EFFICIENT SUB-OPTIMAL INVERSE KINEMATIC SOLUTION FOR REDUNDANT MANIPULATORS

BAHAREH GHOTBI

(B. Eng, Sharif University of Technology)

A THESIS SUBMITTED FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2009

Acknowledgment

I would like to express my warm and sincere thanks to Professor Poo Aun Neow who introduced me to the field of robotics and gave me the opportunity to pursue my studies in National University of Singapore. His support and trust helped in many aspects during my studies in Singapore. I specially want to thank Prof. Sam Ge for his guidance during my research in Social Robotics Laboratory. His perpetual energy and enthusiasm in research was the biggest motivation for me in my work. The generous support from Agency for Science, Technology and Research (A*STAR) is greatly appreciated for granting me the Graduate Scholarship for master studies. I owe my sincere gratitude to Dr. Ong Fook Rhu and Dr. John-John Cabibihan who gave me the opportunity to conduct my experiments under their guidance. I was delighted to interact with Prof. Oussama Khatib and Prof. Francis Quek during their visits to NUS. Their innovative ideas greatly influenced my work.

During this two years I have collaborated with many colleagues for whom I have great regard. I would regret my master studies years in NUS if I did not join the team for TechX challenge. The associated experience broadened my perspective on the practical aspects of robotics and I am grateful to come across several life-long

friends during that time. Thanks dear Brice, Aswin and Pey Yuen. All my lab mates at Social Robotics Lab made it a happy environment to work. My warm thanks go to Dr. Kenneth Pinpin for sharing his useful experiences and dear He Wei for his kind supports.

I owe my loving thanks to my mom, dad and my dear brother, Borna. They have always been a constant source of encouragement in my life. It would have not been bearable if I did not have the three of you in my life. My dear friends in Singapore and Iran made my first experience of independent life happy and memorable.

Summary

An efficient solution to Inverse Kinematic problem is presented in this work. The motivation for this research is the advancements in Social Robotics during the last decade which require robots to interact with human. The conventional methods for manipulator trajectory planning provide tools for smooth and accurate motion of the arm but they fail to simulate the natural motion of human arm. The emphasis of this work would be mostly on providing natural motion of manipulators for social applications such as handshaking. Furthermore, the presented algorithm would be applicable to all manipulators with various geometries and degrees of freedom. Therefore, it is tried to follow the same logic as human to plan the arm motion. The main concept behind this algorithm would be as follows. The torque sent to each joint actuator is proportional to the instantaneous contribution of that joint in driving the end-effector towards the target. Using this concept, the driving command for each joint is computed at each cycle and would be sent to the actuators after applying some control strategies to ensure the smoothness of the motion. This algorithm is applied to highly redundant manipulators in simulation studies to verify its effectiveness. As an example, simulation studies on a ten degrees of freedom arm is presented in this thesis

as a model of the human arm. Furthermore, series of experiments are conducted to compare the motion of the arm in human and the simulated model. The results of this study are shown in graphs as well as numbers, using analysis tools such as Dynamic Time Warping.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Human body and mind have always been an inspiration for researchers to find the most efficient designs and control strategies for robots to eventually enable them to help human in harsh environments and difficult tasks in efficient ways. However, the extreme complexity of biological systems has led engineers to come up with more simple and practical solutions. Inverse kinematic problem is a good example of these simplifications in which, for many industrial robots during the recent decades the closed form solutions have been derived manually using the geometrical relations of the manipulator multibody system. In the past, it has been reasonable to follow the classical inverse kinematic methods, since once the solution was found for a single manipulator structure many other robots with the same structure could utilize the solution. Nowadays robots are built as complex as human body and the wide variety

of designs reduces the chance of finding available inverse kinematic solution in the literature which can be applied to a new robot. Inverse kinematic computation is an inevitable part of robotic control and in most cases the only part which has to be solved manually. Although many solutions has been given for different configurations of robot manipulators, the need for a general solution is extremely felt.

This work, presents a new approach in solving inverse kinematic problem based on the concept of flow. The presented algorithm is developed by simplifying the human decision making process in moving the arm to reach a target in the space. The existing solutions for inverse kinematics either use mathematical tools for each specific manipulator to derive the closed form solution for each joint variable to reach the desired position or use numerical methods. Generally analytical methods are preferable to their numerical counterparts since they offer all the solutions and are computationally faster and more reliable. These methods are very accurate and the end-effector can reach the final position within any desired error threshold of the target location. However, they have two major shortcomings: (i) as mentioned before, the solution has to be computed separately for every different manipulator using complex mathematical derivations, which is prone to mistakes and requires supervision to obtain meaningful results and (ii) the analytical methods give no guarantee that the manipulator would not fail to reach to the final configuration due to self collisions since they only consider the initial configuration to compute the final joints value and no information on the trajectory to be traveled is provided. Therefore, in order to avoid collision with obstacles, setting many midpoints is required. In this work we offer a numerical method which is inspired by human arm motion. As human, when we aim to stretch our arm

towards a specific point in the space, we gradually move our hand on the tangent path of the connecting line between the hand and the target. As the hand is moving in this path, each of arm's joint coordinates needs to be updated to support this motion. The presented algorithm describes the rules governing this consecutive joints coordinate updating.

The main contributions of this research are: 1) The algorithm converges to solution for any number of degrees of freedom as long as the target is reachable within the manipulator workingspace. 2) There is no need for any manual pre-calculations and the algorithm can be applied to any manipulator configuration as long as the forward kinematic formulation is provided. 3) Potential field method can be easily applied to the presented algorithm and compared to conventional inverse kinematics methods no additional trajectory planning and midpoint setting is required. 4) The resulting motion has more similarity to natural human arm motion.

1.2 Thesis Organization

This thesis presents a novel method in solving the Inverse Kinematic problem and the method is verified by experimental results and simulation studies. A complete literature review of the relevant research areas is presented in chapter two. Description of the proposed algorithm and simulation procedure are covered in chapter three. The experimental setup and result analysis are presented in chapter four. Finally Chapter five summarizes the research contributions and describes the future works.

Chapter 2

Literature Review

The basic expectation from a robot manipulator is to have the ability of following a trajectory consisting of specified points. In the real environment the robot should be able to avoid obstacles in its predefined path and perform an efficient motion. An efficient motion can be defined as selection of the motion parameters in a way to travel the specified path in the shortest time and not violating the joint boundaries and force/torque limits of manipulator actuators. At the same time the planning should avoid singularities in which high force/torque is imposed on actuators.

It can be shown that shortest traverse time for a path requires at least one of the actuators to operate at its maximum or minimum boundaries. Dynamic parameters of manipulator can be converted to path parameters and its first and second derivatives. Furthermore the actuator bounds can be transformed to acceleration boundaries on the path as a function of position and velocity. In this section difficulties in the area of motion planning are presented and the most significant contributions of researchers

in the past two decades are described.

2.1 Classic Inverse Kinematic Methods

One of the fundamental challenges in control of manipulators motion is solving the inverse kinematics problem. The objective of this task is to find all the possible joint variables, linear or angular, to enable the end-effector to reach a desired position and orientation in the space. With this basic ability it is possible to integrate a sequence of desired points to form a desired trajectory in order to have the end-effector perform a desired motion. Hence, the inverse kinematics solution involves determining the manipulator configuration corresponding to each desired point. A more exact formulation of inverse kinematic problem would be that given position and orientation of the end effector relative to the base frame, compute all possible sets of joint angles and link geometries which would result in the given position and orientation of the end effector. Different methods are proposed to solve the inverse kinematics problem: matrix, vector, and numerical methods. The matrix method [1, 2] makes use of the homogeneous transformation matrix in Denavit-Hartenberg notation. In this method in a recursive manner, the unit vector of orientation of i^{th} link is obtained, $i = n \cdots 1$. Another matrix based method takes advantage of the rotation matrix property. Although these algorithms offer exact solution to the problem and cover all the possible solutions, they are not convenient for a general manipulator control. The reason is that, these methods can only be derived offline and they lead to nonlinear complex relations for higher degrees of freedom which makes them not practical to solve. The vector method

[3] is also a recursive algorithm. This method utilizes the vector properties such as cross and inner products to find the orientation of joint axes and manipulator's links relative to each other. This method is following D-H parameters as well and basically shares the same logic as the previously mentioned method. The difference is only in the mathematical techniques to deal with this geometrical problem. For the class of manipulators which the inverse kinematics solution cannot be obtained in explicit form, numerical methods are used. Most of these methods are based on an inverse Jacobian and utilizing the Newton-Raphson method [4]. Inverse kinematic is expressed as a nonlinear problem for which it is necessary to discuss issues such as existence of solutions, multiple solutions, and the method of solutions. Discussion on these issues are brought in the following subsections.

2.1.1 Existence of Solutions

For a solution to exist it must lie within the manipulator workspace. Workspace is divided into Dexterous workspace D and Reachable workspace R.

- Dexterous workspace: The subset of space is which the robot end effector can reach with all orientations, i.e. at each point in D, the end effector can be arbitrarily oriented.

- Reachable workspace: The subset of space in which the robot can reach in at least one orientation, therefore, $D \subseteq R$.

For example, a planar arm with three revolute joints has a large dexterous workspace in the plane, while if the arm has two revolute joints with equal links in length the

dexterous workspace would be reduced to the origin point only. Also, a robot with less than six joints cannot attain a general goal in position and orientation in three dimensional space. Joint limits is another constraint which has to be satisfied in solving an inverse kinematic problem. Generally, a manipulator will be considered solvable if the joint variable associated with a given position and orientation can be determined by an algorithm.

2.1.2 Multiple Solutions

A common problem that can occur when solving inverse kinematic is multiple solutions, because the system has to be able to choose one. The number of solutions depends on the number of joints in the manipulator as well as the link parameters a_i, α_i, θ_i, and d_i. For a general manipulator with 6 DOF, there are up to 16 solutions. As an example, the PUMA 560 can reach certain goals with 8 different arm configurations. The solution to the problem of multiple solutions is to introduce decision criteria such as minimizing the weighted amount that each joint is required to move or avoiding collision with obstacles. In the case of PUMA 560, after all eight solutions have been computed, some or all of them may have to be discarded due to joint limit violations, and from the remaining valid solutions, usually the one closest to the present manipulator configuration is chosen.

2.1.3 Method of Solutions

Unlike linear equations there are no general algorithms which lead to the solution of the nonlinear coupled problem of inverse kinematics. Depending on the geometry of

the system, closed form or numerical solutions can used to solve the inverse kinematic problem.

2.1.3.1 Closed form solution

Closed form solutions exists for special manipulator geometries; for example decoupled manipulators and more generally when the degrees of freedom of the characteristic polynomial is less or equal to 4 and problem only involves one unknown. Therefore the inverse kinematic problem would be in the form of a root finding problem of a 4^{th} order polynomial [5]. Generally for decoupled robots analytical solution to inverse kinematic is available. The reason being that in the process of decoupling a subset of joints are found to be responsible for a subset of manipulator tasks and therefore, this results in reducing the system to a lower order subsystem, i.e. 3^{rd} order, for which closed form solutions are guaranteed. This requires the identification of decoupled task and decoupled robot subsystems that the decoupled task can be assigned to. In some manipulators due to their geometry, decoupling is guaranteed. Five groups of decoupled robot geometries are those having

1. Any three translational joints,
2. Any three co-intersecting rotational axes,
3. Any 2 translational joints normal to a rotational joint,
4. Translational joint normal to 2 parallel joints,
5. Any three rotational joints parallel.

The first two geometries are identified by Pieper [6], and the last three geometries are identified by Ang [7]. As an example, robots with spherical wrists are commonly used in industry and belong to the second group of geometries, i.e. three rotational axes intersection at a common point. Virtually all industrial manipulators are designed sufficiently simple so that a closed form solution exists. PUMA 560 is a robot with six revolute joints which has its last three joints axes intersecting at a common point, providing the sufficient condition to have inverse kinematic closed form solution. Closed form solutions, themselves are divided into Algebraic solutions and Geometric solutions. Algebraic solutions are obtained by solving trigonometric nonlinear equations, while geometric solutions are obtained by reducing the larger problem to a series of plane geometry problems.

2.1.3.2 Numerical solution

For robots that do not have decouple geometries an analytical solution does not exist and only numerical solutions relying on iterative procedures can be useful. There are some important requirements for numerical algorithms such as convergence, insensitivity to initial estimates, and provision for multiple solutions. In numerical algorithms there are m equations and n unknowns and the algorithm starts with an initial estimate for n unknowns. The error due to the non accurate value chosen by initial guess is computed and based on the method chosen in each algorithm, it tries to modify the estimates to reduce error. The most common methods are based on the Newton-Raphson approach. In this method the forward kinematic may be interpreted as

$$f(\theta) - x = 0, \tag{2.1}$$

with $x \in \mathbb{R}^n$. The solution at each iteration by the Newton- Raphson method would be

$$\theta^{(k+1)} = \theta^{(k)} - J^{-1}\theta^{(k)}(f(\theta^{(k)}) - x), \tag{2.2}$$

with $k = 0, 1, ...$ being the iteration index and $[J]_{ij} = [\partial f_i / \partial \theta_j] \in \mathbb{R}^{n \times n}$ being the Jacobian matrix. The iteration is stopped for k_{max} or for $\left\|\theta^{(k+1)} - \theta^{(k)}\right\| < \epsilon$. In this method one must pay attention to the possibility of singularity, i.e. $det(J) = 0$, in the robot workspace. Other proposed numerical methods are exploiting polynomial continuation [8], dyalitic elimination [9]or neural networks [10, 11]. However, numerical methods are generally cost intensive and slow algorithms. Apart from classical methods mentioned above, special methods in kinematic analysis are introduced in recent years. In a work by A. Khawaja et al. [12] a unified approach based on Genetic Algorithm is presented. This straightforward algorithm lacks the proof of parameter convergence which reduces the method's efficiency and accuracy. Furthermore, the proposed algorithm is not applicable for real-time computation. Other approaches to inverse kinematic problem are Interval Methods for solving systems of non-linear equations which have been explored by a variety of authors [13, 14]. They have already been used to solve some kinematic problems proving to be robust, but sometimes slow compared to continuation methods [15].

2.1.3.3 Examples of Common Methods

The following methods are some of the most successful approaches to inverse kinematic problem which are subgroups of analytical or numerical solutions or a combination of them.

Resolved Motion Rate Control of Manipulators In the case of redundant manipulators several methods have been suggested to resolve the redundancy. Whitney [16] shows that the operator can obtain control of motion easily along "world coordinates" if the control actions are modified by the inverse of the arm's Jacobian matrix. Since we are dealing in most cases with velocity commands to actuators, rather than position commands, it is necessary to understand the properties of Jacobian matrix and its benefits in finding inverse kinematics solution. In order to have the basic understanding on how Jacobian is used in this problem, a very simple example is presented. Imagine we have to control a manipulator manually by using joysticks. At the beginning we would actuate each joint separately to examine the direction and speed of end-effector due to that actuator. This is analog to one column of matrix J in $\dot{x}(t) = J(\theta(t))\dot{\theta}(t)$, where $x(t) = f(\theta(t))$.

To describe the concept of inverse Jacobian in control, assume that we have a non-redundant arm in which $n = m$, where n is dimension of task space, $x(t)$, and m is the dimension of joint space, $\theta(t)$. For each component of $\dot{x}$, the corresponding actuator commands are calculated through $J^{-1}(\theta)$ in the following equation:

$$\dot{\theta} = J^{-1}(\theta)\dot{x}. \tag{2.3}$$

However, we are sometimes dealing with redundant manipulators and even one manipulator can operate as redundant or non-redundant system at different tasks. For example, by defining different task space dimension we can make a non-redundant manipulator redundant only by choosing not to control come components of x. On the other hand, by freezing some of the actuators in accomplishing a task, a redundant manipulator is turned to a non-redundant one. With the above explanation we have to be able to develop control strategies based on non-square Jacobian matrices. If $m > n$, meaning that the Jacobian is not invertible, we need to define an optimality criterion for the manipulator motion to reduce the solution to a unique one. The types of these criteria are well explained in 2.2.3. Traditionally the criterion is defined to be total kinetic energy

$$H = \frac{1}{2} \int \dot{\theta}^T A \dot{\theta} dt, \tag{2.4}$$

and in some cases to reduce the cost of computation, it was approximated to instantaneous kinetic energy

$$H = \frac{1}{2} \dot{\theta}^T A \dot{\theta} dt. \tag{2.5}$$

Using Lagrange multipliers and with assumption of known $\dot{x}$, the optimal $\dot{\theta}$ can be found to be

$$\dot{\theta} = \dot{x}^T [J(\theta) A^{-1} J(\theta)^T]^{-1} J(\theta) A^{-1}. \tag{2.6}$$

This method is the same as solving equation 2.3 using pseudo-inverse in a way that $[\dot{x} - J\dot{\theta}]^T A^{-1} [\dot{x} - J\dot{\theta}]$ is minimized [17]. In this formulation the role of A is to give

higher weight to those components of task space which have higher priority. However, considering redundancies in the system the generalized solution which includes null space motion and projection of the Jacobian to the null space using the generalized inverse is:

$$\Delta\theta = J^{\dagger}\Delta\vec{x} + (I - J^{\dagger}J)\varphi, \tag{2.7}$$

where for any vector φ we still obtain a value for $\Delta\theta$ which minimizes the value $J\Delta\theta - \vec{x}$. Therefore, φ is desired joint motion that is projected into null space of the Jacobian.

The logic behind using the Jacobian to find the IK solution is an iterative solution. Assume that $p_{current}$ and p_{goal} are known. Defining $\Delta p = p_{goal} - p_{current}$, we have:

$$\Delta\theta \simeq J^{-1}\Delta p, \tag{2.8}$$

and

$$\theta_{current} = \theta_{previous} + \Delta\theta, \tag{2.9}$$

where, if Δp is chosen to be a small step, eventually $p_{current}$ converges to p_{goal}. Figure (2.1) illustrates the concept of Gradient Descent for the function $f(x) = g$.

Although f and consequently J are theoretically not guaranteed to be always invertible, in practice, a physical chain will never be exactly in a configuration that results in a singularity. The performance of IK solutions when a chain is near a singularity configuration varies widely.

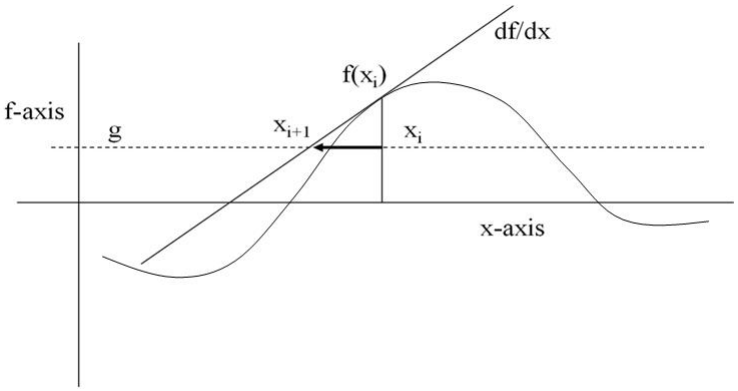


Figure 2.1: Gradient Descent for the function $f(x) = g$.

Basically there are two established techniques for Jacobian Based Inverse Kinematics: pseudo inverse with explicit optimization and the extended Jacobian method. First we describe the pseudo-inverse method. Pseudo-Inverse methods is using Moore-Penrose inverse for non-square Jacobian. Liegeois [18] suggested a more general form of optimization with pseudo-inverses by minimizing an explicit objective function g in the null space of J:

$$\dot{\theta} = J^{\#}\dot{x} - \alpha(I - J^{\#}J)\frac{\partial g}{\partial \theta}. \tag{2.10}$$

As a general problem, pseudo-inverse methods are not conservative, i.e., a closed path in end effector space does not guarantee a closed path in joint space [20]. Additionally, it is not always easy to determine the constant α in (2.10) which controls the influence of the optimization function g. These problems motivate researchers to use Extended Jacobian method.

In Extended Jacobian method Lagrangian approach is used to solve the constraint optimization problem. Let matrix L span the null space of the Jacobian matrix, and

therefore the optimality condition would be

$$L^T \frac{\partial \phi}{\partial q} = 0. \tag{2.11}$$

Projection of the objective function gradient onto the null space in an extended Cartesian space vector is introduced by Baillieul [19] to be

$$X = \begin{bmatrix} f(q) \\ L^T \frac{\partial \phi}{\partial q} \end{bmatrix}, \tag{2.12}$$

and the solution would be optimum since an extended desired Cartesian position is defined to be

$$X_d = \begin{bmatrix} x_d \\ 0 \end{bmatrix}. \tag{2.13}$$

An extended forward kinematic model relates the new task vector X to the joint angle vector q by

$$X = F(q) = \begin{bmatrix} f(q) \\ L^T \frac{\partial \phi}{\partial q} \end{bmatrix}, \tag{2.14}$$

and therefore the corresponding extended Jacobian matrix, $J_e = \frac{\partial F}{\partial q}$, would be square. Furthermore, with this representation the kinematic system is not redundant anymore because the dimensions of the augmented task space and joint space are the same. The limitation with this solution is that the analytical expression of the inverse geometric

model results from a very complicated or even impossible inversion of a non-linear function of constraints with variables changing rapidly during the arm movement.

IK solution using Jacobian transpose is also proposed by some researchers [21, 22], but it does not have the Inverse Jacobian method's popularity. The basic idea is very simple: use the transpose of J instead of the inverse of J. That is, we set $\Delta\theta$ equal to

$$\Delta\theta = \alpha J^T \Delta p, \tag{2.15}$$

for some appropriate scalar α. Now, of course, the transpose of the Jacobian is not the same as the inverse; however, it is possible to justify the use of the transpose in terms of virtual forces. Computing transpose it is much faster than computing the inverse or pseudo-inverse and it has the effect of localizing the computations. To compute Δp_i for joint i, we compute the column i in the Jacobian matrix, J_i , and then just we use:

$$\Delta p = J^T \Delta\theta. \tag{2.16}$$

With the Jacobian transpose (JT) method, we can just loop through each DOF and compute the change to that DOF directly. With the inverse (JI) or pseudo-inverse (JP) methods, we must first loop through the DOFs, compute and store the Jacobian, invert (or pseudo-invert) it, then compute the changes in DOFs, and then apply the changes. Therefore, the JT method is far friendlier on memory access, and computationally efficient. However, if one prefers quality over performance, the JP method is recommended.

Modular Architecture for Inverse Kinematics Tourassis and Ang [23] formulated a fast method that is applicable to general 6 DOF geometries. The method is based on creating a system of three nonlinear equations in three unknowns using the known forward kinematics and inverse kinematics of the first three joints and the last three joints separately. According to this article, there are basically two classes of robot for which decoupling is possible. The first class are robots which the arm joint motions do not produce orientational side-effects. The second group are robots which involves the wrist geometry. In fact, the geometries identified by Pieper [6] which are capable of decoupling are subset of these two classes of manipulator geometries, for which decoupling is a special case of modularity.

Human-Like Motion with Minimizing Potential Energies In this approach it is examined how human muscles deal with problem of positioning. To find the physiological characteristics and constraints that shape human arm motion the potential energies associated with its motion were studied [24, 25]. These characteristics are then mapped for robotic control with potential energies as a factor to prioritize task-level control framework. To this end, the defined muscle effort criterion characterizes effort expenditure in terms of musculoskeletal parameters. Furthermore, muscle fatigue or strength can be simulated within the muscular effort criteria by altering the muscle parameters in the model.

2.2 Motion Planning In Humanoid Robots

Despite the smooth and easy movements of arm in human and animals which makes it the main portion of the body to interact with physical world, manipulator motion planning and control have progressed very slowly in the field of robotics and still has not met the expectations. As an interacting tool with the environment, the most important task of human arm is to pick and carry goods and travel specific trajectories with its end effector and for that purpose it is important to have control on the state of the end effector. Based on some studies on human arm it was observed that human motor planning is done in part by minimizing the "cost" associated with certain "uncomfortable" joint angles during a trajectory [26]. In that work, although it is not explicitly mentioned that the human controls the arm on a joint by joint basis, it implies that joint variables are meaningful parameters in monitoring and controlling the arm motion. The big differences between the human arm and the robot manipulator are in their driving motors. In human arm the movement does not come from motors but from muscles that are attached to the limb themselves. Therefore one muscle may influence the movement of two or more joints at the same time. The other obvious difference lies in the feedback signals sent to both systems' controllers. In human there is no sensory organ which can send the instant values of joints variable and velocities, but according to biological research this form of feedback is used in a more complex control system instead of its direct use, such as visual feedback. A research by Hogan [27] on the mechanics of arm motion suggested biologically sensible "spring-like" model for limb movement. Under this assumption, joint variables are not specified directly, but instead, are the result of the parameters of the spring (equilibrium position, stiff-

ness, and damping) and properties of environment and the arm (gravity, inertia, end load). This assumption has its own limitations. Although it can simulate the biological movements, but the sensory information from the system is much less complex than the information provided by human muscles.

Recently the energy consumption model for muscle models of manipulators has been of special interest. Since the motor energy consumption in robots is not a critical problem, this question raises that why it has been of close attention in motion optimization and robot learning. Adam's research [28] has given credit to this model by stating that existence of this model will help the robot in understanding the human motions and has a great influence on its learning for two reasons. Firstly, it helps the robot to have a sense of tiredness. Although the robot is not expected to run out of energy during its tasks, but this helps the robot to differentiate between the failures of human in some actions due to fatigue or other reasons. Secondly, having an understanding of limited energy, prevents the robot to demonstrate superhuman abilities. For example, the robot would find out that there are different reactions inside, during an intensive work from a slower but lengthy motion. All these factors together, make the robot movement more natural. Furthermore sharing the same optimization indexes with human, makes most of the human actions meaningful for robot and makes the learning process easier.

One of the necessary requirements for the robot is to be able to integrate its sensors data of the joint's variables and the data from its vision system which sees the location of the limb with respect to its global frame in the body. This relation is referred to Forward Kinematics in robotics. In the other hand, robot should be able to choose

a set of joint variables to lead its end effector towards the target detected by its vision system. This problem is known as Inverse Kinematic problem and has been one of the early challenges in the field of robotics. Humanoid robots have interesting associated inverse kinematic problems because their redundancy causes the solution of this problem not to be unique and therefore it is possible for the robot to touch or grasp an object in different ways, i.e. with the elbow up or down or shoulder and wrist bent in different directions. Redundancy is an advantage for the humanoid since it allows the system to avoid obstacles and joint limits, as well as actuator limits and singularities. But, on the other hand, redundancy makes the control and learning procedure very complicated. To solve this problem several techniques are proposed which we are going to describe the most important ones here, very briefly.

One approach to solve the inverse kinematics of redundant manipulators suggested by Heuristic methods is to freeze DOFs to eliminate redundancy. However a smarter approach is to utilize the redundancy in a way to fulfill our expectation from the manipulator motion as much as possible. Therefore redundant manipulators can be used to optimize additional constraints and solve inverse problem by imposing an optimization criterion with a global optimum:

$$f = f(\theta). \tag{2.17}$$

This optimization problem can be solved either globally or locally. In the following subsection advantages and limitations of these two approaches are discussed.

2.2.1 Global vs. Local Approaches in Constraint Optimization Problems

As discussed before, the problem of inverse kinematics and collision avoidance for redundant manipulators can be formulated as a constraint optimization problem since a minimum distance should be kept between the manipulator body and obstacles. Global and local optimization methods are introduced to solve this problem, each of them having some advantages as well as limitations. Global methods [29], as it is indicated by their name, are optimizers over the entire path. They are calculated only once for a given map of obstacles which itself imposes a high computation cost on the planner due to the high dimensionality of configuration space (usually 6 dimension), and complexity of configuration space obstacles. They are therefore, not applicable to dynamic environments and online applications which require fast and highly interactive operations. In contrast, in local methods [30], which are feasible in real time, joint variables are set successively as the robot moves along its trajectory, only based on local information. They only compute optimal changes in θ, $\Delta\theta$, for small changes in x, Δx. Furthermore, it is possible to integrate online path planning into trajectory control. This makes the local methods the only solution for dynamic environments.

2.2.2 Pseudo-Distance

Pseudo-distance is an alternative concept to reduce the cost of Euclidean distance computation for complex objects avoidance. Once we choose our method of optimization, we have to develop the optimization factors leading to introduce the optimization

function, called $f(\theta)$ here. Recalling that the collision avoidance is one of the key constraints on the manipulator, Euclidean distance has been usually used in the literature to formulate this constraint [31]. It is very difficult to obtain the shortest distance to complex objects, since based on the sensor information derivation of the exact equation of their surface in the space is not possible. Therefore the minimum distance has to be evaluated for each particular case and also finding the minimum distance to a complex object is itself the result of an optimization method. An alternative solution is the polyhedral approach [32, 33], which gives a rough estimation of the surface of difficult objects. Another useful concept is to use pseudo-distance instead of exact distance. Concept of distance in collision avoidance is just a tool to avoid the robot parts to pass a boundary around the object. Therefore, if the robot is outside that boundary, its distance to the obstacle in not important and this approach would reduce the computation cost significantly. The idea is to introduce a function describing the object surface, or estimating the surface with a hyper-surface of known analytical expression, and check the position of the point with respect to the surface. There are two factors in selection of surfaces to pay attention. These surfaces should be differentiable and simple, and at the same time should describe the object closely to avoid reduction of workspace. The method proposed by Perdereau et al. [34] describes the surface or envelope of an object as a hyper-surface whose analytical expression is known. Therefore, the problem of surface function is resolved and a large number of obstacles may be approximated in a very simple way. The expressions of hyper-surfaces are defined by ellipsoids as given in equation 2.18:

$$S(x,y,z) = \left(\frac{x}{f_1(x,y,z)}\right)^{2u} + \left(\frac{y}{f_2(x,y,z)}\right)^{2v} + \left(\frac{z}{f_3(x,y,z)}\right)^{2w}, \tag{2.18}$$

where (x, y, z) represents the coordinates of the point from which it is desired to calculate the pseudo-distance to the ellipsoid, f_1, f_2, f_3 are functions giving the desired shape (cylinder, cone . . .) obtained from the approximated ellipsoid, u, v and w represent parameters to fine tune the desired shape to the envelope of the ellipsoid. A closer envelope implies a better description of the obstacle at the cost of a longer computation time. Examples of some Super-Quadratic Surfaces are shown in Figure 2.2.

In approximating the surface, we should balance between closer surface to the object which increases the complexity, and reduction of workspace due to inexact approximations. To describe the idea of pseudo-distance, consider an object closely approximated by a surface equation:

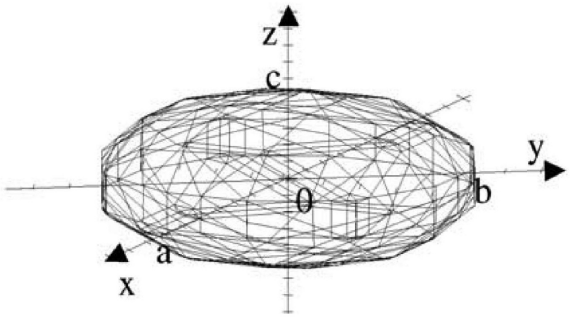
$$S(X) = 0. \tag{2.19}$$

To check the collision of the object with any point of the robot, it is sufficient to evaluate $S(X_0)$, where X_0 is the coordinate of any point in the Cartesian space and compare the result with the following conditions:

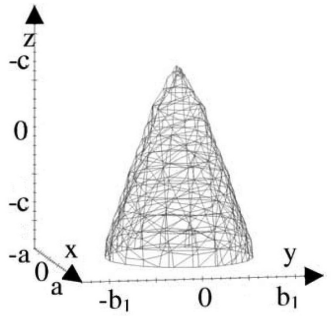
$S(X_0) < 0$ if X_0 is inside the object,

$S(X_0) = 0$ if X_0 is on the object surface,

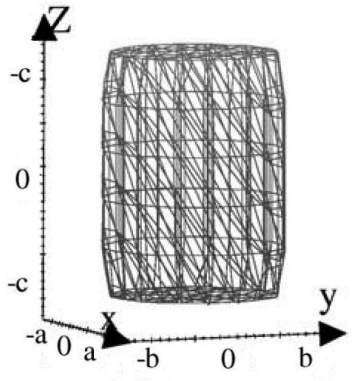
$S(X_0) > 0$ if X_0 is outside the object.



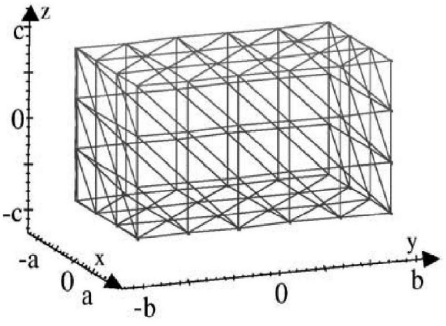
$$S(x, y, z) = \left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 - 1.$$



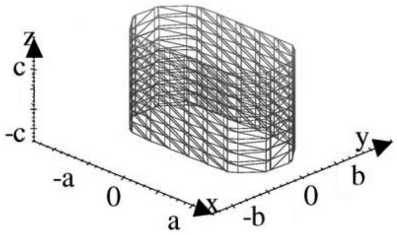
$$S(x, y, z) = \left(\frac{x}{mz + d}\right)^2 + \left(\frac{y}{mz + d}\right)^2 + \left(\frac{z}{c}\right)^{2n} - 1.$$



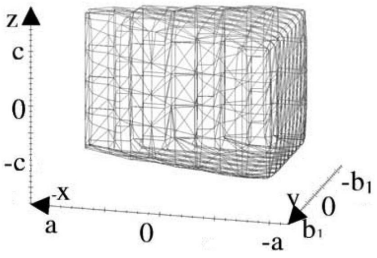
$$S(x, y, z) = \left(\frac{x}{r}\right)^2 + \left(\frac{y}{r}\right)^2 + \left(\frac{z}{h}\right)^{2n} - 1.$$



$$S(x, y, z) = \left(\frac{x}{a}\right)^{2n} + \left(\frac{y}{b}\right)^{2n} + \left(\frac{z}{c}\right)^{2n} - 1.$$



$$S(x, y, z) = \left(\frac{x}{a}\right)^{2l} + \left(\frac{y}{b}\right)^{2m} + \left(\frac{z}{c}\right)^{2n} - 1.$$



$$S(x, y, z) = \left(\frac{x}{m_1 z + d_1}\right)^{2n} + \left(\frac{y}{m_2 z + d_2}\right)^{2n} + \left(\frac{z}{c}\right)^{2n} - 1.$$

Figure 2.2: Super-Quadratic Surfaces with Ellipsoid as Basic Shapes.

Applying these conditions eliminates the necessity of exact calculation of Euclidean distance and the resulting pseudo-distance is used as a basis of the constraint function which will be discussed later.

At this stage, the pseudo-distance should be utilized for manipulator's part collision avoidance. The question rises here is that how many and which points of the manipulator should be examined by function $S(X_0)$. Assuming a manipulator with several links, each link can be approximated with a vector of points. Although by making this assumption the volume of the link is ignored, it can be taken into account as a safety margin in constraints definitions. It is necessary to find the closest point of a single link to the surface to reduce the cost of computation in the following stages. The closest point of a link is the result of one parameter optimization, which has a unique solution due to the convexity of object surface approximated by elliptic basis. To form this one parameter optimization problem, coordinates of any point of the link L_1L_2 is given in the reference frame as follows:

$$\vec{OX} = \vec{OL_1} + \lambda \vec{L_1L_2}, \tag{2.20}$$

where λ is the optimization variable between 0 and 1. In order to find the optimum solution, λ_m, which gives the closest point of the link to the object, $\vec{OX}$ is inserted into the function S, and therefore S would be function of λ only and the rest of parameters are constants. Then, in order to find the optimum λ, $\frac{d(S(\vec{OX}))}{d\lambda} = 0$ is derived and solved for λ. Small or zero λ_m indicates that the minimum distance is close to or on the L_1 point and vice versa.

In a work by C. J. Ong and E. G. Gilbert [35] concept of Growth distance for

separation and penetration is presented which has desirable properties and is computationally efficient. However, the physical meaning of penetration is not as clear as separation. Generally, penetration depth refers to the depth of intersection for object models. In past research penetration distance is defined in a different way which roughly speaking was the shortest relative translation of the two objects, measured by Euclidean norm, that causes them to move apart from each other to have no interior point in common. In the work by C. J. Ong and E. G. Gilbert, however, Growth distances are a measure of how much each of the objects must be grown, outward from fixed seed points in their interior, so that they just touch. Then, the difference between penetration distance and separation distance is that in the former the grown objects are smaller than the actual object and in the later the grown object is larger than the actual object. As mentioned, using the concept of growth distance have some desirable properties such as invariance with respect to the choice of coordinate system in which two objects are represented, and simple characterization of the derivatives of the distance with respect to the configuration variables.

An approach for indoor robot navigation in relatively small environments is using Distance Transform Methodology (DT), proposed by Jarvis [36]. In this method the destination cell in the tessellated map is given a distance propagation cost of zero for all time instance transform. The cells corresponding to obstacles are given infinity distance propagation cost and the rest of the cells are initially assigned with a large distance propagation cost, but their value would be updated at each iteration of a loop for evaluating the propagation distance cost for every cells. In this loop distance propagation cost of each cell is derived based on its previous cost and also the propagation

cost of the surrounding cells.

2.2.3 Optimization Criteria

The necessary condition for imposing optimization criterion on manipulators is their redundancy. Basically, redundancy means that robot is able to do internal movements without influencing the end effector pose. This extra freedom allows the robot to perform auxiliary tasks such as, obstacle avoidance and optimization of the robot's energy. Generally the main task for manipulators is path tracking. There is a vast literature treating the issue of selecting the optimization criterion. Saramago and Ceccarelli [37] solved the problem of manipulator motion, taking into account the robot actuating energy and grasping forces in the manipulator gripper. The energy was calculated by considering mechanical power spent in actuators for manipulator motion, and energy for gripper actions. The optimization problem was formulated through physical constraints, input torque/force constraints and payload limits. Gasparetto and Zanotto [38] proposed new method which worked through an objective function containing a term proportional to the integral of the squared jerk (to ensure that the trajectory is smooth) and the second term, proportional to the total execution time. Therefore, there is no need to define the total execution time before running the algorithm. With respect to trajectory optimization techniques, they tried to reach to the minimum execution time, minimum energy and minimum jerk. Saramago and Steffen [39] introduced a multi-objective function using optimal traveling time and the minimum mechanical energy of the actuators. The problem of minimum cost trajectory planning was also studied by Chettibi et al. [40]. They minimized the cost

function for the motion between two points in the operational space taking into account dynamic equations of motion as well as bounds on joint positions, velocities, jerks and torques. Zha [41] presented a unified approach to optimal pose trajectory planning for robot manipulators in Cartesian space using a genetic algorithm (GA)-enhanced optimization. Aspragathos [42] reported two techniques for manipulator Cartesian trajectory generation. Both techniques generate an approximation of a given robot hand trajectory under bounded position deviation. The first technique is based on bisection pattern determining enough knot points to generate a trajectory of the hand tip of a manipulator under bounded position deviation. The other technique is based on the raster scanning to find a minimal set of knot points on a given Cartesian curve in order to generate a trajectory with bounded position approximation error. Between two knot points, spline functions were used.

Above methods are efficient only in environments without obstacles. In other environments the most important auxiliary task would be obstacle avoidance. The literature available for collision avoidance is both applicable to mobile robots and robot manipulators. Traditional methods such as Artificial Potential Field introduced by Khatib [43] propose a concept efficient for both cases. Agirrebeitia et al. [44] used the concept of artificial potential fields for the planning of mobile robot motion for highly redundant multibody systems for 2D and 3D environments, as well as static or dynamic obstacles. Valero et al.[45] used an algorithm capable of obtaining a sequence of feasible robot configurations between the given initial robot configuration and the robot goal configuration to plan the trajectory for industrial robots in work spaces with obstacles. Saramago and Steffen [46] approached this problem in the operational

space of manipulator. They considered actuator constraints, joint limits, non-linear manipulator dynamics and obstacle avoidance in building a multi-criterion function to optimize traveling time and mechanical energy of the actuators.

Yao and Gupta [47] presented an example of constraint manipulator, where the end effector was constrained to move in a vertical plane in order to move a glass of liquid to a desired place. They address the problem of path planning with general end effector constraints (PPGEC) with two approaches. One of the approaches called Adapted-RGD, will be presented here. This method is adapted from a randomized gradient descent (RGD) method originally proposed for closed-chain robots [48]. They modified the model to use for open chain manipulators by breaking the closed chain and imposing closure constraint at the break point which adds to the existing constraints of the end effector. First the definition of Pose is given as a pair $(P,O) \in SE(N)$, where $P \in R^N$ and $O \in SO(N)$ are the position and orientation of end effector in global frame. The end effector constraint is denoted by

$$G_i(K(\tau)) = 0 \quad \forall \tau \in [0,1], \quad i = 1,2,\ldots,m, \tag{2.21}$$

where $K(\tau)$ is is the end effector pose and all the G_i functions are continuous in task space. As an example the constraint function for the manipulator to move in vertical plane would have the form of

$$G(K(\tau)) \quad = \quad \begin{cases} ax(\tau) + by(\tau) + cz(\tau) = 0 \\ O(\tau). \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \end{cases} \tag{2.22}$$

The proposed solution for this constraint problem is composed of two stages: Generate Feasible Configuration and Connect Feasible Configurations. The former is responsible for generating feasible configurations by means of randomized gradient descent method which randomly selects a set of joint variables, q_t, and searches in q_t neighborhood for the next selection q_{t+1}, which reduces the cost function. The cost function can be defined as integration of distance to goal position and orientation frame. Cost of position error is represented by $h_1(q)$ which is the distance between the current end effector frame to the goal end effector frame,

$$h_1(q) = D(P_e, P_g) = \sqrt{(x_e - x_g)^2 + (y_e - y_g)^2 + (z_e - z_g)^2}. \tag{2.23}$$

Cost in orientation error is represented by $h_2(q)$ which is the coordinate frame distance between the current end effector frame to the goal end effector frame,

$$h_2(q) = D(N_e, N_g) = \sqrt{d(\hat{x}_e, \hat{x}_g)^2 + d(\hat{y}_e, \hat{y}_g)^2 + d(\hat{z}_e, \hat{z}_g)^2}, \tag{2.24}$$

where N_e and N_g are the current frame and goal frame of end effector respectively and

$d(\hat{x}_e, \hat{x}_g)$, $d(\hat{y}_e, \hat{y}_g)$ and, $d(\hat{z}_e, \hat{z}_g)$ are the distances between the vertices of frame axes unit vectors.

Luo et. al. [49] presented a comprehensive analysis of general constrains and performance indexes. In this work trajectory of the robot is regarded as a sequence of points which form a cubic polynomial. The position, velocity, acceleration and, jerk at each point of this polynomial is expressed in terms of $P_i(t), \dot{P}_i(t), \ddot{P}_i(t)$ and $\dddot{P}_i(t)$ $(t \in [t_i, t_{i+1}])$. Assuming that $\ddot{P}_i(t)$ is known the other derivations as well as position can be expressed in terms of $\ddot{P}_i(t)$ and h_i which is the i^{th} time interval. Using Lagrange formulation, the generalized force of robot manipulator at each joint, u_i, is obtained. At this stage the performance index can be defined. It is chosen to be a function which optimizes the operation time and mechanical energy consumption:

$$\min TF = \zeta_1(\sum h_i) + \zeta_2(\alpha \centerdot \sum (u_i^T \dot{P}_i)^2 h_i), \tag{2.25}$$

where ζ_1 and ζ_2 are weighting coefficients, and $\zeta_1 + \zeta_2 = 1$.

The constraints associated with the movement of the manipulator are given as:

1. Kinematic constraint of robot manipulator: This constraint takes care of the velocity, acceleration, and jerk limitations in the system, expressed as:

$$\left\{ \left| \dot{P}_{ij}(t) \right| \leq VC_j; \left| \ddot{P}_{ij}(t) \right| \leq AC; \, |\dddot{P}_{ij}(t)| \leq JC; \quad \forall t \in [t_i, t_{i+1}] \right\}. \tag{2.26}$$

2. Dynamic constraint of robot manipulator: The generalized force of joint j is

limited to the force/torque constraint FC_j:

$$\{|U_{ij}(t)| \leq FC_j; \quad \forall t \in [t_i, t_{i+1}]\}. \tag{2.27}$$

3. Collision avoidance: A minimum allowable distance between robot and object is set and in order to simplify the calculation of the object and robot distance, the object surface is approximated by a polyhedron. Furthermore, edges of the polyhedron are dissociated into points and the minimum value among distance between one feature point and the robot would be the minimum distance of object and robot. This constraint is formulated as

$$\left\{\min_{\lambda}\left\{\left\|\widetilde{f}(P_{ij}(t)) - q_{\lambda}^{B}\right\|;\, q_{\lambda}^{B} \in FP(B),\, \lambda = 1, 2, \ldots, |FP(B)| > \varepsilon\right\}\right\}, \tag{2.28}$$

and the proposed algorithm to solve this problem is Evolutionary Programming (EP).

With this review on various optimization criterion we are going to explain the potential field method more in detail, as it is the main inspiration of the algorithm proposed in this work.

2.2.4 Potential Field Method

In the artificial potential field approach, the obstacles to be avoided are represented by a repulsive artificial potential, and the goal is represented by an attractive potential so that a robot reaches the goal without colliding with obstacles. Unlike many search methods, this approach is computationally suitable for real-time implementation of

higher degrees of freedom manipulators. The artificial potential approach, however, has been limited due to the existence of local minima and its inability to deal with arbitrarily shaped obstacles. Numerous investigators have attempted to use potential functions in various robotic applications. These works mainly modified the potential field method in three categories: 1) addressing the problem of undesired local minima, for which the main class of treatments include: i) the redefinition of the potential functions with no or a few local minima. Other examples are generalized potentials [50] or Laplacian approach [51]. Furthermore, some special potential functions can be used to solve the local minima problem, for example repulsive potential functions with circular thresholds [52], or Gaussian shapes [53] and the navigation function [54, 55]. These methods improve the field in a way that it would only have one global minimum and no local minima. The problem with these methods is that they can only be utilized in environments with simple geometries or those which have been bounded by hyperplanes conservatively; otherwise configuration space construction would be needed. Also some numerical methods based on grid map is proposed by [56]. ii) The utilization of efficient search techniques. In this approach a variety of searching algorithms have also been applied for escaping from local minima, for example valley-guided and random which also use grid map and have high computation costs and therefore not suitable for real-time implementations. 2) Viewing potential functions as an path-planning aid [57, 58]. 3) Improving the use of potential functions to time-varying situations to avoid moving obstacles.

In the famous paper by Khatib [43], it is stated that the control would take place in operational apace rather than joint space as there would be no need for geometric

and kinematic transformation. The manipulator equation of motion in the task space is given in the form of

$$F = \Lambda(x)F^* + \mu(x,\dot{x}) + p(x), \tag{2.29}$$

where Λ is the symmetric matrix of the quadratic form in expression of kinetic energy, $T(x,\dot{x}) = \frac{1}{2}\dot{x}\Lambda(x)\dot{x}$, and F^* is the single unit mass which is the command vector to the control system. As described above, an artificial potential field exerts repulsive and attractive forces from obstacles and the goal respectively. The field can be represented by

$$U(x) = U_d(x) + U_o(x) + U_g(x), \tag{2.30}$$

where U_d, U_o, and U_g are the potential energies due to the desired goal, obstacles, and gravity, respectively. The decoupled end effector command vector corresponding to each potential energy is the gradient of that potential in Cartesian space. Therefore, the attractive force which pulls the end effector towards the goal and the force repulsing the end effector from the obstacle can be written as

$$F_d^* = -\nabla U_d(x), \tag{2.31}$$

and

$$F_o^* = \nabla U_o(x). \tag{2.32}$$

The potential field concept is defined on the basis of Euclidean distance and in this work a conventional PD servo is used as a proportional term. The attractive potential can be defined as

$$U_d(x) = \frac{1}{2} k_p (x - x_d)^2. \tag{2.33}$$

There are some factors to consider in defining the potential formulation. The field should be positive continuous and differentiable, because the forces are derived from the first derivation of the potential field. It should also be zero at the goal position in other to make it the global minimum. In order to make the system asymptotically stabilize, dissipative forces proportional to $\dot{x}$ are added. The force exerted on the end effector from the goal can be written as

$$F_m^* = -k_p(x - x_d) - k_p \dot{x}. \tag{2.34}$$

The specifications of the potential function corresponding to obstacles are that the designed U_o function should be continuous, non-negative, and attaining to infinity at the obstacle's surface. Furthermore, to avoid turbulence, the field must be set to zero at a certain distance ρ_0, from an object. The proposed potential function which satisfies all the mentioned conditions would be

$$U_o = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho} - \frac{1}{\rho_0})^2 & if\ \rho \leqslant \rho_0 \\ 0 & if\ \rho > \rho_0. \end{cases} \tag{2.35}$$

The corresponding force derived from the potential field is

$$F^*_{(o,psp)} = \begin{cases} \eta(\frac{1}{\rho} - \frac{1}{\rho_0})\frac{1}{\rho^2}\frac{\partial \rho}{\partial x} & if\ \rho \leqslant \rho_0 \\ 0 & if\ \rho > \rho_0, \end{cases} \tag{2.36}$$

where PSP stands for Point Subjected to Potential, and $\frac{\partial \rho}{\partial x}$ is partial derivative of the distance between PSP and obstacle. The above potential function is proven to be one of the best designed functions. Perdereau et. al [34], investigated the different forms of the potential function and listed their limitations as bellow. First the constraint is defined as an inequality

$$h(q) = d_s^2 - d_m^2 \leq 0, \tag{2.37}$$

where d_s is the Euclidean distance from one point of robot link to the obstacle, and d_m is the minimum distance allowed between them. This minimum distance can also be responsible for the volume of the link as stated in subsection 2.2.2.

- External penalty functions: which is zero if the constraint is satisfied and has positive value if the constraint is violated, for example $p(h(q)) = h(q)^2\Gamma(h)$, $\Gamma(h)$ being the Heaviside function, which has the following properties as shown in Figure (2.3):

$$p(h) > 0 \qquad if\ h(q) > 0 \tag{2.38}$$

$$p(h) = 0 \qquad if\ h(q) \leq 0. \tag{2.39}$$

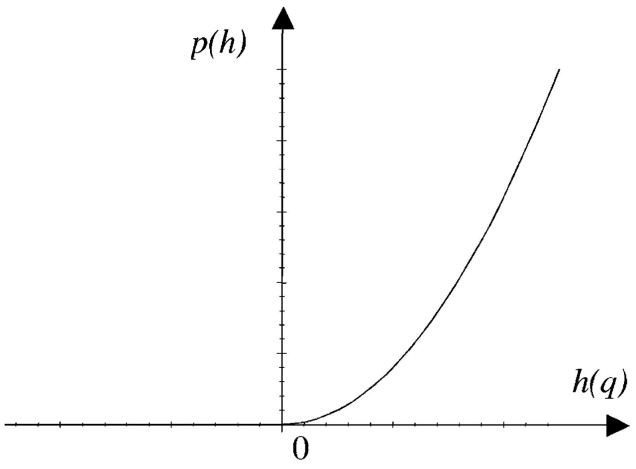


Figure 2.3: External Penalty Functions.

This form of penalty function has undesired behaviors in dynamic environments and exerts sudden torques to the actuators.

- Internal penalty functions: This type of penalty functions, acts even when the constraint is not violated to avoid further collisions. However, existence of this potential field in the whole space and at all the times would reduce the work space. This function has the following properties as illustrated in Figure (2.4):

$$p(h) \;>\; 0 \qquad if\ h(q) < 0 \tag{2.40}$$

$$p(h) \to \infty \qquad if\ h(q) \to 0. \tag{2.41}$$

One example of such penalty functions can be

$$p(h) = -\frac{1}{h(q)}. \tag{2.42}$$

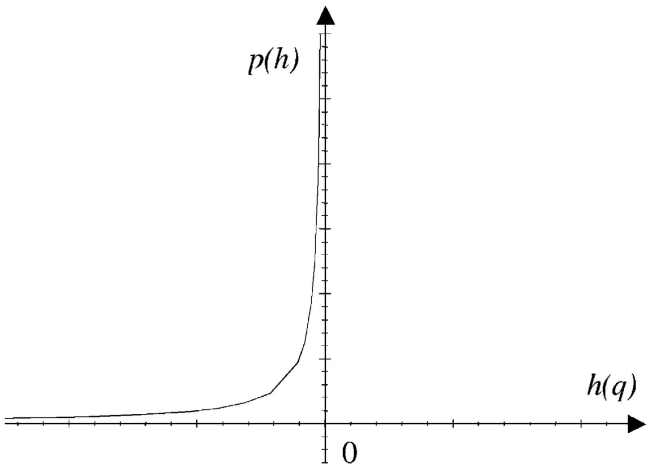


Figure 2.4: Internal Penalty Functions.

- Limited internal penalty functions: They reduce the area of influence of the field by limiting it to a range h_0, so that the robot would not feel the field of obstacles far away. Figure 2.5 illustrates this function. The following penalty function is one of the most important examples of this type which is also proposed by Khatib as mentioned earlier:

$$p(h) = \begin{cases} \frac{1}{2}(\frac{1}{h(q)} - \frac{1}{h_0})^2 & if\ h_0 < h(q) < 0 \\ 0 & if\ h(q) \leq h_0. \end{cases} \tag{2.43}$$

2.3 Summary

In this chapter a comprehensive study on various solutions for Inverse Kinematic problem is presented. Different aspects of this study can be summarized in comparison of global and local approaches, optimization criteria and their indices, as well as analyt-

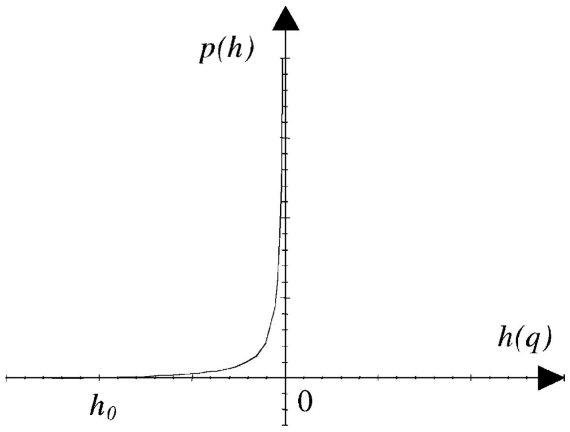


Figure 2.5: Limited Internal Penalty Functions.

ical and numerical methods. It has been shown that each of these approaches is more efficient in some conditions compared to other methods.

Chapter 3

Inverse Kinematics: Flow Algorithm

Social robotics is advancing in many aspects to change the perception of robots from industrial machines with predefined actions to intelligent accompanies in daily life, capable of dealing with human environments. In such environments which are designed for human comfort, robots must be equipped with same tools as human, such as dexterous hands, to be efficient and at the same time, behave naturally. One of the basic skills for a daily human life is object manipulation which is now one of the burdens in the way of development of social robotics. Although the area of manipulator motion planning has been the focus of researchers for more than four decades, all those efforts were mainly dedicated to achieve precise movements of end effector on a given trajectory in machining tasks. The good advancements in that stage have increased the expectations from robot manipulators to merge the planning with artificial intel-

ligence in real-time. It is obvious that all the desired motions of the manipulator in performing its daily tasks in the real environment are not predictable. Especially in human environments there is a high possibility of interaction with human which itself increases the unpredictability. Therefore, the manipulator motion planning must be a real-time module in the robot and it is necessary to compute the inverse kinematic solution in real time as well.

3.1 Algorithm Description

In common inverse kinematic methods, either the geometric relations are used to find the relation of the end effector position in task space with the manipulator configuration in joint space, or the Jacobian matrix is used to relate the small movements of the end effector to small changes in manipulator joints. Result of these methods, together with numerical solutions, which are all fully described in 2.1 and 2.1.3.3, is not similar to human arm motion.

Although many optimization criteria are defined to improve the efficiency of mentioned methods and all of them are necessary for industrial environments, but they are not sufficient for social robots. Since no criteria is introduced to constraint the robot to perform human like movements so far, in this work we proposed a methodology for solving the manipulator inverse kinematic problem and compared the result with human movements under the same condition. Based on experimental data obtained from both human and simulation, and study of the correlation between these two sets of data, we look for closeness of the resulted motion of manipulator to that of human

arm. This method has another significant advantage in providing the solution for any degrees of freedom manipulator with no change in the algorithm and no need for any manual pre-calculations.

To visualize the results, we have used the Robotics Toolbox for MATLAB (Release 7.1), by Corke [61]. In this package we are able to simulate the manipulator and run the algorithm in MATLAB to see the motion of the simulated arm. To simulate the manipulator, each link is defined by its Denavit-Hartenberg parameters in the function LINK. The initial joint values are set in vector q and the desired target position is given in the global frame in the task space, $P_{des} = [x, y, z]'$. The forward kinematic matrix ${}^{n}_{0}A$ is formed using the D-H parameters and the position of the end effector in base frame would be the last column of the transformation matrix, $P_{end} = {}^{n}_{0}A(1:3, 4)$. The index of closeness to the target is defined to be the Euclidean distance between the end effector and the target position in 3D space, $r_{end,des} = P_{des} - P_{end}$. The main loop in the program checks the value of $r_{end,des}$ and repeats the algorithm until this distance becomes less than the given threshold $err_{allowed}$. Description of the algorithm is as follows: For joint i, starting from the end effector to the first joint, the position of end effector and the target are obtained in the coordinate fame attached to joint i, named $r_{joint(i),end}$ and $r_{joint(i),target}$ respectively. By subtracting these two vectors, the resulting vector would indicate the path that the end effector must travel, in case of no obstacle, to reach the target expressed in the coordinate frame attached to joint i.

$$S_{r(joint(i),path)} = r_{joint(i),target} - r_{joint(i),end}. \tag{3.1}$$

Image of the resulting vector in the plane S perpendicular to the joint axis i, Z_i, can easily be found by eliminating the z component of vector $r_{joint,path}$.

$$S_{r(joint(i),path)} = [r_{joint(i),path}(1), r_{joint(i),path}(2), 0]'. \tag{3.2}$$

On the other hand, the only contribution of a revolute joint i in moving the end effector towards the target is along a vector, perpendicular to both Z_i and $r_{joint(i),end}$:

$$V_{joint(i),possible} = Z_i \times r_{joint(i),end}. \tag{3.3}$$

If the joint is prismatic its displacement would be directly mapped to the end effector and therefore the possible path that the end effector can travel due to the displacement in the prismatic joint i would be

$$V_{joint(i),possible} = Z_i. \tag{3.4}$$

At this stage, we have two vectors, one representing the required motion of the end effector in the coordinate frame of joint i, and one the possible path of the end effector generated by change in joint i only. Therefore, the last step would be projecting $S_{r(joint(i),path)}$ on the unit vector parallel to $V_{joint(i),possible}$ to find out the magnitude and direction of the unscaled driving command to be sent to joint i, using inner product. The scaling factor K, then, would be arbitrary and depending on the system.

$$Drive(i) = K.S_{r(joint(i),path)}.V_{joint(i),possible}. \tag{3.5}$$

Note that the step in which we project $r_{joint(i),path}$ in the plane perpendicular to Z_i can be skipped, since equation (3.5) does account for that projection as well, however, it is incuded to clarify the method. Next, the resulting input is sent to the control system and further control strategies are made to send the appropriate command to the joint i actuator. Once the command is sent, the value of joint i would be updated from the encoder signals. Again, the transformation matrices should be updated and from the new ${}^{n}_{0}A$, coordinates of P_{end} would be obtained. The closeness index is recomputed and if it is not yet below the threshold, the loop is repeated. In Figure (3.1), a seven degrees of freedom hand is shown as an example. In this figure, the above mentioned vectors are shown for one of the shoulder joints and in Figure (3.2) the vectors are shown for the elbow joint. The simulation is run for a 10 DOFF arm which the three extra DOFs account for a spherical joint modeling the capsular. In the literature however, 7 DOF arm are common to model human arm but our model is made more complex to show the efficiency of the proposed algorithm. In Figure (3.3) history of joint angles from the initial to the final configuration is shown. Table (3.1) lists the definition of each joint in the simulation together with their initial values.

Figure (3.4) shows the graphical simulation of the humanoid arm and respective position of the target in the space. In the following chapter the efficiency of the proposed method in providing a computationally low cost algorithm which results in a more natural motion is verified by comparing the algorithm results with experimental results.

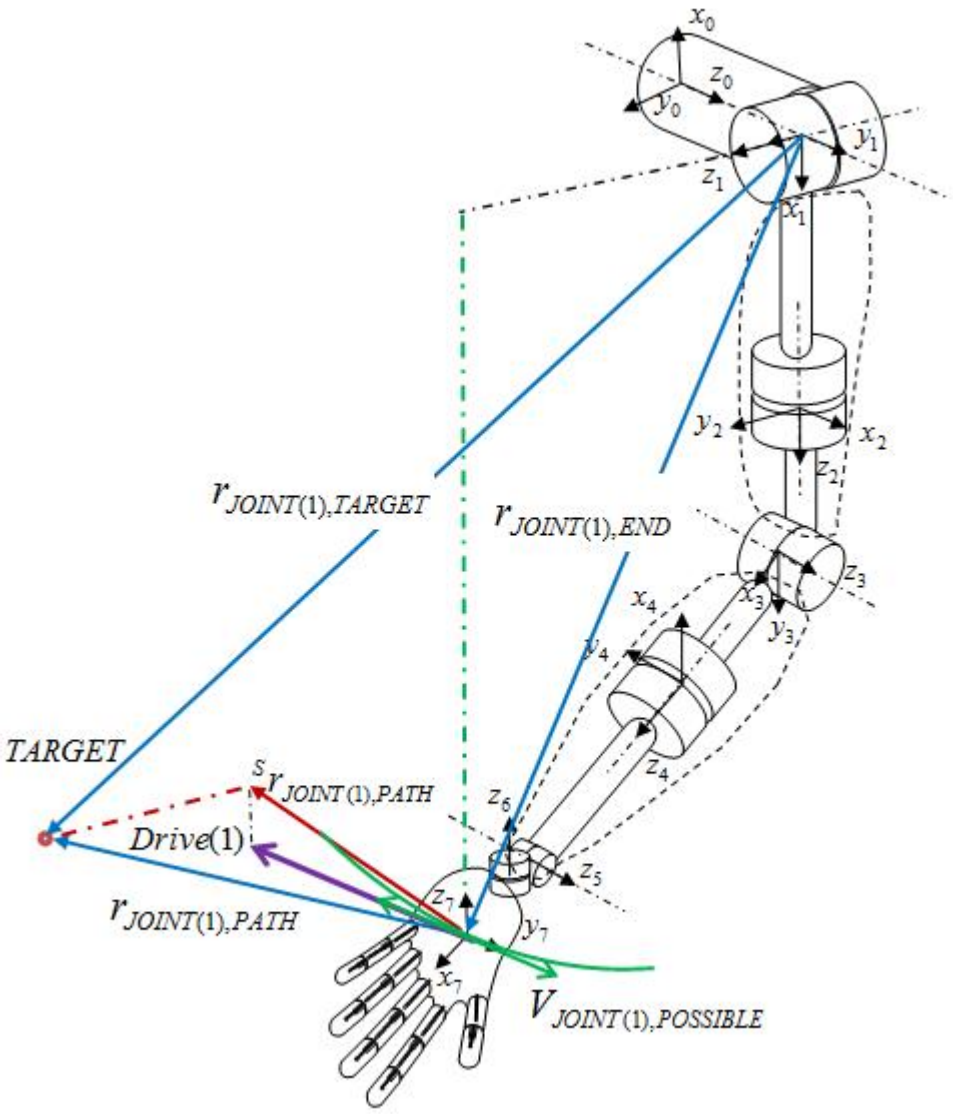


Figure 3.1: Flow Algorithm Key Vectors Illustration for Joint 1 (Shoulder).

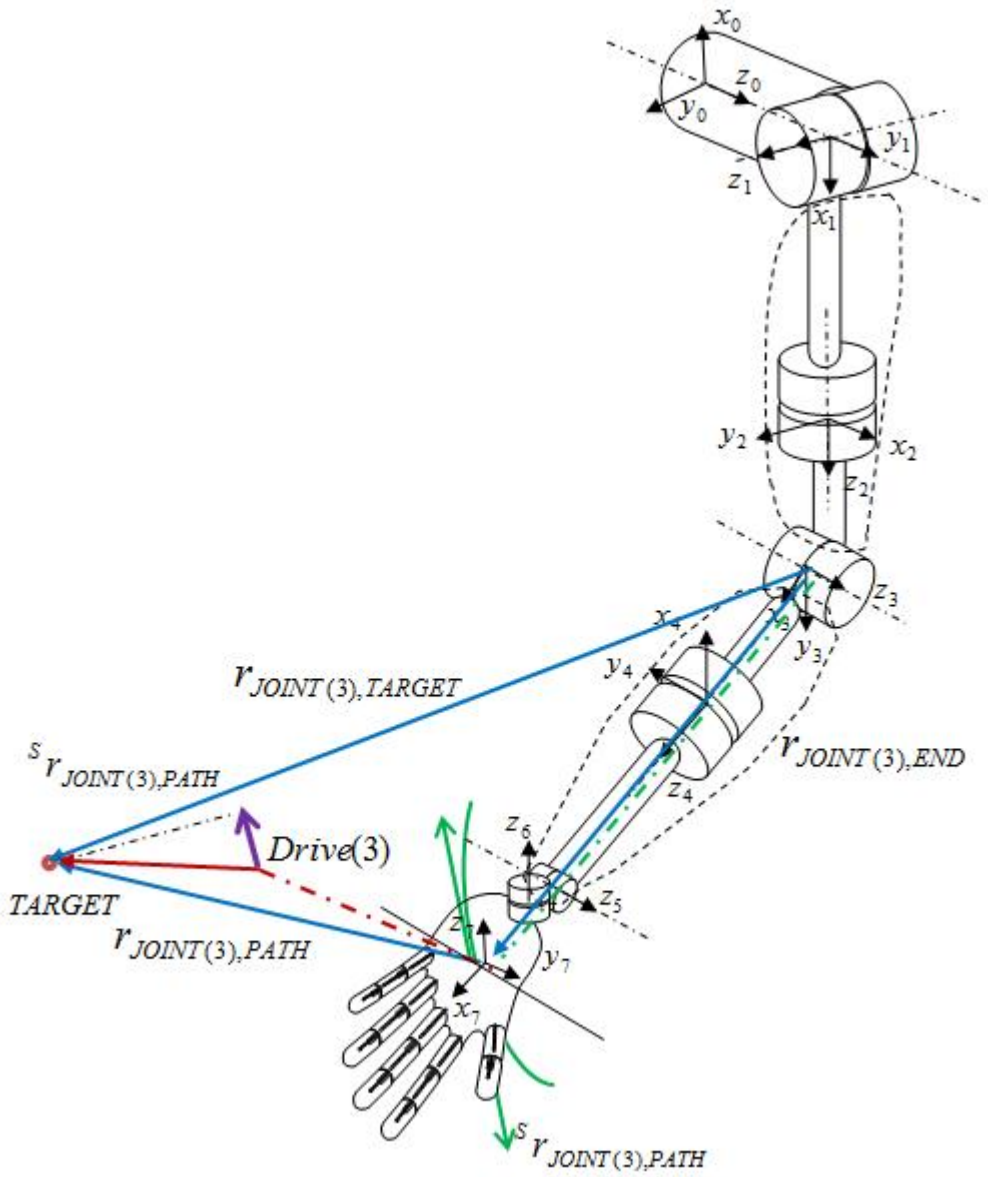


Figure 3.2: Flow Algorithm Key Vectors Illustration for Joint 3 (Elbow).

Joint Variable	Resulting Motion	Initial Value
q_1	Capsular joint: Shoulder horizontal flexion/extension	$\pi + \pi/12$
q_2	Capsular joint: Shoulder vertical flexion/extension	$\pi/2$
q_3	Capsular joint: Full-range shoulder free motion	$\pi/2$
q_4	Shoulder adduction/abduction	$\pi/2 - \pi/12$
q_5	Shoulder flexion/extension	1
q_6	Shoulder interior/exterior rotation	$\pi/2$
q_7	Elbow flexion/extension	0
q_8	Elbow rotation (supination/pronation)	0
q_9	Wrist flexion/extension	0
q_{10}	Wrist ulnar/radial deviation	0

Table 3.1: Joints Variable Definition and Initial Values

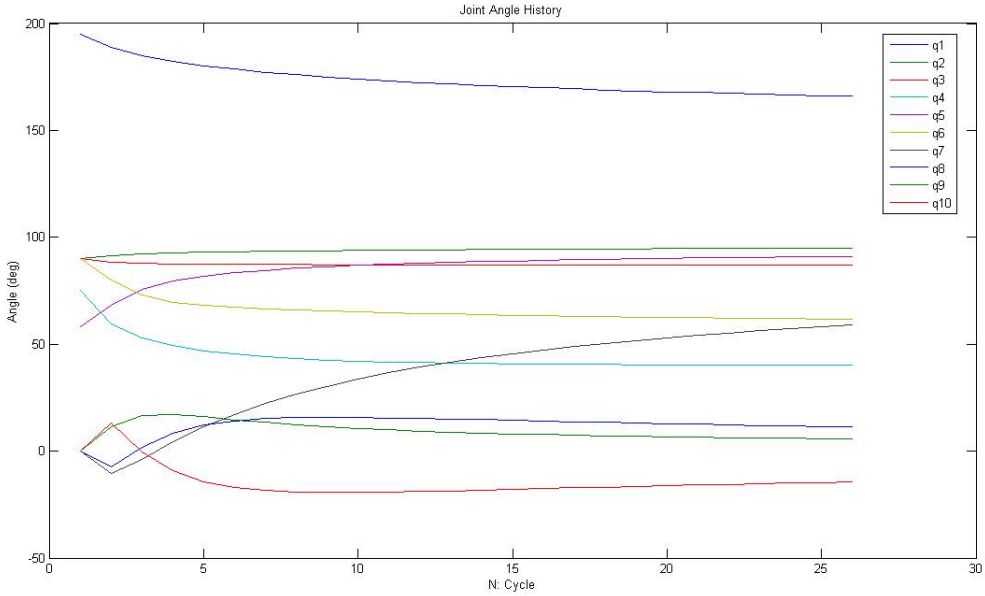


Figure 3.3: Joint Angle History.

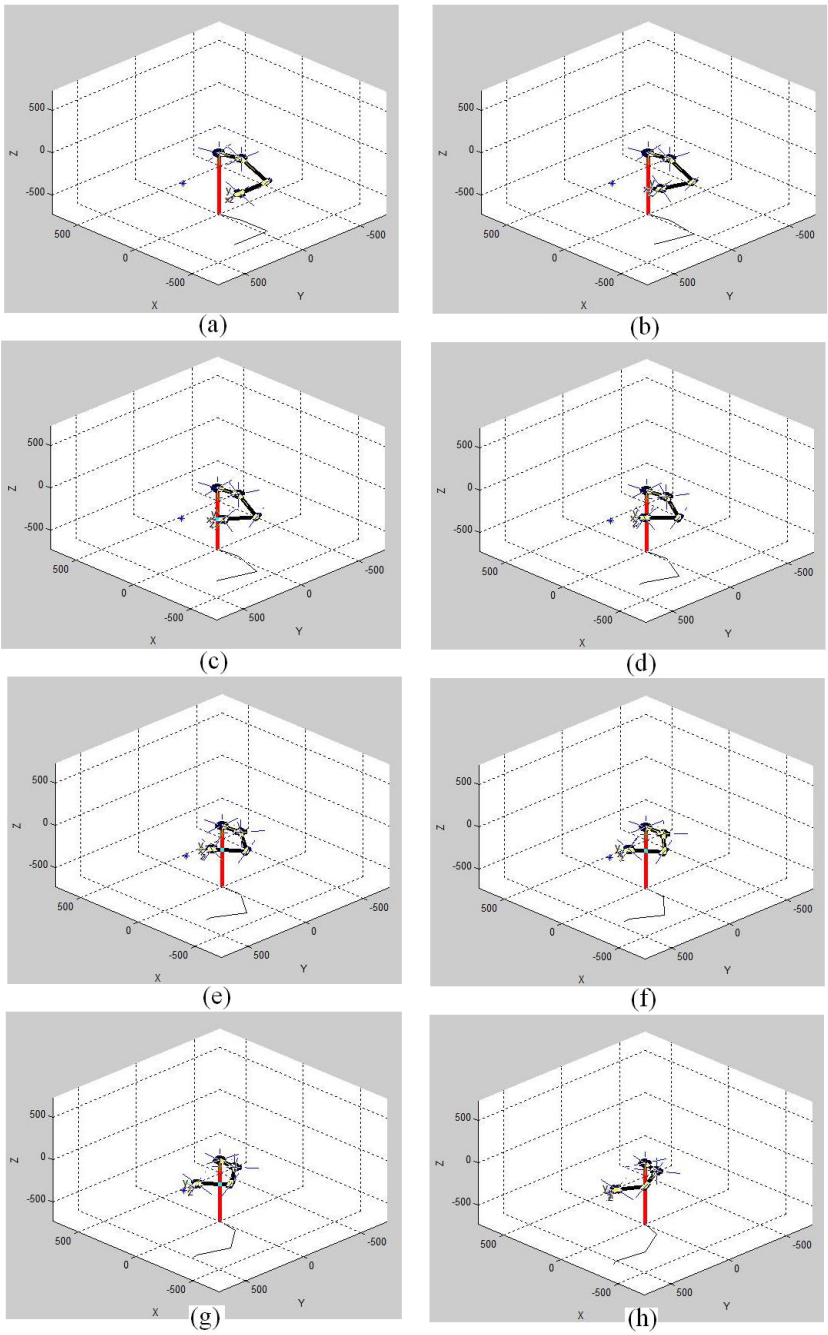


Figure 3.4: Simulated Humanoid Arm Approaching the Target.

3.2 Convergence

An iterative algorithm to solve the inverse kinematic problem is presented. However, the numerical results are not supposed to be perfect after changing joint angles a finite amount. The error inherent in the method is not obvious but it is well known that, if the interval size is reduced to zero, result will reach the ideal value. The order of error associated to this method can be obtained by using Taylor series expansion. Assume the kinematic function f in vector form is expanded as follows:

$$f(q+\Delta q) = f(q) + \frac{\partial f(q)}{\partial q}\Delta q + \frac{1}{2!}\frac{\partial^2 f(q)}{\partial q^2}\Delta^2 q + ... \tag{3.6}$$

In the presented algorithm, $f(q+\Delta q) - f(q)$ indicates the end effector distance to the target, which is called $r_{joint(i),path}$. If the vector indicating the position of the end effector with respect to the origin of frame i is shown by $r_{joint(i),end}$, then according to principles governing vectors in rotating frames, the derivative of kinematic function with respect to i^{th} revolute joint variable would be $\frac{\partial f(q)}{\partial q_i} = Z_i \times r_{joint(i),end}$ or Z_i for prismatic joints.

Let us define the deviation d as the error between the desired value $f(q)+\Delta f$ and the actual value produced by numerical calculations. Since in Flow algorithm is based on the first-order term of (3.6), the deviation in this case would be quadratic in Δq.

$$d = [f(q)+\Delta q] - f(q+\Delta q) = -\frac{1}{2!}\frac{\partial^2 f(q)}{\partial q^2}\Delta^2 q + ... = O(\Delta^2 q). \tag{3.7}$$

Based on the above principles it is useful to plot the deviation against various step sizes of Δf, i.e. to plot d corresponding to $\Delta f/n$ for increasing values of n.

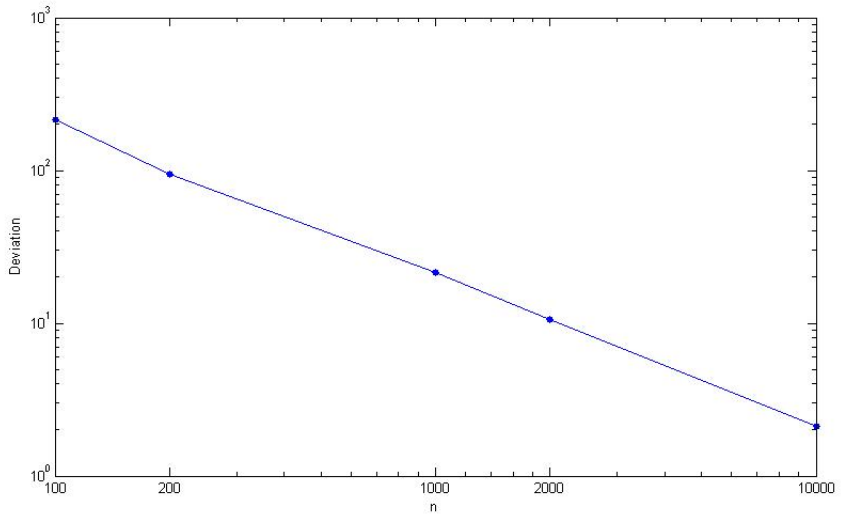


Figure 3.5: Deviation d of the 10 degrees of freedom manipulator for various step sizes in the Flow algorithm.

Therefore, we need to run the program with different values of n and measure the average deviation. The result is illustrated in figure (3.2) which shows the convergence rate of 0.107. In the next section other properties of the Flow algorithm are discussed.

3.3 Discussion

Although convergence is one of the most important properties of a numerical method, there are other important issues associated with numerical methods of solving inverse kinematic such as singularities, existence of solution, and multiple solutions. In analytical methods, however, the mentioned issues can be predicted and appropriate decision can be made to overcome problematic situations. Singularity is a well studied topic in robotics. Many authors have developed useful algorithms to handle singular-

ities. In a work by K. Anderson and J. Angeles [62, 63], many aspects of this issue is studies. However, detailed investigation on handling singularities is not in the scope of this thesis. It is probable that the desired end effector position which is given as the input, not necessarily be in the robot workspace and therefore the algorithm would not converge. To deal with this problem we can either check if the assigned task is in the workspace or stop the algorithm if the error in the position of the end effector was not decreasing for m consequent iterations, where m must be chosen appropriately. In complex manipulators with mobile base the cost of computing the workspace is rather high and watching convergence rate of the algorithm would be less costly. Otherwise, it is a better idea not to start the algorithm unless the existence of the assigned task in the workspace of manipulator is verified. Issue of existence of multiple solutions mainly occurs in analytical methods, where based on some criteria one solution must be chosen. In most of applications the solution which requires less effort to reach, i.e., the closest solution is chosen. In Flow algorithm like many other numerical methods the number of solutions is not obtained and the algorithm by its nature, most of the times finds the closest solution, although this claim is not proven in this work.

3.4 Summary

In this chapter the proposed algorithm for solving inverse kinematic problem is presented. This algorithm which works based on the concept of motion flow through manipulator linkages provides the required torques to be sent to each joint actuator and receives the resulting end effector position as the system feedback. Simulation

studies on ten degrees of freedom manipulator verify the effectiveness of the proposed algorithm. This algorithm can be called sub-Optimal in a sense that it would not satisfy all optimization criteria that could be defined for a manipulator task. The main objective is to generate human like motion in arms but many other criteria could be introduced that this algorithm fails fulfilling them. However the algorithm is efficient due to its small computation cost even for highly redundant manipulators.

Chapter 4

Experimental Results

In order to prove the efficiency of this algorithm, series of experiments are conducted and under the same condition the motion of the simulated arm and the human arm, as an efficient model, are compared. The simulated arm has totally ten degrees of freedom distributed as follows: The complex motion of Capsular, the spherical joint of shoulder and wrist are modeled by three successive rotational joints axes intersecting in one point for each of them. Elbow is considered as a single degree of freedom hinge joint modeled by one rotational joint. As mentioned before, the simulation toolbox used is the Robotics Toolbox for MATLAB which can graphically illustrate the arm motion.

4.1 Experimental Setup

The experimental set up consists of six Vicon cameras which can detect the three dimensional motion of markers attached to the body. Therefore, thirteen markers

Figure 4.1: Experimental Setup: Six Vicon Cameras Covering the Whole Scene.

were attached to the left arm of the experimenter and she has been asked to stretch her hand towards a marked target in the space. For each target the experiment has been conducted five times to reduce the errors and different locations for the target is assigned in order to observe human motion planing in different situations. The experimental setup is shown in Figure (4.1) where cameras (Figure (4.2)) are placed in different locations to cover the whole arm in its different configurations.

A marked triangle shown in Figure (4.3) is used for calibration to specify the location and orientation of the global frame.

The most important part of the experiment is specifying marker's position on the arm. In this experiment, since the relative angular movement of each part of the arm is important, markers are put so that each part has at least three markers to form its coordinate system. Furthermore, for each joint at least two markers are needed to define a line passing the joint center. The exact position of the joint center can be

Figure 4.2: The Vicon Camera.

Figure 4.3: The Reference Triangle to Specify the Global Frame.

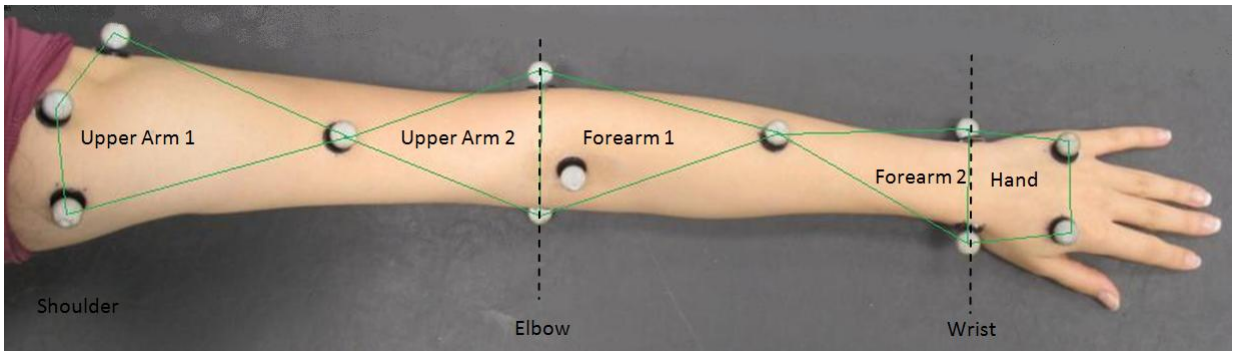


Figure 4.4: Markers Configuration on the Arm.

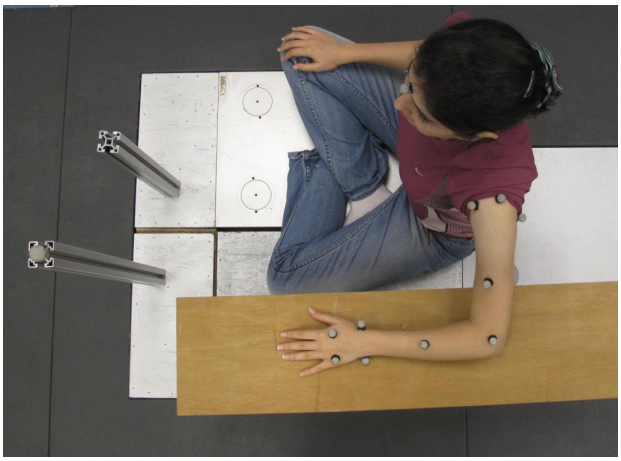


Figure 4.5: Initial Position of the Arm.

found by defining a Virtual Point between two markers at each side of the joint. The markers configuration and parts of the arm are shown in Figure (4.4).

Seven different locations for the target are specified which roughly covers most of the human arm workspace. The initial position and six target locations are shown in Figures (4.5) and (4.6) respectively.

The choice of cameras placement is in a way to make each marker visible by at least two cameras in order to get the three dimensional trajectory of the marker. Each

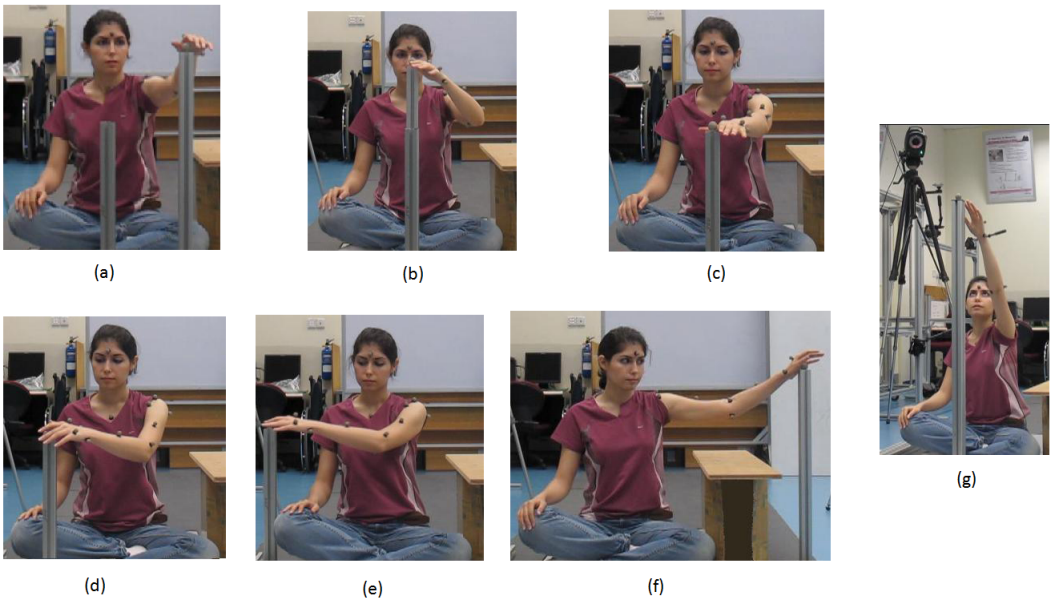


Figure 4.6: Selected Targets in Experiment (a) to (g).

camera send its two dimensional information with the rate of 60 HZ to the Vicon software and once recording is finished, system compiles all information from the six cameras and gives the three dimensional model of the markers grid per frame. In Figure (4.7) the model obtained from the first experiment in states of rest and extended arm is shown.

The purpose of this experiment is to compare some variables in motion of human arm with their correspondents in the simulated arm motion. These variables can be joint angles and position of some parts of the arm in the space. Therefore, in order to find the relative angle and the Cartesian position of each part of the arm, it is necessary to define a frame on each rigid part. As it can be seen in Figure (4.7), the four defined frames are: The global frame attached to the trunk with its origin on

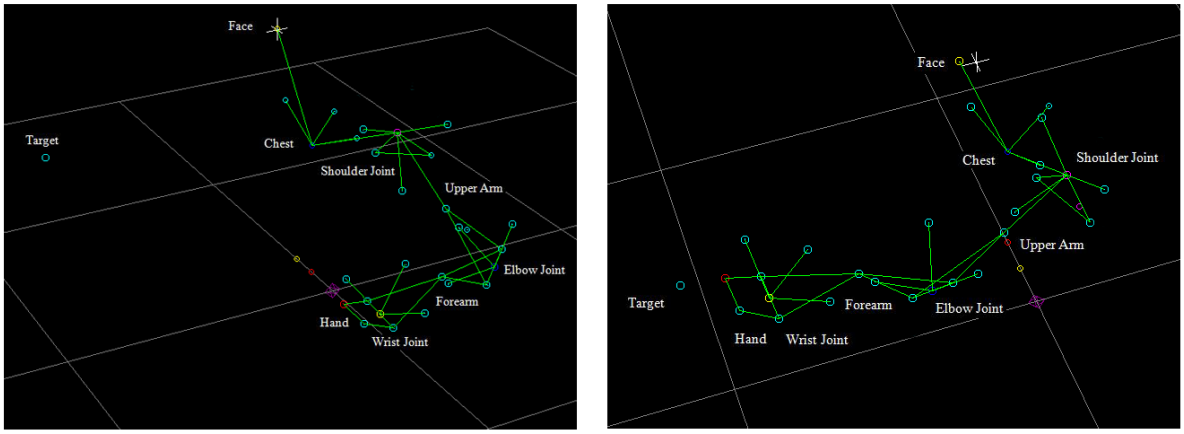


Figure 4.7: Markers Grid in Vicon Work Station Interface. Captured for the First Experiment: (a) Arm at Rest, (b) Arm Stretched.

the chest, the upper arm frame having its origin on a virtually defined point on the intersection of three lines passing through the markers of the Capsular on the shoulder, the forearm frame with its origin on a virtual point between two markers at each side of the elbow and its X frame parallel to the joint axis, and finally, the hand frame defining its origin on a virtual point in the middle of the wrist and its X axis parallel to the wrist tilting joint axis. The position of every point of the arm in the Vicon Work Station is presented in the global frame and all the angles are the relative angles between two consecutive frames.

4.2 Experiment Description

We choose to compare the elbow angle, elbow position and the finger tip position through the trajectory of the hand motion from the state of resting to the state of reaching the target. The reason of this selection is to have a nearly exact comparison

which is only possible if the variables we have chosen are easy to obtain in experimental data. In the case of the shoulder joint, it is known that the motion of the shoulder is very complex and it would involve too much simplification if we define three revolute joints with intersecting axes and derive three angles to explain the shoulder motion in the experimental results. Although the simulated model has the three revolute joint structures, it still can produce a roughly similar motion in the upper arm. We do not observe the motion of the wrist either, since it is interpreted from the results that this joint does not have significant variations though the trajectory. Therefore, instead of comparing the shoulder and wrist axes, the position of the elbow and the finger tip would give us the same result in task space. To keep the result analysis short and consistent, we present the comparisons only form the the first experiment where the target is place in front of the experimenter with some deviations towards the left.

4.3 Analysis of Results

The extracted data from experiments would be compiled in Vicon Workstation. As an example the resulted time series of the elbow joint angles is illustrated in Figure (4.8). In this figure angles along the three elbow joint frame axis are shown at each sample and it can be observed that the only dominant motion of the elbow is along one axis and the non-zero values for other two angles are due to non-rigidity of upper arm and forearm and probable errors in capturing the motion.

First, we compare experimental and simulation results of the elbow position in 3D space from experiment and simulation. Figure (4.9) illustrates the position of the

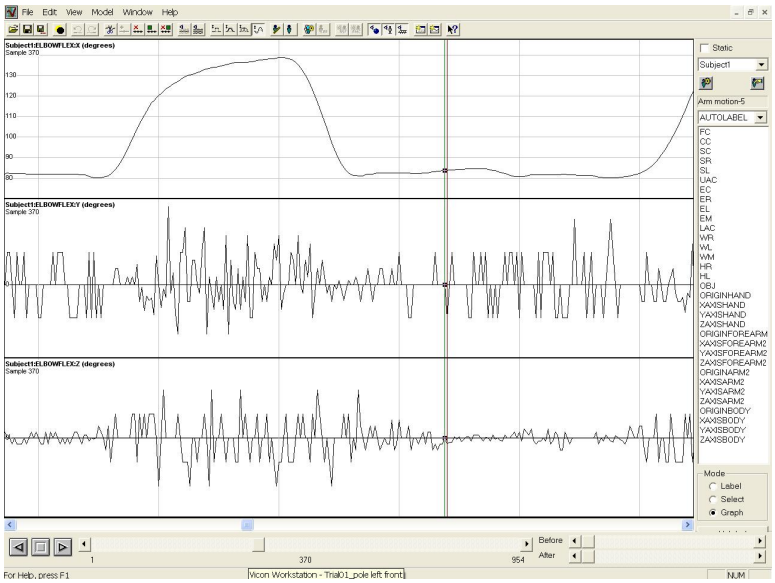


Figure 4.8: Vicon Workstation: Upper arm and Forearm Relative Angles Along 3 Elbow Joint Frame axis.

forearm frame origin which is attached to the elbow, from the rest position until the finger tips touches the target and the following graph in Figure (4.10) presents the trajectory of the elbow in the simulated arm for the same task. Similarly, elbow joint variable and finger tip position obtained from experiment and proposed algorithm are shown in Figure (4.11) to (4.14).

These figures show the similarity of the selected variables in the simulated model and the natural motion. However, it is necessary to express this similarity in a quantitative way. There are several algorithms for comparing time series, such as cross correlation and Euclidean distance. Cross correlation is mainly meant to measure the strength of the relationship between variables; if two variables are correlated, they have dependency and the changes in one can be predicted by changes in the other variable. This measurement expressed by correlation coefficient r, is independent of

the variables magnitudes and can have any value between -1 and +1. Correlation coefficient of +1 indicates a perfect positive relationship, zero indicates no relationship and -1 indicates perfect negative relationship. The correlation coefficient is defined as

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\frac{x_i-\mu_x}{\sigma_x}\frac{y_i-\mu_y}{\sigma_y}. \tag{4.1}$$

In this equation, μ and σ denote the sample mean and the sample standard deviation. In signal processing assuming that we have sampled signals, for example the elbow joint values for each sample time of experiment data and simulated arm in our case, the cross correlation between these two signals is defined as

$$R_{xy}(m) = \frac{1}{N}\sum_{n=1}^{N-m+1} y(n)x(n+m-1), \tag{4.2}$$

where $m = 1,2,3,...N+1$ and N is the length of the signals. In this definition $R_{xy}(m) = 0$ indicates no dependence of signals and the high $R_{xy}(m)$ means that two signals have high similarity at lag m.

For two reasons, cross correlation is not a suitable tool for comparing our results. One reason is that in cross correlation the length of two series must be the same, whereas due to the differences in sampling rate of cameras in capturing the motion and number of loops in algorithm, and even not consistent motion of human in each movement, the number of samples vary a lot between different series. The other reason is the concept of cross correlation which as mentioned above, is to determine the similarity of two series in their general shape. Therefore in this method two series of sine function with different amplitudes have a good correlation if they share the same

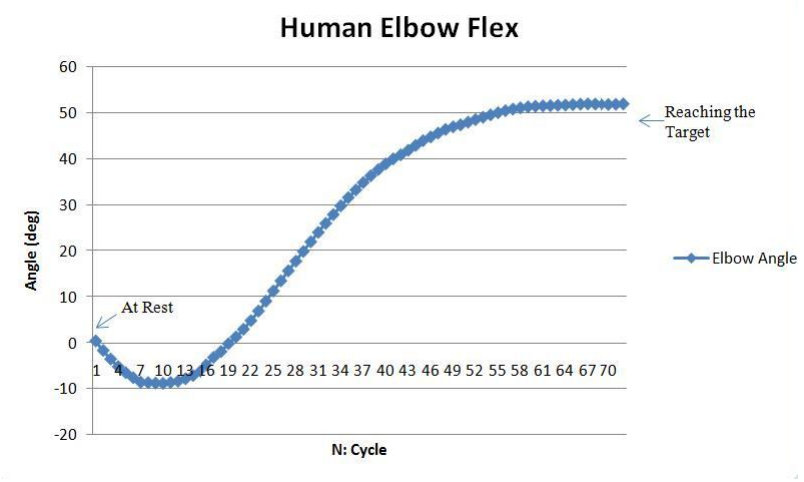


Figure 4.9: Elbow 3D Position History from Experimental Results.

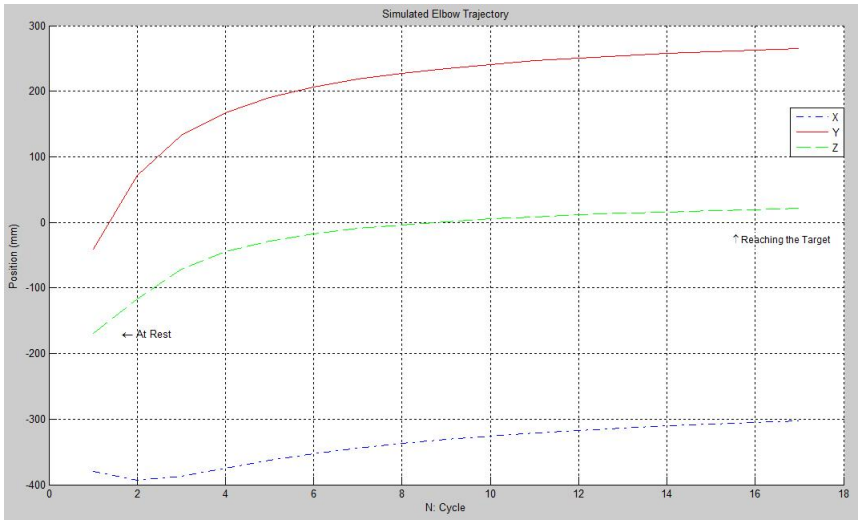


Figure 4.10: Elbow 3D Position History of the Simulated Arm.

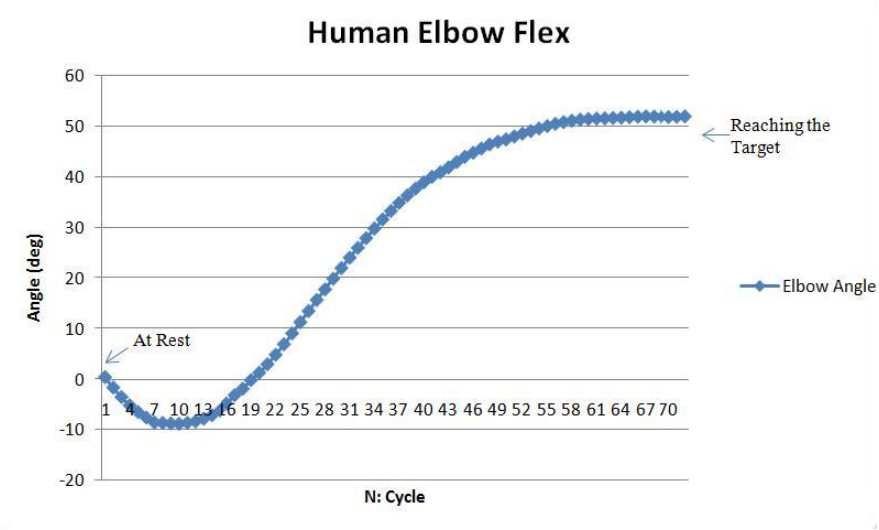


Figure 4.11: Elbow Joint Variable History from Experimental Results.

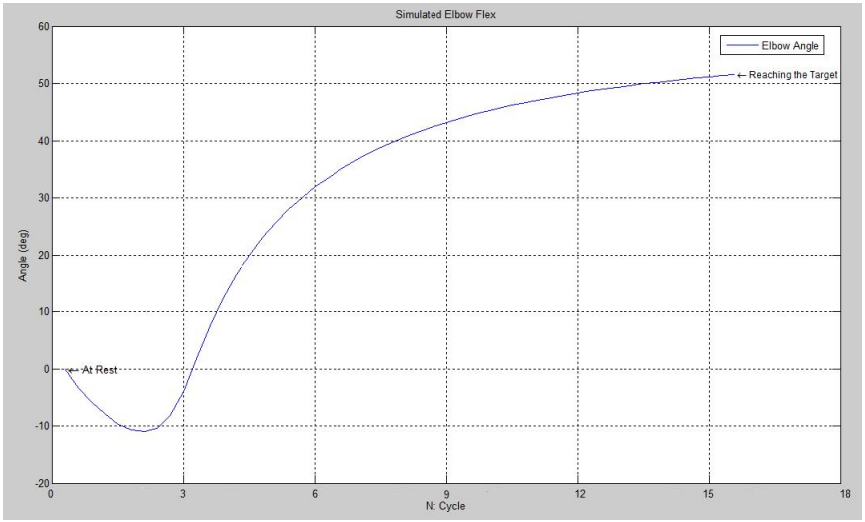


Figure 4.12: Elbow Joint Variable History of the Simulated Arm.

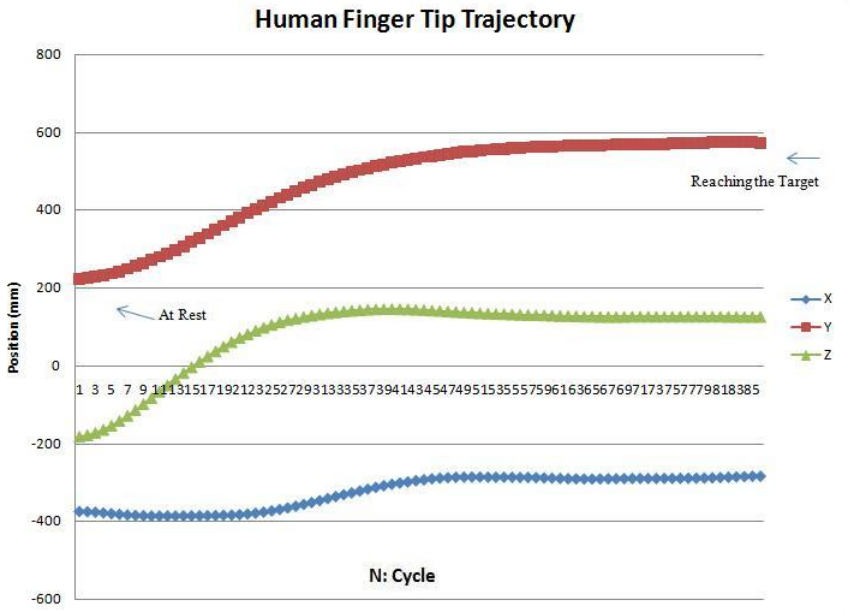


Figure 4.13: Finger Tip 3D Position History from Experimental Results.

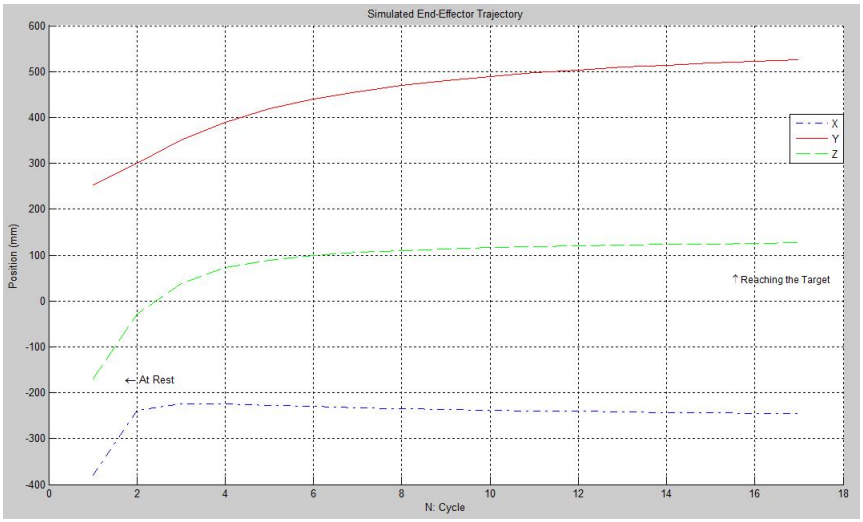


Figure 4.14: Finger Tip 3D Position History of the Simulated Arm.

frequency. Studying the series from this point of view, although can result in good understanding of the motion characteristics, requires large number of experimental results to cover all the possible configurations and is not the purpose of this work. To reduce the number of required experiments we keep the initial and final positions consistent for the simulation and experiment in the task space and try to make the geometric parameters of the model match the human arm. To solve the problem of different number of samples we use a well known method mainly meant for matching purposes in the field of speech and motion recognition called Dynamic Time Warping. This technique can compensate for different length of two sample series while preserving sample's order. The purpose of this method is to find optimal match between the two sample series with certain restrictions. The advantage of DTW compared to other simple matching techniques lies in computing the distance between samples which are not from the same times. DTW searches the neighboring samples to select matches and therefore minimizes the total distance. Assuming we have two sequences with different length as

$$X = x_1, x_2, \ldots, x_p, \ldots, x_m,$$

and

$$Y = y_1, y_2, \ldots, y_q, \ldots, y_n,$$

where x_p and y_q are the p^{th} and q^{th} samples of series X and Y, and m and n are the length of series X and Y respectively. To compare X and Y a recursive function is

used as follow

$$\text{Dist}(X,Y) = \text{cost}(m,n),$$

$$\text{cost}(1,1) = \|x_1 - y_1\|,$$

$$\text{cost}(p,q) = D(p,q) + \min\{\text{cost}(p,q-1), \text{cost}(p-1,q-1), \text{cost}(p-1,q)\}$$

where $\text{D}(p,q) = \|x_{p_k} - y_{q_k}\|$, is the Euclidean distance in desired space (joint or task space) and $\|p_k - q_k\| \leq w$, where w is the maximum number of warping. Therefore, the similarity would have reverse relationship with $\text{Dist}(X,Y)$:

$$\text{similarity}(X,Y) = \frac{1}{\text{Dist}(X,Y)}.$$

To obtain the value of the allowed limit of accumulated distance, $\text{Dist}(X,Y)$, in order to the keep motion natural, first we compare the results of five repeated motion of human arm in reaching the same target and observe their deviation from the first attempt. Figure (4.15) to (4.18) show the comparison of the x component of the elbow trajectory between the first attempt to reach the target and the following four attempts to reach the same target. In the other hand, Figure (4.19) to (4.23) compare the result of the algorithm with each of the tests in the experiment. In all the Figures, the left subplot presents the original series and the right subplot shows the matched

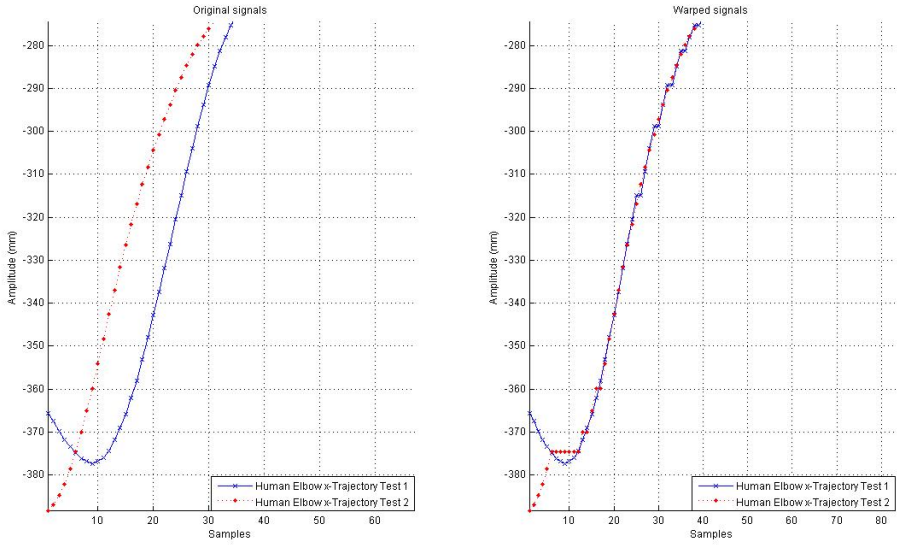


Figure 4.15: Comparison of Human Elbow x-Trajectory in Test 1 and 2, Using DTW Technique.

samples of two series after applying the DTW technique.

The values of the accumulated Euclidean distance for the series are shown in the following tables. By comparing the values of distances between two experimental tests, and between an experimental test and simulation, it can be seen that they are in the same order. We can conclude that this order of difference is inevitable even between two successive human arm motion, and therefore, the model could predict the human arm motion with a good approximation.

4.4 Summary

The comparison of the simulated model motion which follows the Flow algorithm, with real human motion is presented in this chapter. The graphs illustrate the similarity of

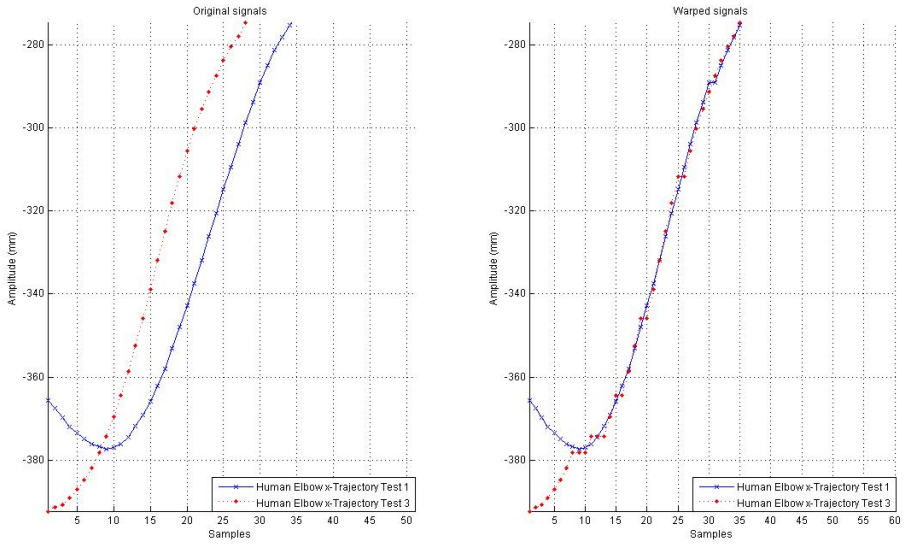


Figure 4.16: Comparison of Human Elbow x-Trajectory in Test 1 and 3, Using DTW Technique.

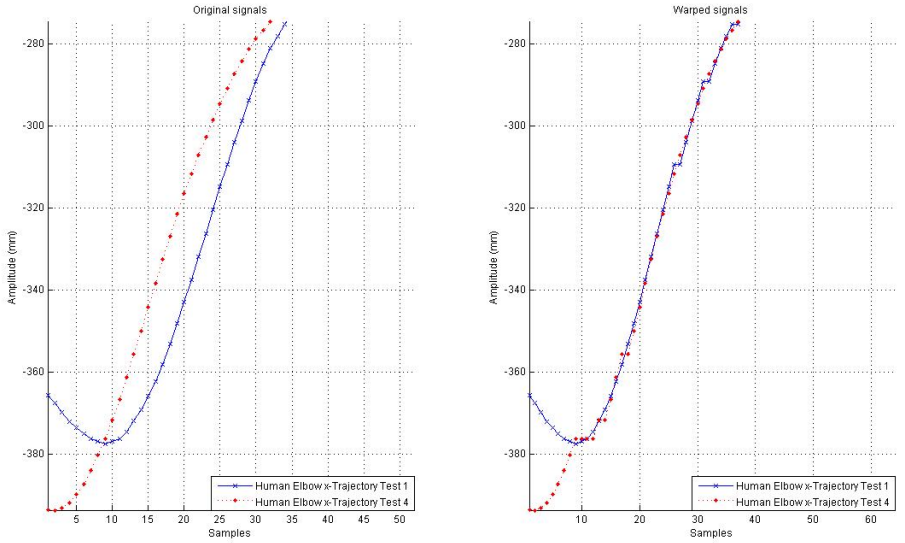


Figure 4.17: Comparison of Human Elbow x-Trajectory in Test 1 and 4, Using DTW Technique.

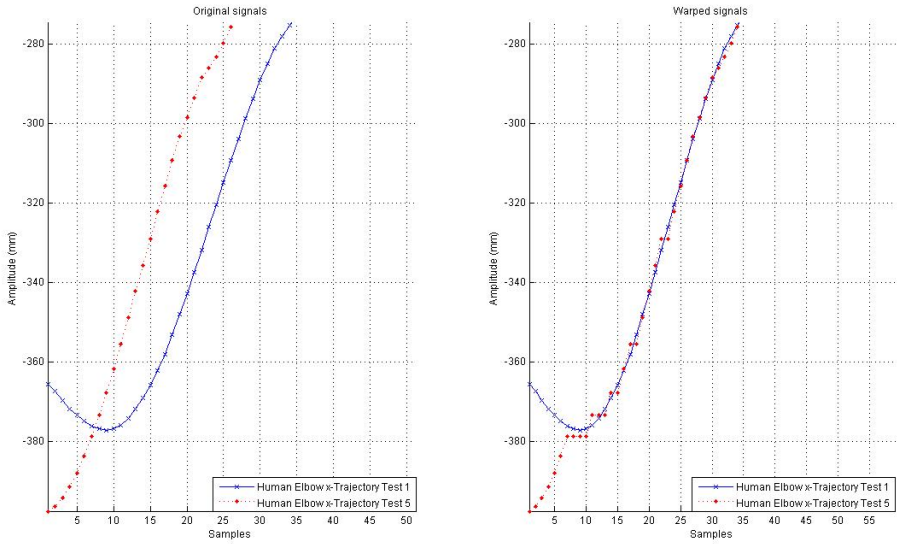


Figure 4.18: Comparison of Human Elbow x-Trajectory in Test 1 and 5, Using DTW Technique.

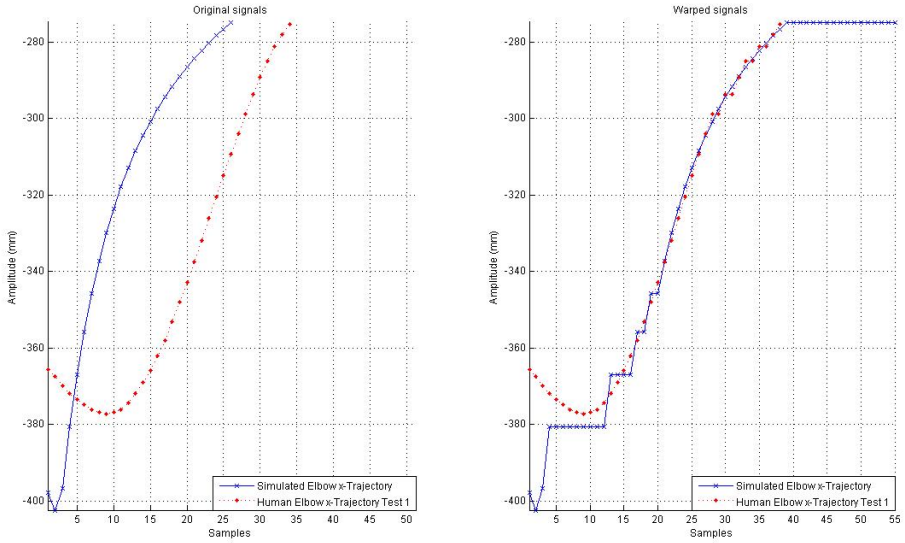


Figure 4.19: Comparison of Simulated and Human Elbow x-Trajectory Test 1, Using DTW Technique.

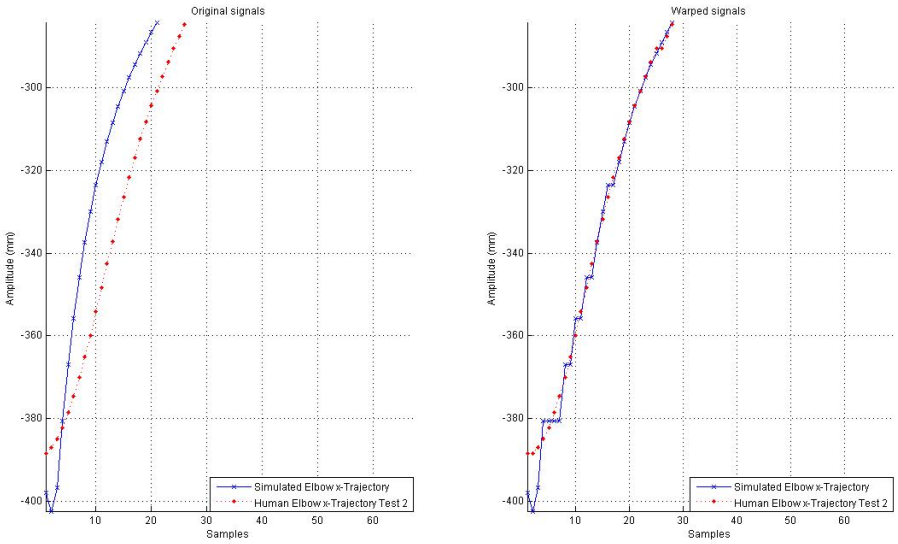


Figure 4.20: Comparison of Simulated and Human Elbow x-Trajectory Test 2, Using DTW Technique.

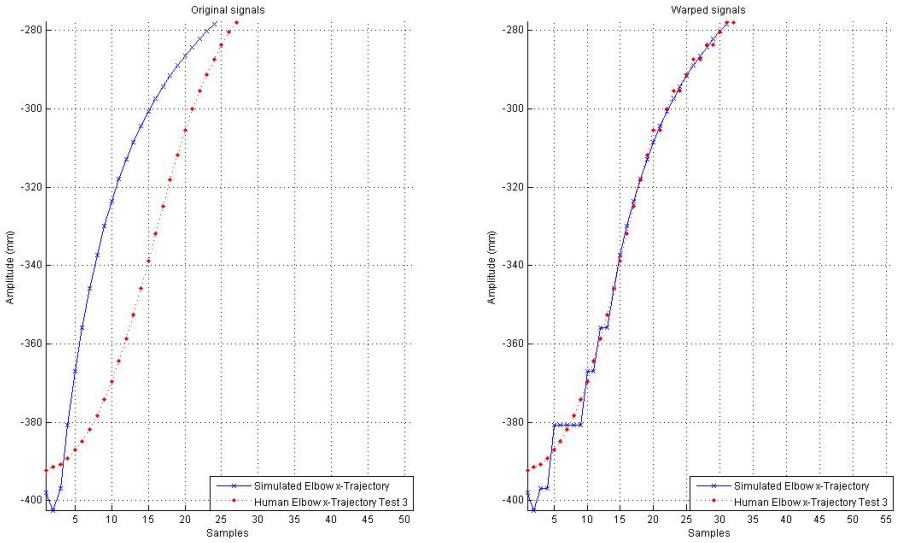


Figure 4.21: Comparison of Simulated and Human Elbow x-Trajectory Test 3, Using DTW Technique.

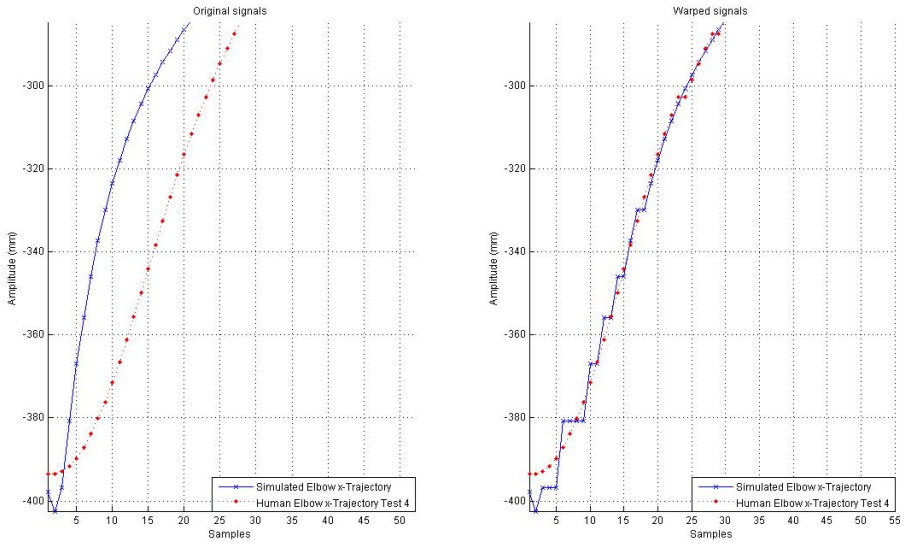


Figure 4.22: Comparison of Simulated and Human Elbow x-Trajectory Test 4, Using DTW Technique.

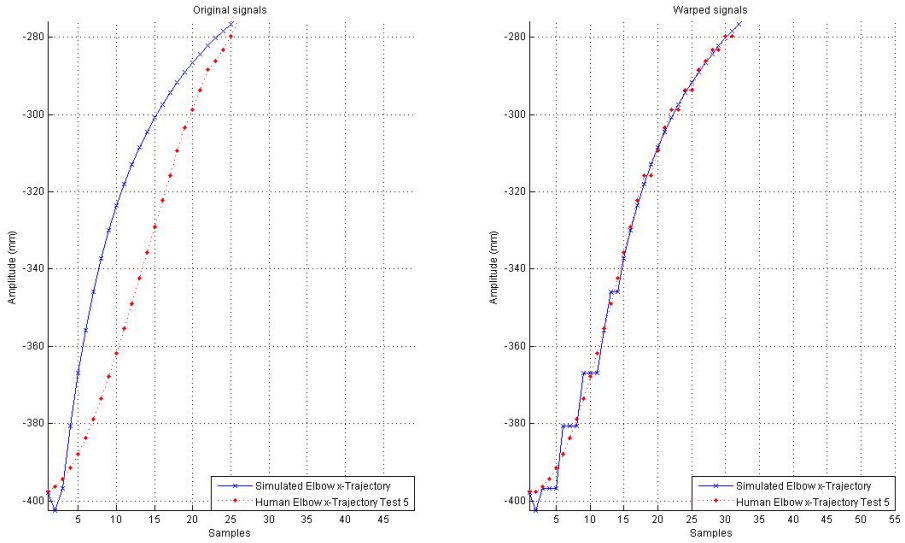


Figure 4.23: Comparison of Simulated and Human Elbow x-Trajectory Test 5, Using DTW Technique.

Accumulated Euclidean Distance (mm)		Test 1	Test 2	Test 3	Test 4	Test 5
X	Test 1	0	1661.3	2432.8	3309.1	3224.2
	Flow Algorithm	4985.1	1207.5	3789.5	3830.4	2814.2
Y	Test 1	0	1805.2	1480.4	1423.2	2701
	Flow Algorithm	3417.7	36328	1363	1266.2	1349.5
Z	Test 1	0	1026	1393.8	1702.6	1802.9
	Flow Algorithm	1268.1	3285.9	4309.2	1285.2	2381.8

Table 4.1: Accumulated Euclidean Distance of Elbow Position Series

Accumulated Euclidean Distance (deg)		Test 1	Test 2	Test 3	Test 4	Test 5
Elbow Joint Angle	Test 1	0	127.27	381.32	44.24	355.89
(θ_7)	Flow Algorithm	255.01	132.08	124.92	182.43	130.87

Table 4.2: Accumulated Euclidean Distance of Elbow Joint Angle Series

Accumulated Euclidean Distance (mm)		Test 1	Test 2	Test 3	Test 4	Test 5
X	Test 1	0	2990.6	2857	3728.3	3691.4
	Flow Algorithm	1320.2	2724.6	2396.4	2937.5	2814.2
Y	Test 1	0	1320.3	2207	2022.4	1917.7
	Flow Algorithm	1586.6	1498.1	951.1	968.4	1058.3
Z	Test 1	0	1909.6	1054.1	4365.7	37641
	Flow Algorithm	1706.2	14097.4	1939.9	4662.4	2457.3

Table 4.3: Accumulated Euclidean Distance of End effector Position Series

two cases in their general shapes, and the quantitative comparison can be obtained by the accumulated Euclidean distance presented in tables. It can be concluded from the results that motion of the simulated arm is semi-natural and comparable with human motion.

Chapter 5

Conclusion and Future Work

5.1 Contributions

Inverse Kinematics problem is an essential part of manipulator motion planning and control which involves solving non-linear complex equations. In most cases, especially for high degrees of freedom manipulators no closed-form solution is available. Furthermore, once the solution is obtained using conventional numerical methods or by means of Jacobian matrix, the resulting motion of the arm would not be natural. In this work we have introduced a novel method in solving the inverse kinematic problem called the "Flow" Algorithm, which has the following advantages compared to the existing techniques in the literature.

1. Flow Algorithm provides the inverse kinematic solution for any degrees of freedom manipulator with any geometry as long as the forward kinematics is given. Despite the Inverse Geometric methods which are available for smaller degrees of

freedom chains, in the proposed method, there would be no need for any manual calculations or parameters adjustment for different geometrical manipulators.

2. This algorithm provides the value of each joint at every time step until the target is reached, whereas in Inverse Geometric method only the final joint angles are provided and the internal motion of the redundant manipulator through the trajectory should be obtained by running the planner module concurrently.

3. This algorithm overcomes the important problem of non-reachable solutions, meaning that the final configuration of the manipulator is reachable from the initial configuration while preventing self-collision and not violating joint limits. This is achieved, since for the redundant manipulators, the selection of the final configuration in Flow Algorithm is highly dependent of initial joint values.

4. Singularity problem which is a big issue in Inverse Jacobian methods can be easily overcome by adding simple conditions in the main loop of the algorithm, such as preventing the robot manipulator to approach some predefined singular postures and find alternative paths. However, in traditional inverse kinematics avoiding singular postures may cause failure in reaching the computed desired joint coordinates.

5. Using the proposed algorithm, more natural movement of the arm is achieved which is an essential factor in social robotics.

In this work, apart from introducing the Flow Algorithm, a comprehensive study of available methods for solving inverse kinematic problem as well as most important

planning techniques and optimization criteria are provided in the Literature Review chapter. The efficiency of the proposed method is shown by experimental results which show close similarities between the result of this algorithm and real human arm motion. Kinematic model of a seven degrees of freedom manipulator using D-H parameters is provided as an example of forward kinematics which is the required input to the Flow Algorithm.

5.2 Future Work

This thesis opens up some interesting directions for further investigations, which are described below.

The proposed algorithm has the potential to be merged with many planning techniques if needed. Some of the required features to be added are self-collision and obstacle avoidance which are fully studied in chapter two and the potential field method is suggested by the author to be used in the Flow Algorithm. Furthermore, in grasping applications, apart from the position of the end effector, its orientation is playing an important role. In this work, we have reduced the problem to simple reaching which only takes care of position errors. Although, any desired orientation of the end effector can be achieved by adding a spherical joint at the wrist, in order to have a natural motion, the desired end effector orientation should be considered in the inverse kinematic algorithm to avoid any unnatural configurations of the manipulator in grasping.

There are some considerations to be made to conduct more accurate experimental studies in future. In this work we have studied the human arm motion in reaching

different points in the space from one fixed initial position. It is essential to repeat the experiments for different starting poses such as fully stretched or bent arm in sitting or standing positions and study their effects in the motion planning. Regarding the experimenter, it is better to choose a person totally unfamiliar with the field being asked to do the experiments with a cover story to avoid any conscious decision making in the arm motion. It is also suggested to repeat the experiment for large number of people to find the main characteristics of human motion.

In order to have more convincing results it is necessary to use some other tools to analyze the experimental data as well. One of the proposed data analysis techniques is using the cross correlation method to find the main characteristics of the motion. Furthermore, it is advised to get the results of other inverse kinematic solutions in the literature and compare them with the proposed algorithm in terms of providing natural motion in real robot manipulators.

Bibliography

[1] J. J. Craig, Introduction to Robotics. Addison-Wesley, 1989.

[2] M. Raghavan and B. Roth, A General Solution for the Inverse Kinematics of All Series Chains. In Proc. 8th CISM-1FTOMM Symposium on Robots and Manipulators Romansy-90, Kracow, Poland, 1990, pp. 24-31.

[3] J. Knapczyk and P. A. Lebiediew, Theory of Spatial Mechanisms and Manipulators. Warsazava, WNT, 1990.

[4] J. Angeles, Rational Kinematics. New York: Springer Verlag. 1988.

[5] C. Mavroidis, F.B. Ouezdou, P. Bidaud, Inverse Kinematics of Six Degree of Freedom General and Special Manipulators Using Symbolic Computation, Robotica, vol. 12, pp. 421–430, 1994.

[6] D. Pieper, The Kinematics of Manipulators under Computer Control, Ph. D. dissertation, stanford University, Stanford, CA, 1968.

[7] V.D. Tourassis and M.H. Ang Jr., Task Decoupling in Robot Manipulators, Journal of Intelligent and Robotic Systems, vol. 14, pp. 283-302, 1995. (Technical Report in 1992).

[8] L.W. Tsai, A. Morgan, Solving the Kinematics of the Most General Six-and-Five-Degree-of-Freedom Manipulators by Continuation Methods, Transactions of ASME, Journal of Mechanisms, Transmission and Automation in Design, vol. 107, pp. 189–200, 1985.

[9] M. Raghavan, B. Roth, Solving Polynomial Systems for the Kinematics Analysis and Synthesis of Mechanisms and Robot Manipulator, Journal of Mechanical Design, vol. 117, pp. 71–79, 1995.

[10] A. Rouvinen, H. Handroos, Comparison of Three Neural Network Methods in Solution of Inverse Kinematics of Large Flexible Redundant Manipulators, Proceedings of the International Symposium on Robotics (ISR 2000), Montreal, Canada, 2000, pp. 168–173.

[11] W. Zhang, The inverse Kinematics for the Orientation of a Robot Arm Based on Neural Network, Journal of Nanjing University of Aeronautics & Astronautics, vol. 29 (1), pp. 46–50, 1997.

[12] A. Khwaja, M. O. Rahman and M. G. Wagner. Inverse Kinematics of Arbitrary Robotic Manipulators Using Genetic Algorithms. In Advances in Robot Kinematics: Analysis and Control, J. Lenarcic and M. L. Justy (editors), pp. 375-382, Kluwer Academic Publishers. 1998.

[13] E. Hansen, Global Optimization Using Interval Analysis. Marcel Dekker, New York, 2003.

[14] R. B. Kearfott, Rigorous Global Search: Continuous Problems. Klumer Ac. Pub., Netherland, 1996.

[15] P. Van Hertenryck, D. McAllester and D. Kapur, Solving Polynomial Systems Using a Branch and Prune Approach. SIAM J. Numer. Anal., vol. 34(2), pp. 797-827, 1997.

[16] D.E. Whitney. Resolved Motion Rate Control of Manipulators and Human Prostheses. Proc. IEEE Transactions on Man–Machine Systems MMS, vol. 10(2), pp. 47–53. 1969.

[17] R. Penrose, The Best Approximation to the Solution of Matrix Equations. Proc. Cambridge Phil. Soc., vol. 52, pp. 17-19. 1956.

[18] A. Liegeois, Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. Proc. IEEE Transactions on Systems, Man, and Cybernetics, vol.7(12), pp. 868-871. 1977.

[19] J. Baillieul, Kinematic Programming Alternatives for Redundant Manipulators. In Proc. IEEE International Conference on Robotics and Automation. 1985.

[20] C.A. Klein and S. Ahmed, Repeatable Pseudo inverse Control for Planar Kinematically Redundant Manipulators. In Proc. IEEE Transactions on Systems , Man, Cybernetics, vol. 25(12), pp. 1657-1662. 1995.

[21] A. Balestrino, G. De Maria, and L. Sciavicco, Robust Control of Robotic Manipulators. In Proc. 9th IFAC World Congress, vol. 5, pp. 2435-2440, 1984.

[22] W. A. Wolovich and H. Elliot, A Computational Technique for Inverse Kinematics. In Proc. 23rd IEEE Conference on Decision and Control, 1984, pp. 1359-1363.

[23] V. D. Tourassis and J. M. H. Ang, A Modular Architecture for Inverse Robot Kinematics, IEEE Trans. Robotics Auromur., vol. 5, pp. 555-568, 1989.

[24] O. Khatib, J. Warren, V. De Sapio, and L. Sentis, Human-like Motion from Physiologically-based Potential Energies, Advances in Robot Kinematics, vol. 1, pp.149–163, 2004.

[25] O. Khatib, E. Demircan, V. De Sapio, L.Sentis, T. Besier, S. Delp, Robotics-based Synthesis of Human Motion, Journal of Physiology, vol. 103(3-5), pp. 211-219, 2009.

[26] H. Cruse and M. Brüwer, The Human Arm as A Redundant Manipulator: The Control of Path and Joint Angles. Biological Cybernetics, vol. 57, pp.137-144, 1987.

[27] N. Hogan, Mechanical Impedance of Single- and Multi-Articular Systems. In Multiple Muscle Systems: Mechanics and Movement Organization, ed. by J. M. Winters and S. L-Y. Woo, Springer-Verlag, 1990.

[28] Adams, B. Meso, A Virtual Musculature for Humanoid Robots. M.Eng Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. 1999.

[29] Y. Nakamura and H. Hanafusa, Task Priority based Redundancy Control of Robot Manipulators. In Robotics Research, 2nd Int. Symposium, ed. by H. Hanafusa and H. Inoue, MIT, 1985, pp. 155-162.

[30] Baillieul, J. and D.P. Martin, Resolution of Kinematic Redundancy. In Proc. Symposia in Applied Mathematics. 1990, American Mathematical Society. pp. 49-89.

[31] B. Faverjon and P. Tournassoud, A Local Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom. In Proc. IEEE International Conference on Robotics and Automation, Raleigh, NC, 1987, pp. 1152–1159.

[32] M. Sharir and A. Schorr, On Shortest Paths in Polyhedral Spaces. In Proc. 16th ACM Symposium on Theory of Computing, Washington, DC, 1984, pp. 144–153.

[33] D. Wang and Y. Hamam, Optimal Trajectory Planning of Manipulators with Collision Detection and Avoidance, Proc. International Journal of Robotics Research, vol. 11(5), pp. 460–468, 1992.

[34] V. Perdereau, C. Passi and M. Drouin, Real-Time Control of Redundant Robotic Manipulators for Mobile Obstacle Avoidance. Proc. Robotics and Autonomous Systems. vol. 41, Elsevier, pp. 41–59, 2002.

[35] C. J. Ong and E. G. Gilbert, Growth distances: New measures for object separation and penetration. IEEE Trans. Robot. Automat., vol. 12(6), pp. 888-903, 1996.

[36] R. A. Jarvis, Distance Transform Based collision-Free Path Planning for Robot. Advanced Mobile Robots, world scientific Publishing, pp. 3-31, 1994.

[37] S.F.P. Saramago and M. Ceccarelli. Effect of Basic Numerical Parameters on a Path Planning of Robots Taking into Account Actuating Energy. Mechanism and Machine Theory vol. 39, pp. 247–260, 2004.

[38] A. Gasparetto and V. Zanotto, A New Method for Smooth Trajectory Planning of Robot Manipulators. Proc. Mechanism and Machine Theory, vol. 42, pp. 455–471, 2007.

[39] S.F.P. Saramago and V. Steffen Jr., Optimization of the Trajectory Planning of Robot Manipulators Taking into Account the Dynamics of the System. Proc. Mechanism and Machine Theory vol. 33, pp. 883–894, 1998.

[40] T. Chettibi, H.E. Lehtihet, M. Haddad and S. Hanchi. Minimum Cost Trajectory Planning for Industrial Robots. Proc. European Journal of Mechanics A/Solids, vol. 23, pp. 703–715, 2004.

[41] Xuan F. Zha, Optimal Pose Trajectory Planning for Robot Manipulators. Proc. Mechanism and Machine Theory, vol. 37, pp. 1063–1086, 2002.

[42] N.A. Aspragathos, Cartesian Trajectory Generation under Bounded Position Deviation. Proc. Mechanism and Machine Theory, vol. 33, pp. 697–709, 1998.

[43] O.Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. Proc. International Journal of Robotics Research, vol. 5(1), pp. 90–98, 1986.

[44] J. Agirrebeitia, R. Avilés, I.F. de Bustos and G. Ajuria, A New APF Strategy for Path Planning in Environments with Obstacles. Proc. Mechanism and Machine Theory, vol. 40, pp. 645–658, 2005.

[45] F. Valero, V. Mata and A. Besa. Trajectory Planning in Work spaces with Obstacles Taking into Account the Dynamic Robot Behaviour, Proc. Mechanism and Machine Theory, vol. 41, pp. 525–536, 2006.

[46] S.F.P. Saramago and V. Steffen Jr., Optimal Trajectory Planning of Robot Manipulators in the Presence of Moving Obstacles. Proc. Mechanism and Machine Theory, vol. 35, pp. 1079–1094, 2000.

[47] Z. Yao and K. Gupta, Path Planning with General End-Effector Constraints, Proc. Robotics and Autonomous Systems, vol. 55, pp. 316–327, 2007.

[48] S. Lavalle, J. Yakey and L. Kavraki, A Probabilistic Roadmap Approach for Systems with Closed Kinematic Chains. In Proc. IEEE International Conference on Robotics and Automation, 1999, pp. 1671–1676.

[49] X. Luo, XP. Fan, H. Zhang and T. Chen, Integrated Optimization of Trajectory Planning for Robot Manipulators Based on Intensified Evolutionary Programming. In Proc. IEEE International Conference on Robotics and Biomimetics, 2004, pp. 546-551.

[50] B. H. Krogh, A Generalized Potential Field Approach to Obstacle Avoidance Control. In Proc. SME Conference of Robotics Research, Aug. 1984, Bethlehem, PA.

[51] C. I. Connolly, J. B. Burns, and R. Weiss, Path Planning Using Laplace'S Equation. In Proc. IEEE Int. Conference of Robotics Automat., Cincinnati, OH, May 1990, pp. 2102-2106.

[52] J. R. Andrews and N. Hogan, Impedance Control as a Framework For Implementing Obstacle Avoidance in a Manipulator. In Control of Manufacturing Processes and Robotic Systems, ed by D. Hardet and W. Book, pp. 243-251. ASME, Boston, 1983.

[53] D. E. Koditschek, Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations. In Proc. IEEE Int. Conf. Rob. Autom., March 1987, Raleigh, NC.

[54] E. Rimon and E. Koditschek, The Construction of Analytic Diffeomorphisms for Exact Robot Navigation on Star Worlds. In Proc. ZEEE Int. Conf. Rob. Autom., 1989, pp. 21-26.

[55] E. Rimon and D. E. Koditschek, Exact Robot Navigation in Geometrically Complicated but Topologically Simple Spaces. In Proc. IEEE Int. Conf. Rob. Autom., Cincinnati, OH, 1990, pp. 1937-1942.

[56] J. Barraquand, B. Langolis and J. Latombe, Numerical Potential Field Techniques for Robot Path Planning. Proc. ZEEE Trans. Syst. Man Cybern., vol. 22(2). 1992.

[57] J. Barraquand, B. Langlois and J. C. Latombe, Robot Motion Planning with Many Degrees of Freedom and Dynamic Constraints. In Proc. 5th Int. Symp. Robotics Res., Tokyo, Aug. 1989, pp. 74-83.

[58] C. W. Warren, Global Path Planning using Artificial Potential Fields. In Proc. IEEE Int. Conf Robotics Automat., Scottsdale, AZ, May 1989, pp. 316-321.

[59] H. Seraji. Configuration Control of Redundant Manipulators: Theory and Implementation. Proc. IEEE Transactions on Robotics and Automation, vol. 5(4), pp. 472-490. August 1989.

[60] K. Kreutz-Delgado, M. Long and H. Seraji, Kinematic Analysis of 7- DOF Manipulators. Proc. Int. Journal of Robotics Research, vol. 11(5), pp. 469-481. October 1992.

[61] P. I. Corke. A robotics Toolbox for MATLAB. Proc. IEEE Robot. Autom. Mag., vol. 3(1), pp. 24–32, March 1996.

[62] K. Anderson and J. Angeles, Kinematic Inversion of Robotic Manipulators in the Presence of Redundancies. Int. Journal of Robotics Research, vol. 8(6), pp. 80-97, December 1989.

[63] Anderson, K. Inverse Kinematics of Robot Manipulators in the Presence of Singularities and Redundancies. Ph.D. Thesis, McGill University. 1987.