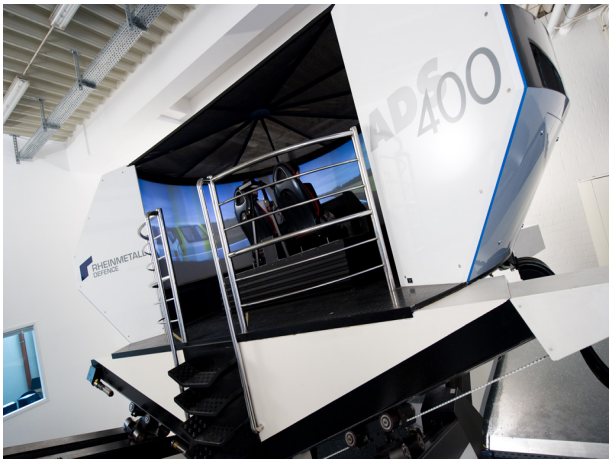
Figure 1.5: Types of parallel manipulators

excavation devices [15]. The other important applications of parallel manipulators are driving simulators and haptic devices for surgical robotics, shown in Figure 1.6.

1.1 Motivation and Objectives

As described earlier, industrial manipulators are widely used in industrial applications. Large load handling capabilities and no battery charging issue give industrial manipulator a prime position in automotive industries. Optimal trajectory planning in robotics has recently gained a lot of attention because of its extensive use not only in industrial applications and in our daily life. An optimal motion of an industrial robot is the key to success because it can help to increase the production rate and system utilization, and to reduce the production cost, cycle times, and energy consumption.

For tractability, the optimal trajectory of robotic manipulators is divided into three stages, shown in Figure 1.7. The first stage is path planning. Extensive amount of work has been done in the domain of path planning [16]. The important problems addressed in path planning include collision avoidance and getting the smooth path.



(a) Driving simulator (Rheinmetall AG)

(b) Flight simulator (ATR 42-300)

(c) Haptic device (Novint Falcon)

Figure 1.6: Use of parallel manipulator for different applications

Obstacles in path planning can be divided into two types:

- Dynamic obstacles
- Static obstacles

Mostly, dynamic obstacles are associated with mobile robots in which robots have to move from one place to another place in the presence moving objects like human beings. In industrial applications, mostly we deal with static obstacles. The workplace of robotic manipulators are very well-defined and normally there is cage around the industrial manipulators to avoid unwanted interference of living things and to avoid accidents. In last few decades, researchers have proposed a lot of algorithms to handle static and dynamic obstacles in path planning. These algorithms are discussed in detail in Section 2.1.

The second stage of path optimization is trajectory optimization for a given geometric path obtained from the path planning step. The general goal of trajectory optimization

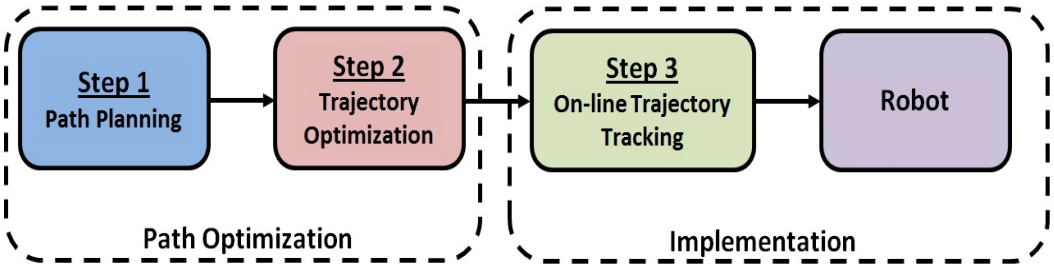


Figure 1.7: Division of path optimization problem for robotic manipulators for tractability

is to find the position of the robotic manipulator as a function of time while minimizing the desired objective function in parallel. Most commonly used objective functions for robotic manipulators are time minimization and energy minimization. Sometimes for different applications, more complex objective functions are also considered in trajectory optimization problem. The optimization in case of robotics becomes more difficult because of non-linearities and coupling in robot dynamics.

Just like any other mechanical system, robotic manipulators have some physical and dynamical limitations and these limitations which act as equality or inequality constraints have to be considered during trajectory optimization. Otherwise, in absence of constraints, the solution for trajectory optimization problem will be trivial. Normally, these constraints are defined as lower and upper bounds on robot joints' velocities, accelerations, forces and torques. Besides these inequality constraints, additional constraints can be defined as minimum/maximum rate of change of acceleration, torque, etc. The violation of equality or inequality constraints may cause damage of robotic manipulators or any serious accident.

The numerical solution of optimal control problems for robotic manipulators under given constraints can be calculated by either direct or indirect optimization methods [17]. The indirect methods are based on the calculus of variations or the maximum principle. The optimal solution can be obtained by using gradient methods, Newton's method or multiple shooting method. In direct methods, the optimal control problem is transformed into a nonlinear programming problem. This nonlinear programming problem is solved by using either a penalty function method or methods of augmented or modified Lagrangian functions such as Sequential Quadratic Programming (SQP) methods. The two disadvantages of direct method are the less accurate solutions as compare to indirect method and the possibility to find local minima. To improve the low accuracy of the direct method and to increase the convergence of indirect method a hybrid approach must be used and it is necessary to combine direct with indirect methods [17].

One of the major problems in existing techniques to solve the optimization problem for robotic manipulator is to divide the optimal problem into different stages for tractability. Unfortunately, the simplicity obtained from the division into path planning and

trajectory optimization comes at the cost of efficiency. Combining the path planning and trajectory optimization would increase the overall efficiency and accuracy of the robotic manipulator as both steps will incorporate the robot's dynamical properties, but on the other hand it will increase the complexity and computational time.

1.2 Main Contributions

In this work, the focus lies solely on solving the path optimization problem by considering the industrial parallel robots, more specially Delta parallel robot. Delta parallel robot is specifically designed for fast transportation of light objects and its motors are kept fixed at the base, thus reducing the robot's structure active mobile mass.

The path planning problem for Delta parallel robot is solved by considering known static obstacles, as in industrial applications most of the obstacles are static. Although Delta parallel robot can be used for movement in 3 dimensions (3D) but for relatively simple movements because its forearms put restrictions to use it in 3D for complex movements in presence of obstacles. In this thesis, path planning for Delta parallel robot is considered in 2D and it is solved by using Probabilistic Roadmap method (PRM) and Genetic Algorithm (GA). The advantages and disadvantages of path planning using above mentioned techniques are discussed and the obtained results are compared.

In this thesis, the predefined geometrical path for robotic manipulators is optimized by different optimization techniques. The results obtained by different optimization techniques are not only compared in terms of accuracy but also in terms of computational complexity, computational time, and variable dependencies. Moreover, some existing algorithms have been modified and new criteria have been developed to reduce the computational time as well as to remove the possibility of getting trapped in a local minima. Improvements as a result of modification and proposed new criteria are demonstrated by numerical examples and the results are compared with other state-of-the-art methodologies.

One of the major contributions of this work is to perform the path planning and trajectory optimization simultaneously. Previously, the path planning and trajectory optimization steps were performed separately to reduce the computational complexities and computational cost. In this work, these steps are combined using a newly developed methodology. In the new methodology, combining both steps together incorporate the information about the robot's dynamics or dynamical model in path planning as well as in trajectory optimization steps. Numerical examples and the comparison of the obtained results with other optimization methodologies show the effectiveness of combining path planning and trajectory optimization together using new methodology.

Finally, path planning and trajectory optimization are implemented on the real hardware to show the practicability and applicability of the discussed and proposed schemes. Experiments are performed on a Delta parallel robot available at Heinz Nixdorf Insti-

tute, University of Paderborn.

1.3 Thesis Outline

This thesis consists of seven chapters. The first chapter gives the brief introduction to the types of robots, advantages and disadvantages of serial and parallel industrial manipulators, and the applications of parallel manipulators. Next, the motivation and objective of this work are described, followed by the main contributions of this thesis.

Chapter 2 starts with the literature survey about different algorithms and methodologies for path planning and trajectory optimization for robotic manipulators. Different path planning techniques suitable for Delta parallel robots are discussed in details. Trajectory optimization techniques are discussed in terms of computational time and variable complexities. At the end, the scientific gaps and open questions in currently existing path planning and trajectory optimization techniques are discussed.

Chapter 3 introduces the Delta parallel robot used in this thesis. Inverse and forward kinematics of Delta parallel robots are discussed that are required for path planning. The dynamical model of the Delta parallel robot is also derived using the virtual work principle because it has to be incorporated in the trajectory optimization step. The total kinetic energy of Delta parallel robot is formulated by discussing the kinetic and potential energy separately. Finally, the Jacobian matrix describes the relationship between velocities in Joint space and Cartesian space.

Chapter 4 deals with the path planning for Delta parallel robots in two dimensions ($2D$). Probabilistic Roadmap (PRM) and Evolutionary Algorithm (EA) based methods are used for path planning in the presence of known static obstacles. Computational efficiency, limitations and accuracy of these methods are elaborated by two numerical examples.

Chapter 5 discusses the trajectory optimization of robotic manipulators for a given geometrical path using different optimization methodologies. Three different methods, namely Phase-Plane method, Dynamic Programming (DP) and Discrete Mechanics and Optimal Control (DMOC) are used for trajectory optimization. A modification in the already existing algorithm and a joint selection criterion is proposed to reduce the computational time and to overcome the local minima problem. The comparisons of these methods are given by two numerical examples applied on the Delta parallel robot.

Chapter 6 outlines the simultaneous path planning and trajectory optimization using a newly developed methodology. It describes the problem formulation, and equality and inequality constraints to handle the both steps together. The practicability and validation of the proposed idea is shown by two numerical examples. Optimal results obtained by Discrete Mechanics and Optimal Control (DMOC) are compared with the conventional methods in which path planning and trajectory optimization are

performed separately.

Chapter 7 summarizes the results and conclusions with some recommendations which may be useful for future studies.

2 Literature Review

Industrial robots are used as a primary means of contemporary automation due to their potential of productivity increase and product quality improvement. Industrial robots are widely used to achieve the demand of high precision and high speed positioning. Typical examples are welding, cutting, pick and place, and gluing. The productivity by industrial robots can further be maximized by path optimization. Optimal motion can help in reducing the machine cycle and increasing the system utilization. In optimization, one determines the control signals that will derive the manipulator from starting point to ending point while minimizing the specified objective function, subject to the constraints on controlling signals and the position of the manipulator.

Normally, the optimal control problem for robotic manipulators is complex because of the non-linearities and complex dynamics. To deal with the complexity, normally the problem is subdivided into three steps as shown in Figure 2.1. Detail discussion about each step is given in the subsections.

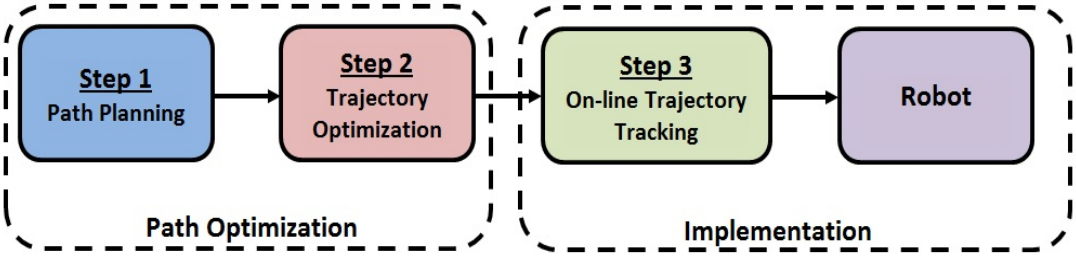


Figure 2.1: Three steps for path optimization for robotic manipulators

2.1 Path Planning

Path planning is the first step of path optimization for robotic manipulators. Most of the path planning techniques consider the manipulator kinematics and do not incorporate the manipulator's dynamics information. While planning motion, the pick and place positions and the intermediate path must be well defined. The main objective achieved in path planning is the collision avoidance from obstacles.

The workspace of robotic manipulators can be divided into two types:

- Workplace without obstacles
- Workplace with obstacles

Path planning in the absence of obstacles is just connecting the starting point to ending

point with a straight line. The main task in this scenario is to set the velocity profile along the straight line.

Nowadays, path planning in the presence of obstacles is a hot area of research. Besides the main objective of collision avoidance, other objectives are also considered in path planning like smooth path, minimum/maximum distance from the obstacle, etc. Path planning in the presence of dynamic obstacles is more complex as compared to path planning in the presence of static obstacles. Path planning in the presence of dynamic obstacles requires a lot of extra work, e.g. modelling of moving object, prediction of the moving object, etc., which makes it difficult. As already discussed, usually environment and obstacles are very well defined in industry and there is no interference of dynamic objects, so in this thesis we will only consider static obstacles.

Global path planning techniques use prior knowledge having a precise and complete description of environment and obstacles and normally these techniques work off-line. An algorithm is called complete if it always finds a solution or determines that none exists. A lot of work has been already done in the field of path planning with static and dynamic obstacles.

A* (pronounced A Star) is a leading path finding algorithm, proposed by Hart et al. [18]. A* finds a least-cost path from starting point to ending point using *best-first* search. The order of search visits nodes in the tree is determined by a knowledge-plus-heuristic cost function. By using heuristics, A* algorithm can achieve better performance. In the standard terminology used when talking about A*, $g(n)$ represents the exact cost of the path from the starting point to any vertex n, and $h(n)$ represents the heuristic estimated cost from vertex n to the goal. In Figure 2.2, the yellow (h) represents vertices far from the goal and teal (g) represents vertices far from the starting point. A* balances the two as it moves from the starting point to the goal. Each time through the main loop, it examines the vertex n that has the lowest $f(n) = g(n) + h(n)$. This method is widely used for path planning because it is generally outperformed by algorithms which can pre-process the graphs to attain better performance, but basically this algorithm was designed as a general graph traversal algorithm.

Taylor [19] formulated the path planning in the presence of uncertainty. His proposed method, called *Skeleton Refinement*, based on numerical uncertainty propagation techniques. This method was further improved by Brooks [20]. Brooks used symbolic propagation techniques for the path planning. Further improvement in skeleton based path planning algorithm is done by Yang and Hong [21]. The gradually increase the area across the vertex to remove the sharp edges and to shorten the path length.

The work that can be considered as “exact” motion planning was proposed by Lozanzo-Pérez and Wesley[22]. The modification of this algorithm was proposed by Udupa [23] and he proposed the first path planning algorithm for polyhedral and polygonal robots for obstacles avoidance without rotation. Lozanzo-Pérez also extended this idea by proposing the approximate cell decomposition approach [24, 25].

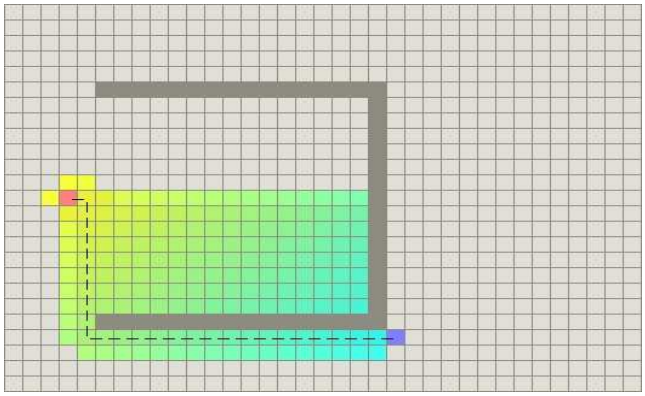


Figure 2.2: Graphical representation of A algorithm*

Dufay and Latombe [26] described an inductive learning based motion planning in the presence of uncertainty. In this approach, the same task is executed several times and the execution traces are combined into more general strategy. Nowadays, the learning based motion planning is a hot research and a lot of work is currently published on it [27, 28, 29].

Another very powerful path planning technique is the Potential Field, proposed by Khatib [30]. The idea of potential field is taken from the nature in which a charged particle navigates in magnetic field towards destination or a small ball rolls from the hill. The same idea is applied to robotics. Robot represents as a point in configuration space is metaphor as a particle moving under the artificial potential field. The target position generates the attractive field that attracts the robot and the obstacle produces the repulsive field that pushes the robot away from it. One of the major drawbacks of this method is the possibility to trap in a local minima. Another drawback is the balance of attractive and repulsive forces and in this case the robot will not move. In mathematical terms, the overall potential field is [31]:

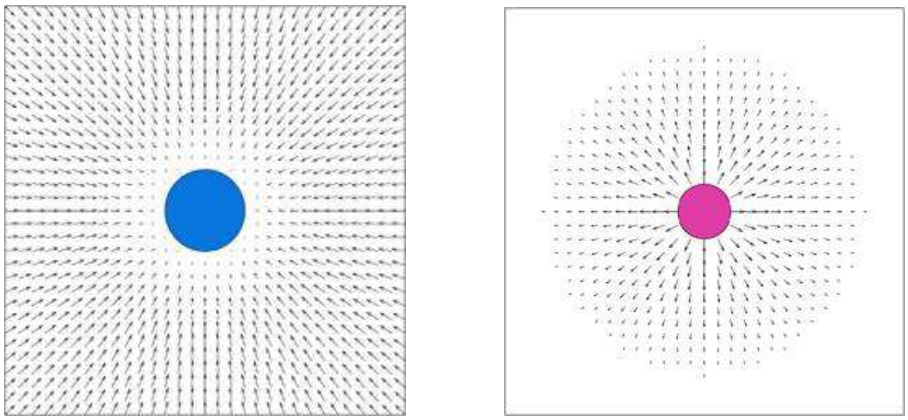
$$U(q) = U_{goal}(q) + \sum U_{obstacles}(q)$$

Many potential field functions are studied in the literature. Mostly, parabolic attractor is used for U_{goal}, given in (2.1).

$$U_{goal} = dist(q, goal)^2 \tag{2.1}$$

Similarly, $U_{obstacles}$ is modeled as potential barrier that approaches to infinity when robot approaches the obstacle.

$$U_{obstacles} = dist(q, obstacle)^{-1} \tag{2.2}$$



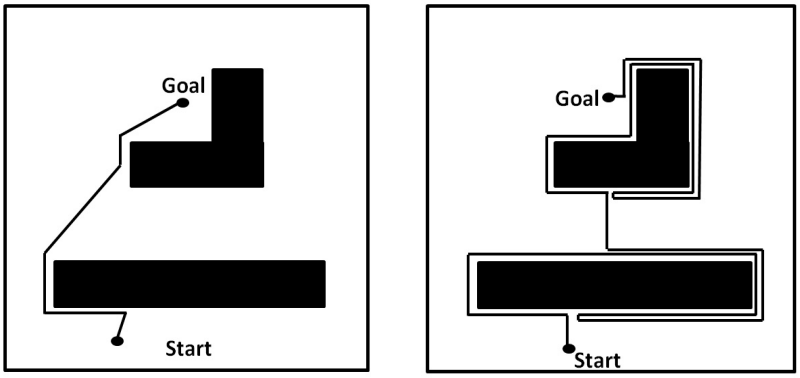
(a) Potential field by Goal position (b) Potential field by Obstacle

Figure 2.3: Potential Fields

Here *dist* is the Euclidean distance. The problem of local minima in Potential Field approach was solved by Barraquand and Latombe [32] and Warren [33]. The inherent limitations of potential field method and problems in using this method for mobile robot navigation are discussed by Koren and Borenstein [34]. This path planning method has been successfully implemented not only for indoor mobile robots but also for outdoor and underwater vehicles [35, 36].

Lumelsky and Stepanov [37] proposed a *Bug Algorithm* for path planning in the presence of obstacles. Bug algorithm assumes only local knowledge of the environment. The Bug's behaviour is very simple to understand, it always follows a wall and moves in a straight line towards goal. Lumelsky presented two types of bug algorithm. In *Bug 0* algorithm, a Bug heads towards the goal and goes straight until it interacts with an obstacle. In case of obstacles, bug follows the obstacles until it has head towards the goal again and then again goes straight towards the goal [37]. In *Bug 1* algorithm, bug circumnavigates the obstacle if it is encountered and remembers how close it is from the goal and it continues to the straight line after returning to the closest point [38]. Graphical representations of Lumelsky's algorithms are shown in Figure 2.4.

Most of the path planning algorithms only used the kinematic information of robots. In kinodynamics, as name implies, the path planning is done subject to simultaneous kinematic and dynamic constraints. This term was coined by Donald et al. [39]. They solved a long standing open problem in optimal control for robotics by providing a provably polynomial time $\varepsilon-$approximation algorithm. They incorporate the safety by introducing a speed dependent obstacle avoidance margin in problem parameter. In solution, the resulting motion is governed by dynamic equations. Pham et al. [40] re-



(a) Bug 0 Algorithm (b) Bug 1 Algorithm

Figure 2.4: Graphical representation of Lumelsky's Bug algorithm

duced the complexity of kinodynamic planning by proposing a method that enables the planning in configuration space (of dimension n) rather than state space (of dimension $2n$). Kinodynamics has been successfully used for path planning of mobile robots with moving obstacles [41, 42, 43], and visual servoing [44].

Rapidly-exploring Random Trees (RRT) is an algorithm to efficiently search nonconvex, high-dimensional spaces. It can easily handle problems with obstacles and differential constraints (nonholonomic and kinodynamic) and has been widely used in autonomous robotic path planning. LaValle is the pioneer of this method [45]. In this method, a rapidly-exploring random tree in state space is built from a starting point and this tree stops further building when it gets sufficiently close to the goal position [46]. RRT method was further refined by bidirectional search to grow two RRTs, one rooted at the initial state and other rooted at the goal state [47]. This method has a key position in autonomous robotic path planning, as it can be used in dynamic environments. A graphical representation of Rapidly-exploring Random Trees is shown in Figure 2.5.

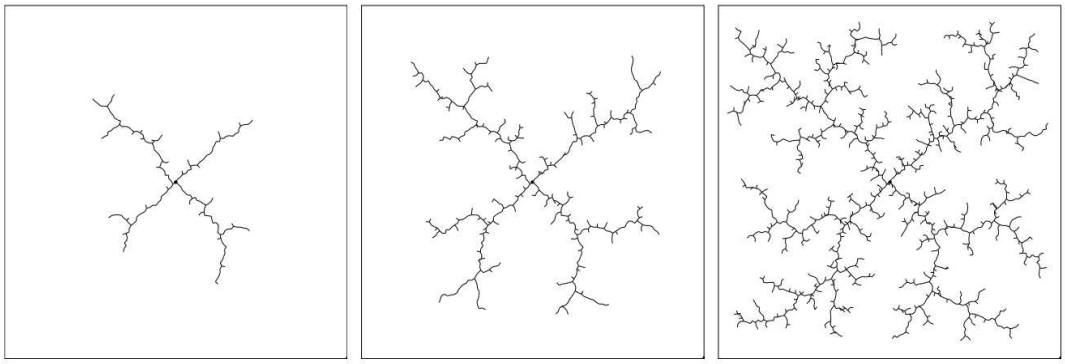


Figure 2.5: Graphical representation of RRT

Most of the path planning algorithms have a priori information about the terrain and the obstacles. Less attention has been paid to the problem of partially known envi-

ronments. Stentz [48] proposed an algorithm, called D* (pronounced D Star), for the exploratory robots that do not have floor plan and terrain map. It is a complete planner in unknown, partial known and changing environments. Widely used D* Lite is an improved version of D* algorithm that is proposed by Koenig and Likhachev [49]. D* Lite implements the same navigation strategy as D* but is algorithmically different. It simplifies the tie-breaking criterion and does not need nested if-statements. Although D* algorithm is a very efficient path planning algorithm, it has a disadvantage of high memory usage problem. Cockburn and Kobti [50] solved the problem by introducing Memory-Bound D* Lite algorithm that is similar to D* Lite, while applying memory usage constraints.

In high dimensional configuration space, path planning becomes more complex and the aforementioned methods are computationally very expensive for high dimensions. Probabilistic Roadmap (PRM) is an efficient way for path planning in high dimensionality. This method was proposed by Kavraki and Latombe [51, 52]. This method consists of two phases: a learning phase and a query phase. It is general and easy to implement and can be applied to n-degrees of freedom robots. PRM is a complete planner and computationally very fast because it uses a relatively weak but fast local planner [53]. PRM with other local planners is discussed by Amato et al. [54]. The improvements to increase the connectivity of roadmaps and to increase the computational efficiency have been discussed by Horsch et al. [55], Amato et al. [56], Wilmarth et al. [57], and Bohlin and Kavraki [58].

In recent years, besides the above mentioned conventional path planning algorithms, a lot of work has been done on the application of swarm intelligence in path planning. Some of the swarm intelligence based algorithms used for path planning are Particle Swarm Optimization (PSO) [59, 60], Ant Colony Optimization (ACO) [61, 62], Artificial Bee Colony optimization (ABC) [63, 64], Firefly Algorithms (FA) [65, 66], and Bat Algorithm (BA) [67].

In artificial intelligence, Evolutionary Algorithms (EA) are widely used for the optimization process. All EA algorithms are inspired by biological evolution and perform processes such as crossover, mutation, selection, inheritance, etc. These algorithms have wide applications and have been successfully applied in field of engineering, sciences, economics, mathematics and bioinformatics. Different EA algorithms are Genetic Algorithm (GA), Differential Evolution (DE) and Neuroevolution. Genetic algorithm (GA) is the most popular type of EA and widely used for optimization. In GA, randomly generated candidate solutions evolve towards better solutions until the termination condition is reached. Tendency to converge towards local optima and difficulties in using on dynamic data are the limitations of GA as compared to other optimization algorithms. GA has been successfully implemented for path planning for mobile robots in the presence of static known obstacles [68, 69, 70]. Tuncer and Yildirim [71] improved the applicability of GA for dynamic path planning of mobile robots by introducing a new mutation operator.

In this thesis, we will more focus on PRM, as it is a complete planner and its computational efficiency is also very good. Mostly, degrees of freedom for industrial robots are very high, as they are developed to perform the complex task. In this case, PRM would be the best choice for path planning because it can be used for n-degrees of freedom robots. Normally the output of the PRM is straight lines with sharp edges and redundant points. The redundant points and the sharp edges can be removed by applying the greedy algorithm and appropriate filter. Besides this, we will also focus on the GA for path planning of industrial parallel robots. The problem associated with GA is the validation of the obtained optimal solution and the only possibility is to compare the solution with other methodologies. In this work, we will compare the results of GA with results obtained from PRM to show its better performance.

2.2 Trajectory Optimization

Trajectory optimization is the second step of path optimization in which a predefined geometrical path, obtained by path planning step, is optimized for a desired objective. The importance of trajectory optimization is clearly obvious from the relationship among task execution, productivity and energy consumption. Research work on trajectory optimization started in late 1960s when Stepanenko [72] discussed the minimum energy control problem for manipulators. Kahn and Roth [73] proposed the near time optimal control for open-loop articulated kinematic chains. The suboptimal solution was expressed in terms of switching curves for each of the system controls. The constraints for the optimization problem were the constant maximum/minimum forces. As this method finds the suboptimal solution, therefore, it was computationally quite efficient compare to the optimal control techniques.

Luh et al. [74, 75] proposed the optimum path planning method using the "*Method of Approximate Programming (MAP)*" in which constraints are applied on Cartesian velocities and accelerations. The Cartesian velocities and accelerations bounds were identified experimentally. This method was computationally expensive because the transformation from Cartesian coordinate points to Joint coordinates points are required. This method doesn't incorporate any information and constraints related to dynamics of the robot that is another disadvantage of this method. The computation time problem is solved by introducing a "*Direct Approximate Programming Algorithm (DAPA)*". In *DAPA*, the solution is made feasible by contracting the distance between two solutions obtained by two iterations.

Kim and Shin [76, 77] developed the method for minimum-time planning under realistic conditions. Their method was similar to Luh's method but developed in Joint space which reduced its computation time. Kim's proposed method accurately calculates the deviation at each corner point which expresses the manufacturing reality more accurately and clearly than in case of implicit bounds. In this method, global problem is transformed into local optimization problem.

Bobrow et al. [78] and Shin et al. [79] separately developed similar methods to calculate the optimal trajectory for robotic manipulators for a given specified path. Their method uses a phase plane plot of the arc length s and its derivative to get the optimal solution in which dimensionality of the problem is reduced by converting dynamical equations to a set of second order differential equations using path parameter, defined as s. The phase plane plot of $(s, \dot{s})$ sometimes referred as Velocity Limit Curve (VLC) or the switching curve. In their method the bounds on the actuator torques were transformed to the acceleration bounds along the path. This technique was further modified by Rajan [80], Shin and McKay [81], Pfeiffer and Johanni [82], Slotine and Yang [83], McCarthy and Bobrow [84], Shiller and Lu [85], and Tarkiainen and Shiller [86]. Constantinescu and Croft [87] smoothed the time-optimal path obtained by Bobrow's method by imposing limits on the torque rates and in implementation phase the limits on the actuators' torques translate into state-dependent limits on the pseudo-acceleration($\ddot{s}$). However, there are some drawbacks associated with this method. First, it can only solve the time minimization problem. Secondly, the joints' torques can be changed instantaneously which is impractical for real applications.

In Bobrow's proposed method, the assumption of maximum acceleration or deceleration may fail at some critical points where the maximum acceleration causes the trajectory to cross the limit curve. Crossing the limit curve, however, violates at least one of the actuator constraints, and may results in deviations of the manipulator from the desired path. Shiller and Lu [88] presented an algorithm to explore the critical points and critical arcs and time optimal motions along critical arcs may be singular in the acceleration in the sense that it is neither maximum nor minimum. Singularity is the major problem in maneuvering of robotic manipulators that may cause the serious accidents at workplace. Chen and Desrochers [89] discussed the singularity issues in time-optimal problem. He converted the problem into non-singular optimal control problem by introducing a perturbed energy term in the performance index. Other issues related to singularities in robotic manipulators are discussed by Kieffer [90], Nenchev et al. [91], and Tsumaki et al. [92].

Besides the optimization techniques, numerical techniques are also very important for optimization because numerical techniques play a vital role in computational efficiency. Betts and Huffman [93] improved the computational efficiency by combining a nonlinear programming algorithm with discretization of the given trajectory dynamics that result large and sparse matrices. In this method, constraints are treated as algebraic inequalities that are satisfied by nonlinear programming.

Dynamic Programming (DP) is a very useful optimization technique which can solve the problem of optimization for highly nonlinear systems under strong constraints [94]. Incorporating any arbitrary constraints in optimization problem is very easy in DP compared to other aforementioned optimization techniques which require complex transformations to incorporate arbitrary constraints. Although Dynamic Programming requires heavy computational power, the following reasons motivate the researchers to

use it for optimization problem. First, the optimal solution is calculated offline. Therefore, the computational cost is not a big problem. Second, the computational power of computer is increasing day-by-day which makes Dynamic Programming suitable for optimal control problems. Another problem associated with DP is the use of interpolation to calculate the control and performance index. The use of interpolation can only be justified in case of continuous optimal control problems.

One of the major advantages of Dynamic Programming is that it always gives the global optima and does not trap in local minima. Researchers have used it widely for many applications including trajectory optimization for robotic manipulators. For the first time, Vukobratović and Kirćanski [95] proposed the Dynamic Programming algorithm to find the energy optimal trajectory for non-redundant robotic manipulators for a given path. Vukobratović calculated the optimal velocity distribution for a given manipulator tip trajectory.

Shin and McKay [96] formulated the dynamic programming algorithm to solve the optimal trajectory planning in terms of path parameter s. First, they converted the dynamics of the robots to a second order differential equation using the path parameter. The optimal pseudo-velocity ($\dot{s}$) is calculated using Dynamic Programming to get optimal solution of the given cost function under constraints. The main advantage of this method is the reduction in dimensionality as the dynamics of the robots are expressed in terms of the path parameter which reduces the state space from $2n$ to two dimensional space for an n-joint manipulator. This Dynamic Programming algorithm can be used for any arbitrary cost function.

Another Dynamic Programming algorithm for optimal trajectory generation for robotic manipulators was proposed by Singh and Leu [4]. Instead of reducing the dimensionality of the problem by path parameter (s), the optimization problem was solved in Joint space. This algorithm does not consider all joints in parallel but at first step consider only a single non-stationary joint as a reference on the robot. This reference joint must be non-stationary throughout the movement of robotic manipulator from starting point to ending point. By Dynamic Programming, this problem is reduced to search over the velocity of one moving manipulator link. Once a non-stationary joint is optimized, the other joints have the same time step between two discretized points.

Yen [97] approached the optimization problem in another way and proposed an Inverse Dynamic Based Dynamic Programming (IDBDP) method for optimal point-to-point trajectory planning of robotic manipulators. This Inverse Dynamic Based DP offers some advantages over conventional DP approach, i.e., it eliminates the interpolation requirement, and the requirement of integration of motion equations is also avoided. These advantages make this algorithm computationally more efficient as compared to aforementioned Dynamic Programming algorithms. Another modification in Dynamic Programming algorithm for robotic manipulators was done by Field and Stepanenko [98]. He proposed an Iterative Dynamic Programming approach to minimum energy trajectory planning. In this modified Dynamic Programming approach a series of dy-

namic programming passes over a small reconfigurable grid size. This approach has advantages of avoiding poor local minima and curse of dimensionality, and providing parallel structure to reduce computational time.

Besides these modifications, there are still some drawbacks and problems in Dynamic Programming algorithms. These problems and the proposed solutions are discussed in Section 2.4 and Section 5.2.2, respectively.

Recently, a new optimization technique based on the discrete variational mechanics of mechanical systems has been developed for optimization. This is called *Discrete Mechanics and Optimal Control* (DMOC) and was introduced by Junge et al. [99]. The theory of discrete variational mechanics has widely been used in the optimal control problems since 1960s [100, 101, 102]. In this method, Marsden and West directly discretized the variational structure of mechanical systems [103]. Ober-Blöbaum et al. [5] gave a detailed discussion over the connection between optimal control and variational mechanics. DMOC uses structure preserving approach to compute the solution for an optimal control problem. The other important aspects of DMOC like preservation of momentum maps, conservation of modified energy, and the ease in implementation are also discussed by Ober-Blöbaum [104, 5].

In DMOC, the equality constraints for a finite dimensional optimization problem are obtained by the direct discretization of the Lagrange-d'Alembert principle. The resulting finite dimensional nonlinear optimization problem can be solved by standard nonlinear optimization techniques like *Sequential Quadratic Programming* (SQP). DMOC has already been applied successfully to some interesting applications, e.g., compass gait biped [105], double pendulum on a cart [106], pitcher's arm [107], image analysis [108], and low thrust transfer and the optimal control of formation flying satellites [109, 110].

A new direction of research is to use DMOC to find the optimal control for multi-body systems with holonomic constraints. This is called *"Discrete Mechanics and Optimal Control for Constrained Systems"*(DMOCC) [111]. DMOCC refers a connection between Null-Space method and DMOC [112, 113]. As a result of combining Null-Space method and DMOC, we get reduced equations that describe a time step method. These reduced equations are used as constraints in calculating the optimal solution using DMOC by an optimization algorithm. Until now, the contact and the collision in the simulation of multi-body systems are neglected. In more recent work [114], the collision avoidance as well as the planned contact between bodies are taken into account to find the optimal control trajectory. Specific algorithms can be used to detect the contact between bodies by means of oriented hyper-surfaces.

Although DMOC has been successfully implemented for many useful application and developed in a number of ways, till now it has not used for trajectory optimization for robotic manipulators. In this thesis, we used DMOC to optimize the predefined geometrical path for robotic manipulators. We also used DMOC to solve the generalized

path optimization problem for robotic manipulators and to solve path planning and trajectory optimization simultaneously instead of dividing the problem into two steps. This is a quite interesting application of DMOC in the robotics domain because till now there is no method, to the best of author's knowledge, to solve the path planning and trajectory optimization simultaneously.

In Chapter 6, we discussed the application of DMOC for the simultaneous path planning and trajectory optimization and applicability is shown by two numerical examples.

2.3 On-line Trajectory Tracking

On-line trajectory tracking is the last step of path optimization for robotic manipulators and mostly it is counted in practical implementation section. In trajectory tracking, manipulator is guided along the planned trajectory. Usually, the on-line trajectory trackers are linear controllers and generally these trackers keep the manipulator close to the desired trajectory. Sometimes, on-line controllers incorporate the manipulator dynamics inside the control loop for accuracy which may cause the overwhelming control computers.

Computed torque control is one of the famous control schemes for robotic manipulators, as it incorporates the manipulator's dynamics which increases its accuracy and gives more insight into the manipulator's structure. It is a special application of feedback linearization of nonlinear systems. A comparison of computed torque driven methods with conventional position servo is given by Markiewicz [115]. This scheme is not only applicable for rigid manipulators but it is equally applicable for flexible link robots [116]. Recently, a lot of work has been done on using adaptive and fuzzy techniques in conjunction with computed torque control [117, 118, 119, 120, 121, 122, 123].

There is a lot of work on the use of Neural Network (NN) for feedback control of plants. In early practices, the tuning algorithms for closed-loop control were not available and standard back propagation weight tuning was used [124]. Many NN controlled techniques are combined with adaptive control approaches [125, 126], proposing NN feedback control topologies such as indirect identification-based control, inverse dynamics control, and series-parallel techniques, etc. [127].

Most of the time, industrial robots have to follow a predefined geometrical path but sometimes they are also used for grinding, assembly and some other tasks in which manipulator interact with environment. As a consequence of this interaction, an interaction force is developed between the robot manipulator and the environment and this force must be controlled. In this thesis, the main focus will be on the on-line tracking of an optimized trajectory and we will not discuss the interaction of manipulators with environment.

2.4 Scientific Gaps

In Section 2.2, we have discussed trajectory optimization methods for a given geometrical path. Although, these method are currently used in industry and many other applications, these methods have some flaws and problems. These flaws can be removed and algorithms can be improved by modifications, discussed in Chapter 5. In Chapter 4, we discuss the path planning using PRM and GA. The results obtained from these two methods are not only analysed numerically but also in term of repeatability, possibility of local minima, and flexibility to add additional constraints.

The most widely used time-optimal trajectory planning method is *Phase-Plane Method* that was proposed by Bobrow et al. [78] and Shin et al. [79]. Although, this method is computationally very efficient and can be used for any type of robots, its applications are only limited for time minimization and it cannot be used for multi-objective optimization for any arbitrary cost function. It is the major flaw in this algorithm that restricted it usability. Another problem in this method is that it is difficult to incorporate arbitrary additional constraints in the time-optimal control problem.

As already discussed, the Dynamic Programming algorithm for trajectory optimization in Joint space, proposed by Singh and Leu [4], is another useful algorithm that can handle any arbitrary strong constraints and optimize multi-objective cost function. Besides these advantages, there are two flaws in this algorithm. First, this algorithm optimizes the problem with respect to only one non-stationary joint. The constraints only on the selected non-stationary joint, which is considered as a reference, are taken into account in the optimization phase and the constraints on all other joints are ignored. This might happen that during the optimization of reference non-stationary joint any other joint may violate its constraints, because it is not necessary that all joint actuators are symmetric and have the same bounds. Second, Singh's DP algorithm does not give any rule for selecting the reference non-stationary joint in case of more than one non-stationary joints. In some cases, it happens that there are more than one non-stationary joints when following the given path. In this scenario, it is always a difficult task to choose a non-stationary joint amongst more than one non-stationary joint because choosing a non-stationary joint randomly does not guarantee the global optimal solution and may ends up with a non-optimal solution.

One of the major gaps that is still present in state-of-the-art techniques is division of the path optimization problem for robotic manipulators into stages for tractability. Unfortunately, the simplicity obtained from the division into path planning and trajectory optimization comes at the cost of efficiency. Combining the path planning and trajectory optimization would increase the overall efficiency and accuracy of the robotic manipulator as both steps will incorporate the robot's dynamical properties. In this thesis, we have proposed a novel method to perform the path planning and trajectory optimization for robotic manipulators simultaneously using *Discrete Mechanics and Optimal Control (DMOC)*. The problem formulation and the constraints for path planning and trajectory optimization are discussed in detail in Chapter 6. Two numerical

examples are also discussed to show the applicability and practicability of the proposed scheme.

3 Delta Parallel Robot

It was in early 80's when Prof. Reymond Clavel[1] and his research team at École Polytechnique Fédérale de Lausanne came up with the idea of parallel arm robot [128]. The main purpose of this new robot was to manoeuvre light weight objects at high speed. It was a great landmark in the era of the development of high speed industrial robots. In pick and place operations, it can work with an acceleration of $15G$. Its high speed and robust structure not only introduces it in packaging industry but also in surgery, medical and pharmaceutical industry. Recently, Delta parallel robots are being used in haptic devices and 3D printers.

In the research of path planning and trajectory optimization, robotic manipulators are required to verify the theoretical results and for practical implementation. Practical results not only show the applicability of the proposed techniques but also give confidence to the industrialist to use it directly in their applications. In this thesis, we will focus on the *D*4-500 Delta parallel robot from CODIAN Robotics[2], shown in Figure 3.1. The geometrical and mechanical parameters, given in Table 3.1, are used for different numerical examples of path planning and trajectory optimization discussed in this thesis.

Figure 3.1: D4-500 Delta parallel robot from CODIAN Robotics

[1] Reymond Clavel: www.people.epfl.ch/reymond.clavel

[2] Codian Robotics B.V.: www.codian-robotics.com

Table 3.1: Dynamic coefficients and actuator characteristics of DELTA parallel D4-500 robot

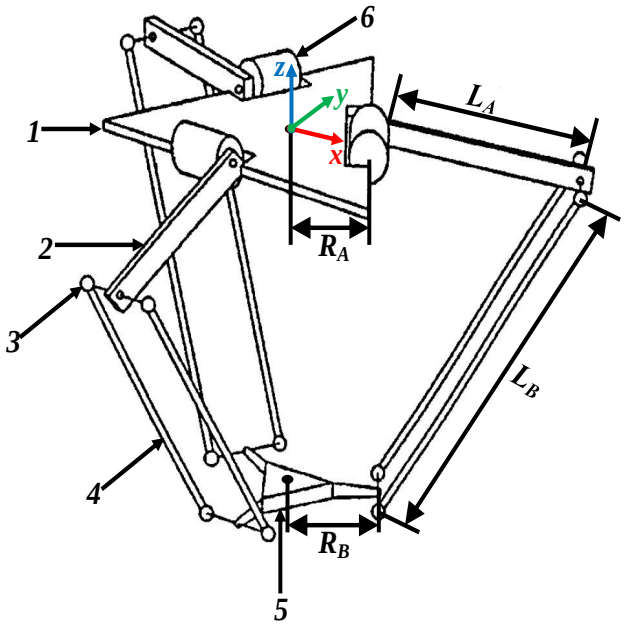
Parameter	Description	Value
L_A	Length of arm	0.15 m
L_B	Length of forearm	0.4 m
R_A	Distance from the center of base to the motor joint	0.1 m
R_B	Distance from the center of travelling plate to the joint	0.04 m
τ_{min}	Minimum joint torque	-35.2 Nm
τ_{max}	Maximum joint torque	35.2 Nm
m_a	Mass of the arm	0.14 Kg
m_b	Mass of the forearm (single rod)	0.124 Kg
m_c	Mass of travelling plate	0.222 Kg
m_{elbow}	Mass of the elbow	0.042 Kg
I_m	Inertia of the motor	3.96×10^{-5} Kg.m^2
$X_{workspace}$	Workspace in x-direction	-110.74mm to 110.74mm
$Y_{workspace}$	Workspace in y-direction	-110.74mm to 110.74mm
$Z_{workspace}$	Workspace in z-direction	-505.4mm to $-$ 283.9mm

3.1 Delta Parallel Robot Structure

The Delta parallel robot is a parallel robot, i.e. multiple kinematic chains are connected from base to end effector. The general structure of Delta parallel robot is shown in Figure 3.2. The key concept behind the Delta parallel robot design is the use of parallelograms in structure. The parallelogram structure keeps the moving platform (labeled as 5) always parallel to the fix base (labeled as 1) and provides only translational motion for moving platform in x-, y-, and z-direction.

In Delta parallel robot, the actuators (labeled as 6) are mounted to the fixed base. The input links (labeled as 2) of three parallelograms are connected to the joint actuators by revolute joints (labeled as 3). The other end of these parallelograms are connected to a small triangular moving platform that can move in $3D$. Normally, Delta parallel robot is a direct drive robot and it requires simpler control mechanism as compared to the indirect driven robots. The power from motor to the moving platform is transmitted through rigid rods instead of gearbox or pulley that makes this mechanism more accurate and less noisy.

As all the actuators are connected to the fixed base, arms can be made of a light weight material so that these arms do not have to carry extra weight of actuators. This light weight travelling mechanism gives the advantage of high speed to Delta parallel robots



1	Fix base
2	Input link
3	Revolute joint
4	Forearm
5	Travelling plate
6	Actuator

Figure 3.2: General schematic of Delta parallel robot [3]

and they can achieve high accelerations (50G in experimental environments and 15G in industrial operations).

3.2 Kinematics of Delta Parallel Robot

In this section, we will develop the relationship between Joint coordinates and Cartesian coordinates based on the knowledge of robot links and movement of joints. While developing the kinematic equations, it is assumed that all links of robot's structure are rigid bodies and joints provide either pure translational or pure rotational movement. Kinematics of robotic manipulators plays a vital role in path planning. In terms of complexity, kinematics of parallel robots is more complex as compared to serial robots because of the analytical difficulty presented by the joint variable interdependencies and the complex relation between Joint coordinates and Cartesian coordinates. Details of different points on Delta parallel robot used for forward and inverse kinematics are shown in Figure 3.3. The definition of the vectors used for the formulation of forward and inverse kinematics of Delta parallel robot is shown in Figure 3.4.

3.2.1 Inverse Kinematics

In inverse kinematics, the joint parameters of robotic manipulator are calculated from the given position of end-effector in Cartesian space. Before going to start the calculations for inverse kinematics, it is better to explain some terms that will be used extensively in kinematics formulation. These terms are defined in Figure 3.5(a) and Figure 3.5(b).

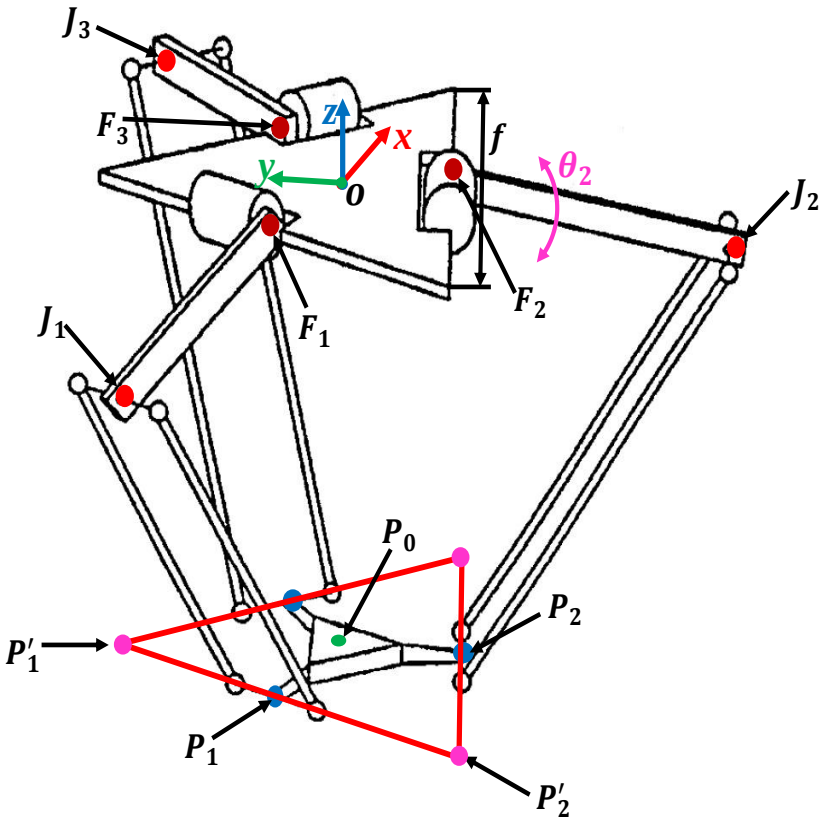


Figure 3.3: Description of different points used for forward and inverse kinematics

In Figure 3.5(a), f is length of one side of the equilateral triangle that inscribed the circle formed by the three actuators points, P_0 is the position of the end-effector in Cartesian space with respect to the coordinate system that has origin (O) at the center point of the fixed base. The input link of Delta parallel robot connects to the forearm at point J_i, $i \in \{1, 2, 3\}$. The index i is used to identify the arm number. In Figure 3.5(b), e is the length of one side of the equilateral triangle that inscribes the circle formed by points P_1, P_2 and P_3 at bottom.

To solve the inverse kinematics, two circles are drawn that intersect at two points as shown in Figure 3.5(a) and Figure 3.6. One circle is drawn with center F_2 and radius L_A. The second circle is drawn with center at point P_2' and radius $|\vec{\xi}_{P_2'J_2}|$ that is equal to $\sqrt{L_B^2 - P_{0_x}^2}$. The calculation of this radius is shown below:

$$\vec{P_0} = [P_{0_x}, P_{0_y}, P_{0_z}]^T$$

$$|\vec{\xi}_{P_0P_2}| = \frac{e}{2}\tan(30^o) = \frac{e}{2\sqrt{3}}$$

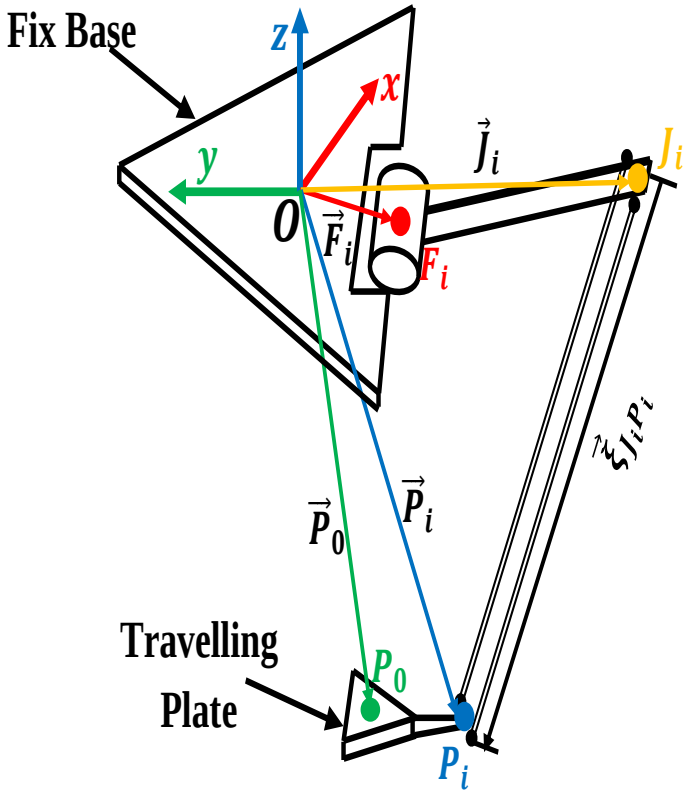


Figure 3.4: Definition of different vectors used for the forward and inverse kinematics of Delta parallel robot

$$\vec{P_2} = \left[P_{0_x}, P_{0_y} - \frac{e}{2\sqrt{3}}, P_{0_z}\right]^T$$

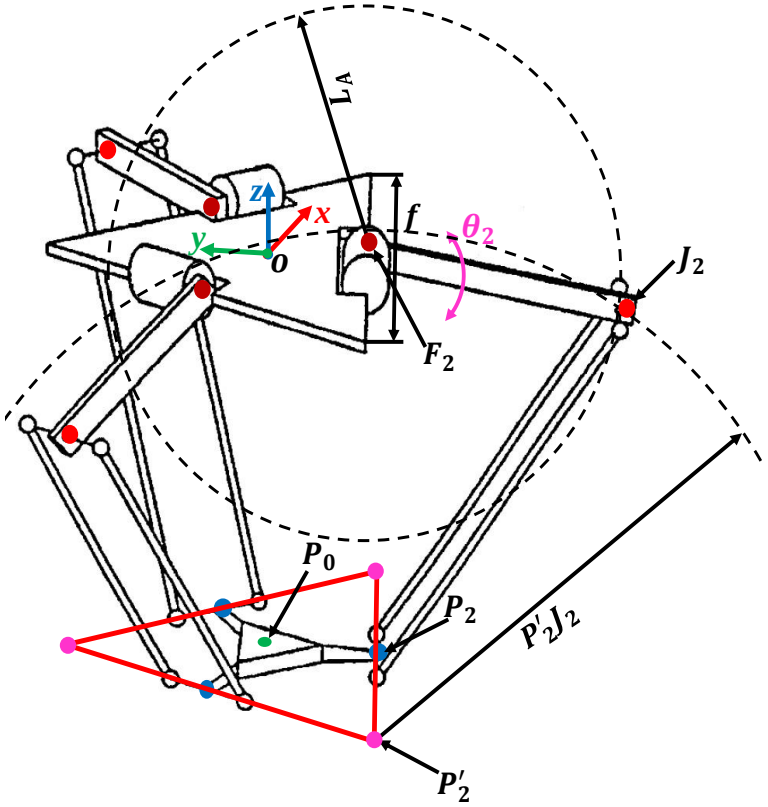
$$\vec{P_2'} = \left[0, P_{0_y} - \frac{e}{2\sqrt{3}}, P_{0_z}\right]^T$$

$$|\vec{\xi}_{P_2'J_2}| = \sqrt{|\vec{\xi}_{P_2J_2}|^2 - |\vec{\xi}_{P_2P_2'}|^2} = \sqrt{L_B^2 - P_{0_x}^2} \tag{3.1}$$

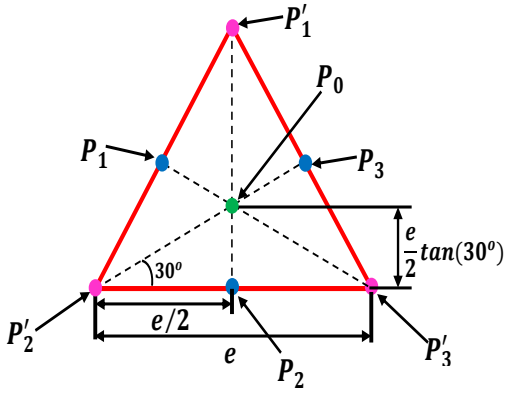
As one can see that these two circles will intersect at two points and we are interested in the point of intersection which has smaller value of y-coordinate. By selecting P_2 point, the reference frame has been reduced only to YZ-plane and link F_2J_2 can only move in this plane. From Figure 3.6, the coordinates of point $\vec{F_2}$ are $\left[0, -f/(2\sqrt{3}), 0\right]^T$.

Equation (3.2) and (3.3) represent the circles with radius L_A and $(L_B^2 - P_{0_x}^2)$ respectively, as shown in Figure 3.6.

$$(J_{2_y} - F_{2_y})^2 + (J_{2_z} - F_{2_z})^2 = L_A^2 \tag{3.2}$$



(a) Geometrical description of different terms for inverse kinematics



(b) Close view of the bottom of Delta parallel robot showing end-effector

Figure 3.5: Geometrical description of different terms, center points and radius of circles and close view of end-effector.

$$\Rightarrow \left(J_{2_y} + \frac{f}{2\sqrt{3}}\right)^2 + J_{2_z}^2 = L_A^2$$

$$(J_{2_y} - P'_{2_y})^2 + (J_{2_z} - P'_{2_z})^2 = L_B^2 - P_{0_x}^2 \tag{3.3}$$

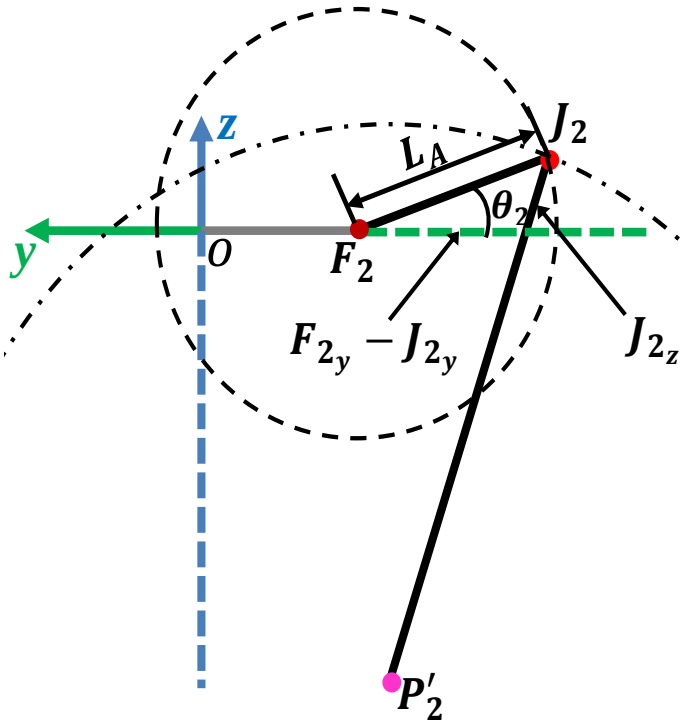


Figure 3.6: Side view of Figure 3.5(a) for better understanding (YZ Plane view)

$$\Rightarrow \left(J_{2_y} - P_{0_y} + \frac{e}{2\sqrt{3}}\right)^2 + \left(J_{2_z} - P_{0_z}\right) = L_B^2 - P_{0_x}^2$$

Value of J_{2_y} and J_{2_z} can easily be calculated by simultaneously solving (3.2) and (3.3). All other variables in these two equations are either given as input or can be measured from the geometrical structure of Delta parallel robot. Once these values are known, simple trigonometry can be used to calculate angle θ_2 as given in (3.4).

$$\theta_2 = \arctan\left(\frac{J_{2_z}}{F_{2_y} - J_{2_y}}\right) \tag{3.4}$$

The symmetrical structure of Delta parallel robot gives the advantage to calculate the remaining angles θ_1 and θ_3 by applying the same procedure. To calculate angle θ_1 and angle θ_3 we have to rotate the reference frame by an angle of 120^o in counter-clockwise and clockwise direction, respectively. The only challenge is to calculate the new coordinates of point P_0 in x and y-direction which can easily be calculated using corresponding rotation matrix.

3.2.2 Forward Kinematics

In forward kinematics, the position of end-effector of robotic manipulator is calculated from the given information of joint parameters. Unlike serial robots, the forward

kinematics of parallel robots is more complex and difficult to find as compared to the inverse kinematics.

As the angles θ_1, θ_2 and θ_3 are known in forward kinematics problem, so the coordinates of points J_1, J_2 and J_3 can be found easily. The forearms J_1P_1, J_2P_2 and J_3P_3 can freely rotate around points J_1, J_2 and J_3, respectively. In order to calculate the forward kinematics, points J_1, J_2 and J_3 are moved to J'_1, J'_2 and J'_3 using transition vector $\vec{\xi}_{P_1P_0}$, $\vec{\xi}_{P_2P_0}$ and $\vec{\xi}_{P_3P_0}$ respectively, as shown in Figure 3.7.

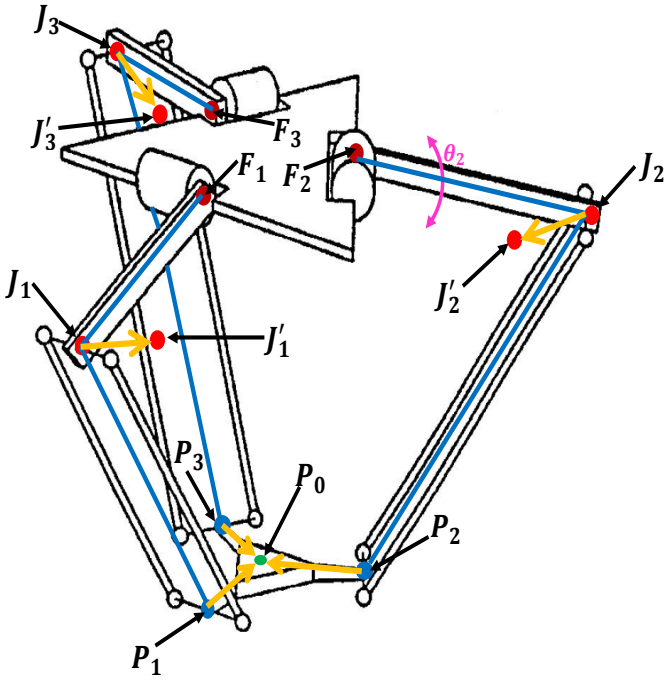


Figure 3.7: Transition of joints to calculate the forward kinematics

As a result of this transition, all three circles with radius L_B and center at J'_1, J'_2 and J'_3 will intersect at point P_0. The coordinates of point $\vec{\mathbf{P}_0} = [P_{0_x}, P_{0_y}, P_{0_z}]^T$ can be obtained by solving the three equations of circles simultaneously as given in (3.5).

$$(P_{0_x} - J'_{i_x})^2 + (P_{0_y} - J'_{i_y})^2 + (P_{0_z} - J'_{i_z})^2 = L_B^2, \quad i \in \{1, 2, 3\} \tag{3.5}$$

In Figure 3.8, the top view of Delta parallel robot is shown. From this top view, as a result of transition from J_i to J'_i different vectors can easily be calculated as given in (3.6),(3.7), and (3.8).

$$|\vec{\xi}_{OF_i}| = \frac{f}{2}\tan(30^o) = \frac{f}{2\sqrt{3}}, \quad i = 1, 2, 3 \tag{3.6}$$

$$|\vec{\xi}_{J_iJ'_i}| = \frac{e}{2}\tan(30^o) = \frac{e}{2\sqrt{3}}, \quad i = 1,2,3 \tag{3.7}$$

$$|\vec{\xi}_{F_iJ_i}| = L_A\cos(\theta_i), \quad i = 1,2,3 \tag{3.8}$$

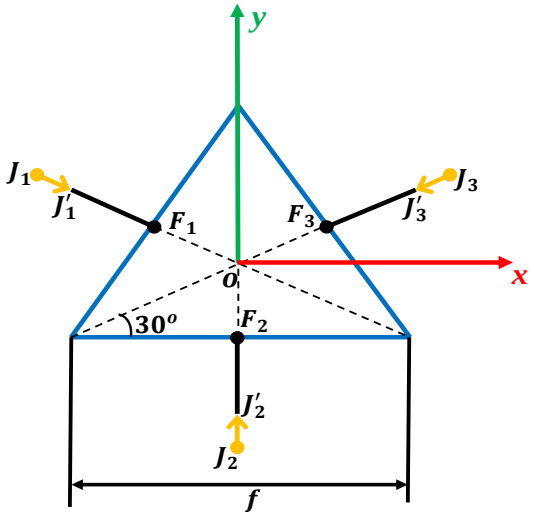


Figure 3.8: Top view of Delta parallel robot to calculate different transition vectors

According to (3.5), three circles are drawn from point J'_1, J'_2 and J'_3. The coordinates of the center of circles can be calculated using the aforementioned transition vectors. The coordinates of points J'_1, J'_2 and J'_3 are given in (3.9), (3.10), and (3.11) respectively.

$$\vec{J'_1} = \left[\left(\frac{(f-e)}{2\sqrt{3}} + L_A\cos(\theta_1)\right)\cos(30^o), \left(\frac{(f-e)}{2\sqrt{3}} + L_A\cos(\theta_1)\right)\sin(30^o), -L_A\sin(\theta_1)\right] \tag{3.9}$$

$$\vec{J'_2} = \left[0, \ -(f-e)/2\sqrt{3} - L_A\cos(\theta_2), \ -L_A\sin(\theta_2)\right] \tag{3.10}$$

$$\vec{J'_3} = \left[\left(\frac{(f-e)}{2\sqrt{3}} + L_A\cos(\theta_3)\right)\cos(30^o), \left(\frac{(f-e)}{2\sqrt{3}} + L_A\cos(\theta_3)\right)\sin(30^o), -L_A\sin(\theta_3)\right] \tag{3.11}$$

After calculating all transition vectors and coordinates of the center of circles, Equation (3.5) can be solved simultaneously to obtain the position of end-effector in three dimensions.

3.3 Jacobian Matrix

The Jacobian matrix, **J**, plays an important role in robotic manipulator's dynamic model as well as trajectory optimization. It describes the relationship between Cartesian space velocities and Joint space velocities as given in (3.12).

$$\dot{\vec{P}}_0 = \mathbf{J}\,\dot{\vec{\theta}} \tag{3.12}$$

Unlike serial robots, systematic approach cannot be used for parallel robots to find this matrix. The Jacobian matrix for Delta parallel robot was first calculated by Codourey [129]. In this method partial derivatives were computed numerically. Jacobian matrix for parallel robots can also be calculated by linking Cartesian space variable to Joint space variables by a set of constrained equations. Guglielmetti [130] was the first person who applied this approach to calculate the Jacobian matrix for Delta parallel robot. A simplified version of Guglielmetti's formulation is addressed by Codourey [131]. In this thesis, we are adopting the Codourey's method to calculate the Jacobian matrix.

In Figure 3.9, single revolute joint of Delta parallel robot and the nodes at arm and forearm are shown in order to derive the constraints equations. Length of forearm of Delta parallel robot can be used as a constrained equation to derive the Jacobian matrix as given in (3.13).

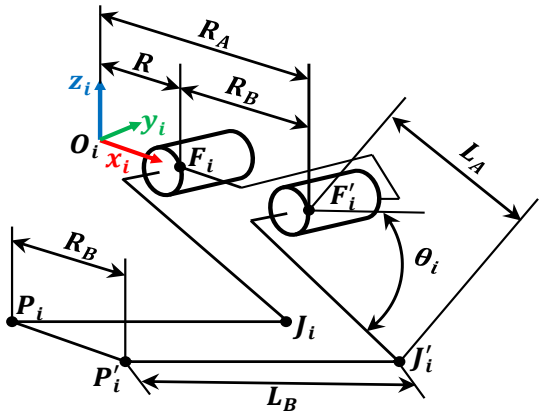


Figure 3.9: Single revolute joint of Delta parallel robot and the nodes at arm and forearm

$$\|\vec{\xi}_{J_iP_i}\|^2 - L_B^2 = 0, \quad i \in \{1,2,3\} \tag{3.13}$$

By defining term $\vec{s_i}$ as vector $\vec{\xi}_{J_iP_i}$, Equation (3.13) can be written as

$$\vec{s_i}^T \cdot \vec{s_i} - L_B^2 = 0, \quad i = 1,2,3. \tag{3.14}$$

In terms of vectors, term s_i can be described by

$$\vec{s_i} = \vec{\xi}_{J_iP_i} = \vec{P_i} - \left(\vec{F_i} + \vec{\xi}_{F_iJ_i}\right) \tag{3.15}$$

$$= \begin{bmatrix} P_{0_x} \\ P_{0_y} \\ P_{0_z} \end{bmatrix} - \mathbf{R}_i^R \left(\begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} L_A \cos(\theta_i) \\ 0 \\ -L_A \sin(\theta_i) \end{bmatrix} \right), \quad i = 1,2,3. \tag{3.16}$$

In (3.15), $[P_{0_x}, P_{0_y}, P_{0_z}]$ is the position of the end-effector $\vec{P_0}$, and R is the absolute reference frame as shown in Figure 3.9. Due to the symmetry in Delta parallel robot structure, each arm can be treated separately. Each arm is separated by an angle of 120^o degrees and the place of corresponding frame for each arm is same as R but rotated by an angle of $\varphi_i = 120^o \cdot (i-1)$, for arm $1, 2,$ and 3, respectively. The transformation matrix between frame $\mathbf{R}_i^R$ is just a rotation matrix as given in (3.17).

$$\mathbf{R}_i^R = \begin{bmatrix} \cos(\varphi_i) & -\sin(\varphi_i) & 0 \\ \sin(\varphi_i) & \cos(\varphi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

Taking the time derivative of (3.14) and simplifying by applying the commutative property will lead to (3.18).

$$\vec{s_i}^T \vec{\dot{s}}_i = 0, \quad i = 1,2,3. \tag{3.18}$$

The time derivative of term s_i is given by

$$\vec{\dot{s}_i} = \begin{bmatrix} \dot{P}_{0_x} \\ \dot{P}_{0_y} \\ \dot{P}_{0_z} \end{bmatrix} + \mathbf{R}_i^R \begin{bmatrix} L_A \sin(\theta_i) \\ 0 \\ L_A \cos(\theta_i) \end{bmatrix} \dot{\theta}_i = \vec{\dot{P}_0} + \vec{b_i}\dot{\theta}_i, \quad i = 1,2,3. \tag{3.19}$$

By rearranging (3.19) and using the definition of (3.15), the following final form is

obtained:

$$\begin{bmatrix} \vec{s_1}^T \\ \vec{s_2}^T \\ \vec{s_3}^T \end{bmatrix} \dot{\vec{P}}_0 + \begin{bmatrix} \vec{s_1}^T \vec{b}_1 & 0 & 0 \\ 0 & \vec{s_2}^T \vec{b}_2 & 0 \\ 0 & 0 & \vec{s_3}^T \vec{b}_3 \end{bmatrix} \dot{\vec{\theta}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.20}$$

In (3.20), $\dot{\vec{\theta}} = [\dot{\theta}_1,\ \dot{\theta}_2,\ \dot{\theta}_3]^T$ is the vector representing Joint space velocity. By rearranging this equation, Jacobian matrix for Delta parallel robot can be given as

$$\dot{\vec{P}}_0 = \mathbf{J}\,\dot{\vec{\theta}}$$

where

$$\mathbf{J} = -\begin{bmatrix} \vec{s_1}^T \\ \vec{s_2}^T \\ \vec{s_3}^T \end{bmatrix}^{-1} \begin{bmatrix} \vec{s_1}^T \vec{b}_1 & 0 & 0 \\ 0 & \vec{s_2}^T \vec{b}_2 & 0 \\ 0 & 0 & \vec{s_3}^T \vec{b}_3 \end{bmatrix} \tag{3.21}$$

In case of serial robots, Jacobian matrix is only a function of $\vec{\theta}$. But for parallel robots, Jacobian matrix depends on information of joint space $\vec{\theta}$ as well as the position of end-effector $\vec{P_0}$.

In this section, we discussed the forward kinematics, inverse kinematics, dynamical model and the Jacobian matrix for the Delta parallel robot. These calculations provide a base for the path planning and trajectory optimization and different numerical examples are solved using these calculations. Although, we have taken some assumptions for simplification, e.g. friction is neglected, links are considered uniform, etc., but the derived models give meaningful results and can be used for real world applications.

4 Path Planning

As described earlier, path planning algorithms are used to find a path from starting point to ending point. Different path planning algorithms are discussed in Section 2.1. Normally, path planning is performed in Cartesian space and obstacles are also defined in Cartesian space. This gives flexibility in incorporating the workspace limitations and obstacle avoidance in Cartesian space as compared to Joint space. Although path planning can also be performed in Joint space but it will increase complexity and computational time. Obstacle avoidance is not the only optimization criterion that is considered in path planning. Multiple criteria e.g., path smoothness, shortest path and safest path can also be considered according to requirement.

In this work, we will perform path planning for the Delta parallel robot in known two dimensional ($2D$) environment using Probabilistic Roadmap (PRM) and Genetic Algorithm (GA). Although Delta parallel robot can be used for 3 dimensional ($3D$) movement, its forearms put restrictions to use it only in $2D$ for path planning in the presence of obstacles. An analysis and comparison of these two methods is given by two numerical examples.

4.1 Path Planning using Probabilistic Roadmap (PRM)

Probabilistic Roadmap (PRM) is a complete path planner so that it always finds a solution or determine that none exists. Nowadays, robotic manipulators are becoming complex day by day with high number of degrees of freedom (DOF) to perform complex tasks. Increase in the number of DOFs increase the complexity in path planning. In this situation, PRM would be the best choice for path planning as it can efficiently plan the path for robots having n-degrees of freedom. PRM can be divided into two phases: a learning/construction phase and a query phase. The details of these two phases are given in the following subsections.

4.1.1 Learning Phase

The main purpose of learning phase is to explore all the allowable workspace of robotic manipulator and to place the random configurations inside the allowable workspace. The learning or construction phase of PRM consists of following steps:

Step-1: In first step, a random configuration is created in workspace of robotic manipulator. Large number of these random configurations will increase the smoothness of the path but at a cost of increase in computational time.

Step-2: In next step, it is checked that either the random configurations are in the al-

lowable workspace or inside the obstacles. If these random configurations are inside the obstacles then there configurations are neglected.

Step-3: In 3rd step, random configurations are connected to their neighboring configurations, typically either the k nearest neighbors or the neighbors that have the distance less than some predetermined distance.

Step-4: In next step of construction phase, the connections from all configurations to their nearest configurations are checked and if any of the connections passes through the obstacles then that connection is neglected.

Step-5: At the last stage, we get a complete mesh in the allowable workspace of robotic manipulator in which random configurations are connected to one another.

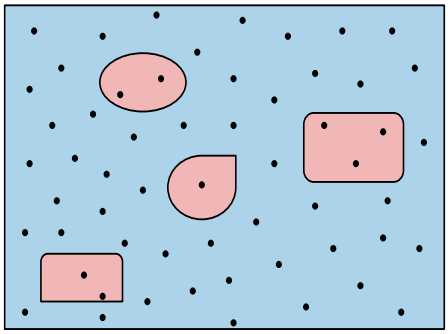
Figure 4.1 shows the different steps of learning or construction phase in PRM. The pseudo-code or algorithm for learning phase of PRM is given in Algorithm 1 (see Appendix A.2.1.1) that summarizes the construction phase described in aforementioned five steps.

4.1.2 Query Phase

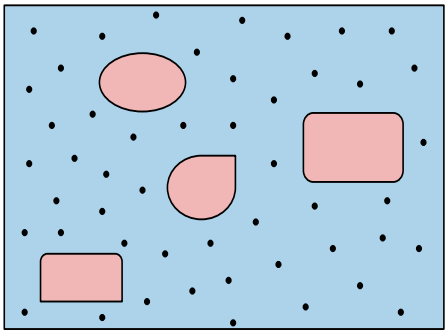
In query phase of PRM, a path from the initial position to the goal position in the allowable workspace is found using the roadmap obtained from the construction phase. In the first step of query phase, initial and goal positions are connected to the nearest configurations. In the next step, Dijkstra's algorithm [132] can be applied to find out the shortest path from initial position to goal position. A pictorial representation of query phase of PRM is shown in Figure 4.2, and the pseudo-code of Query phase is given in Algorithm 2 (see Appendix A.2.1.2).

One of the problems associated with path obtained from PRM is the sharp edges as one can see in Figure 4.2. These sharp edges are difficult to follow by robotic manipulators and sometimes at these sharp edges the torques constraints are violated because of the high transition in Joint space velocities and accelerations. These sharp edges can be removed by applying low pass filter. The order of the filter should be very low in other case the originality of the obtained path will be lost.

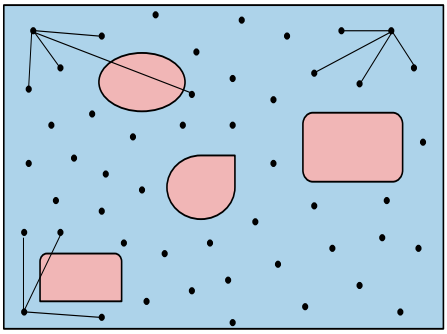
As clear from its name, PRM is a probabilistic method and one can get different results for different runs. In order to get the best results, PRM is run multiple times and then the best solution is selected. As discussed earlier, PRM is computationally quite efficient algorithm and can calculate the path in fraction of seconds. The performance of PRM is shown in Section 4.3 with the help of two examples.



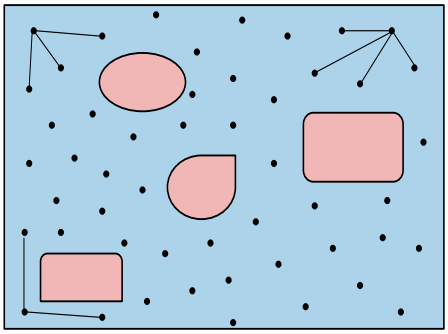
(a) Step-1: Generation of random configuration in the workspace of robotic manipulator



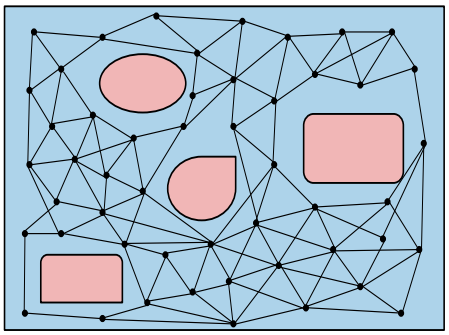
(b) Step-2: Random configurations are neglected if they are inside the obstacles



(c) Step-3: Each configuration is connected to its nearest neighborhood



(d) Step-4: Connections between two configurations are neglected if they pass through obstacles



(e) Step-5: Complete mesh of random configurations connected to one another

Figure 4.1: Learning phase of probabilistic roadmap algorithm

4.2 Path Planning using Genetic Algorithm

Genetic Algorithm (GA) is a parallel and global search technique in which population of candidate solutions, called phenotypes or individuals, are evolved towards better solution. In order to solve a problem, we search all the feasible solutions, called Search Space, and each point in Search space represents one feasible solution. Looking for a solution is then like searching for some extremes in Search Space. In this case,

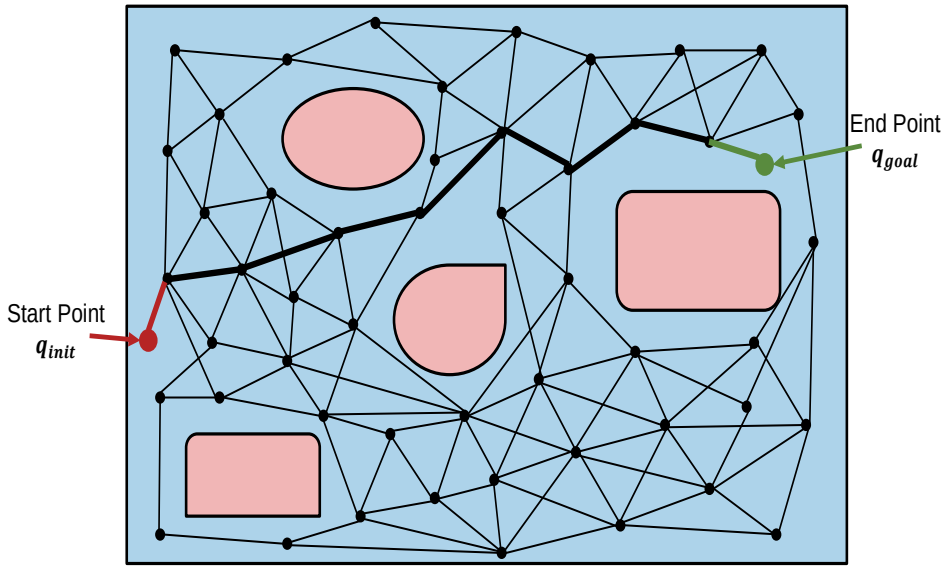


Figure 4.2: Pictorial representation of query phase of PRM. The q_{init} and q_{goal} are first connected to the roadmap and then the shortest path is found out using Dijkstra's algorithm

searching becomes very complicated as it is difficult to define the starting search point and region of interest in the Search Space. GA is one of the algorithms that is suitable for this type of search. There are few terms associated with GA that must be explained properly in order to implement for path planning in the presence of obstacles.

4.2.1 Population Initialization

Generally in GA, population is generated randomly. It might be possible that randomly generated chromosomes lie inside an obstacle which may includes infeasible paths. An alternative way to avoid this problem is to define the initial population intelligently within a specific range that can make sure that initially generated chromosomes are not lying inside the obstacles and solution generated by each chromosome is feasible. Number of iterations towards optimal solution can be reduced by starting GA with feasible initial population.

In case of using GA for path planning, the random initial population would be the random paths from starting point to ending point while avoiding the obstacles. The each path will be considered as a chromosome and the each discrete node of the path will be considered as a Gene. The random paths must be initialized intelligently so that the paths must avoid the obstacles.

4.2.2 Cost Function

The main objective in path planning is to find an optimal path from initial position to goal position in the presence of obstacles. Different optimization criteria can be defined e.g., minimum time, minimum energy, shortest distance, etc. Generally, shortest path is focused in path planning. The fitness function (f) in case of path planning using GA is given below:

$$f = \sum_{i=1}^{n-1} d(P_0[i], P_0([i+1]) \tag{4.1}$$

where

$$d(P_0[i], P_0[i+1]) = \sqrt{(P_{0_x}[i+1] - P_{0_x}[i])^2 + (P_{0_y}[i+1] - P_{0_y}[i])^2}$$

In (4.1), p_i is the i-th gene of chromosome, n is the length of the chromosome, d is the distance between two consecutive nodes, and x_i and y_i are the robot's position in x- and y-coordinates.

4.2.3 Selection Method

In GA, the main idea to get the optimal solution is that the best genes on the chromosome should be survived and new generations must evolve from it. In selection procedure, the best genes are selected from the population. The selection process consists of three steps. First, objective function value with respect to each chromosome is evaluated. Based on the objective function values, fitness values are assigned to each chromosome. There are many methods about selection of the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.

4.2.4 Crossover

In crossover, the two selected chromosomes are combined to form two offspring. If there is no crossover, offspring are exact copy of parents. One can select the different percentage of crossover between two selected chromosomes. Generally, crossover rate should be high as about 80% to 90%. An example of crossover is shown in Figure 4.3.

Chromosome 1	110101|11011001
Chromosome 2	111101|01010011
Offspring 1	110101|01010011
Offspring 2	111101|11011001

Figure 4.3: Single point crossover in chromosomes

In case of path planning using GA, the crossover would be to mix two parent paths and to generate two now offspring paths. Special conditions must be applied in the crossover so that the new offspring paths must avoid the obstacles.

4.2.4.1 Elitism

There is always high probability that after crossover and generating new offspring, parent chromosomes will be lost. Elitism is the method in which copies of the parent chromosomes or sometimes a few best chromosomes are kept and after generating the offspring, copies of the best chromosomes beside the offspring are added to new generation. Elitism increases the performance of GA as it always keeps the best solutions in each iterations.

4.2.5 Mutation

After the crossover, mutation takes place in the new offspring. In mutation, a bit randomly changes in the offspring or a random small change in the gene, depending upon the coding of chromosome. Mutation expands the search space to large regions so that all solutions do not fall into the local optimum of the problem, thus ensuring global search. Normally, mutation rate should be low and the recommended rate is about 0.5% to 1%. Mutation operation in offspring is shown in Figure 4.4.

Original Offspring 1	110**1**0111011001
Original Offspring 2	111**1**01 0101**1**0011
Mutated Offspring 1	110**0**0111011001
Mutated Offspring 2	111**1**1 0100**0**011

Figure 4.4: Mutation operation in offspring

4.2.6 Pseudo-Code for Genetic Algorithm

A pseudo-code of Genetic Algorithm for path planning is given in Algorithm 3 (see Appendix A.2.2). In this algorithm, selection, crossover and mutation depends upon the type of encoding used in GA. In this thesis, we used the decimal coding for path planning using GA.

From the pseudo-code of GA one can see that it is a metaheuristic search algorithm but it is also an iterative process. One can get different results for different runs of algorithm even for similar initial conditions; it happen due to the presence of stochastic operations, i.e., crossover and mutation. Usually multiple runs are required and then results are inferred based on statistics. We cannot say that the solution obtained by GA is global optimum so we have to compare the results obtained by GA with other techniques.

4.3 Numerical Examples

In this section, two path planning problems are solved using PRM and GA. The purpose of this section is to show the applicability of these two methods to solve path planning problem and to compare these two techniques in terms of different properties. Geometrical constraints of Delta parallel robots are considered in these examples and path planning is performed within the allowable workspace of $D4-500$ Delta parallel robot as given in Table 3.1.

In these examples, path planning for Delta parallel robot is considered only in two dimensions ($2D$). The applicability of PRM and GA is not limited to $2D$ and these techniques can also be used for path planning in $3D$.

4.3.1 Example 1

In first numerical example, path is planned from initial position to final position in the presence of a single obstacle. The initial and final positions are set to be $[P_{0_x}^{initial}, P_{0_y}^{initial}] = [-0.08, -0.08]$ and $[P_{0_x}^{final}, P_{0_y}^{final}] = [0.08, 0.08]$, respectively. The initial and final positions are chosen to be at the extreme corner positions in order to use the maximum allowable workspace of D4-500 Delta parallel robot. A rectangle with coordinates $[0, 0], [0.02, 0], [0.02, 0.02]$, and $[0, 0.02]$ is defined as obstacle.

4.3.1.1 Solution using GA

As already discussed that GA is a metaheuristic method, so in order to get the best solution, we have to run GA algorithm multiple times and the best solution is selected based on the statistics.

GA is very sensitive to parameters like size of population, rate of elitism, rate of mutation, etc., which play an important role in rate of convergence and computational time. The values of different parameters used in Example 1 are given in Table 4.1. These values are finalized after trial and error.

Statistics of different runs of GA with same parameters are shown in Table 4.2. From this table, we can see that the best solution obtained after five runs is 0.2299 m, average value is 0.2309 m, standard deviation is 9.810×10^{-4} m, and variance is 9.624×10^{-7} m. The path obtained for Example 1 using GA is shown in Figure 4.5. From this figure, one can see that the path is passing very close to the obstacle in order to minimize the distance and to give the best results.

4.3.1.2 Solution using PRM

For the comparison purpose, Example 1 is also solved by PRM. As appeared from its name, PRM is a probabilistic based method and the results are not repeatable and it is not guaranteed that the obtained solution is optimal one. For these reasons, we ran

Table 4.1: Values of Genetic algorithm's parameters used in Example 1

Parameter	**Value**
Number of iterations n	200
Size of population α	15
Rate of elitism β	0.2
Rate of mutation γ	0.015
Crossover fraction	0.75

Table 4.2: Statistics of five runs of GA to solve Example 1

No.	**Path Length**	**Average**	**Standard Deviation**	**Variance**
1	0.2300	0.2309	9.810×10^{-4}	9.624×10^{-7}
2	0.2320			
3	0.2299			
4	0.2303			
5	0.2321			

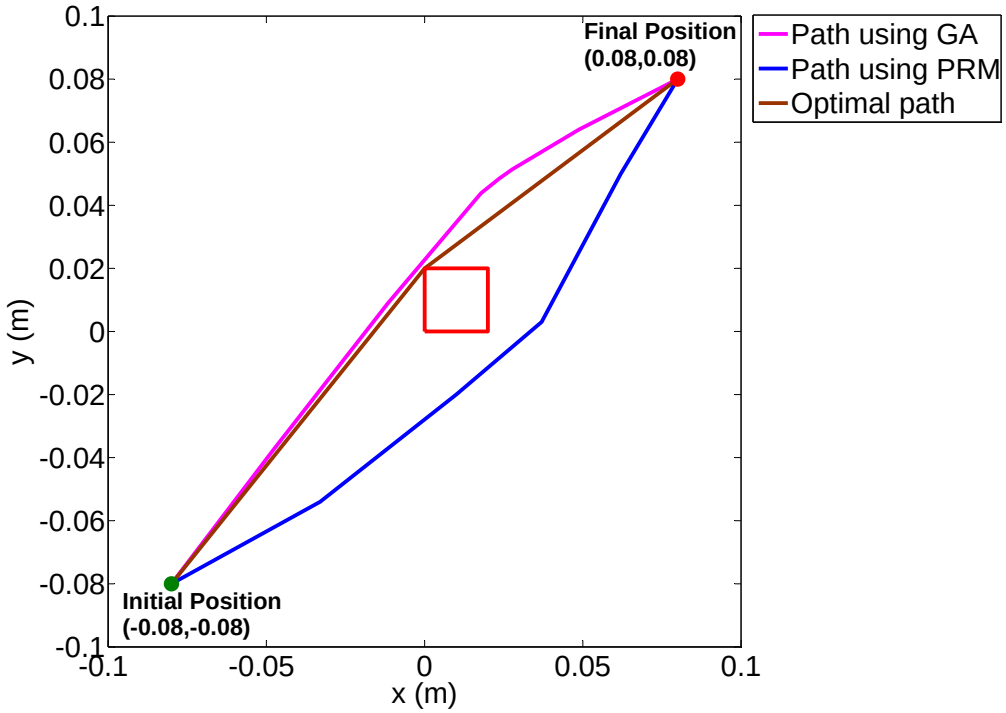


Figure 4.5: Solution obtained for Example 1 using GA and PRM

PRM multiple times to get the best solution based on statistics.

Table 4.3 shows the statistics of five different runs of PRM to solve Example 1. From this table, one can see that the best solution obtained after five runs is 0.2322 m, average value is 0.2336 m, standard deviation is 9.907×10^{-4} m, and variance is 9.816×10^{-7}m.

Table 4.3: Statistics of five runs of PRM to solve Example 1

No.	Path Length	Average	Standard Deviation	Variance
1	0.2350	0.2336	9.907×10^{-4}	9.816×10^{-7}
2	0.2322			
3	0.2343			
4	0.2329			
5	0.2337			

The path obtained for Example 1 using PRM is shown in Figure 4.5. From this figure and statistical analysis, it is clear that the solution obtained from GA is better than PRM as GA searchers all over the workspace. A comparison between GA and PRM in terms of other properties is given in Section 4.4.

4.3.2 Example 2

In Example 2, the practicability of PRM and GA in the presence of multiple known obstacles is discussed. In this example, the initial and final positions are set to be $[P_{0_x}^{initial}, P_{0_y}^{initial}] = [-0.08, -0.08]$ and $[P_{0_x}^{final}, P_{0_y}^{final}] = [0.08, 0.08]$, respectively. Two obstacles of different shapes are placed in the Delta parallel robot's workspace. One obstacle is rectangle with coordinates $[0.02, 0]$, $[0.04, 0]$, $[0.04, 0.02]$, and $[0.02, 0.02]$ and second obstacle is a circle with center at $[-0.04, 0]$ m and with a radius of 0.02 m.

4.3.2.1 Solution using GA

Just like Example 1, we ran GA multiple times to get the best solution because of its metaheuristic and irreproducible property. We ran the GA algorithm five times and based on the statistical analysis of these multiple runs we select the best solution. The value of different parameters used in GA to solve Example 2 are given in Table 4.4.

Statistics of the results obtained from different runs of GA with same parameters are shown in Table 4.5. From these statistics, one can see that the obtained solution is always different in each run. The best solution obtained after five runs is 0.2346 m. The average value, standard deviation and variance of these five runs are 0.2355 m, 7.5525×10^{-4} m, and 5.704×10^{-7} m, respectively. Figure 4.6 shows the path obtained

Table 4.4: Values of Genetic algorithm's parameters used in Example 2

Parameter	**Value**
Number of iterations n	50
Size of population α	25
Rate of elitism β	0.16
Rate of mutation γ	0.015
Crossover fraction	0.75

for Example 2 using GA. From this figure, one can get confused that path is touching the obstacle and not fulfilling the basic constraint of path planning. In actual, the path is passing through very close to the obstacle in order to get the minimum path length.

Table 4.5: Statistics of five runs of GA to solve Example 2

No.	**Path Length**	**Average**	**Standard Deviation**	**Variance**
1	0.2367	0.2355	7.5525×10^{-4}	5.704×10^{-7}
2	0.2355			
3	0.2349			
4	0.2346			
5	0.2360			

4.3.2.2 Solution using PRM

Just like Example 1, Example 2 is also solved by PRM to compare the obtained results with the results obtained using GA. We ran the PRM algorithm five times to get the best solution based on the statistics. The statistics of these five runs are given in Table 4.6.

According to statistics shown in Table 4.6, the best solution is 0.2546 m and the average value, standard deviation and variance of these five runs are 0.2563 m, 1.30×10^{-3} m, and 1.732×10^{-6} m, respectively. The path shown in Figure 4.6 corresponds to the best solution obtained using PRM, given in Table 4.6.

By comparing the statistics of five runs and the path obtained by GA and PRM, one can easily analyse that in case of GA not only the path length is smaller but also the covariance and standard deviation is smaller as compared to PRM. This proves the better performance of GA over PRM. In next section some other properties of PRM and GA, e.g. parameter dependencies, possibility of local minima, computational

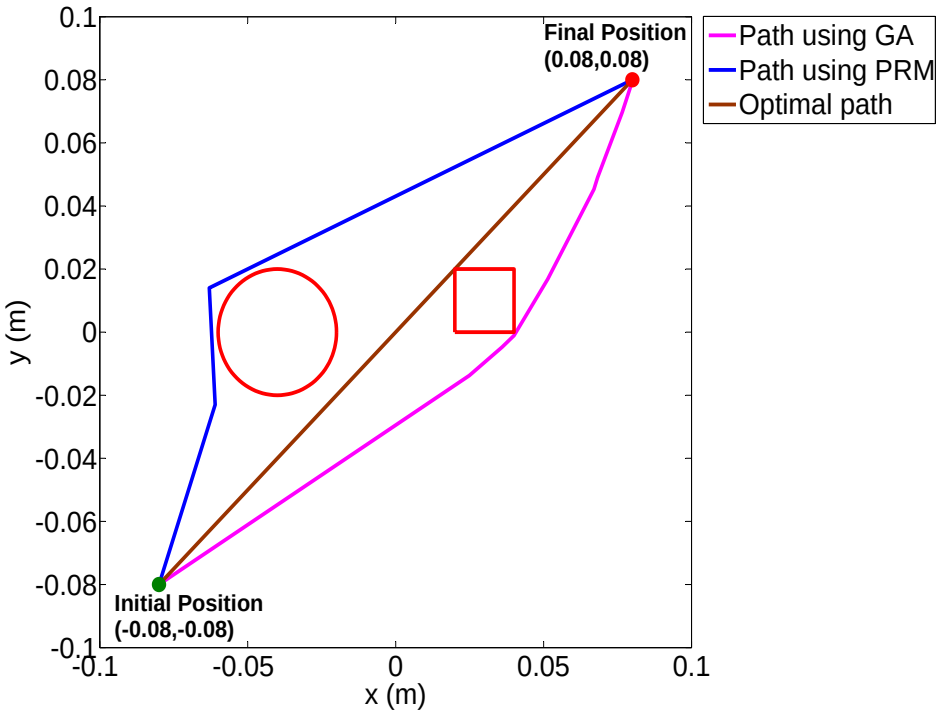


Figure 4.6: Solution obtained for Example 2 using GA and PRM

Table 4.6: Statistics of five runs of PRM to solve Example 2

No.	Path Length	Average	Standard Deviation	Variance
1	0.2581	0.2563	1.30×10^{-3}	1.732×10^{-6}
2	0.2553			
3	0.2575			
4	0.2560			
5	0.2546			

efficiency, possibility of additional constraints, etc., are discussed.

4.4 Comparison between GA and PRM

In numerical example section, we have compared the numerical results obtained by PRM and GA. Based on the obtained geometrical path and statistical analysis one can see that GA outperforms PRM. Besides these obtained geometrical path and statistical analysis, some other factors e.g., sampling, possibility of local minima, computational efficiency, etc., must be considered as well.

As already discussed in the Section 4.1, PRM is a complete planner. This property of

PRM makes it computational expensive as compared to GA, because it has to explore all the allowable workspace and calculate the possible connections from each configuration to its neighborhood configurations. The other possible reason which makes PRM computationally expensive is the implementation of Dijkstra's algorithm after calculating the all possible configurations in workspace. Number of random configurations in allowable workspace also plays an important role in computational efficiency of PRM. As higher will be the number of random configurations the more time it will take.

On the other hand, GA is computationally efficient as compared to PRM. The only computationally expensive step in GA is to calculate the cost function for all chromosomes. Luckily for path planning using GA, the objective function is the distance from initial position to final position which is not a complex function. This makes GA computationally efficient compare to PRM. Another advantage of GA over PRM is that arbitrary constraints can easily be incorporated in path planning. For example, by applying the angle conditions between three consecutive points we can remove the sharp edges and we can also set the minimum/maximum distance of the path from the known obstacles.

If we discuss the possibility of local minima in these two algorithms, we are on a safe side in PRM algorithm as it explores the complete allowable workspace. On the other hand, there are chances of finding local minima in GA. But the possibility of local minima is very small as GA is a metaheuristic process, so its random values can be anywhere in the allowable workspace. But it is not guaranteed that the whole workspace will be explored. Mutation is another step that reduces the possibility to stuck in local minima by changing the values randomly.

In this chapter, we discussed the path planning for robotic manipulators using GA and PRM. One can see that both methods have some advantages and disadvantages. Although the numerical analysis shows the superior performance of GA but PRM has advantage of complete planner. One can use any of the methods according to its application. If the working environment is not so complex and there are few obstacles then GA is recommended for path planning. But for robotic manipulators with high Degrees of Freedom (DOFs) or for complex working environment, PRM is recommended for path planning.

5 Trajectory Optimization

Trajectory optimization is the most important and trickiest step of path optimization problem for robotic manipulators. In trajectory optimization, position of each joint is calculated as a function of time within the constraints on the joints' torques and angular velocities in parallel with optimizing the given objective function. Normally, trajectory optimization is performed in Joint space. For this purpose, the given geometrical path in Cartesian space is transformed to Joint space using inverse kinematics. In this chapter, configuration vector is used for problem formulation and optimal solution calculations for trajectory optimization problem. As trajectory optimization is preferred to solve in Joint space, so configuration vector will consist of joint angles, i.e. $\vec{q} = [\theta_1\ \theta_2\ \theta_3]$.

Section 2.2 discussed the state-of-the-art trajectory optimization techniques for robotic manipulators. In this chapter, the following three techniques are discussed in details:

1) Phase-Plane Method

2) Dynamic Programming (DP)

3) Discrete Mechanics and Optimal Control (DMOC)

Besides the explanation and implementation of above mentioned trajectory optimization techniques for robotic manipulators, limitations, flaws in these algorithms, comparisons of these algorithms and the modifications to remove the flaws are also discussed.

5.1 Trajectory Optimization using Phase-Plane Method

Phase-Plane method is one of the famous and widely used trajectory optimization technique for robotic manipulators. This method is not only applicable for industrial robotic manipulators but equally applicable for all other types of robots. This method was proposed by Bobrow et al. [78] and Shin et al. [79] at the same time.

This is an indirect method which uses a phase plane plot to get the optimal solution. In this method, dimensionality of the problem is reduced by converting dynamical equations to a set of second order differential equations using path parameter, defined s from here on. Phase-Plane method can be divided into three sections. First step is the transformation of Joint space to path parameter. Second step is the problem formulation for trajectory optimization. In third step, Phase-Plane algorithm is used to find the optimal solution. In next sections, these three steps are discussed in details.

5.1.1 Transformation of Joint Space to Path Parameter

It is preferred to perform trajectory optimization in Joint space as it reduces the computational time and it is easy to satisfy the constraints of joint actuators. In robotic manipulators, Joint space consists of $n \times 1$ vector, where n represents the numbers of joint actuators. For large value of n it is difficult to handle the problem as well as it is computationally expensive. Dimension reduction is important feature of Phase-Plane method, which reduces the problem to two dimensions ($2D$). The steps involved in mapping, dimensionality reduction and mapping of torque constraints to acceleration bounds are given below:

Step-1: The first step in Phase-Plane method is to reduce the dimensionality. For this purpose, the vector of Joint coordinates is represented as a function of path parameter, denoted as s. This path parameter must be monotonically increasing and continuously differentiable.

$$\vec{q} = \vec{f}(s) \tag{5.1}$$

In (5.1), $\vec{f}$ is a $n \times 1$ vector function, where n represents the number of joint actuators and in case of Delta parallel robot n is equals to 3. There are a lot of options for path parameter, s. In this thesis, we are using arc length as path parameter because it is easy to calculate and it is always differentiable and monotonically increasing.

Step-2: In next step, the dynamical equation of robotic manipulator is mapped from Joint space to new path parameter function space. For this purpose, we took the first and second derivatives of (5.1).

$$\dot{\vec{q}} = \vec{f'}(s)\dot{s} \tag{5.2}$$

$$\ddot{\vec{q}} = \vec{f''}(s)\dot{s}^2 + \vec{f'}(s)\ddot{s} \tag{5.3}$$

In (5.3), prime denotes the derivative with respect to s and dot denotes the time derivative. Once we have the derivatives of joint angles in terms of path parameter function, equation representing the dynamical model of robotic manipulator in terms of Joint space, Equation (A.15), can be converted in terms of path parameters s. This transformation is given below:

$$\vec{\tau} = \mathbf{M}(\vec{q})\ddot{\vec{q}} + \dot{\vec{q}}^T\mathbf{C}(\vec{q})\dot{\vec{q}} + \vec{G}(\vec{q}) \tag{5.4}$$

$$= \mathbf{M}(f)\left\{\vec{f''}\dot{s}^2 + \vec{f'}(s)\ddot{s}\right\} + \left\{\vec{f'}(s)\dot{s}\right\}^T\mathbf{C}(f)\left\{\vec{f'}(s)\dot{s}\right\} + \vec{G}(f) \tag{5.5}$$

Equation (5.6) can be written as:

$$\vec{\tau} = \vec{m}(s)\ddot{s} + \vec{c}(s)\dot{s}^2 + \vec{g}(s) \tag{5.6}$$

where $\vec{m}(s)$, $\vec{c}(s)$ and $\vec{g}(s)$ are $n \times 1$ vectors given by

$$\vec{m}(s) = \mathbf{M}(f)\vec{f'}$$
$$\vec{c}(s) = \vec{f}'^{T}\mathbf{C}(f)\vec{f'} + \mathbf{M}(f)\vec{f''}$$
$$\vec{g}(s) = \vec{G}(f)$$

Step-3: In trajectory optimization problem for robotic manipulators, the constraints are given in terms of joints torques. In Phase-Plane method these constraints on joints' torques are transformed to acceleration bounds along the path.

We can find out the upper and lower bounds on Joint torques from the data sheet of joint actuators and they can be describe as

$$\tau_{i,min} \leq \tau_i \leq \tau_{i,max},\ i = 1, 2, \ldots, n. \tag{5.7}$$

In (5.7), i represents the number of actuators. In case of Delta parallel robot $i = 1, 2, 3$. We can write (5.6) in terms of torque constraints as follows:

$$\tau_{i,min} \leq m_i(s)\ddot{s} + c_i(s)\dot{s}^2 + g_i(s) \leq \tau_{i,max} \tag{5.8}$$

In (5.4), $\mathbf{M}$ was positive definite matrix and it is also assumed that path is regular, i.e. $\vec{f'}(s)$ is nonzero. Based on these two assumptions, the projected inertia vector $\vec{m}(s)$ is therefore nonzero. Therefore, the corresponding acceleration inequality can be written as:

$$L_i(s, \dot{s}) \leq \ddot{s} \leq U_i(s, \dot{s}),\ i = 1, 2, 3. \tag{5.9}$$

where

$$L_i(s, \dot{s}) = \begin{cases} \left(\tau_{i,min} - c_i(s)\dot{s}^2 - g_i(s)\right)/m_i, & \text{if } m_i > 0, \\ \left(\tau_{i,max} - c_i(s)\dot{s}^2 - g_i(s)\right)/m_i, & \text{if } m_i < 0. \end{cases} \tag{5.10}$$

$$U_i(s,\dot{s}) = \begin{cases} \left(\tau_{i,max} - c_i(s)\dot{s}^2 - g_i(s)\right)/m_i, & \text{if } m_i > 0, \\ \left(\tau_{i,min} - c_i(s)\dot{s}^2 - g_i(s)\right)/m_i, & \text{if } m_i < 0. \end{cases} \tag{5.11}$$

The upper and lower limits of acceleration for robotic manipulators are defined in (5.9) and they must be satisfied in order to fulfill the constraints.

5.1.2 Problem Formulation

In robotic manipulators, the most commonly used objective function is the time required to manoeuvre from starting position to ending position. The objective function for time minimization problem is defined as follows:

$$J = \int_{t_0}^{t_f} dt \tag{5.12}$$

We can use the basic definition of $\dot{s}$ to observe that

$$\dot{s} = \frac{ds}{dt} \Rightarrow dt = \frac{ds}{\dot{s}} \tag{5.13}$$

Finally, the cost function for time minimization can also be written is terms of path parameters given in (5.14).

$$J = \int_{s_0}^{s_f} \frac{ds}{\dot{s}} \tag{5.14}$$

In (5.12) the final time was not fixed to time minimization problem but after converting the objective function in path parameter the upper limit is known and it is the length of the given geometrical path. Equation (5.9) defines intervals of admissible accelerations. If the intersections of these intervals are nonempty, it defines a set of admissible accelerations. The set is defined by:

$$\mathcal{L}(s,\dot{s}) = \max_i L_i(s,\dot{s}) \tag{5.15}$$

$$\mathcal{U}(s,\dot{s}) = \min_i U_i(s,\dot{s}) \tag{5.16}$$

Unfortunately, it is the limitation of Phase-Plane method that it can only handle time minimization problem and it cannot handle any other objective function. It is not difficult to prove that the solution of above mentioned problem must be bang-bang in terms of input variable [133]. Now the constraints have been transformed from joints' torques $\vec{\tau}$ to path parameter acceleration $\ddot{s}$. In this case, the problem of finding optimal control reduces to finding the switching points.

5.1.3 Phase-Plane Method

In Phase-Plane method, we have to find the optimal acceleration to minimize the objective function while satisfying the constraints i.e., trajectory must lie in the admissible set where $\mathcal{L}(s, \dot{s}) \leq \mathcal{U}(s, \dot{s})$. The boundary of these two sets is called Velocity Limit Curve (VLC) which is defined by equation (5.17). According to (5.17), VLC is comprises of pseudo-velocities at each discretized point which makes $\mathcal{L}(s, \dot{s})$ equals to $\mathcal{U}(s, \dot{s})$.

$$\mathcal{L}(s, \dot{s}) = \mathcal{U}(s, \dot{s}) \tag{5.17}$$

5.1.3.1 Case-1 (Single Switching Point)

First we consider the simple case in which there is only one switching point, as shown in Figure 5.1. Switching point is the point where robot switches its movement between acceleration and deceleration.

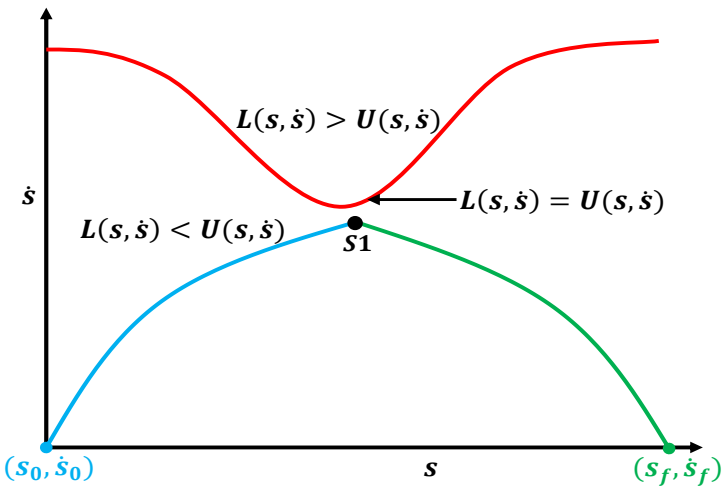


Figure 5.1: Velocity Limit Curve (red line), forward integration (blue line) and backward integration (green line). Switching point occurs at S1 when trajectory switches from acceleration to deceleration.

At first, robot starts moving with the maximum acceleration on predefined geometrical path and then switches to the maximum deceleration. So, first we should integrate the maximum acceleration as given in (5.18). The initial point of this integration on phase plane is $(s_0, \dot{s}_0)$.

$$\ddot{s} = \mathcal{U}(s, \dot{s}) \tag{5.18}$$

In parallel, we integrate the maximum deceleration, given in (5.19), with starting point $(s_f, \dot{s}_f)$.

$$\ddot{s} = \mathcal{L}(s, \dot{s}) \tag{5.19}$$

In simple cases, the forward and backward integration curves do not intersect VLC and intersect each other at switching point, point $S1$ in Figure 5.1. Normally, single switching point occurs only in case of simple movement of robotic manipulators. But for complex movements, there are always more than one switching points, discussed in Case-2.

5.1.3.2 Case-2 (Multiple Switching Points)

In most of the cases, the forward and backward integration curves hit the VLC and in this case we have more than one switching points.

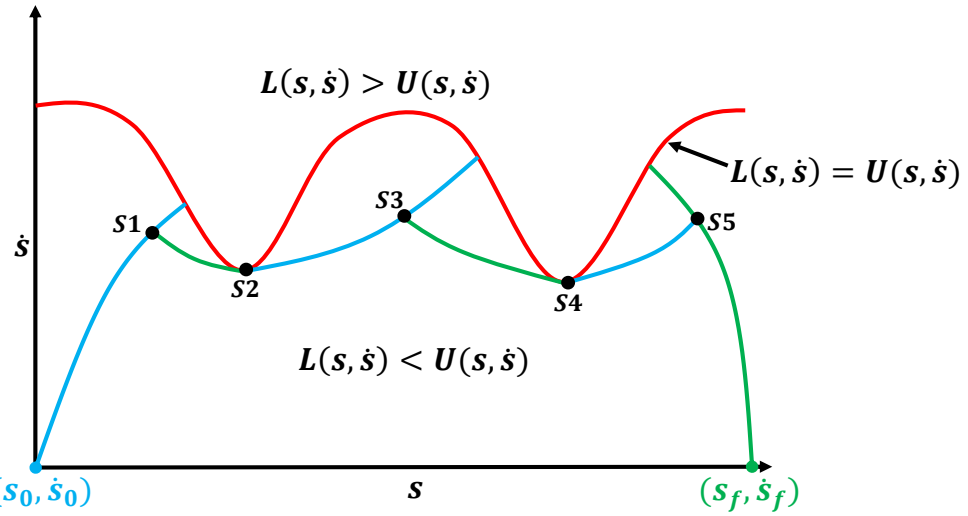


Figure 5.2: Velocity Limit Curve (red line), forward integration (blue line) and backward integration (green line). Switching points occur at S1, S2, S3, S4 and S5 when trajectory switches from acceleration/deceleration to deceleration/acceleration.

The intersection of forward and backward integration curves with VLC is shown in Figure 5.2. Once the integration curves hit the VLC, we have to follow the following

steps in order to get the optimal solution.

Step-1: Consider the VLC between the intersection points of forward and backward integration curves from starting and ending points and find the minima in this segment of VLC (Points $S2$ and $S4$ in Figure 5.2). This minima can be found by gradient method or any other method.

Step-2: From point $S2$, forward integrate $\mathcal{U}(s, \dot{s})$ and backward integrate $\mathcal{L}(s, \dot{s})$ so that these new integration curves intersect either the VLC or the previously obtained integration curves.

Step-3: Repeat Step-1 and Step-2, if the new integration curves again hit the VLC, until we have a continuous curve from starting point $(s_0, \dot{s}_0)$ to ending point $(s_f, \dot{s}_f)$.

The optimal trajectory obtained by Phase-Plane method for the interval $[s_0,\ s_f]$ fulfils the Bellman's optimality principle and the optimal trajectory for subintervals can easily be calculated by integrating (5.18) and (5.19) until the integration curves hit the optimal trajectory. Because of the multiple sections from starting point to ending point, the obtained optimal trajectory is also called the switching curve.

In Section 5.4, the Phase-Plane method is implemented on different geometrical paths using Delta parallel robot and the results are compared to other state-of-the-art techniques.

5.2 Trajectory Optimization using Dynamic Programming

Dynamic Programming (DP) is a general optimization technique used for solving the optimization problems for linear, nonlinear and complex systems. Although DP has the disadvantages of high memory requirement and high computational cost, the advantages of closed-loop solution and finding the global optima supersede these disadvantages.

A detailed review over the use of DP for the trajectory optimization of robotic manipulators is already given in Section 2.2. In this section, DP algorithms for the trajectory optimization problem and the flaws in these algorithms are outlined. One can optimize the predefined geometrical trajectory by two DP algorithms:

1) By using the path parameter and reducing the dimensionality of the problem

2) By solving the problem in Joint space

Details of these two methods are given in next subsection.

5.2.1 Dynamic Programming using Path Parameter

Trajectory optimization of robotic manipulators using DP by reducing the dimensionality of the problem was proposed by Shin and McKay [96]. This algorithm uses the

same method for dimensionality reduction that we have already explained in Section 5.1.1 and uses the dynamical model of robot in terms of path parameter, as given in (5.6).

$$\vec{\tau} = \vec{m}(s)\ddot{s} + \vec{c}(s)\dot{s}^2 + \vec{g}(s)$$

From this equation, one can analyze that the problem has been reduced to two dimensions ($2D$). There are only two variables, path parameter (s) and pseudo-velocity ($\dot{s}$). Path parameter (s) is an independent variable and can be calculated from the information of given geometrical path. On the other hand, pseudo-velocity ($\dot{s}$) is the dependent variable which has to be calculated as a function of path parameter (s) and the optimal value of pseudo-velocity ($\dot{s}$) for the interval $[s_0, s_f]$ is the only parameter that have to be determined by DP.

The initial and the final conditions for the DP can be found easily from the given path information. These terminal conditions are shown in (5.20) and (5.21). In these terminals conditions we have imposed zero angular velocity to actuator joints so that manipulator would be stationary at the start and end points. Someone can has different velocity profile of robotic manipulator at starting and ending points and can incorporate such terminal velocities in (5.20) and (5.21). These are the equality constraints or the terminal conditions for the optimization problem.

$$s(t_0) = s_0, \quad s(t_f) = s_f \tag{5.20}$$

$$\dot{s}(t_0) = 0, \quad \dot{s}(t_f) = 0 \tag{5.21}$$

The most important constraints for robotic manipulators trajectory optimization problem describe the limitation on actuators' torque. Some additional constraints can also be imposed on the joints' velocities as well. The constraints on the actuators' torques and the joints' velocities are given in (5.22) and (5.23), respectively. The upper and lower limits on the actuators' joint velocities and torques can be found out from the actuator's data sheet. Additional constraints can also be imposed like on joints' accelerations, etc.

$$\tau_{i,min} \leq \tau_i \leq \tau_{i,max} \quad i = 1, 2, \ldots, n \tag{5.22}$$

$$\dot{q}_{i,min} \leq \dot{q}_i \leq \dot{q}_{i,max} \quad i = 1, 2, \ldots, n \tag{5.23}$$

In any optimization problem, the most important term is the objective function J that has to be optimized. Normally, the most commonly used objective function is time-optimality. This time-optimality objective function can be combined with other criteria such as energy minimization, fuel minimization or any other arbitrary cost function. It is the flexibility of Dynamic Programming that more general objective functions can be defined. A generalized cost function is shown in (5.24). In (5.24), κ is the weight factor. If $\kappa = 0$, then (5.24) will be an energy minimization problem and, if $\kappa = 1$, then this cost function will present a time minimization problem.

$$J = \int_{t_0}^{t_f} (\kappa + (1 - \kappa)u^2)dt \tag{5.24}$$

In cost function, the final time (t_f) can be fixed or it can be free depending upon the problem. For example, the final time cannot be fixed in case the time optimization problem is considered. The structure and the dynamics of the robotic manipulator are similar to the general optimal trajectory problem which consists of the systems dynamics, terminal conditions and the inequality constraints.

The first step in Dynamic Programming towards finding the optimal solution is the discretization of the given problem. Let's suppose, the calculated arc length is divided into N_p segments. The continuous dynamical equations and the terminal conditions can be described in the discretized form as given in (5.25), (5.26), and (5.27).

$$\vec{\tau}[k] = \vec{m}(s[k])(\dot{s}[k+1] - \dot{s}[k]) + \vec{c}(s[k])\dot{s}^2 + \vec{g}(s[k]), \quad k = 1, 2, \ldots, N_p. \tag{5.25}$$

$$s[1] = s_0, \quad s[N_p] = s_f \tag{5.26}$$

$$\dot{s}[1] = 0, \quad \dot{s}[N_p] = 0 \tag{5.27}$$

As a result of dimensionality reduction and prior information of given geometrical path, the problem is reduced to search over a single variable, i.e., pseudo-velocity $\dot{s}$. The maximum/minimum allowable pseudo-velocity can be calculated from the actuator's data sheets and given information of geometrical path. As we selected positive monotonically differentiable function as path parameter, therefore, minimum allowable value of pseudo-velocity will always be zero. The maximum allowable pseudo-velocity for the considered Delta parallel robot is found to be 5 m/sec, i.e. $\dot{s}_{max} = 5$.

In next step, the allowable range of joint velocity is discretized into N_v segments. So,

the algorithm will search over all the values of pseudo-velocity ($\dot{s}[j], j = 1, 2, \ldots, N_v$) to optimize the objective function for given path.

$$\dot{s}_{min} = \dot{s}[1], \qquad \dot{s}_{max} = \dot{s}[N_v] \tag{5.28}$$

Consider the path movement from point k to $k + 1$, we want to optimize this segment. Assuming that the discretization of given path is very fine, i.e. N_p is large, the travelled distance will become small and acceleration, $m(s), c(s)$, and $g(s)$ do not change significantly over a single interval. For a possible pseudo-velocity at point k, $\dot{s}[j_k]$, and an admissible pseudo-velocity at point $k + 1$, $\dot{s}[j_{k+1}]$, the pseudo-acceleration $\ddot{s}$ at point k can be calculated using (5.29). The graphical representation of (5.29) is shown in Figure 5.3.

$$\ddot{s}[k] = \frac{(\dot{s}[k+1])^2 - (\dot{s}[k])^2}{2(s[k+1] - s[k])} \tag{5.29}$$

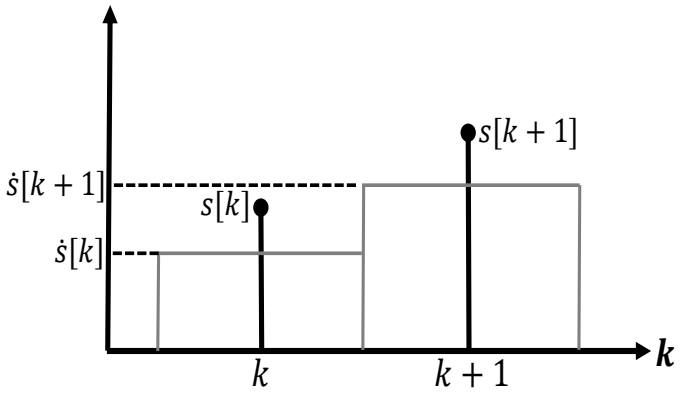


Figure 5.3: Graphical representation of (5.29) *and* (5.30)

The time required to travel from point k to $k + 1$ with pseudo-velocity $\dot{s}(j_{k+1})$ is given by (5.30).

$$\Delta t[k] = \frac{2\,(s[k+1] - s[k])}{\dot{s}[k+1] + \dot{s}[k]} \tag{5.30}$$

Once we have the values of path parameter(s), pseudo-velocity($\dot{s}$) and pseudo-acceleration($\ddot{s}$), the joints' torques required to travel from point k to $k + 1$ can be calculated using (5.25). After calculating the all joints' torques, the second inequality constraint (5.22) is checked. If any of the joint's torque does not satisfy the inequality torque constraint, $\dot{s}[k + 1]$ said to be inadmissible.

The next step after satisfying all the constraints is to calculate the incremental performance index to move from point k to $k+1$. Here $\Phi(\dot{s}[k], k)$ indicates the cost to move from point k to $k+1$ with a pseudo-velocity of $\dot{s}[k]$. Bellman's optimality principle is applied to calculate the minimum performance index:

$$J^f(\dot{s}[k], k) = \min_{\dot{s}[k+1], k=1,2,\ldots,N_p} [\Phi(\dot{s}[k], k)] + J^f(\dot{s}[k+1], k+1) \tag{5.31}$$

Equation (5.31) searches all over the admissible discretized velocities at point k. Thus, every admissible pseudo-velocity at point k gives a unique admissible pseudo-velocity at point $k+1$. In parallel, the pseudo-velocities, pseudo-accelerations and torques can be calculated corresponding to the optimal conditions.

Algorithm 4 in Appendix A.3.1 summarizes the DP algorithm for the trajectory optimization using path parameter. The application of this DP algorithm and the results obtained from different numerical examples are discussed in Section 5.4.

5.2.2 Dynamic Programming using Joint Coordinates

Another DP algorithm for trajectory optimization of robotic manipulators was proposed by Singh and Leu [4]. This method finds the optimal solution in terms of Joint space while the joint displacements could be obtained by solving the manipulator kinematics equations. In this method, the dimensionality of the problem is reduced by considering only a single actuator joint in DP algorithm.

This DP algorithm uses the dynamical equation of robot in terms of Joint space as given in (A.15)

$$\vec{\tau} = \mathbf{M}(\vec{q})\,\ddot{\vec{q}} + \mathbf{C}(\vec{q}, \dot{\vec{q}})\,\dot{\vec{q}} + \vec{G}(\vec{q})$$

The terminal conditions for the DP algorithm can also be found from the given path information like we discussed in Section 5.2.1. But terminal conditions, (5.33) and (5.34), are given in terms of Cartesian space which can be mapped to Joint space using the manipulator kinematics equations.

$$\vec{q} = \Psi(\vec{P}_0) \tag{5.32}$$

$$\vec{P}_0(t_0) = \vec{P}_0^{init}, \quad \vec{P}_0(t_f) = \vec{P}_0^{final} \tag{5.33}$$

$$\vec{P_0}(t_0) = 0, \quad \vec{P_0}(t_f) = 0 \tag{5.34}$$

In (5.34), P_0 represents the position of the TCP of robotic manipulator in Cartesian space. From these conditions, it is obvious that manipulator has zero velocity at terminal positions. Besides the terminal equality constraints, the inequality constraints are given in terms of joint actuators' torque, angles and velocities. Some additional constraints can also be imposed on the joints accelerations as well. These inequality constraints are given in (5.35) and (5.36).

$$\dot{q}_{i,min} \leq \dot{q}_i \leq \dot{q}_{i,max} \quad i = 1, 2, \ldots, n \tag{5.35}$$

$$\tau_{i,min} \leq \tau_i \leq \tau_{i,max} \quad i = 1, 2, \ldots, n \tag{5.36}$$

In case of Delta parallel robot n is equal to 3. The discretization steps for this DP algorithm is same as we already discussed in Section 5.2.1. The dynamical equations and the terminal equality constraints in discretized form are given below:

$$\begin{aligned}\vec{\tau}[k] =& \mathbf{M}(\vec{q}[k])(\vec{\dot{q}}[k+1] - \vec{\dot{q}}[k]) + \mathbf{C}(\vec{q}[k], \vec{\dot{q}}[k])(\vec{q}[k+1] - \vec{q}[k]) \\ &+ \vec{G}(\vec{q}[k]), \quad k = 1, 2, \ldots, N_p\end{aligned} \tag{5.37}$$

$$\vec{P}_0[1] = \vec{P}_0^{init} \quad \vec{P}_0[N_p] = \vec{P}_0^{final} \tag{5.38}$$

$$\vec{\dot{P}_0}[1] = 0 \quad \vec{\dot{P}_0}[N_p] = 0 \tag{5.39}$$

In above equations, N_p are the total number of discretized steps for the given geometrical path. Normally, to get the optimal trajectory of robotic manipulator for a given path using dynamic programming, a search is implied over $\vec{q}$ and $\vec{\dot{q}}$. In case of n-DOF robotic manipulator, the dynamic programming algorithm has to search over $2n$ variables. But due to the prior information of geometrical path, DP algorithm will search over only angular velocities $\vec{\dot{q}} = [\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n]$, and this will reduce the dimensionality of problem from $2n$ to n. The allowable range of joint velocity is discretized into N_v segments. In DP, the algorithm will search over all values of joints' velocities ($\vec{\dot{q}}[j], j = 1, 2, \ldots, N_v$) at each discretized step of given geometrical path to optimize the given objective function.

One possible solution approach could be to use DP in n-dimensions to calculate the optimal velocities of n joints in parallel to get the optimal solution. This solution approach will guarantee the global optimal solution but it would be computationally very expensive and a large memory size is required to store the variables. These two disadvantages raise a question on the applicability of this solution approach.

Another solution approach was proposed by Singh and Leu [4] in which a further reduction in the problem's dimension is achieved by considering only a single actuator joint in optimization process. According to Singh and Leu's algorithm, any non-stationary joint can be selected as a reference. Lets suppose, the joint angles of a robotic manipulator for a given path information are shown in Figure 5.4. In this scenario, Joint-1 and Joint-2 can't be considered as a reference joint because these joints are stationary in the starting and ending phase, respectively. In this case, Joint-3 will be considered as a reference joint as it is non-stationary throughout the movement of robotic manipulator.

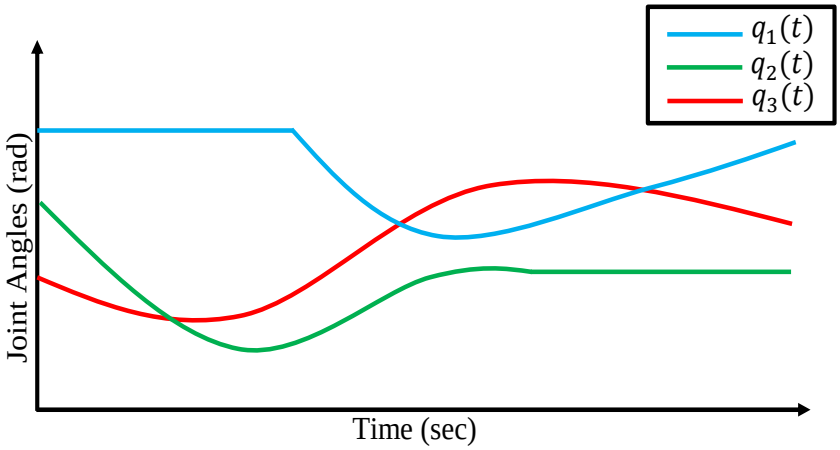


Figure 5.4: Graphical representation of the selection of non-stationary joint to implement Singh and Leu [4] Dynamic Programming algorithm for trajectory optimization

In this script, i^* is used for the index of reference non-stationary joint i.e., q_{i^*}, and k as the index of discrete points along the given path that is discretized into N_p segments, i.e., $k = 1, 2, \ldots, N_p$.

In order to give a brief overview about this algorithm's functionality, let's consider the path movement from point k to $k + 1$ and we want to optimize this segment. Just like the previous DP algorithm, assume that the discretization of given path is very fine (i.e. N_p is large), so the travelled distance will become small and acceleration, $\mathbf{M}(\vec{q})$, $\mathbf{C}(\vec{q}, \dot{\vec{q}})$ and $\vec{G}(\vec{q})$ do not change significantly over a single interval. For a possible velocity ($\dot{q}_{i^*}[k]$) at point k and an admissible velocity ($\dot{q}_{i^*}[k + 1]$) at point $k+1$, the joint acceleration at point k can be calculated using (5.40).

$$\ddot{q}_{i^*}[k] = \frac{(\dot{q}_{i^*}[k+1])^2 - (\dot{q}_{i^*}[k])^2}{2(q_{i^*}[k+1] - q_{i^*}[k])} \tag{5.40}$$

The time required to travel from point k to $k+1$ with joint velocity $\dot{q}_{i^*}[k+1]$ is given by (5.41).

$$\Delta t[k] = \frac{2(q_{i^*}[k+1] - q_{i*}[k])}{\dot{q}_{i^*}[k+1] + \dot{q}_{i^*}[k]} \tag{5.41}$$

The graphical representation of (5.40) and (5.41) is same as shown in Figure 5.3 for the path parametric method. All other joints must also cover the distance from point k to $k+1$ in the same time step. The velocities of other joints, to cover the distance in the same time interval, is calculated using (5.42).

$$\dot{q}_m[k] = \frac{2(q_m[k+1] - q_m[k])}{\Delta t[k]} - \dot{q}_m[k+1] \tag{5.42}$$

If the velocity of any non-reference joint $\dot{q}_m[k], [m = 1, 2, \cdots, n,\ m \neq i^*]$ does not lie in the joint velocity interval, the velocity of the reference non-stationary joint ($\dot{q}_{i^*}[k]$) is considered to be inadmissible and not considered for further calculations. If all the joints satisfy the velocity constraints, the accelerations for all other joints are calculated using (5.40). Once we have the values of displacement, velocity and acceleration of all joints, the joints' torques required to travel from point k to $k+1$ can be calculated using (5.37). After calculating the all joints' torques, the second inequality constraint (5.36) is checked. If any of the joint's torque does not satisfy the inequality torque constraint, $\dot{q}_{i^*}[k]$ said to be inadmissible.

After sorting out all the admissible and inadmissible angular velocities, the incremental performance index to move from point k to $k+1$ for all possible admissible velocities is calculated. Term $\Phi(\dot{q}_{i^*}[k], k)$ indicates the cost to move from point k to $k+1$ with a joint velocity of $\dot{q}_{i^*}[k]$. Equation (5.43) represents the Bellman's optimality principle. Thus, every admissible velocity of joint i^* at point k gives a unique admissible velocity of the same joint at point $k+1$.

$$J^f(\dot{q}_{i^*}[k], k) = \min_{\dot{q}_{i^*}(k+1), k=1,2,\cdots,N_p} [\Phi(\dot{q}_{i^*}[k], k)] + J^f(\dot{q}_{i^*}[k+1], k+1) \tag{5.43}$$

Algorithm 5 in Appendix A.3.2 summarizes the DP algorithm for trajectory optimization using Joint space. The flaws in this DP algorithm and the modifications to improve this algorithm are given in next subsection.

5.2.2.1 Flaws in Singh and Leu's Algorithm

The Dynamic Programming algorithm discussed in Section 5.2.2, proposed by Singh and Leu [4], is practically applicable and can solve the optimal trajectory planning with any arbitrary objective function. Besides the applicability of this algorithm, it has one flaw which require modifications in the algorithm.

First, this algorithm does not give any rule for selecting the reference non-stationary joint in case of more than one non-stationary joints (Step 4 of algorithm). In some cases, it happens that there are more than one non-stationary joints when following the given path. For example, in case of parallel robots, there are always more than one joint that are non-stationary during the movement of the manipulator on the given path. In this scenario, it is always a difficult task to choose a non-stationary joint amongst more than one non-stationary joints because choosing a non-stationary joint randomly does not guarantee the global optimal solution and may ends up with a non-optimal solution. An alternative way is to solve the whole problem with respect to each non-stationary joint and get the optimal solution by comparing the result obtained from each individual case. Solving the problem with respect to each non-stationary joint makes this algorithm computationally expensive.

5.2.2.2 Joint Selection Criterion

In this thesis, we present a novel heuristic based joint selection criterion in case of more than one non-stationary joints.

Criterion 1 *To get the global optimal solution, select the non-stationary joint as a reference (Step 4) which has the lowest sum of absolute differences between pairs of successive discrete angles along the given path.*

$$\text{Reference Joint}(i^*) = \underset{i=1,2,\ldots,n_{dyn}}{\arg\min} \left(\sum_{k=1}^{N_p-1} |q_i[k+1] - q_i[k]| \right) \tag{5.44}$$

In (5.44), n_{dyn} represents the number of non-stationary joints during the movement of the manipulator on given path. The graphical representation of the proposed criterion is shown in Figure 5.5. In this Figure, there are three non-stationary joints, i.e. $n_{dyn} = 3$. Each non-stationary joint is discretized and sum of absolute difference between pairs of successive discrete points. The joint that would give the least sum of absolute difference will be selected as a reference joint for trajectory optimization using Singh and Leu's Dynamic Programming algorithm. It is basically the limitation of the Singh and Leu's algorithm that there must atleast one non-stationary joint to implement this algorithm. This algorithm cannot be applied in the case if there is no non-stationary joint.

The sum of the absolute differences of each non-stationary joint between pairs of suc-

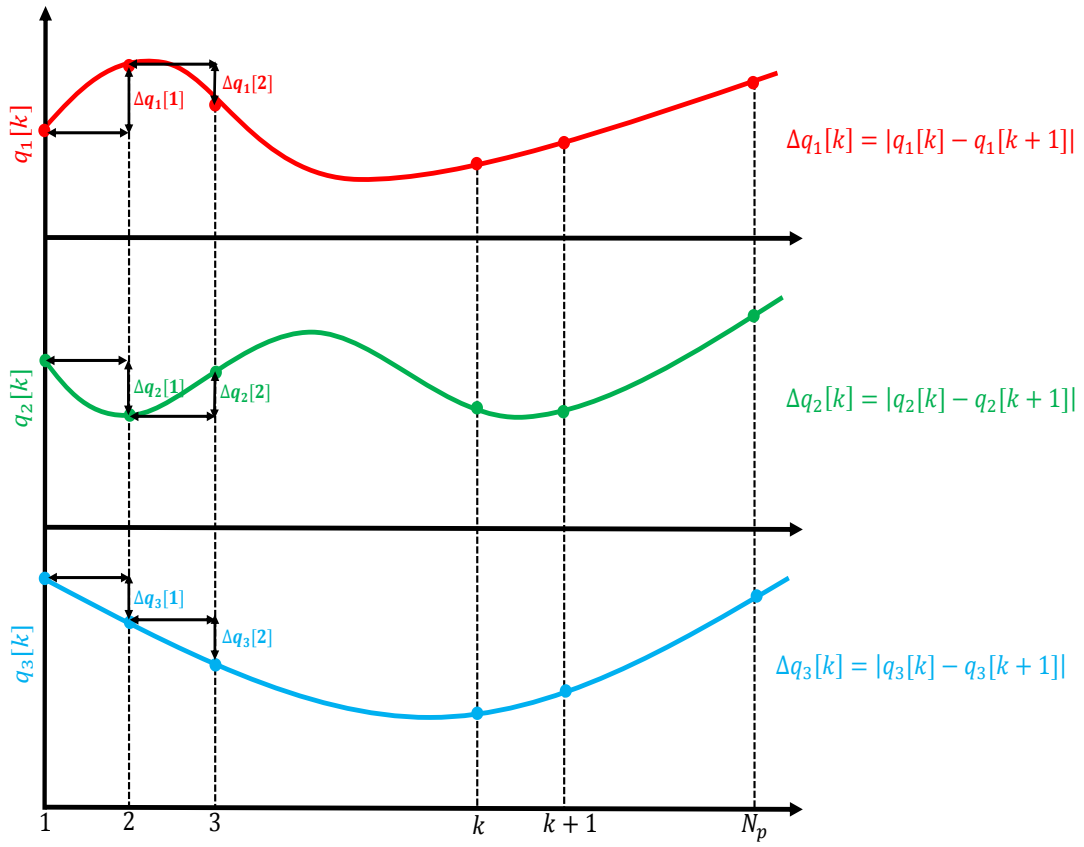


Figure 5.5: Graphical representation of proposed joint selection criterion

cessive discrete angles must be comparable. If the sum of absolute differences of a non-stationary joint is very small and not comparable to the other non-stationary joints, although it will satisfy the above mentioned criterion but the optimal trajectory cannot be calculated by selecting this joint as a reference. In case of large difference between sum of absolute difference of non-stationary joints, the joint that has the least sum of absolute difference will be considered as stationary as compared to other non-stationary joints, because its velocity would be zero as compare to other non-stationary joints.

This proposed joint selection criterion is basically a recommendation to obtain the global optimal solution. Although, this criterion holds in most of the cases but still it is not a strict rule. As discussed in the above paragraph, the sum of absolute differences of each non-stationary joint must be comparable. In this thesis, we have not discussed the threshold or the maximum limit of the difference that must be between the sum of absolute differences of each non-stationary joint. This threshold or the maximum limit of the difference may vary from case-by-case depending upon the given geometrical path and robotic manipulator.

For the validation purpose, this DP algorithm with the proposed joint selection criterion and the modification is applied on a Delta parallel robot and the obtained results are

compared with other state-of-the-art techniques.

5.3 Trajectory Optimization using Discrete Mechanics and Optimal Control

In this section, trajectory optimization using a newly developed technique called *Discrete Mechanics and Optimal Control* (DMOC) will be discussed. DMOC is quite different from the classical approaches to solve the optimal control problem for mechanical systems. In first step of the classical approach, the variational principles are applied to the Lagrange-d'Alembert principle and the Euler-Lagrange equations are obtained. In the second step, discretization or variation for optimal control problem takes place depending upon the direct or indirect approach.

On the other hand in DMOC approach, first the discretization takes place and we convert the continuous Lagrange-d'Alembert principle and continuous objective function to discrete Lagrange-d'Alembert principle and discrete objective function. In second step, discrete variation principles are applied to obtain the discrete Euler-Lagrange equations from the discretized Lagrange-d'Alembert principle. The flow schemes to solve the optimal control problem for mechanical systems using classical control approach and DMOC is shown in Figure 5.6.

As it is a variational based method, so a little understanding of the variational principles of mechanics and variational integrators would help in understanding the equations. The literature on these topics can be found in [103, 134, 135].

Lagrangian function, $L = T - V$, plays an important role in DMOC and it is basically the difference between system's kinetic energy, T, and potential energy, V. By standard definition, the Lagrangian mechanics considers the integral of L along the curve and then calculates a variation δ. This variation must be equals to zero.

$$\begin{aligned}
\delta \int_0^{t_f} L(\vec{q}(t), \dot{\vec{q}}(t))dt &= \int_0^{t_f} \left[\frac{\partial L}{\partial \vec{q}} \cdot \delta \vec{q} + \frac{\partial L}{\partial \dot{\vec{q}}} \cdot \delta \dot{\vec{q}} \right] dt \\
&= \int_0^{t_f} \left[\frac{\partial L}{\partial \vec{q}} \cdot \delta \vec{q} - \frac{d}{dt} \frac{\partial L}{\partial \dot{\vec{q}}} \cdot \delta \vec{q} \right] dt + \left[\frac{\partial L}{\partial \dot{\vec{q}}} \cdot \delta \vec{q} \right]_0^{t_f} \\
&= \int_0^{t_f} \left[\frac{\partial L}{\partial \vec{q}} - \frac{d}{dt} \frac{\partial L}{\partial \dot{\vec{q}}} \right] \cdot \delta \vec{q} \, dt
\end{aligned}$$

In above equations, $\vec{q} = [q_1, q_2, \ldots, q_n]$ are the generalized coordinates of the system considered to solve the problem. In variational approach, it is considered that variation of the integral must be zero for all variations. By this consideration, we get the well-

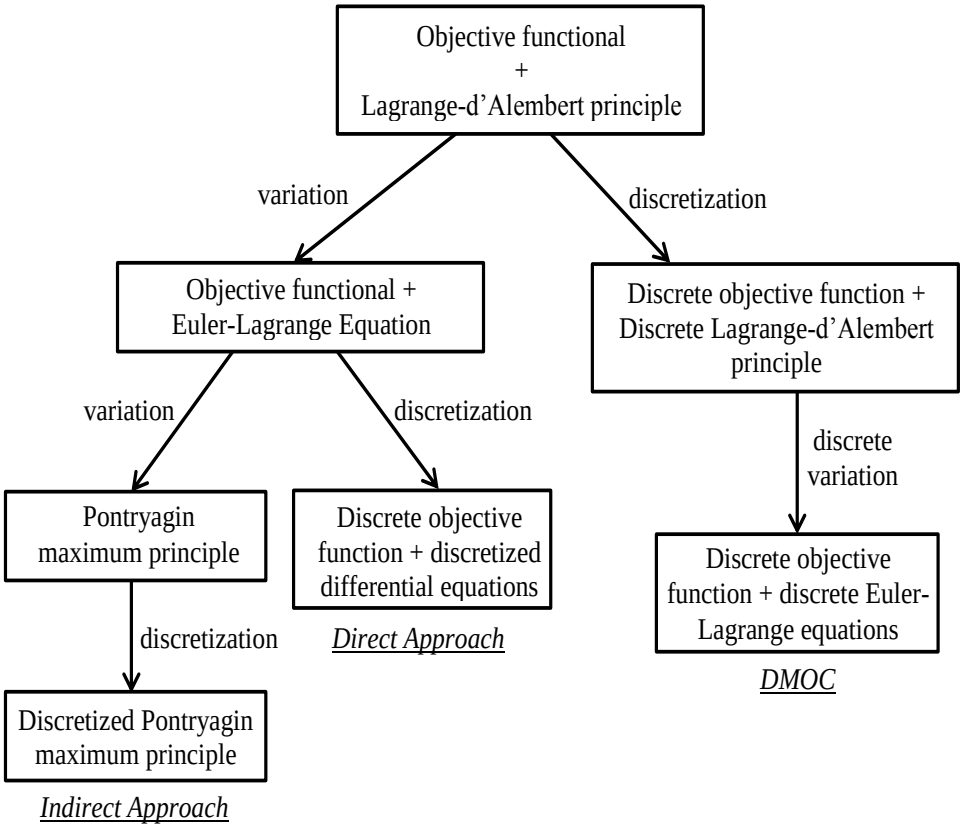


Figure 5.6: Flow scheme to solve the optimal control problem for mechanical systems using classical control approach and DMOC [5]

known Euler-Lagrange equation,

$$\frac{\partial L}{\partial \vec{q}} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\vec{q}}} = 0 \tag{5.45}$$

In Section 5.3.1, the generalized problem formulation to solve the optimal control problem using DMOC is discussed. Discretization is an important step in DMOC and it is discussed in Section 5.3.2. Discretization of the problem, cost function, and the boundary conditions are formulated in Section 5.3.2 and 5.3.3, respectively.

5.3.1 Problem Formulation

Consider Delta parallel robot as a mechanical system having a 3 dimensional configuration manifold Q, $Q \in \mathbb{R}^3$, and the configuration vector $\vec{q} = [q_1, q_2, q_3]$. Here $\vec{q} \in \mathbb{R}^3$ can either be the position of the Tool Center Point (TCP) or the angle of the Joint actuators. As discussed earlier that trajectory optimization for a predefined geometrical path is performed in Joint space, so configuration vector will consist of joints' angles, i.e., $\vec{q} = [\theta_1, \theta_2, \theta_3]$. Delta parallel robot has to move from the initial point $(\vec{q}^0, \dot{\vec{q}}^0) \in \mathcal{T}Q$,

with $\mathcal{T}Q$ the tangent space, to the final point $(\vec{q}^{t_f}, \vec{\dot{q}}^{t_f})$ during a time interval $[0,\ t_f]$. The generalized forces $\vec{f}(\vec{q}(t), \vec{\dot{q}}(t), \vec{u}(t)) \in \mathcal{T}^*_{q(t)}Q$, with $\mathcal{T}^*Q$ cotangent space are responsible to control the system during its movement from initial state to final state and in this equation $\vec{u}(t) \in U \subseteq \mathbb{R}$ is a control parameter. In case of two point bounded value problem (BVP), the configuration vector's components ($\vec{q} = [q_1, q_2, q_3]$), the velocities of configuration vector's components ($\vec{\dot{q}} = [\dot{q}_1, \dot{q}_2, \dot{q}_3]$) and the force are the parameters that have to be calculated in order to optimize the given objective functional $J(\vec{q}, \vec{\dot{q}}, \vec{u})$.

$$J(\vec{q}, \vec{\dot{q}}, \vec{u}) = \int_0^{t_f} C(\vec{q}(t), \vec{\dot{q}}(t), \vec{\tau}(t))dt \tag{5.46}$$

In parallel, the motion $\vec{q}(t)$ and the controlled forces $\vec{\tau}(t)$ must satisfy the Lagrange-d'Alembert principle given in (5.47).

$$\delta \int_0^{t_f} L(\vec{q}(t), \vec{\dot{q}}(t))dt + \int_0^{t_f} \vec{\mathcal{F}}(t) \cdot \delta\vec{q}(t)dt = 0 \tag{5.47}$$

In (5.47), δ represents variations that vanish at the end points.

$$\delta\vec{q}(0) = \delta\vec{q}(t_f) = 0 \tag{5.48}$$

5.3.2 Discretization

In this section, a global discretization method is used to discretize the states and controls for transforming the optimal control problem, stated in (5.46) and (5.47), into a finite dimensional constrained optimization problem [5]. In discretization the state space $\mathcal{T}Q$ of the system is replaced by $Q \times Q$ and discrete Lagrange is a function $L_d : Q \times Q \to \mathbb{R}$ [106]. The discretization grid is defined by $\Delta t = \{t_k = kh \mid k = 0, \ldots, N\},\ Nh = t_f$, where N is the number of steps in the trajectory and h is the step size. The path $\vec{q} : [0,\ t_f] \to Q$ is replaced by a discrete path $\vec{q_d} : \{t_k\}_{k=0}^N \to Q$. The discretization of a path is shown in Figure 5.7.

Similar to path in configuration space, the control path $u : [0,\ t_f] \to U$ is also discretized. A refined grid $\Delta\tilde{t}$, is generated by a set of control points $0 \le c_1 \le \ldots \le c_s \le 1$ such that $\Delta\tilde{t} = \{t_{kl} = t_k + c_l h | k = 0, \ldots, N-1, l = 1, \ldots, s\}$. By using these notations, the defined path is discretized to be $\vec{u_d} : \Delta\tilde{t} \to U$. The intermediate sample of control input $\vec{u_k}$ on $[t_k,\ t_{k+1}]$ as $\vec{u_k} = [u_{k1}, \ldots, u_{ks}] \in U^s$ to be the value of control parameter moving the system from $\vec{q}_k = \vec{q_d}(t_k)$ to $\vec{q}_{k+1} = \vec{q_d}(t_{k+1})$ where $\vec{u}_{kl} = \vec{u_d}(t_{kl})$ for

$l \in \{1, \ldots, s\}$ [106]. In case of robotic manipulators, force $\vec{u}$ and torque $\vec{\tau}$ are the same and these term can be used interchangeably.

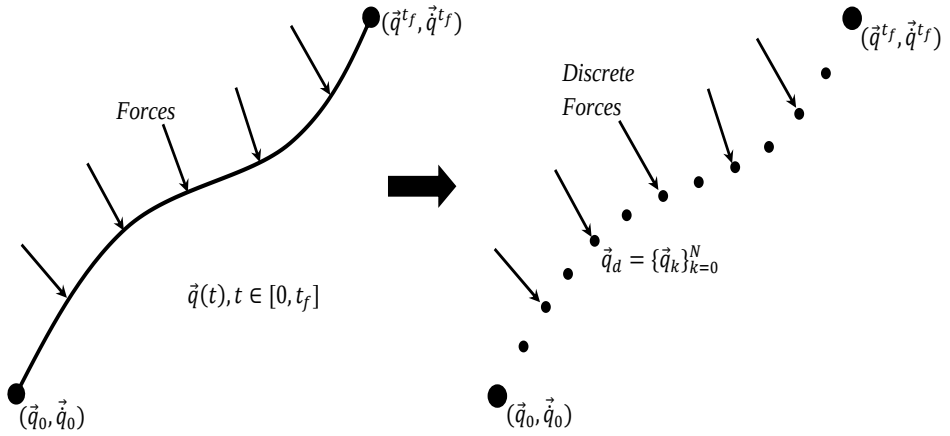


Figure 5.7: Discretization of a path q(t) [6]

Based on the discretization, discussed by Marsden et al. [103], approximate integral of the Lagrangian of the mechanical system, first part of (5.47), for a short time slice $[kh,\ (k+1)h]$ is given by (5.49).

$$L_d(\vec{q}_k, \vec{q}_{k+1}) \approx \int_{kh}^{(k+1)h} L(\vec{q}(t), \dot{\vec{q}}(t))dt, \tag{5.49}$$

and the discrete forces are given by (5.50).

$$\vec{f}_k^{-} \cdot \delta\vec{q}_k + \vec{f}_k^{+} \cdot \delta\vec{q}_{k+1} \approx \int_{kh}^{(k+1)h} \vec{\tau}(t) \cdot \delta\vec{q}(t)dt \tag{5.50}$$

Here $\vec{f}_k^{-}, \vec{f}_k^{+} \in \mathcal{T}^*Q$ are the left and right discrete forces dependent on $(\vec{q}_k, \vec{q}_{k+1}, \vec{u}_k)$, shown in Figure 5.8. The discrete form of the Lagrange-d'Alembert principle can be obtained by combining (5.49) and (5.50).

$$\delta \sum_{k=0}^{N-1} L_d(\vec{q}_k, \vec{q}_{k+1}) + \sum_{k=0}^{N-1} [\vec{f}_k^{-} \cdot \delta\vec{q}_k + \vec{f}_k^{+} \cdot \delta\vec{q}_{k+1}] = 0 \tag{5.51}$$

The discrete Euler-Lagrange equation is equivalent to the system given by (5.52).

$$D_2L_d(\vec{q}_{k-1}, \vec{q}_k) + D_1L_d(\vec{q}_k, \vec{q}_{k+1}) + \vec{f}^{+}_{k-1} + \vec{f}^{-}_{k} = 0 \tag{5.52}$$

The forced discrete Euler-Lagrange equations are the discrete form of the Lagrange-d'Alembert principle, shown in (5.52), and this equation is the equality constraint of the optimization problem. In (5.52), $k = 1, 2, \cdots, N-1$ and D_i is the derivative with respect to the i-th component. For example, $D_2L_d(\vec{q}_{k-1}, \vec{q}_k)$ means the derivative of the discrete Lagrange (L_d) with respect to $\vec{q}_k$.

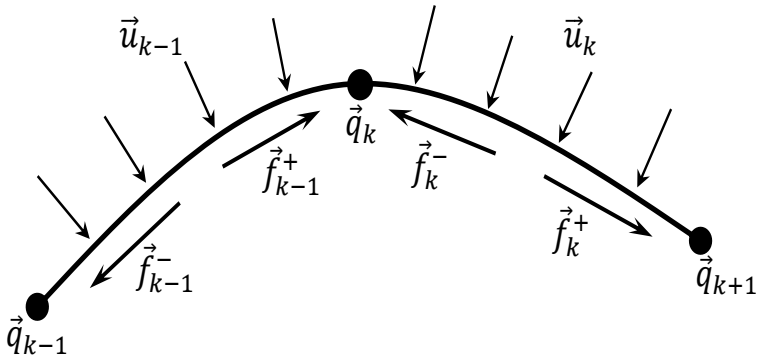


Figure 5.8: Left and right discrete forces [6]

Using the same discretization principle [103], approximation of the cost function (5.46) for a small interval of time $[kh,\ (k+1)h]$ is given by (5.53).

$$C_d(\vec{q}_k, \vec{q}_{k+1}, \vec{u}_k) \approx \int_{kh}^{(k+1)h} C(\vec{q}(t), \dot{\vec{q}}(t), \vec{u}(t))dt \tag{5.53}$$

The overall discrete objective function can be calculated by summing up (5.53) for all discrete steps of the trajectory.

$$J_d(\vec{q}_d, \vec{u}_d) = \sum_{k=0}^{N-1} C_d(\vec{q}_k, \vec{q}_{k+1}, \vec{u}_k) \tag{5.54}$$

From (5.52), one can see that it always requires two nodes to calculate the control input, one is the current node and other is the neighborhood node. This equation can easily be applied to the intermediate discrete points, but it cannot be applied directly to the terminal nodes. For this purpose, we have to derive the terminal conditions separately.

5.3.3 Boundary Conditions

At last stage, the boundary conditions $\vec{q}(0) = \vec{q}^0$, $\dot{\vec{q}}(0) = 0$ and $\vec{q}(t_f) = \vec{q}^{t_f}$, $\dot{\vec{q}}(t_f) = 0$ have to be incorporated in the problem. To this end, the description in $Q \times Q$ is linked to one in $\mathcal{T}Q$ using the discrete Legendre transforms $\mathbb{F}^{f+}L_d : Q \times Q \to \mathcal{T}^*Q$ and $\mathbb{F}^{f-}L_d : Q \times Q \to \mathcal{T}^*Q$ for forced systems, which are described as follows [5]:

$$\mathbb{F}^{f+}L_d : (\vec{q}_{k-1}, \vec{q}_k) \to (\vec{q}_k, \vec{p}_k) \tag{5.55a}$$

$$\vec{p}_k = D_2L_d(\vec{q}_{k-1}, \vec{q}_k) + \vec{f}^{+}_{k-1} \tag{5.55b}$$

$$\mathbb{F}^{f-}L_d : (\vec{q}_{k-1}, \vec{q}_k) \to (\vec{q}_{k-1}, \vec{p}_{k-1}) \tag{5.56a}$$

$$\vec{p}_{k-1} = -D_1L_d(\vec{q}_{k-1}, \vec{q}_k) - \vec{f}^{-}_{k-1} \tag{5.56b}$$

Use of standard Legendre transform will lead to the following two discrete boundary conditions [99].

$$D_2L(\vec{q}^0, \dot{\vec{q}}^0) + D_1L_d(\vec{q}_0, \vec{q}_1) + \vec{f}^{-}_0 = 0 \tag{5.57}$$

$$-D_2L(\vec{q}^{t_f}, \dot{\vec{q}}^{t_f}) + D_2L_d(\vec{q}_{N-1}, \vec{q}_N) + \vec{f}^{+}_{N-1} = 0 \tag{5.58}$$

These equality constraints are the constraints for optimal control problem.

5.3.4 Practical Implementation

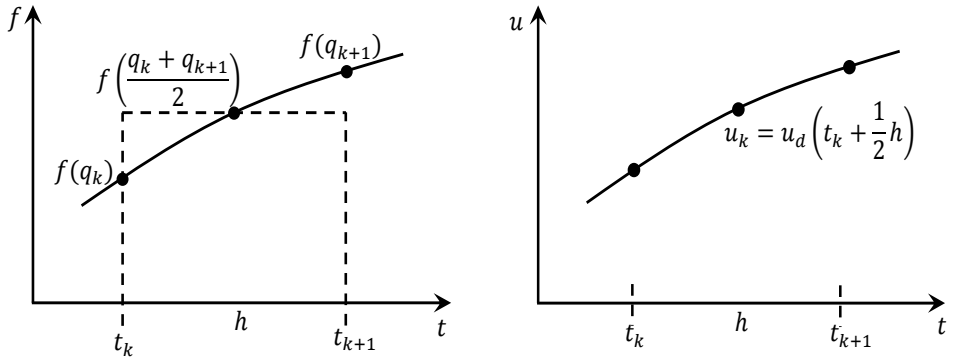
For the sake of compromise between accuracy and efficiency, the midpoint rule is used to approximate the discrete cost function C_d, discrete Lagrangian L_d, and the discrete forces. Constant control parameters are assumed on each time interval with $l = 1$ and $c_1 = 0.50$ for approximating the discrete terms [106].

$$C_d(\vec{q}_k, \vec{q}_{k+1}, \vec{u}_k) = hC\left(\frac{\vec{q}_{k+1} + \vec{q}_k}{2}, \frac{\vec{q}_{k+1} - \vec{q}_k}{h}, \vec{u}_k\right) \tag{5.59}$$

$$L_d(\vec{q}_k, \vec{q}_{k+1}) = hL\left(\frac{\vec{q}_{k+1} + \vec{q}_k}{2}, \frac{\vec{q}_{k+1} - \vec{q}_k}{h}\right) \tag{5.60}$$

$$\vec{f}_k^- = \vec{f}_k^+ = \frac{h}{2} f\left(\frac{\vec{q}_{k+1} + \vec{q}_k}{2}, \frac{\vec{q}_{k+1} - \vec{q}_k}{h}, \vec{u}_k\right) \tag{5.61}$$

The graphical representation of discrete configuration vector and discrete force using midpoint rule are shown in Figure 5.9.



(a) Midpoint rule for configutarion vector (b) Discrete force using midpoint rule

Figure 5.9: Graphical representation of midpoint rule

We can also use the Trapezoidal rule or Simpson's rule to approximate the numerical integrals. The further details of the aforementioned methods can be found in the literature [136, 137, 138, 139].

5.3.5 Trajectory Optimization of Predefined Geometrical Path using DMOC

In this section, problem formulation to use DMOC for trajectory optimization of the predefined geometrical path for robotic manipulators is discussed. In this thesis, we are focusing on the Delta parallel robot and will discuss the problem formulation in this regard. The predefined geometrical path for robotic manipulators can be given either in Cartesian space or in Joint space.

From the computational point of view, it is always efficient to use the Joint space to solve the trajectory optimization problem for a given geometrical path for robotic manipulators. While considering the Joint space, the configuration vector will consist of the joint angles as given in (5.62). Solving the problem in Joint space gives the advantage to implement the constraints on joints' velocities and accelerations more easily as compared to Cartesian space.

$$\vec{q} = [q_1, q_2, q_3] = [\theta_1, \theta_2, \theta_3] \tag{5.62}$$

Just like DP, constraints on the joints actuator's torque and velocity can also be incor-

porated in DMOC.

$$\dot{q}_{i,min} \leq \dot{q}_i \leq \dot{q}_{i,max} \quad i = 1, 2, 3.$$

$$u_{i,min} = \tau_{i,min} \leq \tau_i \leq \tau_{i,max} = u_{i,max} \quad i = 1, 2, 3.$$

If we combine all the constraints, then we have the following set of equality and inequality constraints.

$$\dot{q}_{i,min} \leq \dot{q}_i \leq \dot{q}_{i,max} \quad i = 1, 2, 3. \tag{5.63}$$

$$\tau_{i,min} \leq \tau_i \leq \tau_{i,max} \quad i = 1, 2, 3. \tag{5.64}$$

$$D_2L_d(\vec{q}_{k-1}, \vec{q}_k) + D_1L_d(\vec{q}_k, \vec{q}_{k+1}) + \vec{f}^{+}_{k-1} + \vec{f}^{-}_{k} = 0, \; k = 1, 2, \ldots, N \tag{5.65}$$

$$D_2L(\vec{q}^{0}, \vec{\dot{q}}^{0}) + D_1L_d(\vec{q}_0, \vec{q}_1) + \vec{f}^{-}_{0} = 0 \tag{5.66}$$

$$-D_2L(\vec{q}^{t_f}, \vec{\dot{q}}^{t_f}) + D_2L_d(\vec{q}_{N-1}, \vec{q}_N) + \vec{f}^{+}_{N-1} = 0 \tag{5.67}$$

In aforementioned set of constraints, N represents the number of discretized steps of the given path and i represents the number of joints actuators of Delta parallel robot.

If we consider the approximation of discrete cost function C_d, discrete Lagrangian L_d, and discrete forces, given in (5.59),(5.60), and (5.61) respectively, then the unknown parameters in these equations are the joints' torques/forces ($\vec{u}_k$), and the time step (h) required to move the robotic manipulator from one discretized point to next discretized point, i.e. $[\vec{\tau}_1 \; \vec{\tau}_2 \; \vec{\tau}_3 \; h]$. The other requirements of this Lagrangian based method are the system's kinetic energy and potential energy, that have been discussed in Appendix A.1.2.

Once the problem is formulated to solve using DMOC, the optimal values of above mentioned unknown parameters in nonlinear equations can be calculated using any nonlinear programming package, e.g. SQP. In this thesis, we are using `fmincon` command in MATLAB to solve the problem.

5.4 Numerical Example

In this section, two predefined geometrical paths are optimized using the aforementioned optimization techniques and the results are compared. To solve these numerical examples, the dynamical model and the geometrical parameters of D4-500 Delta parallel robot are considered, as discussed in Chapter 3.

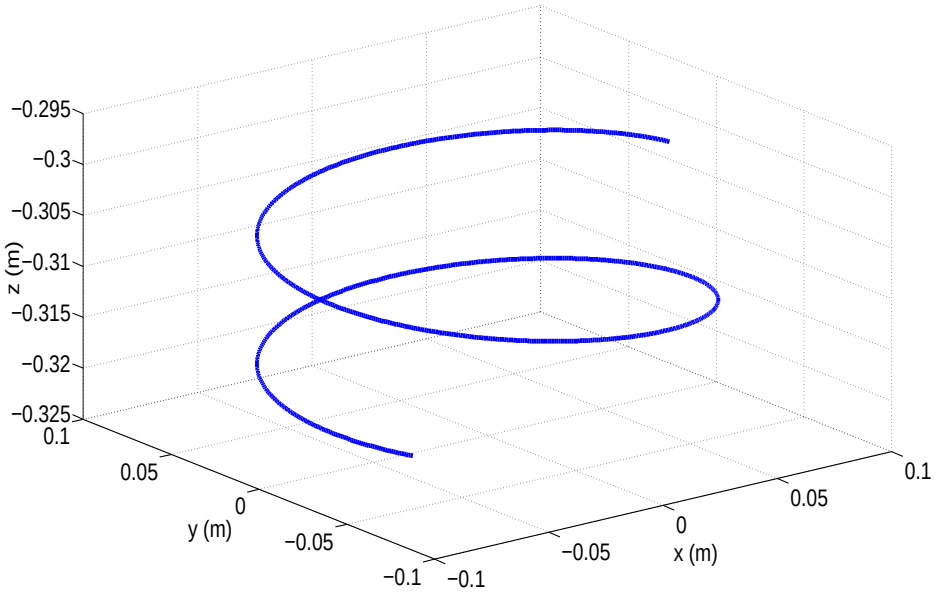


Figure 5.10: Spiral movement of the robotic manipulator considered for the trajectory optimization in Example 1.

5.4.1 Example 1

In Example-1, a spiral movement of the Delta parallel robot is considered. The spiral movement, shown in Figure 5.10, is more complex as compared to the straight line movement that is mostly used in pick and place operations. The objective function in this example is to find the optimal trajectory to minimize the travelling time from starting point to ending point. The final time is free and the step size (h) is the optimization variable in this example.

This trajectory optimization problem is solved by Phase-Plane method, Dynamic Programming using path parameter, Dynamic Programming using Joint space, and DMOC. For the purpose of fair comparison, the predefined path is divided into equal number of discretization steps for each optimization technique. A grid of size 50×50 is used in Dynamic Programming (i.e. $N_p = N_v = 50$) and the predefined path is divided into 50 steps for Phase-Plane method and DMOC (i.e. $N = 50$).

The optimized values of cost function by above mentioned optimization techniques are shown in Table 5.1. To get the optimal solution using the Dynamic Programming algorithm proposed by Singh et al. [4] and to show the validation of the proposed joint selection criterion, presented in Section 5.2.2.2, the problem is solved three times with respect to each non-stationary joint. The different optimal costs are obtained by considering each joint separately as a reference and the global optimal solution is obtained by considering the Joint-2 as reference joint. According to the proposed joint selection criterion, the Joint-2 gives the optimal solution as it has the lowest sum of absolute differences and the sum of absolute differences of Joint-2 is comparable to all other non-stationary joints. This proves the applicability of the proposed joint selection

Table 5.1: Comparison of the optimal results by different optimization techniques (Example-1)

Dynamic Programming using Joint Space			**Dynamic Programming using Path Parameter**	**Phase-Plane Method**	**DMOC**
Reference Joint	Sum of absolute difference	Cost	Cost	Cost	Cost
Joint-1	11.7408	0.4300	0.3972	0.3986	0.3991
Joint-2	11.0382	0.3996			
Joint-3	11.9253	0.4367			

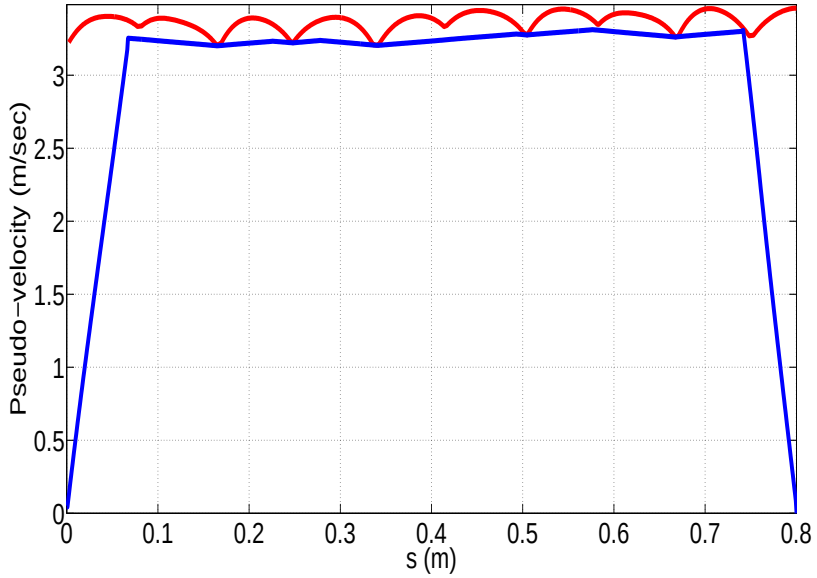


Figure 5.11: Forward and backward integration of acceleration and deceleration for Phase-Plane method. Red line represents the velocity limit curve, blue line represents the optimal pseudo-velocity along arc-length.

criterion.

Figure 5.11 shows the velocity limit curve and the forward and backward integration of the accelerations and decelerations for optimal solution using Phase-Plane method. In this figure, red line shows the velocity limit curve and blue line represents the optimal pseudo-velocity $\dot{s}$. The continuous optimal pseudo-velocity curve consists of multiple segments of forward and backward integration.

From Table 5.1, one can see that the value of optimized cost function by different optimization techniques are comparable. The optimized cost function by different techniques have an average of 0.3986 sec and a standard deviation of 8.9547×10^{-4} sec.

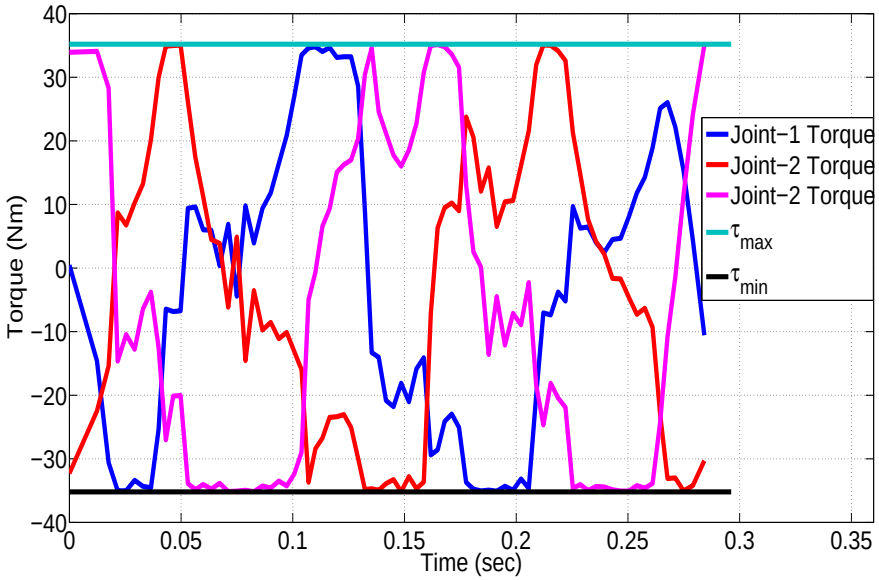


Figure 5.12: Optimal joints' torques for Example-1

The optimized joints' torques to achieve the minimum travelling time are shown in Figure 5.12. According to Pontryagin's maximum principle, at least one of the joints' actuators must operate in saturation region in order to have the optimal value. In this figure, one can clearly see the Bang-Bang control, and throughout the manipulator manoeuvre at least one of the actuator joint is operating in the saturation region. This satisfies the Pontryagin's maximum principle.

5.4.2 Example 2

In second example, a path, shown in Figure 5.13, is considered for optimization using different optimization methods. The objective in this example is to find a time-optimal trajectory while satisfying the constraints on joints' torques and the optimization variable is the time step (h). Just like the Example 1, the final time is free and this problem is also solved by the discussed optimization techniques. A grid of size 80×80 is used in Dynamic Programming (i.e. $N_p = N_v = 80$) and the predefined path is divided into 80 steps for Phase-Plane method and DMOC (i.e. $N = 80$).

The optimized value of cost function by above mentioned optimization techniques are shown in Table 5.2. To get the optimal solution using the Dynamic Programming algorithm proposed by Singh et al. [4] and to show the validation of the proposed joint selection criterion, the problem is solved three times with respect to each non-stationary joint. As per proposed joint selection criterion, Joint-1 must give the global optimal solution as it fulfills the proposed criterion. Its sum of absolute differences between pairs of successive discrete points is lowest and comparable to the all other non-stationary joints' value. The results in Table 5.2 show the validation of the pro-

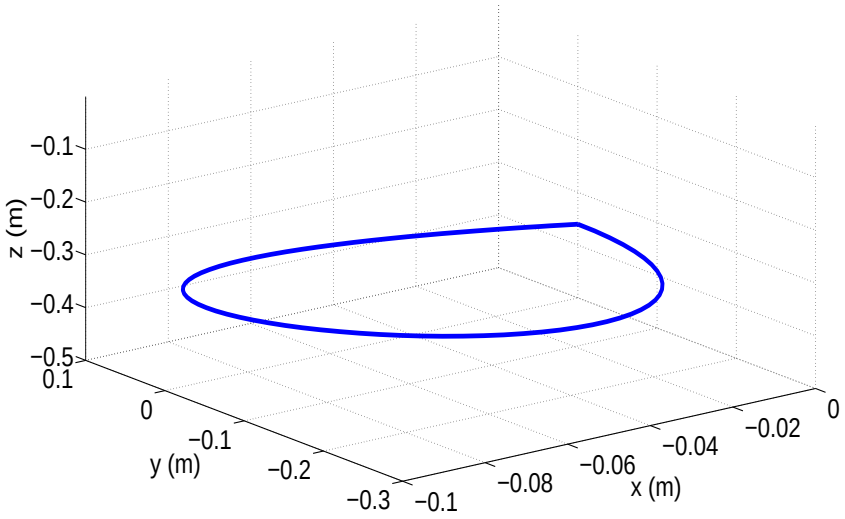


Figure 5.13: Predefined geometric path of the robotic manipulator considered for the trajectory optimization in Example-2.

Table 5.2: Comparison of the optimal results by different optimization techniques (Example-2)

Dynamic Programming using Joint Space			**Dynamic Programming using Path Parameter**	**Phase-Plane Method**	**DMOC**
Reference Joint	Sum of absolute difference	Cost	Cost	Cost	Cost
Joint-1	15.2207	0.2580	0.2680	0.2655	0.2620
Joint-2	28.8829	0.3997			
Joint-3	19.0677	0.2971			

posed joint selection criterion.

The Velocity Limit Curve (VLC) and the forward and backward integrations of accelerations and decelerations to optimize the given geometrical trajectory using Phase-Plane method are shown in Figure 5.14. The forward and backward integration is performed multiple times from switching points in order to get the continuous optimal pseudo-velocity curve, as shown in Figure 5.14 by blue line.

Figure 5.15 shows the optimized torques of the joints' actuators for time-optimal control of the robotic manipulator to follow the predefined geometrical path. The optimal solution satisfies the torque constraints imposed in this problem and all the joints' torques are within the upper and lower limit. From Table 5.2, one can see that the value of optimized cost function by different optimization techniques are comparable.

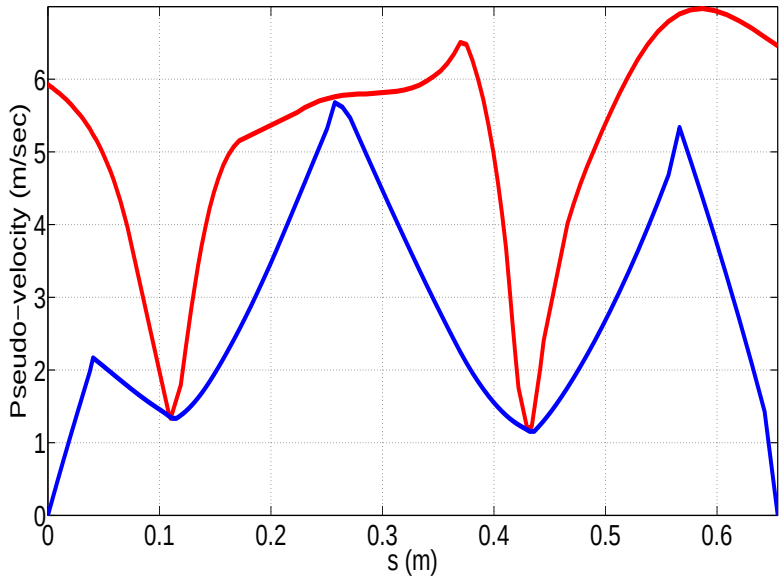


Figure 5.14: Forward and backward integration of acceleration and deceleration using Phase-Plane method for Example-2. Red line represents the velocity limit curve and blue line represents the optimal velocity along arc-length.

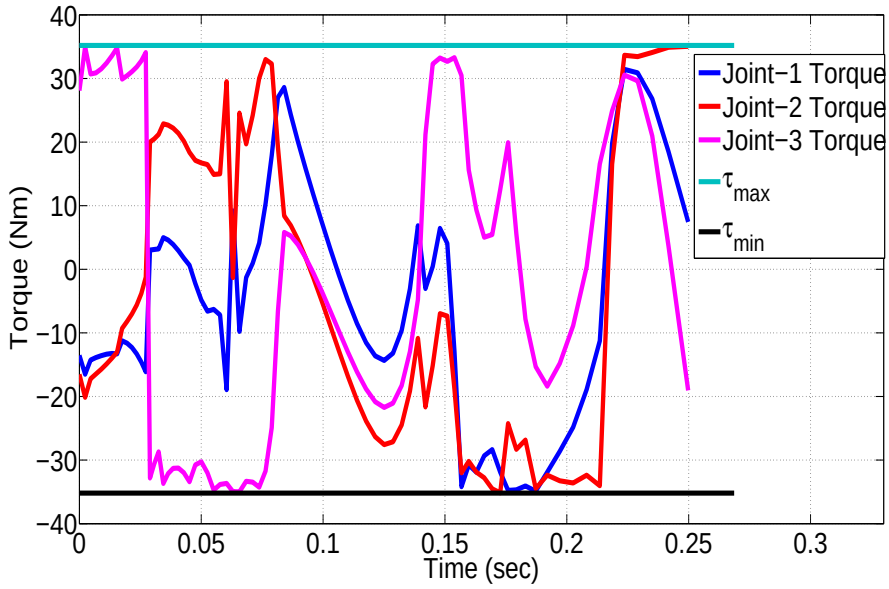


Figure 5.15: Optimal joints' torques for Example-2

The optimized cost function by different techniques have an average of 0.2634 sec and a standard deviation of 3.8×10^{-3} sec.

In this section, we have described two numerical examples to find the optimal trajectory for a robotic manipulator. The optimization problems are solved using four different techniques and the numerical results are compared to each other.

5.5 Comparison of Different Optimization Techniques

In previous sections, two numerical examples are discussed using the Phase-Plane method, Dynamic Programming and Discrete Mechanics and Optimal (DMOC). In this section, the advantages and disadvantages of the above mentioned techniques are discussed for the purpose of comparison.

The Phase-Plane method, proposed by Bobrow et al. [78] and Shin et al. [79], is a very computationally efficient method for all type of robotic manipulators. Computational efficiency is achieved by dimensionality reduction, in which the dynamical equations are converted to a set of second order differential equations using path parameter. One of the major advantages of this method is that it always calculates the global optimal solution. This method follows the Bellman's principle of optimality, i.e., once we have calculated the optimal solution for an interval, the optimal solution for any sub-interval can be found from the calculated optimal solution. Besides these advantages, Phase-Plane method has also some drawbacks. One of the major drawbacks that restricts its applicability is that it can only solve the time minimization problem and it is impossible to optimize the multi-objective cost function. The other drawbacks associated with this method are the limitations to add the arbitrary constraints in optimization problem and the instantaneous change in joints' torques. It is not easy to express any arbitrary constraints in terms of path parameters which restrict the ability of Phase-Plane method to handle arbitrary constraints.

Dynamic Programming is a very useful optimization technique which can solve the optimization problem for highly nonlinear, complex systems under strong constraints. Besides having the advantages of Phase-Plane method, it can be used for multi-objective optimization and additional arbitrary constraints can be handled very easily within the optimization problem. The only disadvantage associated with Dynamic Programming is the computationally efficiency. It is not computationally efficient as it has to explore all the possible combinations which makes it slower. Usually, trajectory optimization for a predefined geometrical path is performed off-line, so computational efficiency is not a big issue.

Another optimization technique that we discussed in this thesis is DMOC. Just like DP, it can also be used for multi-objective optimization and this method can also handle any arbitrary constraints very easily. One of the major disadvantages of this method is the curse of dimensionality. Unlike Phase-Plane method and DP, it does not reduce the dimensionality of the problem and optimize all the variables in parallel. Optimizing all the unknown variables in parallel makes this method computationally expensive and slower. Another issue associated with DMOC is that it is very sensitive to initial conditions. Sometimes, a very small change in initial conditions may end up with a very large change in the final solution and might stuck in the local minima. Just to overcome this problem, the algorithm must be run multiple times with different possible initial values in order to get the global optimal solution. Curse of dimensionality and sensitive to initial conditions are the problems of DMOC that restrict its application

and makes it computationally expensive.

6 Simultaneous Path Planning and Trajectory Optimization

In classical methodologies of path optimization for robotic manipulator, path planning and trajectory optimization are performed separately to reduce the complexity and to increase the computational efficiency. Path planning and trajectory optimization are totally different domains. In first step, path planning is performed in the presence of static obstacles and then the results of path planning step are used as input for trajectory optimization. Mostly, the path planning step uses only the kinematics and the geometrical information of robotic manipulators. On the other hand, trajectory optimization only considers the dynamical model of robotic manipulators and does not handle the kinematic information.

These problems highlight the importance of solving path planning and trajectory optimization simultaneously. Simultaneously solving the path planning and trajectory optimization will incorporate the kinematics information and dynamical model for path planning as well as for trajectory optimization. The structure of path optimization problem in case of simultaneous path planning and trajectory optimization is shown in Figure 6.1.

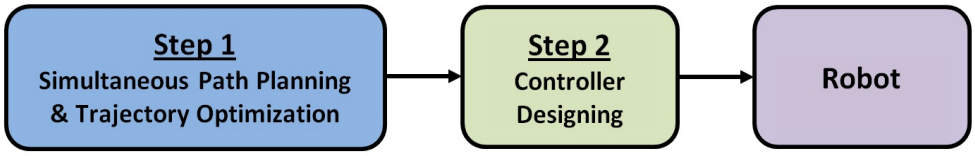


Figure 6.1: Structure of path optimization problem in case of simultaneous path planning and trajectory optimization.

If we have a look on the different optimization techniques discussed in Chapter 5, then one can analyze that DMOC can be used for this purpose. DMOC has the ability that it can optimize several variables in parallel while knowing only the starting point, ending point and the position of the obstacles. Although, DMOC is very sensitive to initial conditions and there is a possibility of finding local minima, but global optimal solution can be obtained after several runs with different initial conditions. The problem formulation for simultaneous path planning and trajectory optimization, constraints and the solution using DMOC are discussed in next sections.

6.1 Problem Formulation

In order to solve the path planning and trajectory optimization simultaneously, there are some requirements that must be known in order to proceed. These requirements

are given below:

- Limits of robotic manipulator's workspace
- Starting point, $\vec{P}_{0_x}^{init} = [P_{0_x}^{init}, P_{0_y}^{init}, P_{0_z}^{init}]$
- Ending point, $\vec{P}_{0_x}^{final} = [P_{0_x}^{final}, P_{0_y}^{final}, P_{0_z}^{final}]$
- Position of obstacles

These requirements are not new and commonly required for path planning.

Simultaneous path planning and trajectory optimization can be solved either in Cartesian space or in Joint space. Normally, position of obstacles and starting and ending points are given in terms of Cartesian space. So, it is preferred to solve simultaneous path planning and trajectory optimization in Cartesian space because it will be much easier to handle obstacle avoidance constraints. Although the same objective can also be obtained while working in the Joint space but in that case we have to use the inverse and forward kinematics formulations multiple times in order to avoid obstacles that will make it computationally expensive. Here $\vec{q} = [P_{0_x}\ P_{0_y}\ P_{0_z}] \in \mathbb{R}^3$ represents the configuration vector that is the position of the Tool Center Point (TCP) of robotic manipulator. As in simultaneous path planning and trajectory optimization we are performing two tasks, so the objective function in this case will also be consisting of two parts as given in (6.1).

$$J_T = L(C^E) + J_d \tag{6.1}$$

The first part of objective function represents the path planning phase and the main objective is to minimize the arc length from starting point to ending point, given in (6.2).

$$L(C^E) = \sum_{k=0}^{N-1} d(f(t_k), f(t_{k+1})) \tag{6.2}$$

In (6.2), C^E is the curve in the Euclidean space and from a partition $t_0 < t_1 < \ldots < t_f$ of the interval $[t_0,\ t_f]$ one can obtain a finite collection of points $f(t_0), f(t_1), \ldots, f(t_{N-1}), f(t_N)$ on the curve C^E. The term $d(f(t_k), f(t_{k+1}))$ calculates the distance between two consecutive points. The second part of objective function represents trajectory optimization phase and it can be any arbitrary cost function, defined in (6.3). There exists a relationship between the arc length, (6.2), and the second part of cost function, (6.3). The relationship between arc length and generalized coordinates can be found using inverse

or forward kinematics.

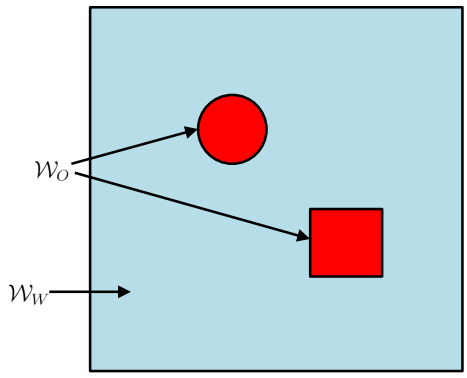
$$J_d(\vec{q}_d, \vec{u}_d) = \sum_{k=0}^{N-1} C_d(\vec{q}_k, \vec{q}_{k+1}, \vec{u}_k) \tag{6.3}$$

Obstacle avoidance is the essential part of path planning. The obstacle avoidance in path planning phase is obtained by incorporating the inequality constraints in optimization problem. The main purpose of these inequalities constraints is that the body of robotic manipulator should not touch the known obstacles. Let $\mathcal{W}_R \subset \mathbb{R}^3$ is the space occupied by the body of robotic manipulator, $\mathcal{W}_W \subset \mathbb{R}^3$ is the allowable workspace of robotic manipulator and $\mathcal{W}_O \subset \mathbb{R}^3$ is the space occupied by the obstacles. To avoid the collision with the known obstacle the following conditions must be satisfied.

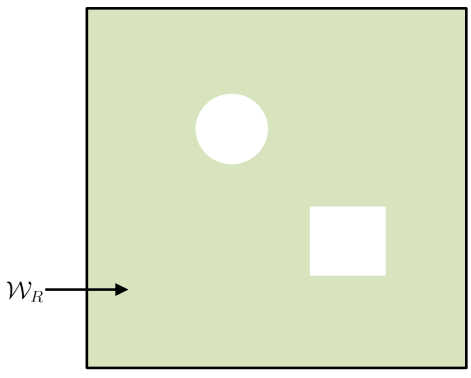
$$\mathcal{W}_O \subset \mathcal{W}_W \subset \mathbb{R}^3 \tag{6.4}$$

$$\mathcal{W}_R \in \{\mathcal{W}_W \setminus \mathcal{W}_O\} \in \mathbb{R}^3 \tag{6.5}$$

The graphical representation of these two constraints is shown in Figure 6.2. In this figure, obstacle avoidance conditions are shown for two dimensions but can be easily extended the same concept for three dimensions that is more practical for real world applications.



(a) Graphical representation of Condition-1 for obstacle avoidance, Equation (6.4)



(b) Graphical representation of Condition-2 for obstacle avoidance, Equation (6.5)

Figure 6.2: Graphical representation of obstacle avoidance constraints in two dimensions

6.2 Solution using Discrete Mechanics and Optimal Control (DMOC)

The problem formulation for simultaneous path planning and trajectory optimization is same as we have already discussed in Section 5.3.1. The discrete Lagrange-d'Alembert equation and terminal conditions would remain the same for this case as well. As described earlier, it would be computationally efficient to work in the Cartesian space, so the configuration vector will consist of the position of robotic manipulator in three dimensions, i.e. $\vec{q} = \vec{P}_0 = [P_{0_x}\ P_{0_y}\ P_{0_z}]$. Here $\vec{q} \in \mathbb{R}^3$ is the position of the Tool Center Point (TCP) of robotic manipulator.

Constraints on the joint actuators velocity and torque are the essential part for any optimal control problem. It is the advantage of DMOC that besides the constraints on joints' velocities and joints' torques, any arbitrary constraints can be easily incorporated in optimization problem. If we combine all the equality and inequality constraints then we have the following set of constraints:

$$\dot{q}_{i,min} \leq \dot{q}_i \leq \dot{q}_{i,max} \quad i = 1, 2, 3. \tag{6.6}$$

$$\tau_{i,min} \leq \tau_i \leq \tau_{i,max} \quad i = 1, 2, 3. \tag{6.7}$$

$$D_2L_d(\vec{q}_{k-1}, \vec{q}_k) + D_1L_d(\vec{q}_k, \vec{q}_{k+1}) + \vec{f}^{+}_{k-1} + \vec{f}^{-}_{k} = 0,\ k = 1, 2, \ldots, N \tag{6.8}$$

$$D_2L(\vec{q}^0, \dot{\vec{q}}^0) + D_1L_d(\vec{q}_0, \vec{q}_1) + \vec{f}^{-}_{0} = 0 \tag{6.9}$$

$$-D_2L(\vec{q}^{t_f}, \dot{\vec{q}}^{t_f}) + D_2L_d(\vec{q}_{N-1}, \vec{q}_N) + \vec{f}^{+}_{N-1} = 0 \tag{6.10}$$

$$\vec{q}^{tf} - \vec{q}_N = 0 \tag{6.11}$$

$$\vec{q}^0 - \vec{q}_0 = 0 \tag{6.12}$$

$$\mathcal{W}_O \subset \mathcal{W}_W \subset \mathbb{R}^3 \tag{6.13}$$

$$\mathcal{W}_R \in \{\mathcal{W}_W \setminus \mathcal{W}_O\} \in \mathbb{R}^3 \tag{6.14}$$

In above set of equations, N represents the number of discretized points between starting and ending points.

6.2.1 Practical Implementation

For the sake of compromise between accuracy and efficiency, the midpoint rule is used to approximate the discrete cost function C_d, discrete Lagrangian L_d, and the discrete forces. The discretized terms are given below:

$$C_d(\vec{q}_k, \vec{q}_{k+1}, \vec{u}_k) = hC\left(\frac{\vec{q}_{k+1} + \vec{q}_k}{2}, \frac{\vec{q}_{k+1} - \vec{q}_k}{h}, \vec{u}_k\right) \tag{6.15}$$

$$L_d(\vec{q}_k, \vec{q}_{k+1}) = hL\left(\frac{\vec{q}_{k+1} + \vec{q}_k}{2}, \frac{\vec{q}_{k+1} - \vec{q}_k}{h}\right) \tag{6.16}$$

$$\vec{f}_k^- = \vec{f}_k^+ = \frac{h}{2} f\left(\frac{\vec{q}_{k+1} + \vec{q}_k}{2}, \frac{\vec{q}_{k+1} - \vec{q}_k}{h}, \vec{u}_k\right) \tag{6.17}$$

From the discretized equations, we can identify seven variables that have to be optimized in this simultaneous path planning and trajectory optimization problem. These seven unknown variables are the position of the TCP in three dimensions or configuration vector, i.e., $\vec{q} = \vec{P}_0 = [P_{0_x}\ P_{0_y}\ P_{0_z}]$, torques of joints' actuators, $\vec{u} = [u_1\ u_2\ u_3]$, and the time step, h.

$$\text{Optimization Variables} = [P_{0_x}\ P_{0_y}\ P_{0_z}\ u_1\ u_2\ u_3\ h] \tag{6.18}$$

In DMOC initial conditions play a vital role in the final optimal solution and it is necessary that initial conditions must be properly chosen. As we have the position of the TCP in the list of the variables that have to be optimized, so we can provide a suboptimal path as initial conditions. This suboptimal path as initial condition will increase the computational efficiency and solution will converge more quickly. If we give some wrong initial guesses, the problem will take long time to converge and it might possible to stuck in local minima.

The above discussed optimization problem can be solved by any nonlinear optimization tool. In this thesis, we are using SQP to solve DMOC. It is very easy to incorporate aforementioned constraints and any other arbitrary constraints in SQP.

6.3 Numerical Examples

In this section, we will discuss two numerical examples to show the applicability of simultaneous path planning and trajectory optimization using DMOC and the results are compared with state-of-the-art techniques. For a fair comparison, the problem is also solved by conventional method in which path planning and trajectory optimization are performed separately. To obtain solution using conventional method, path planning is performed using Genetic Algorithm (GA) and the trajectory optimization is performed using Dynamic Programming [96], because DP programming always gives the global optimal solution.

The dynamical model and the geometrical parameters of Delta parallel D4-500 robot, discussed in Chapter 3, are considered to solve these numerical examples. As already explained in Chapter 4, due to restriction imposed by the forearms and the body, Delta parallel robot can only be used for path planning in two dimensions ($2D$). But the method discussed in this section could be easily used for simultaneous path planning and trajectory optimization for any type of robotic manipulator in $3D$.

SQP is used to solve this nonlinear optimization problem. In MATLAB, we used `fmincon` command for this purpose. By using this command in MATLAB, we can easily incorporate any linear, nonlinear, equality and inequality constraints.

6.3.1 Example 1

In first numerical example, path planning and trajectory optimization from given initial point to final point is performed in the presence of obstacle. The initial and final positions are set to be $[P_{0_x}^{init}, P_{0_y}^{init}, P_{0_z}^{init}] = [-0.08, -0.08, -0.34042]$ and $[P_{0_x}^{final}, P_{0_y}^{final}, P_{0_z}^{final}] = [0.08, 0.08, -0.34042]$, respectively. The initial and final positions are chosen to be at the extreme corner positions in order to use the maximum workspace of D4-500 Delta parallel robot. A rectangle with coordinates $[0, 0], [0.02, 0], [0.02, 0.02]$, and $[0, 0.02]$ is defined as an obstacle. The workspace with initial point, final point, obstacle and initial path guess is shown in Figure 6.3.

The number of discretized points between initial point and final point are considered to be 10, i.e. $N = 10$. The cost function considered in this numerical example is the arc length and the travelling time from starting point to ending point while avoiding the obstacle, given in (6.19).

$$J_T = \sum_{k=0}^{N-1} d(f(t_k), f(t_{k+1})) + \sum_{k=0}^{N-1} \Delta t(k) \tag{6.19}$$

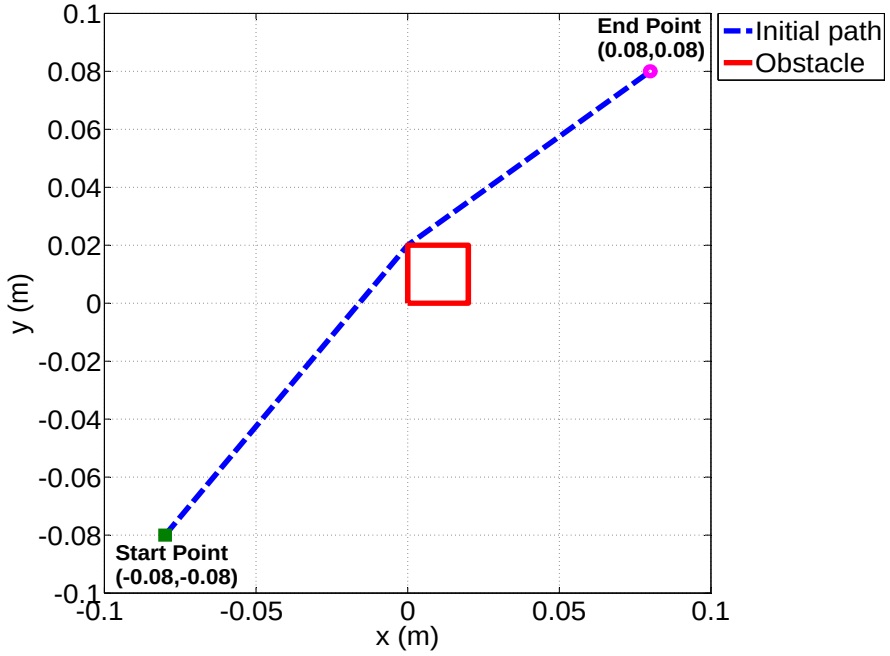


Figure 6.3: Workspace considered for Example-1 with initial position, final position, initial path and obstacle

First, the path optimization problem is solved in a conventional way. GA is used to find a path from initial point to final point in the presence of obstacle. The length of the obtained path using GA is 0.2307 m and the obtained path is shown in Figure 6.5. In the next step, we used the obtained path as an input for the trajectory optimization step. Dynamic Programming using path parameter, discussed in Section 5.2.1, is used for the trajectory optimization purpose. The optimized value of cost function is 0.1689 sec.

For comparison purpose, DMOC is used for simultaneous path planning and trajectory optimization. As we have already discussed, DMOC is very sensitive to initial conditions and very small change in initial values may end up with a totally different solution. In order to show the sensitivity of DMOC on initial values, the problem is solved using sets of different initial values. The set of different initial values for joints actuators' torques ($\vec{\tau} = [\tau_1\ \tau_2\ \tau_3]$) and time step ($h$) are given in Table 6.1. Initially, these values of torques and time step are assigned to every node, i.e. $\tau_i[k] = \tau_i,\ k = 1, 2, \ldots, N,\ i = 1, 2, 3$ and similarly $h[k-1] = h,\ k = 2, 3, \ldots, N$. The initial guess for configuration vector or position of TCP ($\vec{q} = \vec{P}_0 = [P_{0_x}\ P_{0_y}\ P_{0_z}]$) is same for all sets of initial values and is shown in Figure 6.3. The final paths as a result of these initial values are shown in Figure 6.4. In this figure, one can see that there is very small change in the time step in the initial value Set-1, Set-2 and Set-3. However, the final obtained paths are totally different from each other. This shows the sensitivity of the DMOC on initial values. In order to get the global optimal solution, the problem must be solved with all possible combinations of initial values and finally the global optimal solution can be selected by analysing the solutions obtained by sets of different initial values.

Table 6.1: Sets of different initial values to solve Example-1 using DMOC

No.	τ_1 **(Nm)**	τ_2 **(Nm)**	τ_3 **(Nm)**	**h (sec)**
Initial values 1	20	20	20	0.1
Initial values 2	20	20	20	0.25
Initial values 3	20	20	20	0.5
Initial values 4	30	30	30	0.25
Initial values 5	25	25	25	0.25

After analysing the solutions obtained by using sets of different initial values, it is figured out that Set-2 of initial values gives the global optimal solution for Example-1. The obtained path is shown in Figure 6.5 and the arc length of the path found out to be 0.2308 m. The absolute difference between the arc lengths obtained by GA and DMOC is 1×10^{-4} m. This small difference between arc lengths shows the applicability of DMOC for path planning. The optimized travelling time from starting point to ending point using DMOC is calculated to be 0.1693 sec. If we compare the opti-

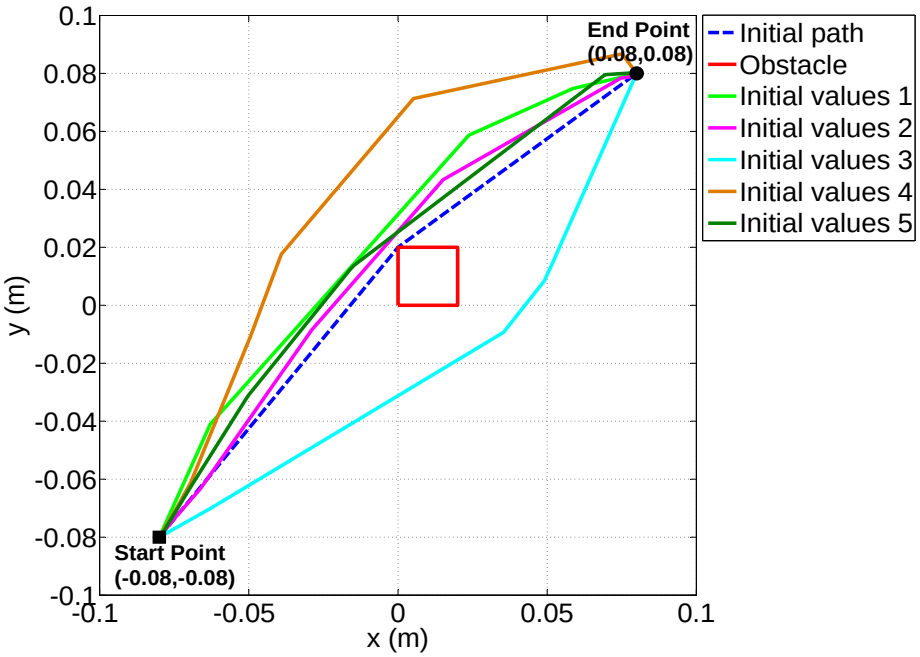


Figure 6.4: Solution obtained by sets of different initial values for Example-1

mized travelling time obtained using DMOC with the result obtained using DP, then the absolute difference is found to be 4×10^{-4} sec.

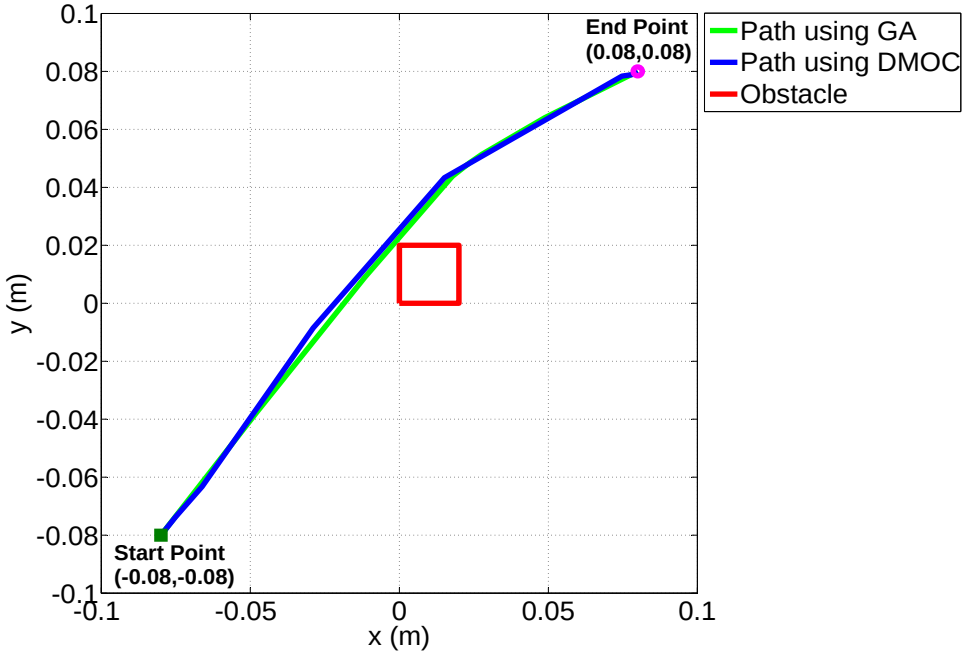


Figure 6.5: Path for Example-1 obtained using GA and DMOC

The numerical results obtained for path optimization problem using conventional method and DMOC are outlined in Table 6.2.

Table 6.2: Comparison of the optimal solutions for path optimization problem using different optimization techniques (Example 1)

Path Planning and Optimization Technique	**Arc Length**	**Minimum Time**
Optimal Solution	0.2281 m	0.1381 sec
Genetic Algorithm (GA) and Dynamic Programming [96]	0.2307 m	0.1689 sec
Discrete Mechanics and Optimal Control (DMOC)	0.2308 m	0.1693 sec

6.3.2 Example 2

In Example 2, just like Example 1, the starting and ending points for simultaneous path planning and trajectory optimization are $[-0.08, -0.08, -0.34042]$ and $[0.08, 0.08, -0.34042]$, respectively. In this example, a circle of radius 0.03 m with center at origin $[0, 0]$ is used as an obstacle. The workspace with initial point, final point, obstacle and initial path guess is shown in Figure 6.6.

The number of discretized points between initial point and final point are considered to be 10, i.e. $N = 10$. The cost function is the arc length and the travelling time from starting point to ending point while avoiding the obstacle, given in (6.19).

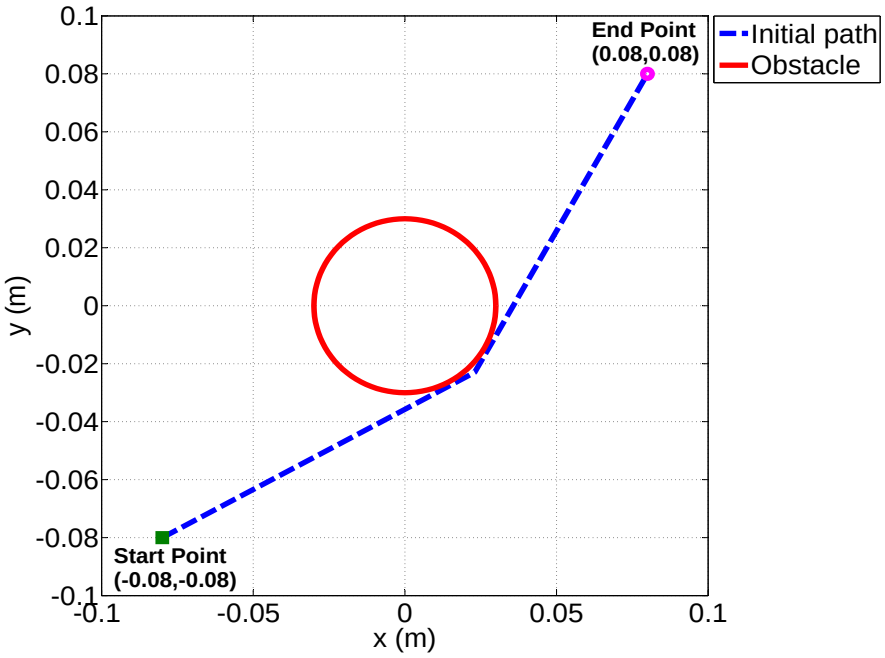


Figure 6.6: Workspace considered for Example-2 with initial position, final position, initial path and obstacle

The results for Example-2 using conventional method and using DMOC are outlined in Table 6.3. First, the problem is solved by the conventional method in which path planning and trajectory optimization is performed separately. For path planning purpose, GA is used. The arc length of the path obtained using GA, shown in Figure 6.8, is 0.2406 m. The next step is to optimize the obtained path. For this purpose, Dynamic Programming using path parameter is used as this method is computationally efficient and always gives the global optimal solution. The optimized travelling time using DP from starting point to ending point is 0.1730 sec.

Table 6.3: Comparison of the optimal solutions for path optimization problem using different optimization techniques (Example 2)

Path Planning and Optimization Technique	**Arc Length**	**Minimum Time**
Optimal solution	0.2354 m	0.1365 sec
Genetic Algorithm (GA) and Dynamic Programming [96]	0.2406 m	0.1730 sec
Discrete Mechanics and Optimal Control (DMOC)	0.2405 m	0.1762 sec

For comparison, the same problem is solved using DMOC and path planning and trajectory optimization steps are performed simultaneously. Just like Example-1, in order to show the sensitivity of DMOC on initial values, the problem is solved with sets of different initial values. The set of different initial values for joints actuators' torques ($\vec{\tau} = [\tau_1\ \tau_2\ \tau_3]$) and time step ($h$) are given in Table 6.4. Initially, these values of torques and time step are assigned to every node, i.e. $\tau_i[k] = \tau_i,\ k = 1, 2, \ldots, N,\ i = 1, 2, 3$ and similarly $h[k-1] = h,\ k = 2, 3, \ldots, N$. The initial guess for configuration vector or position of TCP ($\vec{q} = \vec{P}_0 = [P_{0_x}\ P_{0_y}\ P_{0_z}]$) is same for all sets of initial values and is shown in Figure 6.6. The final paths as a result of these initial values are shown in Figure 6.7. If we compare the Set-1 and Set-3 of initial values then there is a difference of only 0.02 sec in step size. As a result of this small change, the final obtained paths, shown in Figure 6.7 are totally different. In order to get the global optimal solution, the problem must be solved with all possible combinations of initial values and finally the global optimal solution can be selected by analysing the solutions obtained by sets of different initial values.

After analysing the solutions obtained by using sets of different initial values, it is figured out that Set-2 of initial values gives the global optimal solution for Example-2. The obtained path is shown in Figure 6.8 and the arc length of the path found to be 0.2405 m. If we calculate the absolute difference between the arc length of the paths obtained by DMOC and GA, then it is just 1×10^{-4} m. This small absolute difference in arc length shows the applicability of DMOC for path planning and it can be compared

Table 6.4: Set of different initial values to solve Example-2 using DMOC

No.	τ_1 **(Nm)**	τ_2 **(Nm)**	τ_3 **(Nm)**	**h (sec)**
Initial values 1	10	10	10	0.02
Initial values 2	15	15	15	0.02
Initial values 3	10	10	10	0.04
Initial values 4	15	15	15	0.07
Initial values 5	30	30	30	0.25

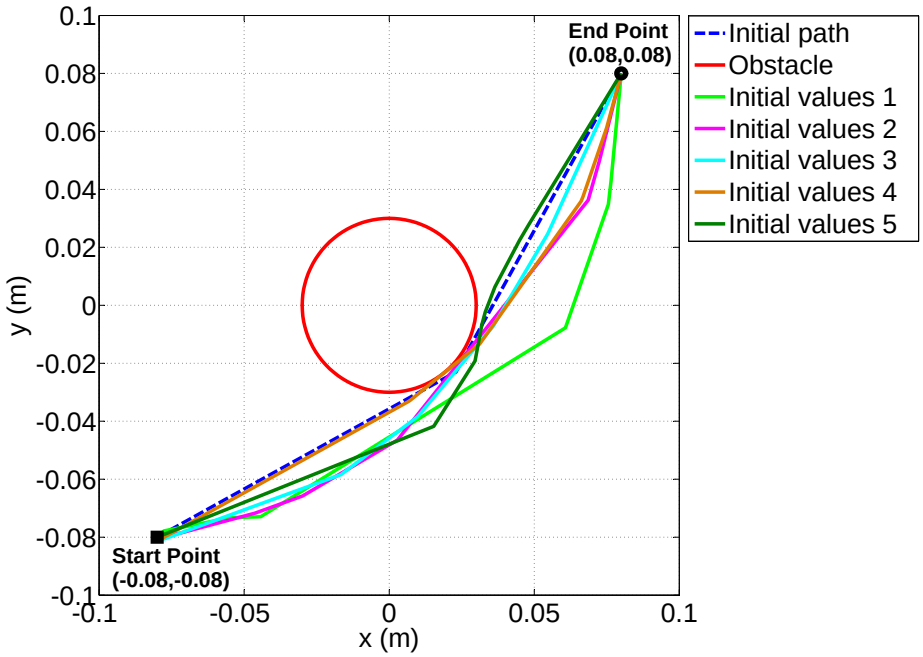


Figure 6.7: Solution obtained by sets of different initial values for Example-2

with any state-of-the-art method. DMOC not only find the path from starting point to ending point but also optimized the trajectory with respect to the given objective function. In this example, the objective function was travelling time and the optimized value found to be 0.1762 sec. The absolute difference between the optimized time using DMOC and Dynamic Programming is 3.2×10^{-3} sec. This small absolute difference in arc length and optimal time obtained using conventional path optimization technique and using proposed methodology shows the applicability of the proposed methodology.

In this section, we discussed the two numerical examples solved by conventional methods and the proposed method in which path planning and trajectory optimization is performed simultaneously. For comparison purpose, the obtained paths by these two methods are also shown in parallel. The numerical analysis and statistics show the

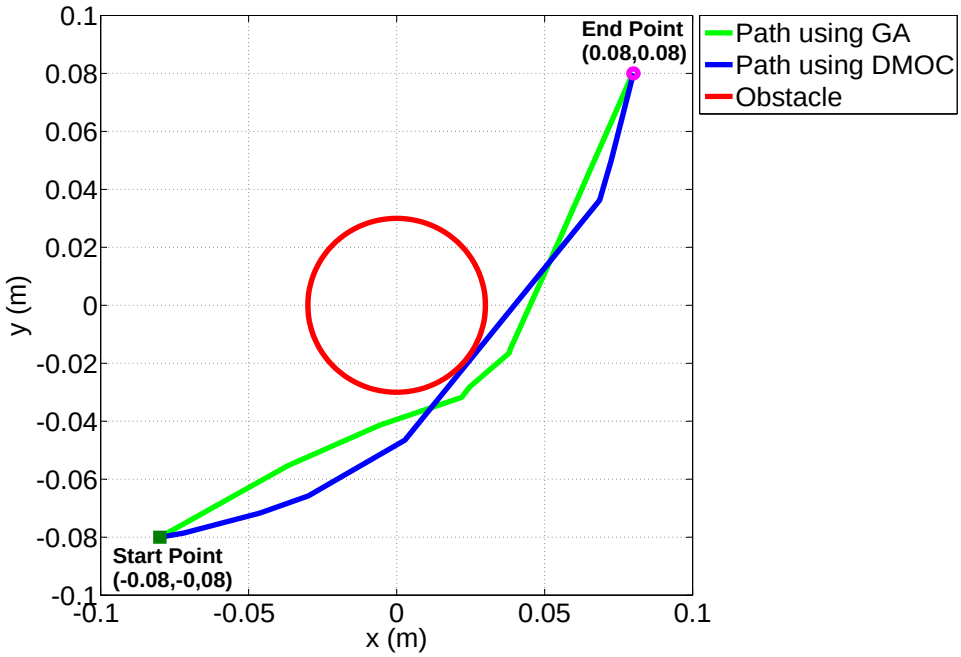


Figure 6.8: Path for Example-2 obtained using GA and DMOC

applicability of the proposed method.

6.4 Comparison of Simultaneous Path Planning and Trajectory Optimization with Conventional Methods

Simultaneous Path Planning and Trajectory Optimization (SPPTO) is a new method that brings the path planning step and trajectory optimization step together on a single platform. Previously, path planning and trajectory optimization steps were performed separately and considered to be completely separate domains and researchers do not know the requirements and the limitations of other domains. This method will give a better insight to the researchers in both steps so that researchers can better understand the limitations and requirements of the problem and can produce more practicable results.

Normally in path planning techniques, discussed in Chapter 4, only the kinematics of the robotic manipulators are considered and the dynamics are ignored. It is one of the biggest advantages of simultaneous path planning and trajectory optimization that information about the dynamics of robotic manipulator are also incorporated in the path planning step that give more practicable and realistic results.

Number of variables dependencies play a vital role in any optimization technique. In case of simultaneous path planning and trajectory optimization, there are $n+4$ variables that have to be optimized in parallel. Here n represents the number of joints actuators' torques. The other four variables are the position of robotic manipulator in three dimensions ($\vec{P}_0 = [P_{0_x}\ P_{0_y}\ P_{0_z}]$) and the time step ($h$) between any two discretized points.

In case of N discretized nodes, there would be total $(n + 4) \cdot N$ points that have to be optimized. If we compare the simultaneous path planning and trajectory optimization using DMOC with Dynamic Programming in terms of variable dependencies, then it would not be a fair comparison, as in Dynamic Programming one only optimizes the predefined geometrical path by optimizing a single variables, as discussed in Section 5.2.1 and Section 5.2.2. On the other hand, in the newly developed methodology we are not only optimizing the trajectory but also doing path planning in parallel.

Computational efficiency of any optimization technique also depends upon the number of discretized points, N, that plays a very important role. Just like Dynamic Programming or Phase-Plane method, the accuracy of the result and computational efficiency of simultaneous path planning and trajectory optimization is also dependent on the number of discretized points N. By increasing the number of discretized points, one can get the better numerical accuracy and numerical results would be more close to the continuous time results but off course at the cost of computational time.

In conventional methods of path optimization, in which path planning and trajectory optimization is performed separately, there is no need of initial conditions and the boundary conditions are also very well defined. On the other hand, the proposed simultaneous path planning and trajectory optimization technique is very sensitive to initial conditions. The proposed method is not only dependent on the terminal conditions but also on the initial guess or initial conditions of the variables that have to be optimized and we have to provide the initial guess for each and every variable at each discretized node that makes it more complex. As already discussed in Section 6.3, a very small change in initial conditions may end up with a totally different result. So, one has to select the initial conditions very intelligently or to solve the problem with all possible initial conditions, if possible.

In this chapter, we presented a novel idea to combine path planning and trajectory optimization steps for general path optimization problem. This proposed method is not limited to Delta parallel robot and can easily be applied to any type of robotic manipulator. Effectiveness and applicability of the proposed methodology is shown by two numerical examples implemented on Delta parallel robot.

7 Summary and Outlook

This chapter summarizes the primary results obtained in this thesis and suggests some directions for future research work.

7.1 Summary and Outlook

Robots are playing a very important role not only in industries but also in our daily life applications. The increasing use of robotic manipulators attracts the attention of researchers in this field. The overall efficiency of any industry can be increased by finding the optimal solution for robotic manipulators in terms of travelling time and energy consumption to increase the production line and to reduce the energy consumption.

The primary contribution of this thesis is to perform path planning and trajectory optimization using newly developed methodologies and to compare the results with existing state-of-the-art techniques. Methodologies are compared not only in terms of numerical solutions but also in terms of computational efficiency, variable dependencies and possibility to find local minima. One of the major contributions is to perform path planning and trajectory optimization in a single step. Previously, these two steps were performed separately for the purpose of tractability.

Path planning and trajectory optimization techniques that are discussed in this thesis are generally applicable and can be applied to all types of robots. In this thesis, Delta parallel robot is considered in the numerical examples of path planning and trajectory optimization. One of the main reasons is that it was available in our lab to check the practicability of different methodologies. Second reason was to have some realistic dynamical data and kinematic parameters instead of assuming any unrealistic robotic parameters. In Chapter 3, the forward and inverse kinematics of Delta parallel robot are discussed that are important for path planning. The dynamical model of Delta parallel robot is derived using virtual work principle that states that input of all inertial forces must be equal to the contribution of all non-inertial forces. Some hypotheses are considered to get a dynamical model that can be used for real time calculations.

Path planning is the first step of generalized solution of path optimization of robotic manipulators. In Chapter 4, path planning is discussed using Probabilistic Roadmap (PRM) and Genetic Algorithm (GA). Although PRM is a complete planner and computationally efficient that can be used for n joint robotic manipulator, but post processing of obtained path and the restriction to incorporate the arbitrary constraints raise a question on its usability. On the other hand, GA gives the option to incorporate arbitrary constraints but it is applicable to robotic manipulators with low number of joints.

In Chapter 5, trajectory optimization for a predefined geometrical path for robotic

manipulators is discussed using three different techniques: Phase-Plane Method, Dynamic Programming and Discrete Mechanics and Optimal Control (DMOC). Phase-Plane method is computationally efficient, always gives the global optimal solution and follows the Bellman's principle of optimality. However, this method is only limited for time minimization problems that restrict its usability. For the purpose of multi-objective optimization and to add additional arbitrary constraints, Dynamic Programming and a newly developed methodology called '*Discrete Mechanics and Optimal Control (DMOC)*' are used. Dynamic Programming is a very useful optimization technique that can be used for multi-objective optimization and can handle any arbitrary constraints. As recently developed robots are complex and have high Degrees of Freedom, so it is computationally expensive to solve the systems with high order dynamics. The other way to handle this problem, adopted in this thesis, is dimensionality reduction of the problem using path parameter that increased the computational efficiency. A heuristic based joint selection criterion is proposed to increase the computational efficiency and to guarantee the global optimal solution while using Dynamic Programming in joint space for trajectory optimization. Third trajectory optimization technique discussed in this thesis is DMOC, based on discrete variational mechanics and the constraints for optimization problem are obtained by discrete Lagrange-D'Alembet principle and Legendre transformation. DMOC has some advantages over classical approaches for optimal control problem, e.g., preservation of momentum maps and conservation of modified energy. In this thesis, DMOC is used to find optimal trajectory for predefined geometrical path of Delta parallel robot. Using DMOC for parallel robots is more complex compared to serial robots because of analytical difficulty presented by the joint variable interdependencies and the complex relationship between Cartesian space and Joint space. DMOC is a Lagrange based optimization method which requires the information of system's kinetic energy and potential energy. The kinetic and potential energy equations of Delta parallel robot, Appendix A.1.2, use the information of joints' angles as well as the coordinate of TCP. In order to solve the simultaneous path planning and trajectory optimization, the energy equations must be converted to Cartesian coordinates which requires the forward kinematics. The conversion of energy equations into Cartesian coordinates makes DMOC computationally expensive to use for simultaneous path planning and trajectory optimization. Aforementioned optimization techniques are compared not only in terms on numerical solution but also in terms of some other properties like, computational efficiency, possibility to stuck in local minima, sensitivity to initial conditions and dependencies on the number of variables.

The major contribution of this thesis is the novel idea about combining the path planning and trajectory optimization in a single step, discussed in Chapter 6. By combining path planning and trajectory optimization into a single step the dynamics of robot are incorporated not only in trajectory optimization but also in path planning step. The other advantage is that all constraints, either on kinematics properties or on dynamical properties, are considered in both steps that give more practicable results. In proposed

general solution of path optimization problem for robotic manipulator, optimal joints' torques, optimal joints' velocities and the position of the TCP of robotic manipulator are calculated in parallel. For this purpose, DMOC is used to solve the optimization problem, because in DMOC the search for optimal solution can be started with initial guesses and multiple variables can be optimized in parallel. The applicability of the proposed novel idea is discussed by two numerical examples and the obtained results are compared with the conventional approach's result in which path planning and trajectory optimization is performed separately.

Different techniques of path planning and trajectory optimization, discussed in this thesis, can be used in general and these techniques are not limited to Delta parallel robot only. Similarly, combining path planning and trajectory optimization in a single step is also a general idea for path optimization problem of robotic manipulators. This idea can be used for any type of robotic manipulator.

7.2 Future Work

In this thesis, we discussed path planning and trajectory optimization for robotic manipulators using different techniques. The practicability and the comparison among different techniques are discussed by numerical examples. Besides that there is a room for further improvements and future work.

In this thesis, DMOC is used for trajectory optimization as well as for simultaneous path planning and trajectory optimization. One of the major problems in using DMOC is the curse of dimensionality. In DMOC, all the unknown variables are optimized in parallel that increases the computational time and complexity. A further research can be carried out in the area of dimensionality reduction for DMOC. This will increase the computational efficiency as well as will reduce the possibility of finding local minima. As a result of dimensionality reduction, DMOC will converge more quickly to optimal solution.

In this thesis, we presented a joint selection criterion to ensure the global optimal solution while using Dynamic Programming in joint space. Unfortunately, it is a heuristic based selection criterion and till now there is no mathematical proof for it. One can further investigate this criterion and can give the mathematical proof.

Although a significant amount of research has been done in field of trajectory optimization of robotic manipulators but still we are lagging in online trajectory optimization techniques. Currently, there is no technique for online trajectory optimization and all the existing techniques are offline. This area requires investigation and research that would be beneficial.

A Appendix

In this Appendix, different algorithms and derivations are presented for the better understanding for different techniques presented in this thesis.

In Appendix A.1, the derivation for the dynamical model of Delta parallel robot is presented using the virtual work principle approach presented by [3]. Besides dynamical model, the kinetic and potential energy equations of Delta parallel robot are also formulated that are necessary for Lagrangian based trajectory optimization techniques.

Algorithms for path planning using PRM and GA are presented in Appendix A.2. These algorithms give better understanding of the path planning methodologies and also help in implementation in computer simulations.

In Appendix A.3 we have presented the algorithms for trajectory optimization of robotic manipulators using dynamic programming.

A.1 Dynamical Model and Energy Equations for Delta Parallel Robot

In this appendix we will present the dynamical model and the kinetic and potential energy equations for Delta parallel robot.

A.1.1 Dynamical Model of Delta Parallel Robot

Dynamical model of robotic manipulators play an important role in trajectory optimization. The main difficulty in calculating dynamical model is that it must be realizable and can be calculated in real-time. Just like kinematics, the dynamical model of parallel manipulators is more complex as compared to the serial manipulators.

One simple way to solve the dynamical model of parallel manipulators is the closed-chain at passive joints. This simplification provides relaxation in the closure conditions. The final dynamical model will be the sum of all individual robots thus created. Kleinfinger [140] uses this technique and applied the Lagrange multipliers to calculate the dynamical model. Another way to derive the dynamical model is to use the Lagrange-d'Alembert virtual work principle. Dynamical model using Lagrange-d'Alembert virtual work principle has been successfully derived by Kokkinis and Stoughton [141], Nakamura [142], and Wang and Chen [143].

Another very useful method to derive the dynamical model of robotic is based on virtual work principle. According to this principle, the contribution of all inertial forces must be equal to the contribution of all non-inertial forces. This method simplifies the problem and is equally efficient for both serial as well as parallel robots [129, 144, 145]. By incorporating all the masses, inertia and non-linearities, the dynamical model

becomes very complicated and computationally expensive that cannot be used for real-time applications. A practicable dynamical model for real-time applications can be derived by neglecting the masses and inertia of the legs, proposed by Ji [146]. In this thesis, we will also derive the dynamical model using virtual work principle presented by Codourey [3].

A.1.1.1 Simplifying Hypothesis

Before going to derive dynamical model, there are few hypothesis to make the model practicable and computationally efficient. The simplifying hypothesis are:

1) The rotational inertia of forearm are neglected.

2) Friction effects and elasticity are neglected.

3) For analytical purposes, the masses of forearm are optimally divided into two parts and placed at two extremities. It is placed $1/3$ at its lower extremity (travelling plate) and $2/3$ at its upper extremity (elbow).

Due to these hypotheses, Delta parallel robot is reduced to 4 body parts: the three upper arms and one travelling plate. This reduction of Delta parallel robot into 4 body parts makes the calculation simple and understandable.

A.1.1.2 Dynamic Parameters

The dynamical model of the robot consists of several dynamic parameters. The intermediate terms are defined to simplify the dynamical model representation. First, the position of the center of the mass of the arm is given by (A.1).

$$CoM = L_A \frac{\frac{1}{2}m_a + m_{elbow} + 2\,\frac{2}{3}\,m_b}{m_a + m_{elbow} + 2\,\frac{2}{3}\,m_b} \tag{A.1}$$

In (A.1), r is chosen to be $2/3$ for an optimal distribution of the mass as explained in hypothesis. The second important dynamical parameter is inertia matrix in joint space, given in (A.2)

$$\mathbf{I}_b = \begin{bmatrix} I_{b_1} & 0 & 0 \\ 0 & I_{b_2} & 0 \\ 0 & 0 & I_{b_3} \end{bmatrix} \tag{A.2}$$

In (A.2), I_{b_i} is the inertia of the each arm that is the sum of the inertia of motor I_m and

the arm. It is given by (A.3).

$$I_{bi} = I_m + L_A^2 \left(\frac{m_a}{3} + m_{elbow} + 2\ r\ m_b \right) \tag{A.3}$$

These dynamic parameters are used to derive the dynamical model of the Delta parallel robot. The values of different dynamic coefficients and geometrical parameters used in these dynamic parameters can be found in Table 3.1.

A.1.1.3 Dynamical Model based on Virtual Work Principle

As explained earlier, virtual work principle based on the assumption that the contribution of all inertial forces must be equal to the contribution of all non-inertial forces. In Delta parallel robot, two kinds of forces act on the travelling plate; the gravity force $\vec{F}_g$ and the inertial force $\vec{F}_{in}$.

$$\vec{F}_g = m_{nt} [0 \ \ 0 \ \ -g]^T \tag{A.4}$$

$$\vec{F}_{in} = m_{nt} \vec{\ddot{P}}_0 \tag{A.5}$$

The contribution of these two forces towards each motor can be calculated by multiplying it with Jacobian matrix as shown in (A.6) and (A.7), respectively.

$$\vec{\tau}_{F_{in}} = \mathbf{J} \vec{F}_{in} = \mathbf{J}\ m_{nt}\ \vec{\ddot{P}}_0 \tag{A.6}$$

$$\vec{\tau}_{F_g} = \mathbf{J} \vec{F}_g = \mathbf{J} m_{nt} [0 \ \ 0 \ \ -g]^T \tag{A.7}$$

In above equations, $\mathbf{J}$ is the Jacobian matrix that we have already calculated in (3.21) and m_{nt} is the total mass as given in (A.8). Mass of the payload attached to the end-effector can also be incorporated in the total mass.

$$m_{nt} = m_c + m_{payload} + 6(1 - r) m_b \tag{A.8}$$

Now we can apply virtual work principle and the application of this principle at the joint level will lead to the following equation:

$$\begin{aligned} &\vec{\tau} + \mathbf{J}^T \vec{F}_g + \vec{\tau}_{Gb} = \mathbf{I}_b \vec{\ddot{\theta}} + \mathbf{J}^T \vec{F}_{in} \\ \Rightarrow\ &\vec{\tau} = \mathbf{I}_b \vec{\ddot{\theta}} + \mathbf{J}^T m_{nt} \vec{\ddot{P}}_0 - \mathbf{J}^T \vec{F}_g - \vec{\tau}_{Gb} \end{aligned} \tag{A.9}$$

In (A.9), $\vec{\tau}$ is the vector of torques acting on the joint actuators for maneuvering, and $\vec{\tau}_{Gb}$ is the torques due to the gravitational forces of the arm and it is given by (A.10).

$$\vec{\tau}_{Gb} = g\ CoM(m_a + m_{elbow} + 2\ r\ m_b)\,[\cos(\theta_1)\ \cos(\theta_2)\ \cos(\theta_3)]^T \tag{A.10}$$

In (A.9), the only two unknowns are $\vec{P}_0$ and $\vec{\ddot{\theta}}$. These two terms are linked together and this link can be calculated by taking the time derivative of (3.20), as given by (A.11).

$$\vec{\ddot{P}}_0 = -\begin{bmatrix} \vec{s}_1^T \\ \vec{s}_2^T \\ \vec{s}_3^T \end{bmatrix}^{-1} \left(\begin{bmatrix} \vec{\dot{s}}_1^T \\ \vec{\dot{s}}_2^T \\ \vec{\dot{s}}_3^T \end{bmatrix} \mathbf{J} + \mathbf{K} \right) \vec{\dot{\theta}} + \mathbf{J}\,\vec{\ddot{\theta}} \tag{A.11}$$

Here $\mathbf{K}$ is defined as below:

$$\mathbf{K} = \begin{bmatrix} \vec{\dot{s}}_1^T \vec{b}_1 + \vec{s}_1^T \vec{\dot{b}}_1 & 0 & 0 \\ 0 & \vec{\dot{s}}_2^T \vec{b}_2 + \vec{s}_2^T \vec{\dot{b}}_2 & 0 \\ 0 & 0 & \vec{\dot{s}}_3^T \vec{b}_3 + \vec{s}_3^T \vec{\dot{b}}_3 \end{bmatrix} \tag{A.12}$$

In (A.12), $\vec{\dot{b}}_i$ is the time derivative of the term $\vec{b}_i$ that is given in (3.19).

$$\vec{\dot{b}}_i = \begin{bmatrix} L_A \cos(\theta_i) \\ 0 \\ -L_A \sin(\theta_i) \end{bmatrix} \dot{\theta}_i, \quad i = 1, 2, 3.$$

Substitution of all these terms into (A.9) will lead to the final equation as given below:

$$\vec{\tau} = \mathbf{I}_b\,\vec{\ddot{\theta}} + \mathbf{J}^T\, m_{nt}\,(\mathbf{J}\,\vec{\ddot{\theta}} + \dot{\mathbf{J}}\,\vec{\dot{\theta}}) - \mathbf{J}^T \vec{F}_g - \vec{\tau}_{Gb} \tag{A.13}$$

$$= (\mathbf{I}_b + m_{nt}\mathbf{J}^T\mathbf{J})\,\vec{\ddot{\theta}} + \mathbf{J}^T m_{nt}\dot{\mathbf{J}}\,\vec{\dot{\theta}} + (-\mathbf{J}^T \vec{F}_g - \vec{\tau}_{Gb}) \tag{A.14}$$

From (A.13), we can easily identify the mass matrix $\mathbf{M}(\theta)$, Centrifugal and Coriolis coefficient matrix $\mathbf{C}(\theta, \dot{\theta})$, and the vector of Gravity terms $\vec{G}(\theta)$ by comparing it with standard dynamical equation given as (A.15).

$$\vec{\tau} = \mathbf{M}(\theta)\,\vec{\ddot{\theta}} + \mathbf{C}(\theta, \dot{\theta})\,\vec{\dot{\theta}} + \vec{G}(\theta), \tag{A.15}$$

where

$$\mathbf{M}(\theta) = \mathbf{I}_b + m_{nt}\mathbf{J}^T\mathbf{J}$$

$$\mathbf{C}(\theta, \dot{\theta}) = \mathbf{J}^T m_{nt}\dot{\mathbf{J}}$$

$$\vec{G}(\theta) = -\mathbf{J}^T\vec{F}_g - \vec{\tau}_{Gb}$$

A more efficient method for calculating the dynamical model could be to use (A.9) directly and calculating $\vec{\ddot{P}}_0$ and $\vec{\ddot{\theta}}$ using numerical differentiation, instead of calculating $\dot{\mathbf{J}}\vec{\dot{\theta}}$ that is computationally expensive. This is not done because, calculating $\vec{\ddot{P}}_0$ and $\vec{\ddot{\theta}}$ by numerical differentiation may lead to the numerical noise especially when the accelerations of the robot are small.

A.1.2 Energy Equations for Delta Parallel Robot

The kinetic and potential energy equations play an important role either to derive the dynamical model using Lagrangian or to optimize a trajectory using Lagrangian based methods, e.g. Discrete Mechanics and Optimal Control (DMOC). In classical mechanics, Lagrangian L is the system's kinetic energy, T, minus potential energy, V.

In case of Delta parallel robot, the three mechanical parts that contribute towards system's energy are: 1) The arm, 2) The forearm, and 3) Moving platform or TCP. So, total kinetic and potential energy will be the sum of each arm and forearm's contribution [147].

$$V = V_c + \sum_{i=1}^{3}(V_{bi} + V_{ai}) \tag{A.16}$$

$$T = T_c + \sum_{i=1}^{3}(T_{bi} + T_{ai}) \tag{A.17}$$

In above equations, V_c is the potential energy of the moving platform, V_{bi} is the potential energy of connecting rods in forearm of leg i, and V_{ai} is the potential energy of the arm i, and same for kinetic energy.

The potential energy of the moving platform is trivial and can be calculated using simple relationship as given in (A.18).

$$V_c = m_c\, g\, P_{0_z} \tag{A.18}$$

The potential energy due to forearm depends upon the moving platform as well as the arm of Delta parallel robot. The effect of these two factors on potential energy of forearm is shown in (A.19).

$$V_{bi} = m_b\ g[P_{0_z} + L_A \sin(\theta_i)],\ i \in \{1, 2, 3\}\,. \tag{A.19}$$

A simple relationship can describe the potential energy due to the arm of Delta parallel robots as arm is the active joint and is not affected by any other joint. The potential energy due to the arm is given by (A.20).

$$V_{ai} = \frac{1}{2} m_a\ g\ L_A \sin(\theta_i),\ i \in \{1, 2, 3\}\,. \tag{A.20}$$

Just like the potential energy, the each component of kinetic energy can be described separately. The contribution towards the system's kinetic energy by moving platform can be described by the simple relationship, given as (A.21), as it is only dependent on the translational velocity.

$$T_c = \frac{1}{2}\ m_c\ (\dot{P}_{0_x}^2 + \dot{P}_{0_y}^2 + \dot{P}_{0_z}^2) \tag{A.21}$$

Kinetic energy due to the forearms not only depends on the translational velocity of moving platform but also on the angular velocity of the respective arm. Equation (A.22) represents the kinetic energy due to forearms.

$$T_{bi} = \frac{1}{2}\ m_b(\dot{P}_{0_x}^2 + \dot{P}_{0_y}^2 + \dot{P}_{0_z}^2) + \frac{1}{2} L_A^2\ m_a\ \dot{\theta_i}^2,\ i = 1, 2, 3. \tag{A.22}$$

Kinetic energy due to the arms consists of two parts, kinetic energy due to inertia and kinetic energy due to the angular velocity of the active link. Equation (A.23) describes the kinetic energy due to each arm of Delta parallel robot.

$$T_{ai} = \frac{1}{6}\ L_A^2\ m_a\ \dot{\theta_i}^2 + \frac{1}{2}\ I_m\ \dot{\theta_i}^2,\ i = 1, 2, 3. \tag{A.23}$$

While deriving the energy equations, it is assumed that the mass of the each connecting rod, m_b, is considered as divided and concentrated at two points.

A.2 Algorithms for Path Planning

In this section we have presented the algorithms for path planning using PRM and GA.

A.2.1 Algorithm for Path Planning using PRM

As discussed in Chapter 4.1, PRM consists of two phases; Learning phase and Query phase. Here we have presented the algorithms for these two phases.

A.2.1.1 Learning Phase

It is the algorithm of the learning phase in which algorithm explores all the allowable workspace of robotic manipulator and to place the random configurations inside the allowable workspace [148]. Learning phase of PRM is explained in details in Section 4.1.1.

Algorithm 1 Algorithm for Learning phase of PRM

INPUT: n number of nodes to put in the roadmap, k number of closest neighbors to examine for each configuration.

OUTPUT: A roadmap $G = (W, E)$

1: $W \leftarrow \emptyset$
2: $E \leftarrow \emptyset$
3: **while** $|W| < n$ **do**
4: **repeat**
5: $q \leftarrow$ *a* random configuration in Q
6: **until** q is collision-free
7: $W \leftarrow W \cup \{q\}$
8: **end while**
9: **for all** $q \in W$ **do**
10: $N_q \leftarrow$ the k closest neighbors of q chosen from W according to $dist$
11: **for all** $q' \in N_q$ **do**
12: **if** $(q, q') \notin E$ and $\Delta(q, q') \neq$ `NIL` **then**
13: $E \leftarrow E \cup \{(q, q')\}$
14: **end if**
15: **end for**
16: **end for**

A.2.1.2 Query Phase

In query phase of PRM, a path from the initial position to the goal position in the allowable workspace is found using the roadmap obtained from the construction phase. In the first step of query phase, initial and goal positions are connected to the nearest configurations. In the next step, Dijkstra's algorithm [132] can be applied to find out the shortest path from initial position to goal position [148].

Algorithm 2 Algorithm for Query phase of PRM

INPUT: The initial configuration q_{init}, the final configuration q_{final}, number of closed neighbors to examine for each configuration k, the roadmap computed in construction phase using Algorithm 1, $G = (W, E)$

OUTPUT: A path from q_{init} to q_{final} or failure

$N_{q_{init}} \leftarrow$ the k closest neighbors of q_{init} from W according to $dist$
$N_{q_{goal}} \leftarrow$ the k closest neighbors of q_{goal} from W according to $dist$
$W \leftarrow \{q_{init}\} \cup \left\{q_{goal}\right\} \cup W$
set q' to be the closest neighbor of q_{init} in $N_{q_{init}}$
repeat
    if $\Delta(q_{init}, q') \neq$ `NIL` **then**
        $E \leftarrow (q_{init}, q') \cup E$
    else
        set q' to be the next closest neighbor of q_{init} in $N_{q_{init}}$
    end if
until a connection was successful or the set $N_{q_{init}}$ is empty
set q' to be the closest neighbor of q_{goal} in $N_{q_{goal}}$
repeat
    if $\Delta(q_{goal}, q') \neq$ `NIL` **then**
        $E \leftarrow (q_{goal}, q') \cup E$
    else
        set q' to be the next closest neighbor of q_{goal} in $N_{q_{goal}}$
    end if
until a connection was successful or the set $N_{q_{goal}}$ is empty
$P \leftarrow$ shortest path (q_{init}, q_{goal}, G)
if P is not empty **then**
    return P
else
    return `failure`
end if

A.2.2 Algorithm for Path Planning using GA

It is the pseudo-code for path planning using Genetic Algorithm. In this algorithm, selection, crossover and mutation depends upon the type of encoding used in GA. In this thesis, we used the decimal coding for path planning using GA.

Algorithm 3 Algorithm for path planning using GA

INPUT: Size of population (α), Rate of elitism (β), Rate of mutation (γ), Number of iterations n, Fitness function $f(\alpha)$, Initial Position q_{init}, Goal position q_{goal}

OUTPUT: A path from q_{init} to q_{final}

Generate random population of α chromosomes.
do
    Evaluate fitness function $f(\alpha)$ for each chromosome α in population
    for i = 1 to **n**
        Number of elitism $ne = \alpha \cdot \beta$
        Select the best ne from the current generation G_i and save in G_i^1.
        Number of crossover $nc = (\alpha - ne)/2$
        for i = 1 to **nc**
            Select two best solutions X_A and X_B from G_i.
            Generate two offspring X_C and X_D using crossover.
            Save X_C and X_D in G_i^2.
        end for
        for j = 1 to **nc**
            Select a solution X_j from G_i^2.
            Mutate each bit of X_j with a rate of γ and generate X'_j
            Update X_j with X'_j in G_i^2
        end for
        update $G_{i+1} = G_i^1 + G_i^2$
    end for
until (the Stop conditions are satisfied **or** the solution is not further improved **or** maximum number of iterations reached)

A.3 Algorithms for Trajectory Optimization

In this Appendix, we have presented the algorithms for trajectory optimization of robotic manipulators using Dynamic Programming.

A.3.1 Dynamic Programming using Path Parameter

In this section, we presented the dynamic Programming algorithm for trajectory optimization of robotic manipulators using path parameter, discussed in Section 5.2.1.

Algorithm 4 Dynamic Programming Algorithm for Trajectory Optimization using Path Parameter

INPUT: Geometrical path, N_p, N_v, Objective function J.

OUTPUT: Optimized cost function, time required to move from one discretize point to next discretize point.

1: Calculate path parameter s from the given geometric path.
2: Calculate function f of dimension $n \times 1$ to map each Joint angle to path parameter, (5.1).
3: Calculate the derivatives df'/ds of the parametric function, (5.2) and (5.3).
4: Discretize the calculated path parameter into N_p segments.
5: Discretize the allowable pseudo-velocity into N_v segments. Now we have a grid, N_p on columns and N_v on rows.
6: **for k** = **1** to $\boldsymbol{N_p}$
7: **for j** = **1** to $\boldsymbol{N_v}$
8: **if k** = $\boldsymbol{N_p}$
9: $Cost[k, j] = 0$ ▷ Set the cost zero to last column
10: **else**
11: $Cost[k, j] =$ `inf` ▷ Set the cost infinity to all column
12: **end if**
13: $P[k, j] = 0$ ▷ Set pointer to zero
14: **end for**
15: **end for**
16: **for k** = $\boldsymbol{N_p}$ to **1**
17: **for j** = **1** to $\boldsymbol{N_v}$
18: Generate the curve that connects point $[k-1, j_{k-1}]$ to $[k, j_k]$.
19: Calculate $\ddot{s}[k_j]$, $\Delta t[k_j]$ and $\tau[k]$ using (5.29), (5.30) and (5.25).
20: **if** Constraints are not satisfied
21: $Cost[k, j] = \inf$
22: **end if**
23: Calculate performance index to move from $[k-1, j_{k-1}]$ to $[k, j_k]$.
24: $Cost[k-1, j_{k-1}]^* =$ Performance index + $C[k, j]$
25: **if** $Cost[k-1, j_{k-1}]^* < Cost[k-1, j_{k-1}]$
26: $Cost[k-1, j_{k-1}] = Cost[k-1, j_{k-1}]^*$
27: **end if**
28: **end for**
29: **end for**
30: Calculate the optimal pseudo-velocity $\dot{s}$ sequence by tracking the pointer.
31: Calculate the corresponding angular velocities (q_i) using inverse parameter function, and the forces/torques of all joints (τ_i), $i = 1, 2, 3$.

A.3.2 Dynamic Programming using Joint Coordinates

In this section, we have presented Algorithm 5 for the trajectory optimization of robotic manipulators using Joint space. In this method, dimensionality of the problem is reduced by considering the non-stationary joint as a reference joint. This method is discussed in details in Section 5.2.2.

Algorithm 5 Dynamic Programming Algorithm for Trajectory Optimization using Joint Space

INPUT: Geometrical path, N_p, N_v, Cost function J.

OUTPUT: Optimized cost function, time required to move from one discretize point to next discretize point.

1: Calculate joints' angles from the given geometric path.
2: Discretize the calculated path parameter into N_p segments.
3: Discretize the allowable pseudo-velocity into N_v segments. Now we have a grid, N_p on columns and N_v on rows.
4: Select a non-stationary joint as a reference joint, i^*
5: **for k** = **1** to $\boldsymbol{N_p}$
6: **for j** = **1** to $\boldsymbol{N_v}$
7: **if k** = $\boldsymbol{N_p}$
8: $Cost[k, j] = 0$ ▷ Set the cost zero to last column
9: **else**
10: $Cost[k, j] =$ `inf` ▷ Set the cost infinity to all column
11: **end if**
12: $P[k, j] = 0$ ▷ Set pointer to zero
13: **end for**
14: **end for**
15: **for k** = $\boldsymbol{N_p}$ to **1**
16: **for j** = **1** to $\boldsymbol{N_v}$
17: Generate the curve that connects point $[k-1, j_{k-1}]$ to $[k, j_k]$.
18: Calculate $\dot{q_{i^*}}[k_j]$, $\Delta t[k_j]$ and $\tau[k]$ using (5.40), (5.41) and (5.37).
19: **if** Constraints are not satisfied
20: $Cost[k, j] =$ `inf`
21: **end if**
22: Calculate performance index to move from $[k-1, j_{k-1}]$ to $[k, j_k]$.
23: $Cost[k-1, j_{k-1}]^*$ = Performance index + $C[k, j]$
24: **if** $Cost[k-1, j_{k-1}]^* < Cost[k-1, j_{k-1}]$
25: $Cost[k-1, j_{k-1}] = Cost[k-1, j_{k-1}]^*$
26: **end if**
27: **end for**
28: **end for**
29: Calculate the optimal pseudo-velocity $\dot{s}$ sequence by tracking the pointer from initial to final state.
30: Calculate the corresponding angular velocities and the torque of all joints.

Bibliography

[1] O. Holland, "The first biologically inspired robots," *Robotica*, vol. 21, pp. 351–363, August 2003.

[2] G. P. Taylor, "An automatic block-setting crane," *Meccano Magazine*, vol. 23, no. 3, 1938.

[3] A. Codourey, "Dynamic modelling and mass matrix evaluation of the delta parallel robot for axes decoupling control," in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 1211–1218, November 1996.

[4] S. Singh and M. C. Leu, "Optimal trajectory generation for robotic manipulators using dynamic programming," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, pp. 88–96, June 1987.

[5] S. Ober-Blöbaum, O. Junge, and J. E. Marsden, "Discrete mechanics and optimal control: An analysis," *Control, Optimisation and Calculus of Variations*, vol. 17, no. 2, pp. 322–352, 2011.

[6] J. Timmermann, *Optimale Steuerung und Mehrzieloptimierung von dynamischen Systemen untersucht am Beispiel des Mehrfachpendels*. PhD thesis, University of Paderborn, Paderborn, Germany, 2013.

[7] W. G. Walter, "An electromechanical animal," *Dialectica*, vol. 4, pp. 206–213, September 1950.

[8] W. G. Walter, "A machine that learns," *Scientific American*, vol. 185, no. 2, 1950.

[9] J. Devol, "Programmed article transfer," June 13 1961. US Patent 2,988,237.

[10] M. Hägele, K. Nilsson, and J. N. Pires, "Industrial robotics," *Springer Handbook of Robotics*, pp. 963–986, 2008.

[11] Y. D. Patel and P. M. George, "Parallel manipulators applications - a survey," *Modern Mechanical Engineering*, pp. 57–64, 2012.

[12] D. Stewart, "A platform with six degrees of freedom," *Institution of Mechanical Engineers*, pp. 371–386, June 1965.

[13] R. Aronson, "A bright horizon for machine tool technology," *Manufacturing Engineering*, vol. 116, pp. 57–70, March 1996.

[14] C. M. Gosselin and J. F. Hamel, "The agile eye: a high-performance three-degree-of-freedom camera-orienting device," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 781–786, May 1994.

[15] T. Arai, R. Stoughton, H. A. K. Homma, T. Nakamura, and K. Nakashima, "Development of a parallel link manipulator," in *Proceedings of Fifth International Conference on Advanced Robotics*, vol. 1, pp. 839–844, June 1991.

[16] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[17] O. V. Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, vol. 37, no. 1, pp. 357–373, 1992.

[18] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[19] R. H. Taylor, *The Synthesis of Manipulator Control Programs from Task-level Specifications.* PhD thesis, Stanford, CA, USA, 1976. AAI7707174.

[20] R. A. Brooks, "Symbolic error analysis and robot planning," *International Journal of Robotics Research*, vol. 1, no. 4, pp. 29–78, 1982.

[21] D.-H. Yand and S.-K. Hong, "A roadmap construction algorithm for mobile robot path planning using skeleton maps," *Advanced Robotics*, vol. 21, no. 1, pp. 51–63, 2007.

[22] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, pp. 560–570, Oct. 1979.

[23] S. M. Udupa, *Collision detection and avoidance in computer controlled manipulators*. PhD thesis, California, USA, 1977.

[24] T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 10, pp. 681–698, 1981.

[25] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. 32, no. 2, pp. 108–120, 1983.

[26] B. Dufay and J.-C. Latombe, "An approach to automatic robot programming based on inductive learning," *International Journal of Robotic Research*, vol. 3, no. 4, pp. 3–20, 1984.

[27] Y. Li, C. Li, and Z. Zhang, "Q-learning based method of adaptive path planning for mobile robot," in *Proceedings of International Conference on Information Acquisition*, pp. 983–987, Auugst 2006.

[28] C. Z. Jie Shao, JunPeng Zhang, "Research on multi-robot path planning methods based on learning classifier system with gradient descent methods," *Springer Handbook of Robotics*, vol. 169, pp. 963–986, 2012.

[29] J.-A. Meyer and D. Filliat, "Map-based navigation in mobile robots:: I. a review of map-learning and path-planning strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 243–282, 2003.

[30] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 500–505, Mar 1985.

[31] H. Safadi, "Spatial representation and mobile robotics," project report, School of Computer Science, McGill University, Canada, April 2007.

[32] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Rob. Res.*, vol. 10, pp. 628–649, Dec. 1991.

[33] C. W. Warren, "Global path planning using artificial potential fields," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 500–505, May 1989.

[34] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1398–1404, April 1991.

[35] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila, "Reactive navigation in outdoor environments using potential fields," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1232–1237, May 1998.

[36] D. Fu-guang, J. Peng, B. Xin-qian, and W. Hong-jian, "Auv local path planning based on virtual potential field," in *Proceedings of the IEEE International Conference on Mechatronics and Automation*, vol. 4, pp. 1711–1716, 2005.

[37] V. J. Lumelsky and A. A. Stepanov, "Manipulator cartesian path control," *IEEE Transactions on Automatic Control*, vol. 33, pp. 1058–1063, 1986.

[38] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1-4, pp. 403–430, 1987.

[39] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM*, vol. 40, pp. 1048–1066, 1993.

[40] Q.-C. Pham, S. Caron, and Y. Nakamura, "Kinodynamic planning in the configuration space via admissible velocity propagation," in *Proceedings of Robotics: Science and Systems*, (Berlin, Germany), June 2013.

[41] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 2427–2433, October 2009.

[42] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock, "Kinodynamic motion planning amidst moving obstacles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 537–543, April 2000.

[43] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[44] M. Kazemi, M. Mehrandezh, and K. Gupta, "Kinodynamic planning for visual servoing," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2478–2484, May 2011.

[45] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., Computer Science Deptartment, Iowa State University, 1998.

[46] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 473–479, May 1999.

[47] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[48] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3310–3317, May 1994.

[49] S. Koenig and M. Likhachev, "D* lite," in *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, July 2002.

[50] D. Cockburn and Z. Kobti, "Memory-bounded d* lite," in *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference*, May 2008.

[51] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2138–2145, May 1994.

[52] L. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, August 1996.

[53] M. H. Overmars and P. Švestkavestka, "A probabilistic learning approach to motion planning," in *In Proc. Workshop on Algorithmic Foundations of Robotics*, 1994.

[54] N. Amato, O. Bayazit, L. Dale, and C. Jones, "Choosing good distance metrics and local planners for probabilistic roadmap methods," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 630–637, May 1998.

[55] T. Horsch, F. Schwarz, and H. Tolle, "Motion planning with many degrees of freedom-random reflections at c-space obstacles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3318–3323, May 1994.

[56] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Obprm: An obstacle-based prm for 3d workspaces," in *Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics on Robotics : The Algorithmic Perspective: The Algorithmic Perspective*, WAFR '98, (Natick, MA, USA), pp. 155–168, A. K. Peters, Ltd., 1998.

[57] S. Wilmarth, N. Amato, and P. Stiller, "Maprm: a probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1024–1031, May 1999.

[58] R. Bohlin and E. Kavraki, "Path planning using lazy prm," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 521–528, April 2000.

[59] K. Saska, M. Macas, L. Preucil, and L. Lhotska, "Robot path planning using particle swarm optimization of ferguson splines," in *Proceedings of the International Conference on Emerging Technologies and Factory Automation*, pp. 833–839, September 2006.

[60] L. Wang, Y. Liu, H. Deng, and Y. Xu, "Obstacle-avoidance path planning for soccer robots using particle swarm optimization," in *Proceedings of the International Conference on Robotics and Biomimetics*, pp. 1233–1238, December 2006.

[61] Y. Z. Cong and S. Ponnambalam, "Mobile robot path planning using ant colony optimization," in *Proceedings of the International Conference on Advanced Intelligent Mechatronics*, pp. 851–856, July 2009.

[62] Y.-T. Hsiao, C.-L. Chuang, and C.-C. Chien, "Ant colony optimization for best path planning," in *Proceedings of the International Symposium on Communications and Information Technology*, vol. 1, pp. 109–113, October 2004.

[63] P. Bhattacharjee, P. Rakshit, I. Goswami, and A. Konar, "Multi-robot path-planning using artificial bee colony optimization algorithm," in *Proceedings of the Third World Congress on Nature and Biologically Inspired Computing*, pp. 219–224, October 2011.

[64] E. Mansury, A. Nikookar, and M. Salehi, "Artificial bee colony optimization of ferguson splines for soccer robot path planning," in *Proceedings of the International Conference on Robotics and Mechatronics*, pp. 85–89, Feburary 2013.

[65] C. Liu, Z. Gao, and W. Zhao, "A new path planning method based on firefly algorithm," in *Proceedings of the 2012 Fifth International Joint Conference on Computational Sciences and Optimization*, pp. 775–778, June 2012.

[66] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang, "A modified firefly algorithm for ucav path planning," *International Journal of Hybrid Information Technology*, vol. 5, no. 3, p. 123, 2012.

[67] G. Wang, L. Guo, H. Duan, L. Liu, , and H. Wang, "A bat algorithm with mutation for ucav path planning," *The Scientific World Journal*, p. 15 pages, 2012.

[68] C. E. Thomaz, M. A. C. Pacheco, and M. M. B. R. Vellasco, "Mobile robot path planning using genetic algorithms," *Foundations and Tools for Neural Modeling*, vol. 1, pp. 671–679, 1999.

[69] C.-K. Loo, M. Rajeswari, E. K. Wong, and M. V. C. Rao, "Mobile robot path planning using hybrid genetic algorithm and traversability vectors method," *Intelligent Automation and Soft Computing*, vol. 10, no. 1, pp. 51–63, 2004.

[70] O. Castillo, L. Trujillo, and P. Melin, "Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots," *Soft Computing*, vol. 11, no. 3, pp. 269–279, 2007.

[71] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," *Computers and Electrical Engineering*, vol. 38, no. 6, pp. 1564–1572, 2012.

[72] Y. Stepanenko, "Dissipative properties and optimal control of manipulators," in *Proceedings. of the Second Symposium on the Control of Artificial Limbs*, June 1970.

[73] M. E. Kahn and B. Roth, "The near-minimum-time control of open-loop articulated kinematic chains," *Journal of Dynamic Systems, Measurement, and Control*, vol. 93, no. 3, pp. 164–172, 1971.

[74] J. Y. S. Luh and M. W. Walker, "Minimum-time along the path for a mechanical arm," in *Proceedings of the Conference on Decision and Control*, pp. 755–759, June 1977.

[75] J. Y. S. Luh and C. S. Lin, "Optimum path planning for mechanical manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 103, no. 2, pp. 142–151, 1981.

[76] B. K. Kim and K. G. Shin, "An efficient minimum-time robot path planning under realistic conditions," in *Proceedings of the American Control Conference*, pp. 296–303, June 1984.

[77] B. K. Kim and K. G. Shin, "Minimum-time path planning for robot arms and their dynamics," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, pp. 213–223, March 1985.

[78] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, pp. 3–17, September 1985.

[79] K. Shin and N. D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, pp. 531–541, June 1985.

[80] V. Rajan, "Minimum time trajectory planning," in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, pp. 759–764, March 1985.

[81] K. Shin and N. D. McKay, "Selection of near-minimum time geometric paths for robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 31, pp. 501–511, June 1986.

[82] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," in *Proceedings of the International Conference on Robotics and Automation*, vol. 3, pp. 1399–1405, April 1986.

[83] J.-J. Slotine and H. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 118–124, February 1989.

[84] J. McCarthy and J. Bobrow, "The number of saturated actuators and constraint forces during time-optimal movement of a general robotic system," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 407–409, 1992.

[85] Z. Shiller and H.-H. Lu, "Computation of path constrained time optimal motions with dynamic singularities," *Journal of Dynamic Systems, Measurement, and Control*, vol. 114, pp. 118–124, March 1992.

[86] M. Tarkiainen and Z. Shiller, "Time optimal motions of manipulators with actuator dynamics," in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, pp. 725–730, May 1993.

[87] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of Robotic Systems*, vol. 17, no. 5, pp. 233–249, 2000.

[88] Z. Shiller and H.-H. Lu, "Robust computation of path constrained time optimal motions," in *Proceedings of the International Conference on Robotics and Automation*, vol. 3, pp. 144–149, May 1990.

[89] Y. Chen and A. Desrochers, "Time-optimal control of two-degree of freedom robot arms," in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, pp. 1210–1215, April 1988.

[90] J. Kieffer, "Manipulator inverse kinematics for untimed end-effector trajectories with ordinary singularities," *International Journal of Robotics Research*, vol. 11, no. 3, pp. 225–237, 1992.

[91] D. Nenchev, Y. Tsumaki, M. Uchiyama, V. Senft, and G. Hirzinger, "Two approaches to singularity-consistent motion of nonredundant robotic mechanisms," in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, pp. 1883–1890, April 1996.

[92] Y. Tsumaki, S. Kotera, D. Nenchev, and M. Uchiyama, "Singularity-consistent inverse kinematics of a 6-dof manipulator with a non-spherical wrist," in *Proceedings of the International Conference on Robotics and Automation*, vol. 4, pp. 2980–2985, April 1997.

[93] J. T. Betts and W. P. Huffman, "Path-constrained trajectory optimization using sparse sequential quadratic programming," *Journal of Guidance, Control and Dynamics*, vol. 16, pp. 59–68, February 1993.

[94] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1 ed., 1957.

[95] M. Vukobratović and M. Kirćanski, "A method for optimal synthesis of manipulation robot trajectories," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, pp. 188–193, June 1982.

[96] K. G. Shin and N. D. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 31, pp. 491–500, June 1986.

[97] V. Yen, "An inverse dynamic-based dynamic programming method for optimal point-to-point trajectory planning of robotic manipulators," *International Journal of Systems Science*, vol. 26, no. 2, pp. 181–195, 1995.

[98] G. Field and Y. Stepanenko, "Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators," in *Proceedings of the International Conference on Robotics and Automation*, vol. 3, pp. 2755–2760, April 1996.

[99] O. Junge, J. Marsden, and S. Ober-Blöbaum., "Discrete mechanics and optimal control," in *Proceedings of the 16th IFAC World Congress*, July 2005.

[100] J. A. Cadzow, "Discrete calculus of variations," *International Journal of Control*, vol. 11, no. 3, pp. 393–407, 1970.

[101] B. W. Jordan and E. Polak, "Theory of a class of discrete optimal control systems," *Journal of Electrical Control*, vol. 17, pp. 697–711, 1964.

[102] C. L. Hwang and L. T. Fan, "A discrete version of pontryagins maximum principle," *Operation Research*, vol. 15, no. 1, pp. 139–146, 1967.

[103] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, vol. 10, pp. 357–514, 2001.

[104] S. Ober-Blöbaum, *Discrete Mechanics and Optimal Control*. PhD thesis, Universität Paderborn, Paderborn, Germany, 2008.

[105] D. Pekarek, A. Ames, and J. Marsden, "Discrete mechanics and optimal control applied to the compass gait biped," in *Proceedings of the Conference on Decision and Control*, pp. 5376–5382, December 2007.

[106] J. Timmermann, S. Khattab, S. Ober-Blöbaum, and A. Trächtler, "Discrete mechanics and optimal control and its application to a double pendulum on a cart," in *Proceedings of the 18th IFAC World Congress*, vol. 18, pp. 10199–10206, 2011.

[107] S. Ober-Blöbaum and J. Timmermann, "Optimal control for a pitcher's motion modeled as constrained mechanical system," in *Proceedings of the International Conference on Multibody Systems, Nonlinear Dynamics, and Control*, pp. 597–606, 2009.

[108] R. I. McLachlan and S. Marsland, "Discrete mechanics and optimal control for image registration," in *Proceedings of the 13th Biennial Computational Techniques and Applications Conference*, vol. 48, pp. C1–C16, 2006.

[109] O. Junge and S. Ober-Blöbaum, "Optimal reconfiguration of formation flying satellites," in *Proceedings of the Conference on Decision and Control and European Control Conference*, pp. 66–71, 2005.

[110] O. Junge, J. E. Marsden, and S. Ober-Blöbaum, "Optimal reconfiguration of formation flying spacecraft - a decentralized approach," in *Proceedings of the Conference on Decision and Control*, pp. 5210–5215, 2006.

[111] S. Leyendecker, S. Ober-Blöbaum, E. Marsden, and M. Ortiz, "Discrete mechanics and optimal control for constrained systems," *Optimal Control Applications and Methods*, vol. 31, no. 6, pp. 505–528, 2010.

[112] P. Betsch, "The discrete null space method for the energy consistent integration of constrained mechanical systems: Part i: Holonomic constraints," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 50-52, pp. 5159–5190, 2005.

[113] P. Betsch and S. Leyendecker, "The discrete null space method for the energy consistent integration of constrained mechanical systems. part ii: multibody dynamics," *International Journal for Numerical Methods in Engineering*, vol. 67, no. 4, pp. 499–552, 2006.

[114] S. Leyendecker, G. Johnson, and M. Ortiz, "Planned contacts and collision avoidance in optimal control problems," *Proceedings in Applied Mathematics and Mechanics*, vol. 12, no. 1, pp. 77–78, 2012.

[115] B. R. Markiewicz, "Analysis of the computed torque drive method and comparison with conventional position servo for a computer-controlled manipulator," tech. rep., Jet Propulsion Laboratory, California Institute of Technology, 1973.

[116] A. Morris and A. Madani, "Computed torque control applied to a simulated two-flexible-link robot," *Transactions of the Institute of Measurement and Control*, vol. 19, no. 1, pp. 50–60, 1997.

[117] R. Middleton and G. Goodwin, "Adaptive computed torque control for rigid link manipulations," *Systems & Control Letters*, vol. 10, no. 1, pp. 9–16, 1988.

[118] T. Martin and E. Yaz, "Robot manipulator control using adaptive computed torque technique," in *Proceedings of the 34th Midwest Symposium on Circuits and Systems*, vol. 2, pp. 947–949, May 1991.

[119] Z. Song, J. Yi, D. Zhao, and X. Li, "A computed torque controller for uncertain robotic manipulator systems: Fuzzy approach," *Fuzzy Sets and Systems*, vol. 154, no. 2, pp. 208–226, 2005.

[120] M. Llama, V. Santibanez, R. Kelly, and J. Flores, "Stable fuzzy self-tuning computed-torque control of robot manipulators," in *Proceedings of the International Conference on Robotics and Automation*, vol. 3, pp. 2369–2374, May 1998.

[121] M. A. Llama, R. Kelly, and V. Santibañez, "Stable computed-torque control of robot manipulators via fuzzy self-tuning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 30, no. 1, pp. 143–150, 2000.

[122] Y. Chen, G. Ma, S. Lin, , and J. Gao, "Adaptive fuzzy computed-torque control for robot manipulator with uncertain dynamics," *International Journal of Advanced Robotic Systems*, 2012.

[123] W.-W. Shang, S. Cong, and Y. Ge, "Adaptive computed torque control for a parallel manipulator with redundant actuation," *Robotica*, vol. 30, no. 3, pp. 457–466, 2012.

[124] P. Werbos, "Backpropagation: past and future," in *Proceedings of the International Conference on Neural Networks*, vol. 1, pp. 343–353, July 1988.

[125] G. C. Goodwin and K. S. Sin, *Adaptive filtering prediction and control*. United States: Prentice-Hall, 1984.

[126] K. J. Aström and B. Wittenmark, *Adaptive Control*. Mineola, N.Y.: Dover Publications, 2 ed., 1995.

[127] F. Lewis, A. Yesildirek, and K. Liu, "Neural net robot controller: structure and stability proofs," in *Proceedings of the 32nd Conference on Decision and Control*, vol. 3, pp. 2785–2791, December 1993.

[128] L. Rey and R. Clavel, *The Delta Parallel Robot*, pp. 401–417. Advanced Manufacturing, Swiss Federal Institute of Technology, IMT-EPFL, Switzerland: Springer London, 1999.

[129] A. Codourey, *Contribution á la commande des robots rapides et précis. Application au robot DELTA à centraînement direct*. PhD thesis, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland, 1991.

[130] P. Guglielmetti, *Model-based control of fast parallel robots: A global approach in operational space*. PhD thesis, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 1994.

[131] A. Codourey, "Dynamic modeling of parallel robots for computed-torque control implementation," *International Journal of Robotics Research*, vol. 17, no. 12, pp. 1325–1336, 1998.

[132] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[133] A. E. Bryson and Y. C. Ho, *Applied Optimal Control: Optimization, Estimation and Control*. New York: John Wiley and Sons, 1975.

[134] I. M. Gelfand and S. V. Fomin, *Calculus of Variations*. Mineloa, N.Y.: Dover Publications, 1991.

[135] C. Lanczos, *The Variational Principles of Mechanics*. New York: Dover Publications, 1970.

[136] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*. Mineola, N. Y. US: Dover Publications, 2nd ed., 2007.

[137] G. S. Rao, *Numerical Analysis*. New Delhi, India: New Age International Publishers, 3rd ed., 2006.

[138] R. L. Burden and J. D. Faires, *Numerical Analysis*. M. A., USA: Richard Stratton, 9th ed., 2011.

[139] B. Hildebrand, *Introduction to Numerical Analysis*. N.Y., USA: Dover Publications, 2nd ed., 1974.

[140] J. F. Kleinfinger, *Modélisation dynamique des robots à chaîne simple, arborescente ou fermée, en vue de leur commande*. PhD thesis, Ecole Naionale Supérieure de Mécanique, Nantes, France, 1986.

[141] T. Kokkinis and R. Stoughton, "Dynamics and control of closed-chain robot arms with application to a new direct-driven robot arm," *International Journal of Robotics and Automation*, vol. 6, no. 1, pp. 25–34, 1991.

[142] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1990.

[143] L.-C. Wang and C. C. Chen, "On the dynamic analysis of general parallel robotic manipulators," *International Journal of Robotics and Automation*, vol. 9, no. 2, pp. 81–87, 1994.

[144] A. Codourey and E. Burdet, "A body-oriented method for finding a linear form of the dynamic equation of fully parallel robots," in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, pp. 1612–1618, April 1997.

[145] C.-D. Zhang and S.-M. Song, "An efficient method for inverse dynamics of manipulators based on the virtual work principle," *Journal of Robotic Systems*, vol. 10, no. 5, pp. 605–627, 1993.

[146] Z. Ji, "Study of the effect of leg inertia in stewart platforms," in *Proceedings of the International Conference on Robotics and Automation*, vol. 1, pp. 121–126, May 1993.

[147] R. E. Stamper, *A Three Degree of Freedom Parallel Manipulator with Only Translational Degrees of Freedom*. PhD thesis, MD, USA, 1997. Ph.D. 97-4.

[148] H. Choset, K. M. Lynch, S. Hutchinston, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion*. Cambridge, M.A., USA: MIT Press, 1st ed., 2005.

VITA

Zeeshan Shareef was born in Karachi (Pakistan), on June 17, 1986. He received his early education at Abbas Govt. Boys Secondary School and D. J. Sindh Govt. Science College in Karachi, Pakistan. He then enrolled at the Institute of Industrial Electronics Engineering (IIEE), NED University of Engineering and Technology (UET), where he got his BE in Industrial Electronics Engineering with distinction in 2007. He worked in a public sector research and development organization in the domain of digital control. He then obtained scholarship and moved to Pakistan Institute of Engineering and Applied Sciences (PIEAS) Pakistan, where he received his MS in Systems Engineering with distinction in 2010. His master thesis was awarded as the Best MS Thesis for session 2008-2010. After completing his MS, he joined Department of Electrical Engineering in Lahore University of Management Sciences (LUMS) as teaching fellow. In November 2011, he got the PhD scholarship from the ministry of Science and Technology of NRW Government. He joined the research group of Control Engineering and Mechatronics at Heinz Nixdorf Institute, Universität Paderborn. His research interests include robotics, trajectory optimization, swarm intelligence algorithms, anti-windup compensator designing and nonlinear control.