University of Cambridge

Technical Report

Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver

Andreas Aristidou and Joan Lasenby

CUED/F-INFENG/TR-632

September 20, 2009

Abstract

Inverse Kinematics (IK) is defined as the problem of determining a set of appropriate joint configurations for which the end effectors move to desired positions as smoothly, rapidly, and as accurately as possible. During the last decades, several methods and techniques, sophisticated or heuristic, have been presented to produce fast and realistic solutions to the IK problem. However, most of the currently available methods suffer from high computational cost and production of unrealistic poses. This report reviews and compares the most popular IK methods regarding reliability, computational cost and conversion criteria, with a novel heuristic and iterative method, called Forward And Backward Reaching Inverse Kinematics (FABRIK). FABRIK avoids the use of rotational angles or matrices, and instead finds each joint position via locating a point on a line. Thus, it converges in fewer iterations, has low computational cost and produces realistic poses. Constraints can easily be incorporated within the FABRIK methodology and multiple chains with multiple end effectors are also easily supported.

Acknowledgements

We would like to thank Richard Wareham and Jonathan Cameron for introducing us to Blender and for their invaluable discussions and help with capturing data and producing video examples. We are also very grateful to Charles Lee for his help with video processing and to Adam Becker for allowing us to use his voice as background to the videos. The environment presented in section 4.3 has been adopted from the work of Jeff Lander [1]; we would like to express our enormous thanks to Jeff for giving us permission to use his code and application.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

THIS report addresses the problem of manipulating articulated figures in an interactive and intuitive fashion for the design and control of their posture. This problem finds its application in the area of robotics, computer animation, ergonomics and the computer games industry. In the area of computer graphics, articulated figures are a convenient model for humans, animals or other virtual creatures from films and video games. The most popular method for animating such models is motion-capture; however, despite the availability of highly sophisticated techniques and expensive tools, many problems appear when dealing with complex figures. Most virtual character models are complicated; they are made up of many joints having a high number of degrees of freedom, thus, it is often difficult to produce a realistic character animation.

1.1 Introduction and Motivation

A *posture* is defined as the skeletal configuration of a figure; for a realistic posture a set of criteria should be satisfied. All character models have natural articulation limits and interpenetration of the body with other objects or themselves is not permitted. In addition, physical laws should be considered as well as numerous personal factors. General constraints can be applied to most articulated figures, however special cases of posture control are needed when large number of degrees of freedom exist.

Inverse Kinematics (IK) is a method for computing the posture via estimating each individual degree of freedom in order to satisfy a given task; it plays an important role in the computer animation and simulation of articulated figures. Inverse Kinematics finds applications in several areas. IK methods have been implemented in many computer graphic and robotics applications, aiming to animate or control different virtual creatures. They are also very popular in the video games industry. The field of computer-aided ergonomics is also concerned with articulated figures, especially human models developed for simulation and prediction purposes. The need for accurate biomechanical modelling and body sizing based on anthropometric data make IK methods a popular approach for fast and reliable solution. IK has been used in rehabilitation medicine in order to observe asymmetries or abnormalities. Recently, IK techniques have also been applied in protein science for protein structure prediction.

In this work, the most popular Inverse Kinematic techniques are reviewed. A new heuristic iterative method, FABRIK, is also presented for solving the IK problem in different scenarios. FABRIK (Forward And Backward Reaching Inverse Kinematics) is an efficient method for solving the IK problem; it uses a forward and backward iterative approach, finding each joint position via locating a point on line. FABRIK has been utilised in highly complex systems with single and multiple targets, with and without joint restrictions. It can easily handle end effector orientations and support, to the best of our knowledge, all chain classes. A reliable method for incorporating constraints is also presented and utilised within FABRIK. The proposed method retains all the advantages of FABRIK, producing visually smooth movements without oscillations and discontinuities, with low computational cost. Several experiments have been implemented for comparison purposes between the most popular manipulator solvers, including multiple end effectors with multiple tasks, and highly constrained joints. The algorithms are tested for reliability, computational cost, realistic movements, reconstruction quality, conversion criteria and number of iterations.

1.2 Literature Review and Motivation

The problem of IK has been extensively studied during recent decades. It was first used in the field of robot technology, where it was utilised to compute the poses of the robots. Since then it has been integrated in the world of animation, computer graphics and in the computer game industry, where it is used for computing the poses of articulated animated figures.

The production of realistic and plausible motions remains an open challenge within the robotics and animation communities. Several models have been implemented for solving the IK problem from many different areas of study. [2] solves the IK task as a problem of finding a local minimum of a set of non-linear equations, defining Cartesian space constraints. However, the most popular numerical approach is to use the Jacobian matrix to find a linear approximation to the IK problem. The *Jacobian solutions* linearly model the end effectors' movements relative to instantaneous system changes in link translation and joint angle. Several different methodologies have been presented for calculating or approximating the Jacobian inverse, such as the Jacobian Transpose, Damped Least Squares (DLS), Damped Least Squares with Singular Value Decomposition (SVD-DLS), Selectively Damped Least Squares (SDLS) and several extensions [3, 4, 5, 6, 7, 8]. Jacobian inverse solutions produce smooth postures; however most of these approaches suffer from high computational cost, complex matrix calculations and singularity problems. An alternative approach is given by Pechev in [9] where the inverse kinematics problem is solved from a control prospective. This approach is computationally more efficient than the pseudo-inverse based methods and does not suffer from singularity problems.

The second family of IK solvers is based on Newton methods. These algorithms seek target configurations which are posed as solutions to a minimisation problem, hence they return smooth motion without erratic discontinuities. The most well known methods are Broyden's method, Powell's method and the Broyden, Fletcher, Goldfarb and Shanno (BFGS) method [10]. However, the Newton methods are complex, difficult to implement and have high computational cost per iteration.

Recently, [11] and [12] proposed a Sequantial Monte Carlo Method (SMCM) and Particle filtering approach respectively. Neither method suffers from matrix singularity problems and both perform reasonably well. However, these statistical methods have high computational cost. Another recent approach is presented in [13], which is a direct extension of [14]. The inputs to this method, which is called Sequential IK (SIK), are end effector positions, such as wrists, ankles, head and pelvis, which are used to find the human pose. The IK problem is then solved sequentially using simple analytic-iterative IK algorithms, in different parts of the body, in a specific order. [13] also presents a comparison with several IK methods regarding their joint average position error, joint average orientation error and the median processing time of each methodology. [15, 16, 17] use mesh-based Inverse Kinematic techniques to configure the animated shapes. Mesh-based IK learns a space of natural deformations from example meshes. Using the learned space, they generate new shapes that respect the deformations exhibited by the examples, yet still satisfy vertex constraints imposed by the user.

A very popular IK method is the Cyclic Coordinate Descent (CCD) algorithm, which was first introduced by [18] and then biomechanically constrained by [19]. CCD has been extensively used in the computer game industry [1] and has recently been adapted for protein structure prediction [20]. CCD is a heuristic iterative method with low computational cost for each joint per iteration, which can solve the IK problem without matrix manipulations; thus it formulates a solution very quickly. However, CCD has some disadvantages; it can suffer from unrealistic animation, even if manipulator constraints have been added, and often produces motion with erratic discontinuities. It is designed to handle serial chains, thus, it is difficult to extend to problems with multiple end effectors. A deeper analysis of each methodology is given in Chapter 3.

[21] presents a real-time method for rope simulation which proposes a 'Follow-the-Leader' (FTL) non-iterative technique which is similar to each individual iteration of FABRIK. However, this method does not use points and lines to estimate joint positions, does not address the IK problem and does not support constraints or joint orientations. Although it has strong similarities to the FTL method, FABRIK is the first technique that uses an iterative forward

and backward method using lines and points for solving the IK problem.

The facts that many of these algorithms produce unrealistic solutions and that the motions of human joints are limited, implies that joint restrictions need to be added. Constraints have been incorporated into several methods in an effort to produce realistic motion movements. Most of the literature incorporates motion constraints by weighting the move of each individual joint. [19] proposed a simple projection of the unconstrained solution onto a feasible posture; this method cannot however guarantee an optimal solution. Fedor in [22] presents a penalty-based method that adds movement restrictions, with the drawback that it often converges to poor results. Recent works have proposed using motion capture data to constrain the motion [15, 23, 24]. [25] and [11] use a learned statistical dynamic model in a constraint-based motion optimisation framework for adding restrictions to their solutions.

1.3 Outline of the Report

The body of this report is divided into 5 chapters. Chapters 1 and 2 introduce readers to the motion, robotic and kinematics problems. Chapter 2 describes the skeletal model of a human, the possible human movements and the variety and features of all human joints.

Chapter 3 presents the Forward and Inverse Kinematics problem and discusses the most popular solutions during the last couple of decades. It is divided into 5 sections; each section presents a solution of the IK problem from a different area. It briefly describes the advantages and disadvantages of each family of methods including the Inverse Jacobian methods, the Newton methods, the Sequential Monte Carlo Methods and the Heuristic methods. It also presents a new heuristic approach, FABRIK, which solves the IK problem in an iterative fashion. FABRIK is faster than any proposed method to date and returns visually the most realistic poses without erratic discontinuities, matrix manipulations or singularity problems.

Chapter 4 presents, compares and discusses the results of the most popular methods proposed in Chapter 3 under several conditions. Each methodology is implemented for several cases (single or multiple targets) and tested for its realistic movement, reliability and computational cost with or without joint restrictions. Chapter 4 also presents several implementations of FABRIK within a hand and a humanoid model and analyses the results in terms of visual realism of movements, reconstruction quality, computational cost and conversion criteria.

Finally, Chapter 5 presents conclusions and suggestions for future work.

2 The Articulated Body Model

In this section, a brief introduction to the human skeleton and joint modelling is presented. Before motion data can be edited by any system, it usually needs to be preprocessed to ensure that correct hierarchical connections and constraints are satisfied. Human body modelling is a problem that arises in ergonomics and in computer graphics applications. It is a complex hierarchical model consisting of many joints, each one having different degrees of freedom (DoF) and various possible restrictions. In fact, the human body consists of more than 200 bones and joints.

2.1 Human Body Modelling

A rigid multibody system consists of a set of rigid objects, called *links*, connected together by *joints*. A *joint* is the component concerned with motion; it permits some degree of relative motion between the connected segments. Virtual body modelling is important for human posture control. A well constrained model can restrict postures to a feasible set, therefore allowing a realistic motion. Most models assume that body parts are rigid, although this is just an assumption approximating reality. The skeletal structure is usually modeled as a hierarchy of rigid segments connected by joints, each defined by their length, shape, volume and mass properties. The skeletal structures are often defined using a parent-child system (see figure 2.1). The size, shape and proportions of the body and its segments are also essential in order to build models with realistic dimensions and proportions.

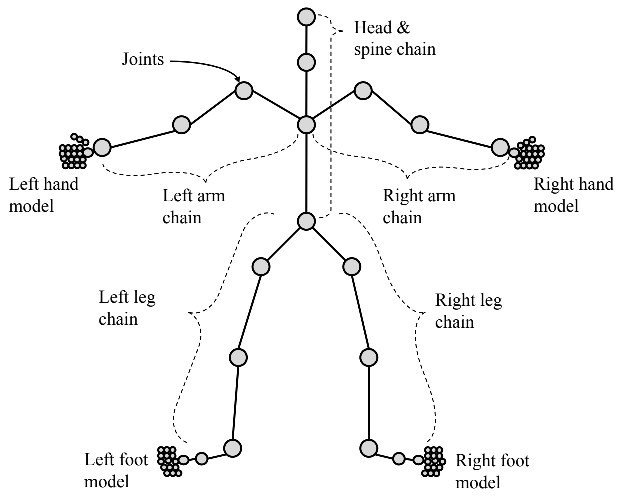


Figure 2.1: *An example of a skeletal structure of a human.*

Figure 2.2 shows examples of a model of a human body and human legs taken by a motion capture system (Phasespace [26]) and graphically processed in Blender [27]. The joints are shown as spheres.

A manipulator such as a robot arm or an animated graphics character is modeled as a *chain* composed of rigid *links* connected at their end by rotating *joints*. Any translation and/or rotation of the i-th joint affects the translation and rotation of any joint placed later in the chain. The chains are built under the assumption that all bones have at most one parent and any number of children. The chains can be formalised as follow: All bones (joints) with no children are marked as *end effectors*; a chain can be built for each end effector by moving back through the skeleton, going from parent to parent, until the root is reached. There are a variety of possible joint types. For a well designed human model, it is essential to study these joint types. Each joint provides a local rotation (and each bone a local translation) with different degrees of freedom (DoF). Different rotation paradigms arise from different joint types. The main human joint types are enumerated below (see also figure 2.3):

1. *The suture joint model* (1 DoF): This is a fixed joint that allows very limited movement. Suture joints can be found in the skull. The bones in the skull are held together with fibrous connective tissue.

2. *The hinge joint model* (1 DoF): The simplest type of joint; it can be found in the elbows, knees and the joints of the fingers and toes. Hinge joints allow movement in only one direction.

3. *The gliding joint model* (2 DoF): Gliding joints permit a wide range of mostly sideways movements - as well as movements in one direction.

4. *The saddle joint model* (2 DoF): A saddle joint is more versatile than either a hinge joint or a gliding joint. It allows movement in two directions.

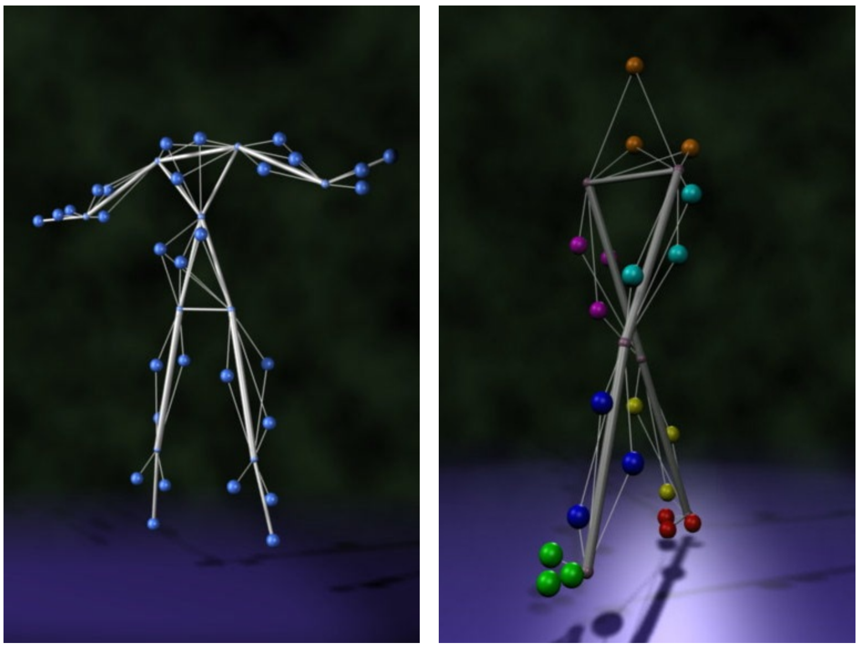


Figure 2.2: *Part of a skeletal animation presenting joints of a human body (left) and human legs (right).*

5. *The pivot joint model* (2 DoF): The pivot joint is a 2 degree of freedom joint and it can be found in the neck allowing a side to side turn of the head.

6. *The ball and socket joint model* (3 DoF): This is the most mobile type of joint in the human body; it allows 3 degrees of freedom. A limited (in the sense of restricted magnitude) version of the ball and socket joint is the Ellipsoidal joint.

It is also possible to work with more general types of joints, and thereby simulate non-rigid objects.

Kinematic joint models must be defined in order to formalise the relative motion of each joint. An analytically and anatomically correct model is necessary to control and constrain the available movements of the human body. These models are mainly characterised by the number of parameters which describe the motion space and are usually constrained by joint limits and joint structure [28, 29]. Because of their complex nature, most of the proposed joint models are simplified or approximated by more than one joint. There are many different models, each one performing different movements. Each specific model can be expressed via multiple joints of different types together with their movements and degrees of freedom. The most well-known models are: *the shoulder model*, a very complex model composed of 3 different joints [30, 31]; *the spine model*, a complex arrangement of 24 vertebrae (usually, for simplicity, the spine is modelled as a simple chain of joints [28, 32, 33, 34]); *the hand model*, this is the most versatile part of the body comprising a large number of joints [35, 36, 37]; *the strength model*, which takes account of the forces applied from the skeletal muscles to the bones [28].

A realistic body appearance is also very important in many graphical applications. Thus, data additional to the skeletal structure must be added for the generation of a more realistic

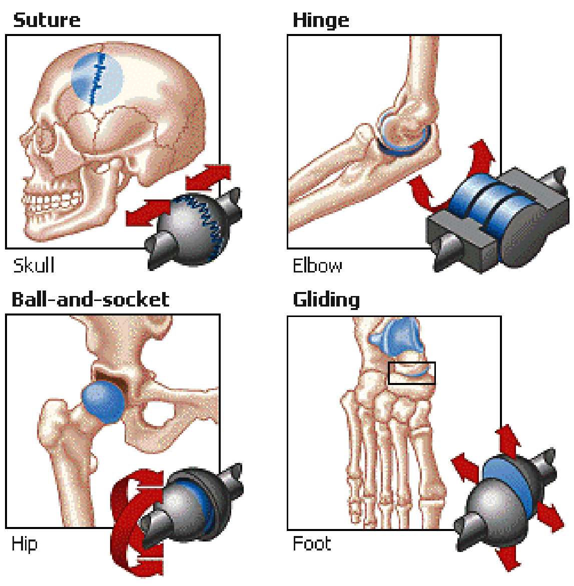


Figure 2.3: *Human joints with their available movements. The images have been taken from the Microsoft Encarta Online Encyclopedia 2008 [40].*

human animation with skin, face, clothes etc [38].

Figure 2.3 shows an example of human joints with their available degrees of freedom. More details about human body and kinematic joint models can be found in [28, 29, 30, 33, 36, 39].

2.2 Motion

Once a body model has been defined, it can then be animated, manipulated or simply used for simulation purposes. Animating articulated figures is highly dependent on their allowed motion. *Motion* is the change in position of an object with respect to a reference. A motion can be achieved when a rotational or translational transformation has been applied in order to move the end effector(s) of a chain to a desired position. There are two main issues related to motion and these are given below.

- *Forward Kinematics* (FK): can be defined as the problem of locating the end effectors' positions after applying known transformations to the chain.

- *Inverse Kinematics* (IK): is described as the problem of determining an appropriate joint configuration for which the end effectors move to desired positions, named *target positions*, as smoothly, rapidly, and as accurately as possible.

During recent decades, many methods have been proposed to solve the IK problem. However, for a complete IK solver it is important to apply restrictions in order to control the joint configurations, according to the joint type. Moreover, we often have models with multiple end effectors and multiple targets. Performing single tasks sequentially is not a practical way of controlling complex figures. Therefore, it is desirable for a resolution technique to be able to manage multiple tasks with an appropriate strategy.

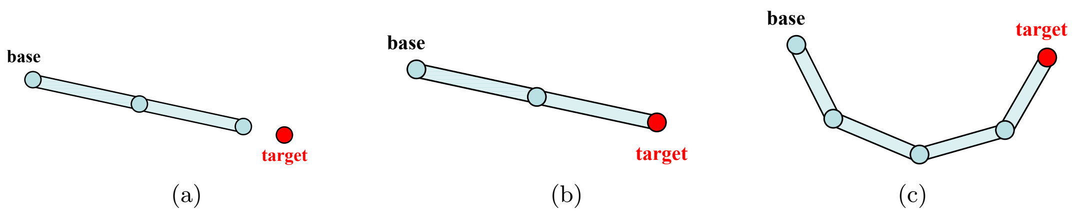


Figure 2.4: *Possible solutions of the IK problem: (a) The target is unreachable; in many cases it is impossible for the linked structure to touch the target, (b) One solution; there instances where there is only one solution to the problem, (c) Many solution; most of the times the IK problem has more than a single solution.*

The FK problem has a unique solution, and its success depends on whether the joints are allowed to do the desired transformation. In contrast, when dealing with IK, it is not always the case that a solution can be achieved. There are instances where the goal is unreachable or when two or more tasks conflict and cannot be satisfied simultaneously. Unreachable targets are the targets which can be further than the chain can reach or can be at a point where no pivoting of links can bend the chain to reach (see figure 2.4). These problems are known as *over-constrained* problems. On the other hand, there are instances where more than a single solution exists. It is up to the IK method to choose the best solution and the IK solver's performance is ranked according to how realistic the solution is and the computational cost of choosing that solution.

2.3 Obtaining joint positions

Motion capture hardware, such as that provided by Phasespace [26], is under constant development, providing real-time acquisition of labelled 3D marker data. These data can be used for reconstruction of the human skeleton allowing accurate real-time feedback via tracking and modelling of human motion. Throughout this work, markered motion capture systems will be used in order to provide information related to the human skeleton and for localisation of the joints. More details of how the joint positions are obtained can be found in [41, 42]

2.4 Mathematical Background

Before proceeding, it is useful to introduce the mathematical background used within this report. Most of the proposed techniques are implemented using object orientations and rotations. Geometric Algebra (GA) provides a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. GA consists of three main operations, the *inner product*, the *outer product*, and the *geometric product*. The multivector basis of GA, with vector elements in $\mathbb{R}^3$, can be defined via outer products of the three orthonormal basis vectors e_1, e_2 and e_3. Table 2.1 presents the multivector orthonormal basis.

The highest grade basis element in GA is referred to as the pseudoscalar and often denoted as $I = e_1 \wedge \ldots \wedge e_n$ for an n-dimensional space. GA can also be used for rotating vectors;

Table 2.1: *The orthonormal basis for computation*

Scalar:	a		
Vector:	e_1	e_2	e_3
Bivector:	$e_{12} = e_1 \wedge e_2$	$e_{13} = e_1 \wedge e_3$	$e_{23} = e_2 \wedge e_3$
Trivector:	$e_{123} = e_1 \wedge e_2 \wedge e_3$		

the scalar plus bivector $R = \exp(-B\phi/2)$, also known as a *rotor*, represents an anticlockwise rotation ϕ in a plane specified by the bivector B. The transformation is given by $x \mapsto RxR^{-1}$. More information about Geometric Algebra can be found in [43, 44].

Rotors in GA are simpler to manipulate than Euler angles and avoid the problem of gimbal lock. Gimbal lock is a common problem associated with Euler angles and occurs because two axes become aligned during rotational operations, producing unexpected behavior since one degree of freedom is lost. GA is also more numerically stable and more efficient than rotation matrices making it popular for applications in computer graphics and robotics.

3 Numerical Solution of the Inverse Kinematic Problem

The IK problem puzzled scholars for many years in the field of robotics technology and computer graphics. During recent decades, several algorithms have been implemented for computing the poses of a robot. The most popular techniques for solving the IK problem are presented in this chapter.

3.1 Introduction

Let the complete joint configuration of the multibody be specified by the scalars $\theta_1, ..., \theta_n$, assuming that there are n joints and each θ_j value is called a *joint angle* (joint configuration may not always be an angle), where θ_j is the angle in the plane of rotation assuming we also have knowledge of the rotation axis. Certain points on the links are identified as *end effectors*. To solve the IK problem, the joint angles must be settled so that the resulting configuration of the multibody places each end effector at, or as close as possible to, its target position. If there are k end effectors, let their positions be denoted as $\mathbf{s}_1, ..., \mathbf{s}_k$ relative to a fixed origin. Each end effector position $\mathbf{s}_i$ is a function of the joint angles. The column vector $(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_k)^T$ can be written as $\vec{\mathbf{s}}$; this can be viewed as a column vector either with $m = 3k$ scalar entries or with k entries from $\mathbb{R}^3$. One way to control the multibody is to specify target positions, one for each end effector. The target positions are also defined by a vector $\vec{\mathbf{t}} = (\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_k)^T$, where $\mathbf{t}_i$ is the target position for the i-th end effector. Let $\mathbf{e}_i = \mathbf{t}_i - \mathbf{s}_i$, be the desired change in position of the i-th end effector (moving to the desired i-th target). This equation can be rewritten as $\vec{\mathbf{e}} = \vec{\mathbf{t}} - \vec{\mathbf{s}}$.

The joint angles are also written as a column vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)^T$. The end effector positions are functions of the joint angles; this fact can be expressed as

$$\vec{\mathbf{s}} = f(\boldsymbol{\theta}) \tag{3.1}$$

or, for $i = 1, ..., k$, $\vec{\mathbf{s}}_i = f_i(\boldsymbol{\theta})$. This is called the *Forward Kinematics* (FK) solution.

The goal of *Inverse Kinematics* (IK) is to find a vector $\boldsymbol{\theta}$ such that $\vec{\mathbf{s}}$ is equal to a given desired configuration $\vec{\mathbf{s}}_d$:

$$\boldsymbol{\theta} = f^{-1}(\vec{\mathbf{s}}_d) \tag{3.2}$$

where f is a highly non linear operator which is difficult to invert.

However, there are instances where a solution to the Inverse Kinematics problem does not exist due to an unreachable target or where the (best) solution is not unique. Even in well-behaved situations, a closed-form equation cannot generally be achieved. Therefore, the use of iterative methods to approximate a good solution to the problem seems to be necessary. Such methods are described in this chapter.

3.2 Jacobian Inverse Methods

The Jacobian J is a matrix of partial derivatives of the entire chain system relative to the end effectors $\mathbf{s}$. The *Jacobian solutions* are a linear approximation of the IK problem (see figure 3.1); they linearly model the end effectors' movements relative to instantaneous system changes in link translation and joint angle. The Jacobian matrix J is a function of the $\boldsymbol{\theta}$ values and is defined by

$$J(\boldsymbol{\theta})_{ij} = \left(\frac{\partial \mathbf{s}_i}{\partial \theta_j}\right)_{ij} \tag{3.3}$$

where $i = 1, ..., k$ and $j = 1, ..., n$. Orin and Schrader in [45] discussed how to calculate the Jacobian matrix entries for different representations of joints and multibodies. The Jacobian matrix entries for the j-th rotational joint can be calculated as follows

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j) \tag{3.4}$$

where $\mathbf{p}_j$ is the position of the joint, and $\mathbf{v}_j$ is the unit vector pointing along the current axis of rotation for the joint. Note that J can be viewed either as a $k \times n$ matrix whose entries are vectors in $\mathbb{R}^3$, or as an $m \times n$ matrix with scalar entries ($m = 3k$).

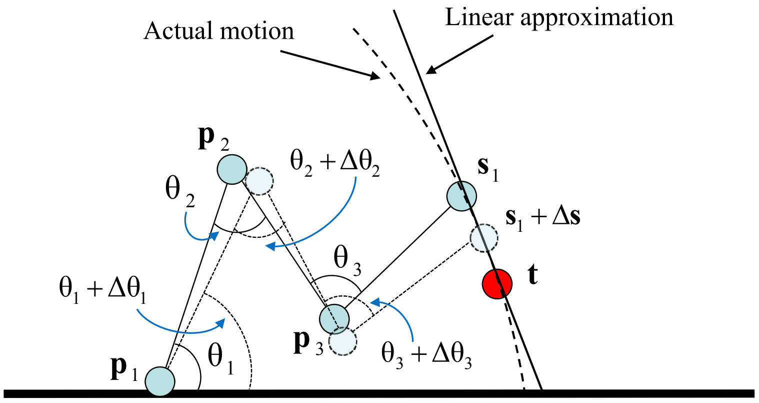


Figure 3.1: *The Jacobian solution is a linear approximation of the actual motion of the kinematic chain.*

Equation 3.1 for forward dynamics can now be written as

$$\dot{\vec{s}} = J(\boldsymbol{\theta})\,\dot{\boldsymbol{\theta}} \tag{3.5}$$

where the dot notation specifies the first derivative with respect to time. Using the current values $\boldsymbol{\theta}$, $\vec{s}$ and $\vec{t}$, the Jacobian $J = J(\boldsymbol{\theta})$ can be computed. We then seek an update value $\Delta\boldsymbol{\theta}$ for the purpose of incrementing the joint angles $\boldsymbol{\theta}$ by $\Delta\boldsymbol{\theta}$:

$$\boldsymbol{\theta} := \boldsymbol{\theta} + \Delta\boldsymbol{\theta} \tag{3.6}$$

The change in end effector positions caused by this change in joint angles can be estimated as

$$\Delta\vec{s} \approx J\Delta\boldsymbol{\theta} \tag{3.7}$$

The idea is that the $\Delta\boldsymbol{\theta}$ value should be chosen so that $\Delta\vec{s}$ is approximately equal to $\vec{e}$, although it also common to choose $\Delta\boldsymbol{\theta}$ so that the approximate movement $\Delta\vec{s}$ in the end effectors (partially) matches the velocities of the target positions.

Thus, the FK problem can be expressed as $\vec{e} = J\Delta\boldsymbol{\theta}$ and the IK problem can be rewritten as $\Delta\boldsymbol{\theta} = J^{-1}\vec{e}$. In most cases, the IK equation cannot be solved uniquely. Indeed, the Jacobian J may not be square or invertible, and even if it is invertible, J may work poorly as it may be nearly singular[1]. Several approaches have been proposed to overcome these problems. Such methods are presented and discussed in the rest of this chapter.

3.2.1 Jacobian Pseudo-inverse

The Jacobian Pseudo-inverse, also known as the *Moore-Penrose* inverse of the Jacobian, sets the value $\Delta\boldsymbol{\theta}$ equal to

$$\Delta\boldsymbol{\theta} = J^{\dagger}\vec{e} \tag{3.8}$$

[1]Singularities occur when no change in joint angle can achieve a desired change in chain end position.

where $J^{\dagger}$ is an $n \times m$ matrix and is called the *pseudo-inverse* of J. It is defined for all matrices J, even ones which are not square or not of full row rank. The pseudo-inverse gives the best possible solution to the equation $J\Delta\boldsymbol{\theta} = \vec{\mathbf{e}}$ in the least squares sense.

The pseudo-inverse has the property that the matrix $(I - J^{\dagger}J)$ performs a projection onto the nullspace of J . Therefore, for all vectors φ, $J(I - J^{\dagger}J)\varphi = 0$. This means that we can set $\Delta\boldsymbol{\theta}$ by

$$\Delta\boldsymbol{\theta} = J^{\dagger}\vec{\mathbf{e}} + (I - J^{\dagger}J)\varphi \tag{3.9}$$

for any vector φ and still obtain a value for $\Delta\boldsymbol{\theta}$ which minimises the value $J\Delta\boldsymbol{\theta} - \vec{\mathbf{e}}$. Several authors have used the nullspace method to help avoid singular configurations [46, 47]. A more sophisticated nullspace method, the *Extended Jacobian* method, was introduced by Baillieul [5]; in this version a local minimum value of a function is tracked as a secondary objective.

The pseudo-inverse method can be derived as follows:

$$J^T J\Delta\boldsymbol{\theta} = J^T\vec{\mathbf{e}} \tag{3.10}$$

Then let $\vec{\mathbf{z}} = J^T\vec{\mathbf{e}}$ and solve the equation

$$J^T J\Delta\boldsymbol{\theta} = \vec{\mathbf{z}} \tag{3.11}$$

It can be shown that $\vec{\mathbf{z}}$ is always in the range of $J^T J$, hence the above equation always has a solution. When J is full row rank, $J^T J$ or JJ^T is guaranteed to be invertible. In this case, the minimum magnitude solution $\Delta\boldsymbol{\theta}$ can be expressed as

$$\Delta\boldsymbol{\theta} = \left(J^T J\right)^{-1} J^T\vec{\mathbf{e}} \equiv J^T\left(JJ^T\right)^{-1}\vec{\mathbf{e}} \tag{3.12}$$

The pseudo-inverse method is widely discussed in the literature, however it often performs poorly because of its instability near singularities.

3.2.2 Jacobian Transpose

The Jacobian transpose method was first used for inverse kinematics in [3, 4]. The idea is to use the transpose of the Jacobian instead of its inverse. Hence,

$$\Delta\boldsymbol{\theta} = \alpha J^T\vec{\mathbf{e}} \tag{3.13}$$

for some appropriate scalar α. Obviously the transpose of the Jacobian is not the same as the inverse; however, [3, 4] justify the use of the transpose in terms of virtual forces. We can easily show that for all J and $\vec{\mathbf{e}}$, $\left\langle JJ^T\vec{\mathbf{e}}, \vec{\mathbf{e}}\right\rangle \geq 0$, where $\langle\mathbf{a}, \mathbf{b}\rangle$ indicates the dot product between vectors $\mathbf{a}$ and $\mathbf{b}$,

$$\left\langle JJ^T\vec{\mathbf{e}}, \vec{\mathbf{e}}\right\rangle = \left\langle J^T\vec{\mathbf{e}}, J^T\vec{\mathbf{e}}\right\rangle = \|J^T\vec{\mathbf{e}}\|^2 \geq 0 \tag{3.14}$$

Therefore, if we update the angles $\Delta\boldsymbol{\theta}$ in eq. 3.13 by a sufficiently small $\alpha \geq 0$, the end effector positions will be changed by $\alpha JJ^T\vec{\mathbf{e}}$. α can be calculated by minimising the new value of the error vector $\vec{\mathbf{e}}$ after each update. Assuming that the end effector position change is equal to $\alpha JJ^T\vec{\mathbf{e}}$, α is chosen to make this value as close as possible to $\vec{\mathbf{e}}$. Thus α is given by

$$\alpha = \frac{\langle \vec{\mathbf{e}}, JJ^T\vec{\mathbf{e}} \rangle}{\langle JJ^T\vec{\mathbf{e}}, JJ^T\vec{\mathbf{e}} \rangle} \tag{3.15}$$

3.2.3 Singular Value Decomposition

The singular value decomposition (SVD) provides a powerful method for utilising the pseudo-inverse Jacobian. Let J be the Jacobian matrix. A singular value decomposition of J consists of expressing J in the form

$$J = UDV^T \tag{3.16}$$

where U and V are orthogonal matrices and D is diagonal. For an $m \times n$ Jacobian matrix, U is $m \times m$, D is $m \times n$, and V is $n \times n$. The non-zero entries of the D matrix are the values $\sigma_i = d_{ii}$ along the diagonal. It is assumed that $m \leq n$ and, without loss of generality, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m \geq 0$. Note that there are cases where $\sigma_i = 0$, for some i. In fact, the rank of J is equal to the largest value r such that $\sigma_r \neq 0$ and $\sigma_i = 0$ for $i > r$, . We use $\mathbf{u}_i$ and $\mathbf{v}_i$ to denote the i-th columns of U and V respectively. Their orthogonality implies that their columns form an orthonormal basis for $\mathbb{R}^m$ (respectively $\mathbb{R}^n$). The vectors $\mathbf{v}_{r+1}, ..., \mathbf{v}_n$ are an orthonormal basis for the nullspace of J. The singular value decomposition of the Jacobian J always exists, and can be formed as

$$J = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \tag{3.17}$$

The transpose, D^T, of D is the $n \times m$ diagonal matrix form. The product DD^T is the $m \times m$ matrix form with diagonal entries d_{ii}^2. The pseudo-inverse, $D^\dagger = \left(d_{ii}^\dagger\right)$, of D is an $n \times m$ diagonal matrix with diagonal entries

$$d_{ij}^\dagger = \begin{cases} 1/d_{ij} & \text{if } d_{ii} \neq 0 \\ 0 & \text{if } d_{ii} = 0 \end{cases} \tag{3.18}$$

The pseudo-inverse of the Jacobian is thus equal to $J^\dagger = VD^\dagger U^T$ and can be rewritten as

$$J^\dagger = \sum_{i=1}^{r} \sigma_i^{-1} \mathbf{v}_i \mathbf{u}_i^T \tag{3.19}$$

3.2.4 Damped Least Squares

The Damped Least Squares method (DLS) was first used for inverse kinematics by [6, 7]. DLS avoids many of the pseudo-inverse method's problems with singularities and can give a numerically stable method of selecting $\Delta\boldsymbol{\theta}$. In the DLS method, instead of finding the minimum vector $\Delta\boldsymbol{\theta}$ that gives a best solution to equation $\vec{\mathbf{e}} = J\Delta\boldsymbol{\theta}$, we find the value of $\Delta\boldsymbol{\theta}$ that minimises the quantity

$$\|J\Delta\boldsymbol{\theta} - \vec{\mathbf{e}}\|^2 + \lambda^2\|\Delta\boldsymbol{\theta}\|^2 \tag{3.20}$$

where $\lambda \in \mathbb{R}$ is a non-zero damping constant. This is given by

$$\left(J^TJ + \lambda^2 I\right)\Delta\boldsymbol{\theta} = J^T\vec{\mathbf{e}} \tag{3.21}$$

It is shown that $J^TJ + \lambda^2 I$ is non-singular, thus the DLS solution is equal to

$$\Delta\boldsymbol{\theta} = \left(J^TJ + \lambda^2 I\right)^{-1} J^T\vec{\mathbf{e}} \tag{3.22}$$

Now J^TJ is an $n \times n$ matrix, where n is the number of degrees of freedom. It is easily proven that $\left(J^TJ + \lambda^2 I\right)^{-1} J^T = J^T\left(JJ^T + \lambda^2 I\right)^{-1}$; the advantages of this transform over the one in eq. 3.22 is that the matrix being inverted is $m \times m$ where $m = 3k$ is the dimension of the space of the target positions, and m is often much less than n. Thus,

$$\Delta\boldsymbol{\theta} = J^T\left(JJ^T + \lambda^2 I\right)^{-1}\vec{\mathbf{e}} \tag{3.23}$$

The damping constant depends on the details of the multibody and the target positions and must be chosen carefully to make equation 3.23 numerically stable. The damping constant should be large enough so that the solutions for $\Delta\boldsymbol{\theta}$ are well-behaved near singularities, but if it is too large, the convergence rate is slow.

3.2.5 Pseudo-inverse Damped Least Squares

The Pseudo-inverse Damped Least Squares uses the singular value decomposition (SVD) under the damped least squares method. Hence, the matrix $JJ^T + \lambda^2 I$ can be rewritten as

$$JJ^T + \lambda^2 I = \left(UDV^T\right)\left(VD^TU^T\right) + \lambda^2 I = U\left(DD^T + \lambda^2 I\right)U^T \tag{3.24}$$

The matrix $DD^T + \lambda^2 I$ is a diagonal matrix with entries $\sigma_i^2 + \lambda^2$. It is clearly non-singular with inverse an $m \times m$ diagonal matrix with non-zero entries $\left(\sigma_i^2 + \lambda^2\right)^{-1}$. Therefore,

$$J^T\left(JJ^T + \lambda^2 I\right)^{-1} = VD^T\left(DD^T + \lambda^2 I\right)^{-1}U^T = VEU^T \tag{3.25}$$

where E is an $n \times m$ diagonal matrix with entries

$$e_{i,i} = \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \tag{3.26}$$

Thus, the pseudo-inverse DLS solution can be expressed in the form

$$J^T \left(JJ^T + \lambda^2 I\right)^{-1} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \mathbf{v}_i \mathbf{u}_i^T \tag{3.27}$$

Comparing the pseudo-inverse DLS with the simple pseudo-inverse method, we observe that in both cases the Jacobian is inverted by an expression $\sum_{i=1}^{n} \tau_i \mathbf{v}_i \mathbf{u}_i^T$. In the case of the simple pseudo-inverse $\tau_i = \sigma_i^{-1}$, whereas for the pseudo-inverse DLS method, $\tau_i = \sigma_i / \sigma_i^2 + \lambda^2$. The simple pseudo-inverse method is unstable as σ_i approaches zero. Pseudo-inverse DLS acts similarly to the more simple version away from singularities, but smooths out the performance of the simple pseudo-inverse method in areas close to singularities.

3.2.6 Selectively Damped Least Squares

The Selectively Damped Least Squares (SDLS) method was presented by Buss and Kim in [8] and is an extension of the pseudo-inverse Damped Least Squares method. SDLS adjusts the damping factor separately for each singular vector of the Jacobian SVD based on the difficulty of reaching the target positions. The damping constants of SDLS depend not only on the current configuration of the articulated multibody, but also on the relative positions of the end effector and the target position. This method converges in fewer iterations and does not require ad hoc damping constants. SDLS also performs better than any other inverse Jacobian method when multiple end effectors exist. The DLS and pseudo-inverse DLS methods are computationally cheaper and easier to code than the SDLS method; however, SDLS offers improved performance for applications where runtime is not restricted and where it is difficult to choose a good damping constant.

3.2.7 Incorporating constraints

There exist several ways to improve the performance and increase the realism of an animation; one of these is to incorporate constraints. However, implementing constraints in the Jacobian family of methods is not straightforward. A simple projection of the unconstrained solution onto a feasible posture has been proposed in [19]. However, it is not guaranteed that the result will lie close to an optimal solution. A penalty-based method adding movement restrictions is presented in [22], with the drawback that this often converges to poor results. The simplest way of incorporating constraints can be achieved by weighting the moves of the individual joints [48]. Given an update vector $\mathbf{p}$ and a weight matrix W, where $W = w^T I$ and w is a vector of weights on the individual joints, the weighted update $\mathbf{p}_w$ is given by $\mathbf{p}_w = W\mathbf{p}$.

3.2.8 Feedback Inverse Kinematics

The Feedback Inverse Kinematics (FIK) method [9] solves the inverse kinematics problem from a control prospective, minimising the difference between demanded and actual Cartesian velocities. Within the feedback loop, the required joint parameters are derived through a control sensitivity function. The algorithm operates as a filter and does not require matrix manipulations (inversion or singular value decomposition). Singularities are handled without the necessity of a damping factor and this makes it computationally more efficient than pseudo-inverse based methods. [9] also describes how manipulator constraints can be applied, weighting both joints and end-effectors to a more feasible set of postures. As with the other Jacobian-based algorithms, it can easily handle problems with multiple end effectors.

3.3 Newton Methods

The Newton family of methods is based on a second order Taylor series expansion of the object function $f(x)$:

$$f(x+\sigma) \approx f(x) + [\nabla f(x)]^T \sigma + \frac{1}{2}\sigma^T H_f(x)\sigma \tag{3.28}$$

where $H_f(x)$ is the Hessian matrix. However, the calculation of the Hessian matrix is very complex and it results in high computational cost for each iteration. Hence, several approaches have been proposed which, instead of calculating the Hessian matrix, use an approximation of the Hessian matrix based on a function gradient value. The most well known methods are Broyden's method, Powell's method and the Broyden, Fletcher, Goldfarb and Shanno (BFGS) method [10, 49].

Since the Newton methods are posed as a minimisation problem, they return smooth motion without erratic discontinuities. It is also straightforward to incorporate joint restrictions. The most obvious method for constraints is the gradient projection method proposed by Zhao in [2]. The Newton methods also have the advantage that they do not suffer from singularity problems, such as that which occurs when finding the Jacobian Inverse; however they are complex, difficult to implement and have high computational cost per iteration.

3.4 IK using Sequential Monte Carlo Methods

Sequential Monte Carlo Methods (SMCM) have been recently introduced for solving IK problems. Courty and Arnaud in [11] proposed such a solution based on the sampling principle. Using a sampling approach, the inverse kinematics problem can be solved with forward kinematics, hence the numerical inversion of the forward operator can be avoided. The problem is cast as a hidden Markov model (HMM), whose hidden state is given by all the parameters that define the articulated figure. Hence, the state space consists of all the possible configurations of the state. The inverse kinematics is then reformulated in a filtering framework. The proposed SMCM IK solver does not require explicit numerical inversion and joint restrictions can be added to the system in an intuitive manner. These can be easily implemented without the need for complex optimisation algorithms. A particle IK solver has also been implemented in [12] which uses a body pose goals set and attempts to satisfy the goals by forming a system of constraints over the linked character bodies.

3.5 Style or Mesh-based Inverse Kinematics

[15] presents a style-based IK method which is based on a learned model of human poses. Given a set of constraints, the proposed system can produce, in real-time, the most likely pose satisfying those constraints. The model has been trained on different input data that leads to different styles of IK; it can generate any pose, but poses are highly related to those which are most similar to the space of poses in the training data. In [16], a mesh-based Inverse kinematics (MESH-IK) has been implemented which, instead of using human styles as training data, learns the space of meaningful shapes from example meshes. Using the learned space, MESH-IK generates new shapes that respect the deformations exhibited by the examples, yet still satisfy vertex constraints imposed by the user. [17] describes an extension of the MESH-IK method which provides interactive control of reduced deformable models via an intuitive IK framework. The collection of transformations compactly represents articulated character movement that has been derived automatically from example data. The IK problem is formulated in a reduced space to achieve an independent resolution performance, meaning the speed of the posing task is a function of the model parameters rather than of character geometry. However, this family of methods requires an off-line training procedure and the results are highly depended on the training data and limited only to those models and movements the system has been trained on.

3.6 Heuristic Inverse Kinematics Algorithms

3.6.1 Cyclic Coordinate Descent

Cyclic Coordinate Descent (CCD) [18, 19] is an iterative heuristic search technique that is suitable for interactive control of an articulated body. CCD is one of the most popular IK iterative algorithms; it has been implemented in many computer graphic and robotics applications and is extensively used for solving the inverse kinematic problem in the computer games industry [1]. CCD has also been effectively used in protein science for protein structure prediction and/or structure determination [20].

CCD is very simple to implement and is extremely fast. It provides a numerically stable solution and it has linear-time complexity in the number of degrees of freedom (DoF). The CCD method attempts to minimise position and orientation errors by transforming one joint variable at a time. The algorithm states that, starting from the end effector inward towards the manipulator base, each joint must be transformed in order to move the end effector as close as possible to the target. This procedure is repeated until a satisfactory solution is obtained. The computational cost for each joint is low and therefore a solution can be formulated very quickly. Figure 3.2 gives a visual solution of the IK problem using the CCD algorithm executing over a number of iterations.

Like other inverse kinematics algorithms, CCD can generate many different resulting postures for a given initial posture. It is then very difficult to choose a feasible posture among many resulting postures. Therefore, manipulator constraints must be incorporated to restrict

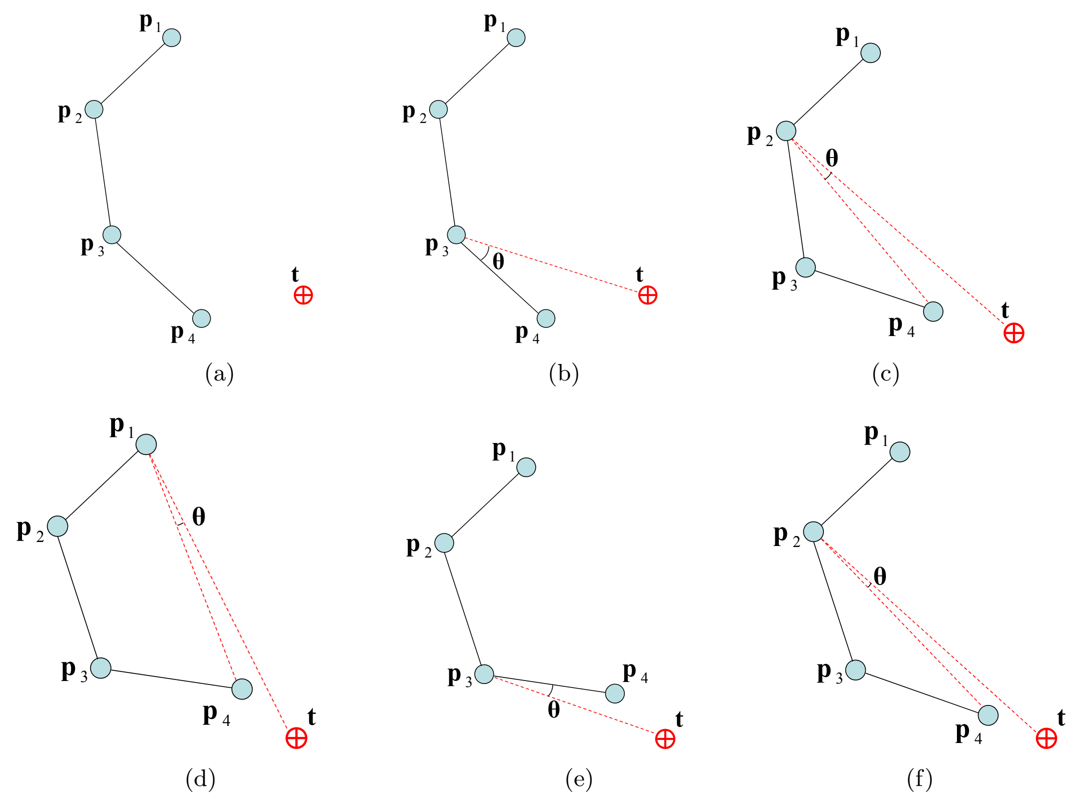


Figure 3.2: *An example of visual solution of the IK problem using the CCD algorithm. (a) The initial position of the manipulator and the target, (b) find the angle θ between the end effector, joint $\mathbf{p}_3$ and the target and rotate the joint $\mathbf{p}_4$ by this angle, (c) find the angle θ between the end effector, joint $\mathbf{p}_2$ and the target and rotate joints $\mathbf{p}_4$ and $\mathbf{p}_3$ by this angle, (d), (e) and (f) repeat the whole process for as many iterations as needed. Stop when the end effector reaches the target or gets sufficiently close.*

motion to a feasible posture. In CCD it is easy to apply local constraints but it is more difficult to implement global manipulation restrictions.

CCD only handles serial chains; however multiple goals are necessary for most graphics and robotics applications. It is therefore difficult to implement the CCD technique for multiple end effectors. [50] describes such a technique, which deals with tree articulated structures. The proposed multiple-chain CCD method can be applied successively over multiple articulated chains; it divides the articulated structure in smaller serial chains and treats each chain independently.

CCD is a very quick method but it is not free from problems. CCD suffers from unrealistic animation, even if manipulator constraints have been incorporated, and often produces motion with erratic discontinuities. CCD also tends to overemphasise the movements of the joints closer to the end effector of the kinematic chains, producing an unnatural movement, even if constraints have been incorporated.

Inductive Inverse Kinematic Algorithm The Inductive Inverse Kinematics (IIK) algorithm [51] is an extension of the CCD algorithm; it uses a Uniform Posture Map (UPM) to control the posture of a human-like 3D character. The UPM is organized through the quantisation of various postures with an unsupervised learning algorithm, and the learning algorithm prevents the generating of invalid output neurons. The IIK algorithm can be formed by implementing a forward kinematic table containing the forward kinematics values of each output neuron. Thereafter, the forward kinematics table is searched to find the point with the smallest distance from the desired point, and to choose the posture vector associated with that point. If the current end point needs to be made closer to its target position, traditional CCD can be used in the final phase of the algorithm. It is guaranteed that the postures generated by the UPM are realistic postures which observe physical constraints. Hence it is possible to get a natural posture by finding a posture whose forward kinematics point is closest to the desired position.

3.6.2 Triangulation Inverse Kinematics

Another method which does not use an iterative approach is presented in [52]. The Triangulation algorithm uses the cosine rule to calculate each joint angle starting at the root of the kinematic chain moving outward towards the end effector. It is guaranteed to find a solution when used with unconstrained joints and when the target is in range. The Triangulation algorithm incurs a lower computational cost than the CCD algorithm, since it needs only 1 iteration to reach the target. However, the results are not realistic. The joints close to the end-effector are usually in a straight line, with the emphasis on rotation of the joints neighbouring the root. The Triangulation IK method can only be applied to problems with a single end effector; kinematic chains with multiple end effectors cannot be solved and it cannot therefore be used for complex character models. Another drawback of this algorithm is that, when constraints are applied, the end effector often cannot reach the target, even if there is a solution. This happens because each joint position is calculated independently without considering the restrictions of the next joint.

3.6.3 Sequential Inverse Kinematics

Sequential Inverse Kinematics (SIK), which is presented in [13], is a direct extension of [14]. The SIK is an analytic-iterative IK method that reconstructs 3d human full-body movements in real-time. The inputs to this method are end effector positions, such as wrists, ankles, head and pelvis (the least possible input in order to be usable within a low-cost motion capture system in real-time), which are used to find the human pose. The IK problem is then solved sequentially using simple analytic-iterative IK algorithms, in different parts of the body, in a specific order. The SIK, according to [13], outperforms many IK methods regarding the joint average position error, the joint average orientation error and the median processing time of each methodology.

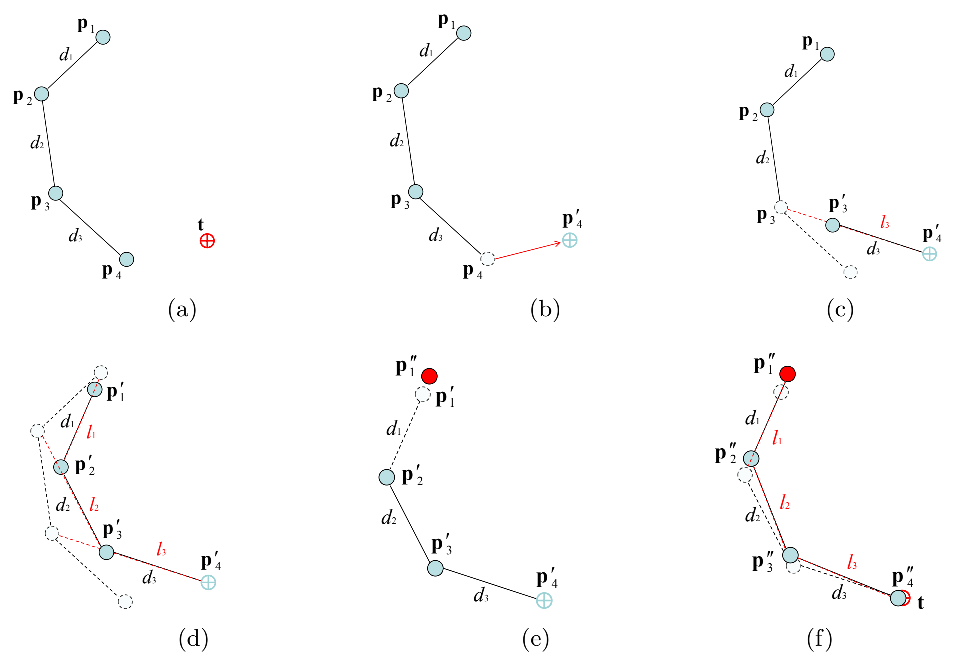


Figure 3.3: *An example of a full iteration of FABRIK for the case of a single target and 4 manipulator joints. (a) The initial position of the manipulator and the target, (b) move the end effector $\mathbf{p}_4$ to the target, (c) find the joint $\mathbf{p}'_3$ which lies on the line l_3, that passes through the points $\mathbf{p}'_4$ and $\mathbf{p}_3$, and has distance d_3 from the joint $\mathbf{p}'_4$, (d) continue the algorithm for the rest of the joints, (e) the second stage of the algorithm: move the root joint $\mathbf{p}'_1$ to its initial position, (f) repeat the same procedure but this time start from the base and move outwards to the end effector. The algorithm is repeated until the position of the end effector reaches the target or gets sufficiently close.*

3.7 FABRIK: A new heuristic IK methodology

In this section, a new heuristic method for solving the IK problem, FABRIK [53], is presented. It uses the previously calculated positions of the joints to find the updates in a forward and backward iterative mode. FABRIK involves minimising the system error by adjusting each joint angle one at a time. The proposed method starts from the last joint of the chain and works forwards, adjusting each joint along the way. Thereafter, it works backward in the same way, in order to complete a full iteration. This method, instead of using angle rotations, treats finding the joint locations as a problem of finding a point on a line; hence, time and computation can be saved.

Assume $\mathbf{p}_1, ..., \mathbf{p}_n$ are the joint positions of a manipulator. Also, assume that $\mathbf{p}_1$ is the root joint and $\mathbf{p}_n$ is the end effector, for the simple case where only a single end effector exists. The target is symbolised as $\mathbf{t}$ and the initial base position by $\mathbf{b}$. A graphical representation of a full iteration of the algorithm with a single target and 4 joints is presented and explained in figure 3.3.

First calculate the distances between each joint $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$, for $i = 1, ..., n-1$. Then, check whether the target is reachable or not; find the distance between the root and the target, $dist$, and if this distance is smaller than the total sum of all the inter-joint distances, $dist < \sum_1^{n-1} d_i$, the target is within reach, otherwise, it is unreachable. If the target is within reach, a full iteration is constituted by two stages. In the first stage, the algorithm estimates each joint position starting from the end-effector, $\mathbf{p}_n$, moving inwards to the manipulator base, $\mathbf{p}_1$. So, let the new position of the end-effector be the target position, $\mathbf{p}'_n = \mathbf{t}$. Find the line, l_{n-1}, which passes through the joint positions $\mathbf{p}_{n-1}$ and $\mathbf{p}'_n$. The new position of the $(n-1)^{th}$ joint, $\mathbf{p}'_{n-1}$, lies on that line with distance d_{n-1} from $\mathbf{p}'_n$. Similarly, the new position of the $(n-2)^{th}$ joint, $\mathbf{p}'_{n-2}$, can be calculated using the line l_{n-2}, which passes through the $\mathbf{p}_{n-2}$ and $\mathbf{p}'_{n-1}$, and has distance d_{n-2} from $\mathbf{p}'_{n-1}$. The algorithm continues until all new joint positions are calculated, including the root, $\mathbf{p}'_1$.

Having in mind that the new position of the manipulator base, $\mathbf{p}'_1$, should not be different from its initial position, a second stage of the algorithm is needed. A full iteration is completed when the same procedure is repeated but this time starting from the root joint and moving outwards to the end effector. Thus, let the new position for the 1^{st} joint, $\mathbf{p}''_1$, be its initial position $\mathbf{b}$. Then, using the line l_1 that passes through the points $\mathbf{p}''_1$ and $\mathbf{p}'_2$, we define the new position of the joint $\mathbf{p}''_2$ as the point on that line with distance d_1 from $\mathbf{p}''_1$. This procedure is repeated for all the remaining joints, including the end effector. In cases where the root joint has to be translated to a desired position, FABRIK works as described with the difference that in the backward phase of the algorithm, the new position of the root joint, $\mathbf{p}''_1$, will be the desired and not the initial position.

After one complete iteration, it is always the case (observed empirically) that the end effector is closer to the target. The procedure is then repeated, for as many iterations as needed, until the end effector is identical or close enough (to be defined) to the desired target. FABRIK always converges to any given chains/goal positions, when the target is within reach. If there are constraints which do not allow the chain to bend enough in order to reach the target or if the target is not within the reachable area, there is a termination condition which compares the previous and the current position of the end effector, and if this distance is less than an indicted tolerance, FABRIK terminates its operation. Also, in the extreme case where the number of iterations has exceeded an indicated value and the target has not been reached, the algorithm is terminated (however, we have never encountered such a situation). Several optimisations can be achieved using Conformal Geometric Algebra (GA) [43, 44] to produce faster results and to converge to the final answer in fewer iterations; Conformal GA has the advantage that basic entities, such as spheres, lines, planes and circles, are simply represented by algebraic objects. Therefore, a direct estimate of a missing joint, when it is between 2 true positions, can be achieved by intersecting 2 spheres with centres the true joint positions and radii the distances between the estimated and the true joints respectively. Another simple optimisation is the direct construction of a line pointing towards the target, when the latter is unreachable. FABRIK is illustrated in pseudo-code in Algorithm 1.

The proposed method has all the advantages of existing iterative heuristic algorithms. The computational cost for each joint per iteration is low, meaning the solution is arrived at very quickly. It is also very easy to implement, since it is simply a problem involving points, distances and lines and always returns a solution when the target is in range. It does not require complex calculations (e.g Jacobian or Hessian matrix) or matrix manipulations (inversion or singular

Algorithm 1: A full iteration of the FABRIK algorithm.

Input: The joint positions $\mathbf{p}_i$ for $i = 1, ..., n.$, the target position $\mathbf{t}$ and the distances between each joint $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$ for $i = 1, ..., n-1$.

Output: The new joint positions $\mathbf{p}_i$ for $i = 1, ..., n$.

```
1.1  % The distance between root and target
1.2  $dist = |\mathbf{p}_1 - \mathbf{t}|$
1.3  % Check whether the target is within reach
1.4  if $dist > d_1 + d_2 + ... + d_{n-1}$ then
1.5      % The target is unreachable
1.6      for $i = 1, ..., n-1$ do
1.7          % Find the distance $r_i$ between the target $\mathbf{t}$ and the joint position $\mathbf{p}_i$
1.8          $r_i = |\mathbf{t} - \mathbf{p}_i|$
1.9          $\lambda_i = d_i / r_i$
1.10         % Find the new joint positions $\mathbf{p}_i$.
1.11         $\mathbf{p}_{i+1} = (1 - \lambda_i)\,\mathbf{p}_i + \lambda_i \mathbf{t}$
1.12     end
1.13 else
1.14     % The target is reachable; thus, set as $\mathbf{b}$ the initial position of the joint $\mathbf{p}_1$
1.15     $\mathbf{b} = \mathbf{p}_1$
1.16     % Check whether the distance between the end effector $\mathbf{p}_n$ and the target $\mathbf{t}$ is greater than a
         tolerance.
1.17     $dif_A = |\mathbf{p}_n - \mathbf{t}|$
1.18     while $dif_A > tol$ do
1.19         % STAGE 1: FORWARD REACHING
1.20         % Set the end effector $\mathbf{p}_n$ as target $\mathbf{t}$
1.21         $\mathbf{p}_n = \mathbf{t}$
1.22         for $i = n-1, ..., 1$ do
1.23             % Find the distance $r_i$ between the new joint position $\mathbf{p}_{i+1}$ and the joint $\mathbf{p}_i$
1.24             $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
1.25             $\lambda_i = d_i / r_i$
1.26             % Find the new joint positions $\mathbf{p}_i$.
1.27             $\mathbf{p}_i = (1 - \lambda_i)\,\mathbf{p}_{i+1} + \lambda_i \mathbf{p}_i$
1.28         end
1.29         % STAGE 2: BACKWARD REACHING
1.30         % Set the root $\mathbf{p}_1$ its initial position.
1.31         $\mathbf{p}_1 = \mathbf{b}$
1.32         for $i = 1, ..., n-1$ do
1.33             % Find the distance $r_i$ between the new joint position $\mathbf{p}_i$ and the joint $\mathbf{p}_{i+1}$
1.34             $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
1.35             $\lambda_i = d_i / r_i$
1.36             % Find the new joint positions $\mathbf{p}_i$.
1.37             $\mathbf{p}_{i+1} = (1 - \lambda_i)\,\mathbf{p}_i + \lambda_i \mathbf{p}_{i+1}$
1.38         end
1.39         $dif_A = |\mathbf{p}_n - \mathbf{t}|$
1.40     end
1.41 end
```

value decomposition), it does not suffer from singularity problems and returns smooth motion without erratic discontinuities. Additionally, it emphasises movements in the joints closer to the chain base ensuring a closer simulation of natural movements than that observed with the CCD method.

3.7.1 FABRIK with Multiple End Effectors

IK solvers are commonly used for solving the IK problem in many areas including computer graphics, gaming and protein science. In reality, most of the multibody models, such as hands, human or legged bodies etc, are comprised of several kinematic chains, and each chain generally has more than 1 end effector. Therefore, it is essential for an IK solver to be able to solve problems with multiple end effectors and targets. The proposed algorithm can be easily extended to process models with multiple end effectors. However, prior knowledge of the model, such as the sub-base[2] joints, and the number and structure of chains is needed.

The algorithm is divided into two stages, as in the single end effector case. In the first stage, the normal algorithm is applied but this time starting from each end effector and moving inwards until the parent sub-base. This will produce as many different positions of the sub-base as the number of end effectors connected with that specific sub-base. The new position of the sub-base will then be the centroid of all these positions. Thereafter, the normal algorithm should be applied inwards starting from the sub-base until the manipulator root. If there are more sub-bases between the previous sub-bases and the root, the same technique should be used. In the second stage, the normal algorithm is applied starting now from the root and moving outwards to the sub-base. Then, the algorithm should be applied separately for each chain until the end effector; if more sub-bases exist, the same process is applied. The method is repeated until all end effectors reach the target or there is no significant change between their previous and their new positions. An example of a model figure having multiple end effectors and multiple sub-bases is presented in figure 3.4.

More sophisticated (and complex) models can be also used, extending the proposed algorithm, taking into account the figure's shape, constraints and properties, producing faster and more realistic results. Such models reduce the number of iterations needed to reach the targets and return more feasible postures.

3.7.2 Applying Constraints to FABRIK

Most legged body models are comprised of joints having biomechanical constraints, which provide natural restrictions on their motion. Such constraints are essential in physical simulations, inverse kinematic techniques and tracking in motion capture systems in order to reduce visually unrealistic movements.

Several biomechanically and anatomically correct models have been presented that formalise the range of motion of an articulated figure. These models are mainly characterised by the number of parameters which describe the motion space and are hierarchically structured. Because

[2]A sub-base joint is a joint which connects 2 or more chains. A pre-analysis of the body can determine exactly where the sub-bases are located.

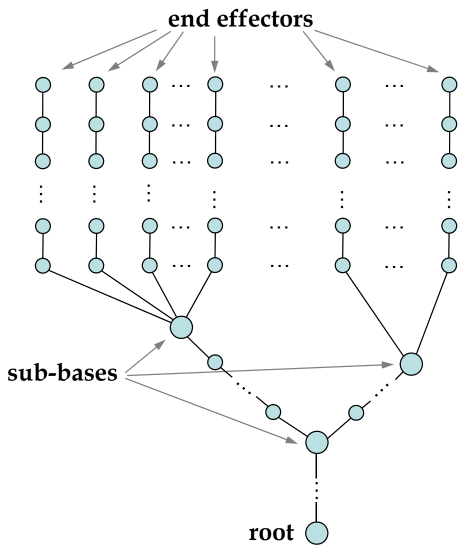


Figure 3.4: *An example of a model figure with multiple end effectors and multiple sub-bases.*

of their complex nature, most of the proposed joint models are simplified or approximated by more than one joint. The most well-known models are: *the shoulder model*, a complex model composed of 3 different joints [30, 31, 54, 55]; *the spine model*, a complex arrangement of 24 vertebrae (usually, for simplicity, the spine is modelled as a simple chain of joints [28, 32, 33, 34]); *the hand model*, this is the most versatile part of the body comprising a large number of joints [35, 36, 37]; *the strength model*, which takes account of the forces applied from the skeletal muscles to the bones [28].

A joint is defined by its position and orientation and, in the most general case, has 3 DoF. A bone rotation can be described by factoring it into two rotations: one "simple rotation", named here as *rotational*, that moves the bone to its final direction vector, and the *orientational*, which represents the twist around this final vector. Thus, the range of movement of a bone can be controlled by dividing the joint restriction procedure into two interconnected phases, a rotational and an orientational phase, contributing equally to the joint restrictions. The essential feature of a joint is that it permits a relative motion between the two limbs it connects. Most of the existing structure models, such as those described above, use techniques which restrict the bone to lie within the rotational and orientational limits of the joint. Blow [56] proposes a loop hung in space, limiting the range of motion of the bone to "reach windows" described by star polygons. Wilhelms and Van Gelder [57], instead of using reach windows, present a 3D "reach cone" methodology using planes, treating the joint limits in the same way as [56]. Also, [33, 58] parameterise realistic joint boundaries of the ball-and-socket joint by decomposing the arbitrary orientation into two components and controlling the rotational joint limits so they do not exceed their bounds. Once a proper parametrisation is defined for each joint of the articulated body, an animation engine is utilised.

In this section, a reliable methodology for incorporating manipulator constraints is described using FABRIK. Since FABRIK is iterative, the joint restrictions can be enforced at each step just by taking the resultant orientation and forcing it to stay within the valid range. FABRIK's

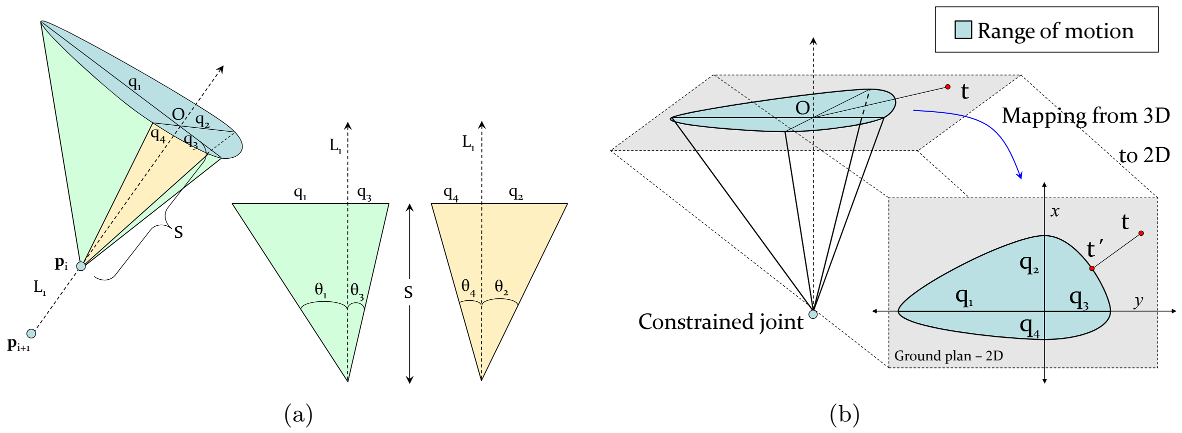


Figure 3.5: *A graphical representation of the implemented constraints and the irregular cone describing the rotational motion bounds. (a) The ball-and-socket joint, $\mathbf{p}_i$, with its associated irregular cone which defines the allowed range of motion. (b) Shows the composite ellipsoidal shape created by the distances q_j mapped from 3D to 2D.*

ability to converge on an answer, if the target is within reach, is not affected by any imposed joint limits.

The main idea behind this methodology is the re-positioning and re-orientation of the target to be within the allowed range bounds; ensuring that these restrictions are always satisfied means a more feasible posture can be achieved. This can be accomplished by checking if the target is within the valid bounds, at each step of FABRIK, and if it is not, to guarantee that it will be moved accordingly. Assume we have a ball-and-socket joint with rotational and orientational limits, restricting the allowed space to a realistic subset. Its orientation is described by the rotor $\mathbf{r}$ and its rotation by the angles $\theta_1, ..., \theta_4$. Figure 3.5 gives a graphical representation of the implemented constraints and the irregular cone describing the rotational motion bounds.

The orientation of the joint can be assigned as follows: Assume we are in the first stage of the algorithm, i.e. we have just calculated the new position of joint $\mathbf{p}'_i$, and we want to find the new position of the $(i-1)^{th}$ joint, $\mathbf{p}'_{i-1}$. Find the rotor expressing the rotation between the orientation frames at joints $\mathbf{p}'_i$ and $\mathbf{p}_{i-1}$ and if this rotor represents a rotation greater than a limit, reorient the joint $\mathbf{p}_{i-1}$ in such a way that the rotation will be within the limits. Repeat the procedure for all the joints on both stages of the algorithm. The methodology is also described in pseudo-code in Algorithm 2.

Once the joint orientation is established, the rotational (2 DoF) limits, described by angles $\theta_1, ..., \theta_4$, can be applied as follows. Firstly, we find the projection O of the target $\mathbf{t}$ on line L_1, where L_1 is the line passing through the joint under consideration, $\mathbf{p}_i$, and the previous joint of the chain, $\mathbf{p}_{i+1}$. Then determine the distance S from the point O to the joint position $\mathbf{p}_i$ and calculate the distances $q_j = S\tan(\theta_j)$, for $j = 1, ..., 4$, as shown in figure 3.5. We then apply a rotation and translation which takes O to the origin and the axes defining the constraints to the x and y axes, as in figure 3.5(b). Working in this 2D plane, we locate the target in a particular quadrant and find the ellipse defined on that quadrant using the associated distances q_j; for

Algorithm 2: The orientational constraints.

```
    Input: The rotor $\mathbf{R}$ expressing the rotation between the orientation frames at joints
           $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$.
    Output: The new re-oriented joint $\mathbf{p}'_{i-1}$.
2.1 Check whether the rotor $\mathbf{R}$ is within the motion range bounds
2.2 if within the bounds then
2.3 |   do nothing and exit
2.4 else
2.5 |   reorient the joint $\mathbf{p}_{i-1}$ in such a way that the rotor will be within the limits
2.6 end
```

example, in figure 3.5(b) we are working with the ellipse which is defined by the angles θ_2 and θ_3 (or the distances q_2 and q_3). Finally, find the nearest point on that ellipse from the target, if the latter is not in the allowed motion range. The nearest point on an ellipse from a point can be found by simultaneously solving the ellipse equation and the equation of the tangent line at the orthogonal contacting point on the ellipse using the Newton-Raphson method, as described in [59]. Obviously, it is not necessary to calculate all the ellipses which define the composite ellipsoidal shape of figure 3.5(b), but only the ellipse related to the quadrant in which the target is located. The last step is to undo the initial transformation which mapped O to the origin. This procedure is illustrated in pseudo-code in Algorithm 3.

Algorithm 3: The rotational constraints.

```
     Input: The target position $\mathbf{t}$ and the angles defining the rotation constraints $\theta_j$ for
            $j = 1, ..., 4$.
     Output: The new target position $\mathbf{t}'$.
3.1  Find the line equation $L_1$
3.2  Find the projection $O$ of the target $\mathbf{t}$ on line $L_1$
3.3  Find the distance between the point $O$ and the joint position
3.4  Map the target (rotate and translate) in such a way that $O$ is now located at the axis
     origin and oriented according to the $x$ and $y$-axis $\Rightarrow$ Now it is a 2D simplified problem
3.5  Find in which quadrant the target belongs
3.6  Find the ellipse which is associated with that quadrant using the distances
     $q_j = S \tan \theta_j$, where $j = 1, .., 4$
3.7  Check whether the target is within the ellipse or not
3.8  if within the ellipse then
3.9  |   use the true target position $\mathbf{t}$
3.10 else
3.11 |   go to the next step
3.12 end
3.13 Find the nearest point on ellipse from the target
3.14 Map (rotate and translate) that point on ellipse via reverse of 3.4 and use that point
     as the new target position
```

This is a versatile and easily visualisable method of restricting where the bone can go. Incorporating this methodology within an IK solver, such as FABRIK, will give us the opportunity to reconstruct or track animated figures with high accuracy. IK algorithms are generally more

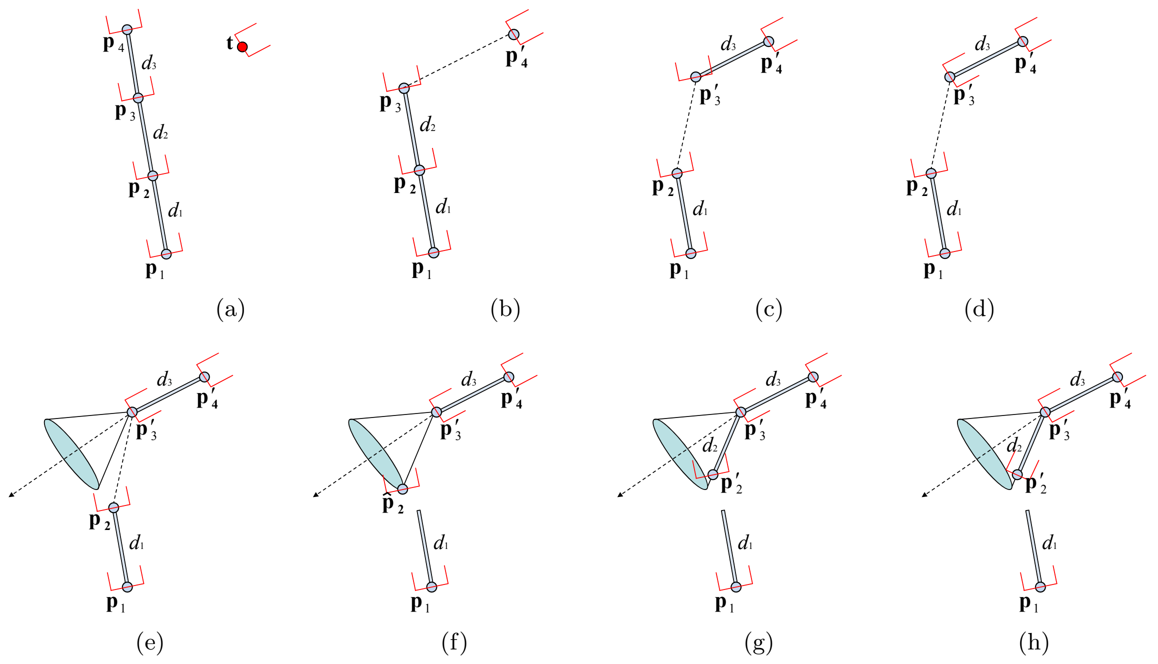


Figure 3.6: *Incorporating rotational and orientational constraints within FABRIK. (a) The initial configuration of the manipulator and the target, (b) relocate and reorient joint* $\mathbf{p}_4$ *to target* $\mathbf{t}$*, (c) move joint* $\mathbf{p}_3$ *to* $\mathbf{p}'_3$*, which lies on the line that passes through the points* $\mathbf{p}'_4$ *and* $\mathbf{p}_3$ *and has distance* d_3 *from* $\mathbf{p}'_4$*, (d) reorient joint* $\mathbf{p}'_3$ *in such a way that the rotor expressing the rotation between the orientation frames at joints* $\mathbf{p}'_3$ *and* $\mathbf{p}'_4$ *is within the motion range bounds, (e) the rotational constraints: the allowed regions shown as a shaded composite ellipsoidal shape, (f) the joint position* $\mathbf{p}_2$ *is relocated to a new position,* $\hat{\mathbf{p}}_2$*, which is the nearest point on that composite ellipsoidal shape from* $\mathbf{p}_2$*, ensuring that the new joint position* $\mathbf{p}'_2$ *will be within the allowed rotational range. (g) move* $\hat{\mathbf{p}}_2$ *to* $\mathbf{p}'_2$*, to conserve bone length, (h) reorient the joint* $\mathbf{p}'_2$ *in order to satisfy the orientation limits. This procedure is repeated for all the remaining joints in a forward and backward fashion.*

effective if the constraints are applied at each step (not at the end of the algorithm), ensuring that the rotational and orientational restrictions are satisfied at each iteration. Thus, the proposed joint constraints can be applied within FABRIK by ensuring that the target, at each step, is moved within the allowed orientational and rotational bounds. Hence, assume that we are in the first stage of the algorithm, and have just calculated the new positions of the joints, $\mathbf{p}'_{i+1}$ and $\mathbf{p}'_i$, and we want to find the new position of the $(i-1)^{th}$ joint, $\mathbf{p}'_{i-1}$. Check if the joint $\mathbf{p}_{i-1}$ satisfies the orientational limits and if so, check whether it is within the composite ellipsoidal shape that describes the allowed range bounds, as illustrated above. If it is not, then $\mathbf{p}_{i-1}$ should be re-oriented and/or re-positioned within the allowed bounds ($\hat{\mathbf{p}}_{i-1}$). Thereafter, $\mathbf{p}'_{i-1}$ can be defined as the point on the line l_{i-1}, which passes through the joint positions $\hat{\mathbf{p}}_{i-1}$ and $\mathbf{p}'_i$ and has d_{i-1} distance from $\mathbf{p}'_i$.

The same technique for constraining joints is applied in the second stage of the algorithm and for each iteration until the target is reached or there is no significant change in the end effectors' positions. The algorithm copes with joints and limbs having 3 DoF, and it can handle cases of joint and limb twist. It is important to recall here that the inter-joint lengths are not changing over time since these distances are implicity kept constant by FABRIK. A demonstration of the process is given in figure 3.6.

One big advantage of the proposed methodology is that no bone requires rotation to lie in any cone or polygon window, such as those described in [56, 57]; it is only necessary to check whether the target is within the composite ellipsoidal shape defined by the restrictions on the motion. It loses none of the advantages of the FABRIK algorithm, incorporating joint limits via only points, lines and basic 2D entities; no rotational matrices need to be calculated, resulting in large savings in computational time. It also produces visually smooth and natural movements without oscillations and discontinuities, and requires low processing time per iteration.

If more information about the allowed range of motion is available, the proposed methodology can be extended to include increased sophistication, supporting more complex joint types. Thus, instead of having an ellipsoidal entity to describe the sub-area in which the target can be placed, a polygonal area can be implemented. If the target is out of range, we would look for the nearest point on the polygon.

The constraining methodology can also be easily modified to support other IK solvers. There are, however, some limitations on what joint types this prototype version can support, since it is assumed that the inter-joint distance remains constant over time. Prismatic, slicing or shifting joints (joint types more usually discussed in robotics) are not supported. Self-collisions can be handled using existing techniques, such as [60]; but more work is needed to ascertain if the FABRIK framework gives any advantages when dealing with self occlusions.

4 Results

THIS chapter presents, compares and discusses the results of the most popular IK methods, as presented in Chapter 3, under several conditions. Each methodology is implemented for several cases (single or multiple targets) and tested for its realistic movement, reliability and computational cost with and without joint restrictions. It also presents several implementations of FABRIK within a hand and a humanoid model and analyses the results in respect of visual realism, reconstruction quality, computational cost and conversion criteria.

4.1 The Experimental Environment

A target database has been created for the validation and testing of the IK methods. The database consists of reachable and unreachable targets, targets with different distances from the end effectors and targets that move smoothly in space with end effectors tracking their position. The tests also consist of reconstructing sequences with different classes of motion in order to process different swivel angles and axial orientations of the root joint. The examples are demonstrated in 6 different kinematic models; a chain with 10 unconstrained joints allowing 3 DoF on each joint; a chain with 10 constrained joints allowing limited angle rotations with 3 DoF; a model containing a 'Y-shape' having 10 unconstrained joints and 2 end effectors; a fully unconstrained and un-modelled hand with 26 joints, 3 DoF on each joint and 5 end effectors; and a 13 joint humanoid model, in a constrained and unconstrained version, with 3 DoF on

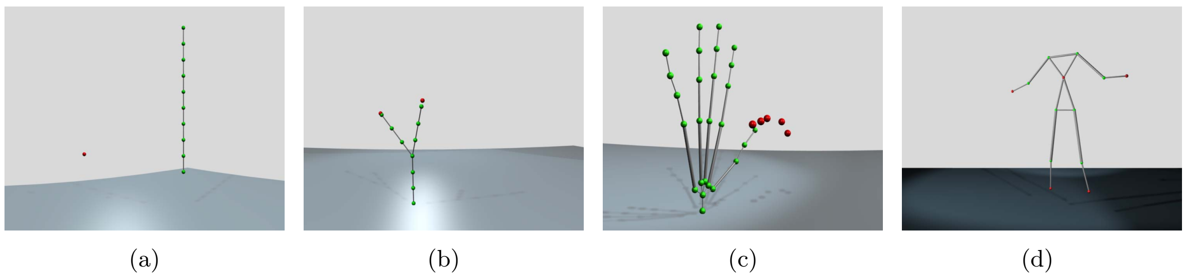


Figure 4.1: *The structure of the models used in our experimental examples. (a) A kinematic chain consisting of 10 joints and 1 end effector. There are 2 kinematic chain models, an unconstrained and a constrained version, (b) a kinematic model with 10 unconstrained joints and 2 end effectors, (c) a hand model with 26 unconstrained joints and 5 end effectors, (d) a 13 joint humanoid model, in a constrained and unconstrained version, with 4 end effectors. The target joint positions (end effectors) are shown in red and the joint positions that the IK solvers have to estimate are shown in green.*

each joint and 4 end effectors. Figure 4.1 shows the different kinematic models used within this work.

IK techniques will mostly work with specified positions and orientations of specific joints, usually the end effectors, since they are more easily specified by the animator and tracked by the motion capture system; thereby, they automatically configure the remaining joints according to different criteria that depend on the model variant and joint type restrictions.

All experiments were run using MATLAB [61] on a computer with a Pentium 2 Duo 2.2 GHz processor. The operating system used is Microsoft Windows Vista service pack 1.0. The results have been animated in video sequences using Blender [27].

4.2 Results

In this section, some of the most popular IK methods have been tested against FABRIK, such as CCD, Jacobian Transpose, Jacobian DLS and Jacobian pseudo-inverse DLS (SVD-DLS). In some of our experiments, we implemented examples with large distances between target and end-effectors; hence, some methods tend to require more iterations to reach the target and thus the convergence differences are more obvious. The DLS parameter values used in our experiments are the parameter values suggested by [8]; the damping constant was set to $\lambda = 1.1$. Several tests and comparisons have been implemented between the proposed algorithms in respect of their computational cost, processing time, convergence, the number of iterations needed to reach the target and the reconstruction quality.

4.2.1 A single end effector

In this section, the methods have been tested on problems with a single end effector and fixed target positions. These experiments did not include any joint constraints, but all methods could be enhanced to enforce rotational and orientational limits. Examples with the resulting postures for each methodology are presented in figure 4.6 and 4.7.

FABRIK produces results significantly faster than all IK methods tested. It is approximately

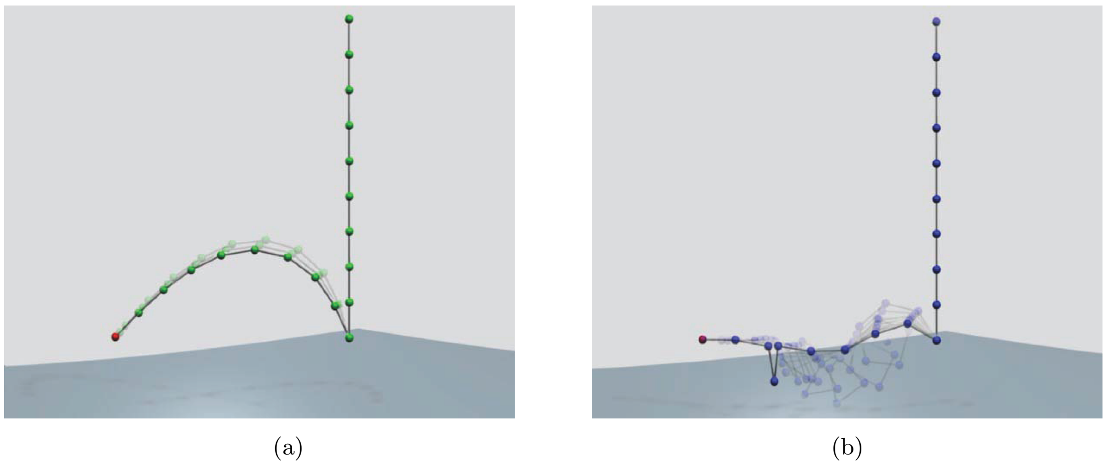


Figure 4.2: *The stages illustrated in order the end effector to reach the target. (a) The FABRIK solution and (b) the CCD solution.*

10 times faster than the CCD method and a thousand times faster than the Jacobian-based methods, for these examples with large end effector movements; FABRIK has the lowest computational cost and, at the same time, produces visually the smoothest and most natural movements. It needs the fewest iterations to reach the target, it converges faster to the desired position and, when the target is unreachable, it keeps the end effector pointing to the target. Figure 4.2 shows an example of an IK solution using FABRIK and CCD; the figure presents all the stages before the kinematic chain reaches the target for both cases. It is clear that FABRIK needs fewer iterations and has a more natural movement to the target. On average, FABRIK needs 15.4 iterations and just 13.2ms to attain a reachable target and 67 iterations and 62ms for an unreachable target. The time and iterations needed to converge to a final answer, when the target is unreachable, can be reduced dramatically when algorithm optimisations are applied (see Alg.1); using optimisations, FABRIK needs just 1 iteration and 0.2ms to return the final chain pose. Obviously, as the target gets closer to the end effector, fewer iterations will be needed to reach the target. From the methods used within this report FABRIK produces the most realistic and smoothest postures (see figures 4.6 and 4.7 for verification).

CCD can also be applied in real-time. It is much faster than any Jacobian-based method; it needs, on average, 26 iterations and 123ms to reach the target when it is within reach. On the other hand, when the target is not reachable, it needs almost 400 iterations and 4sec to converge to its final solution (using the default algorithm without optimisations). However, CCD can often generate unrealistic postures since it can roll and unroll itself before reaching the target (figure 4.2 and 4.3). CCD also tends to overemphasise the movements of the joints closer to the end effector of the kinematic chain. Another drawback of CCD is that it can only handle problems with serial chains. It is not straightforwardly modified in order to solve problems with multiple end effectors, thus limiting its use.

The Jacobian methods return reasonable results; the reconstructed chain poses are visually more natural than CCD. Nevertheless, the biggest advantage of the Jacobian methods over all other methods is that, by default, they can treat problems with multiple end effectors very

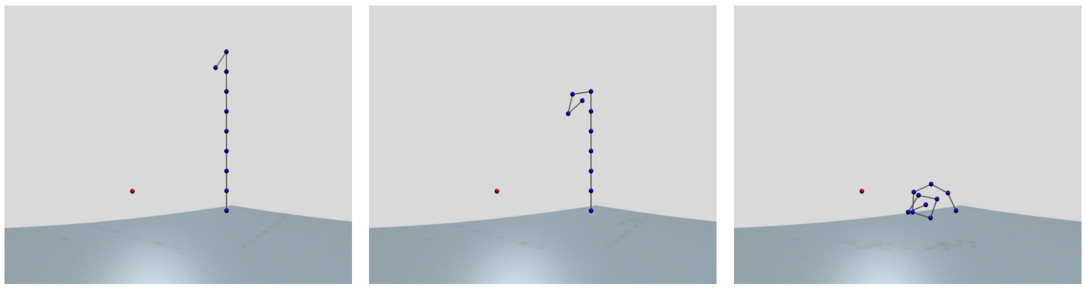


Figure 4.3: *Unnatural joint angles exhibited by CCD; the kinematic chain rolls itself before reaching the target, producing unrealistic poses.*

easily. Constraints can be applied within the Jacobian algorithms, but the way in which these restrictions are incorporated is again not straightforward. Some Jacobian methods also suffer from singularity problems, since matrix inverses need to be calculated. The Transpose and DLS methods do not suffer in this way since they do not use the matrix inverse. The Jacobian methods also incur high computational cost making this family of methods non-ideal for real-time applications. For the examples considered here, the Jacobian Transpose method needs on average more than 1300 iterations and 13sec to reach the target when it is within reach, the DLS needs more than 990 iterations and 10sec and SVD-DLS more than 800 and 9sec. The Jacobian methods generally converge very slowly to their final solutions since they use a linear approximation with a small step. This is more obvious in figure 4.4, where the number of iterations needed to reduce the distance between target and end effector as this changes over time is presented for each methodology. In this example, the original chain is 9000mm long, the distance between target and end effector is 6000mm, and the termination tolerance is 1×10^{-3}mm.

The Triangulation algorithm also incurs lower computational cost than the CCD algorithm and it is substantially faster than the Jacobian methods. However, Triangulation returns the poorest results from the methods used within this report. The kinematic chain does not have a realistic shape; the joints close to the end-effector are usually in a straight line, with the emphasis on rotation of the joints neighbouring the root. Another important drawback of the Triangulation algorithm is that it cannot be adapted for multiple end effectors, it is thus useless for complex character models. The Triangulation algorithm also suffers from an inability to reach a feasible solution when constraints are applied; the end effector often cannot reach the target, even if there is a solution, since each joint position is calculated independently without considering the restrictions of the next joint.

Table 4.1 presents the average runtimes of each of the methods, as well as the number of iterations needed to reach the target when the latter is reachable. Runtimes are in seconds and were measured with custom MATLAB code on a Pentium 2 Duo 2.2 GHz. No optimisations were used for any method reported in the table. It also indicates the time needed per iteration for each method and how many iterations per second each methodology can support. An iteration of FABRIK has the lowest computational cost since, instead of using angle rotations, it treats finding the joint locations as a problem of finding a point on a line. Thus, it can process up to 1164 iterations in one second, requiring 0.85ms per iteration. The time required for a full iteration using CCD is 8.8ms, while the Jacobian Transpose, DLS and SVD-DLS methods need 9.9ms, 10.5ms and 11.5ms per iteration respectively. The same results are reported in table 4.2, but this time for the case where the target is unreachable.

Table 4.1: *Average results when the target is reachable.*

	Number of Iterations	Matlab exe. time (sec)	Frames per second
FABRIK	15.461	0.01328	75.301
CCD	26.308	0.12359	8.091
Jacobian Transpose	1311.190	12.98947	0.077
Jacobian DLS	998.648	10.48501	0.095
Jacobian SVD-DLS	808.797	9.29652	0.107
Triangulation	1.000	0.05747	17.400

Table 4.2: *Average results when the target is unreachable.*

	Number of Iterations	Matlab exe. time (sec)	Frames per second
FABRIK	67.564	0.06207	16.109
CCD	390.135	3.92869	0.254
Jacobian Transpose	6549.000	33.90473	0.029
Jacobian DLS	2881.667	14.87918	0.067
Jacobian SVD-DLS	8808.667	45.97591	0.021
Triangulation	1.000	0.06993	14.299

Figures 4.6 and 4.7 compare the performance of each algorithm for solving inverse kinematics problems; they show the initial configuration and the goal solution obtained with each methodology. The manipulator is fully unconstrained and has no limits on the range of allowed movement for each joint. In each case a position goal is specified for the end effector and the inverse kinematic problem is solved to varying degrees of accuracy. Figure 4.5 plots the convergence of each method, meaning the time taken to achieve the solution with the requested degree of accuracy requested. It is clearly observed that FABRIK converges to the target faster than any other implemented methodology. Table 4.3 indicates on average the time needed per iteration for each method and how many iterations per second each methodology can support. An iteration of FABRIK has the lowest computational cost since, instead of using angle rotations, it treats finding the joint locations as a problem of finding a point on a line. Thus, it can process up to 1150 iterations within a second. The time required for a full iteration using CCD and the Jacobian methods is similar, however CCD can reach the target faster since it needs many fewer iterations. The Triangulation algorithm requires 1 iteration of 48ms to reach the target. Figure 4.5 verifies that FABRIK always converges to the target, if the latter is reachable.

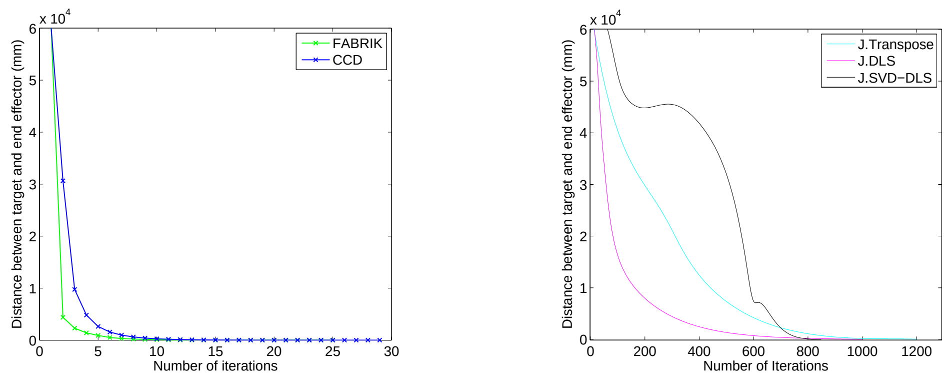


Figure 4.4: *The iterations needed to reach the target against distance between target and end effector.*

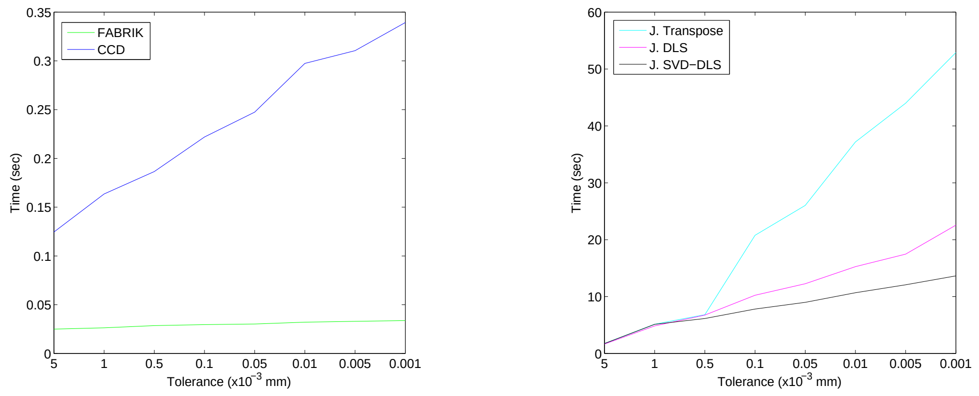


Figure 4.5: *An example presenting the time needed for each methodology to achieve the solution with the degree of accuracy requested.*

Figures 4.6 and 4.7 present examples of implementation over different targets. It is clear that FABRIK produces the most realistic chain postures of any method used within this report.

The FABRIK, CCD, DLS and SVD-DLS methods have also been tested when the target is moving in a sinusoidal trajectory and the end-effector is tracking its position when they are within reach, and keeping the end effector pointing at the targets when they are unreachable. The accuracy of the tracking was measured over a period of a thousand simulation steps. FABRIK tracks the target in real-time producing a smooth and visually natural motion without erratic discontinuities. CCD produces reasonable results within the real-time constraints; however there are instances where the motion produced is not visually realistic. It is important to mention that CCD's performance improves when the target is within a small distance from the end effector's position or the frame rate is high. This happens because the kinematic chain does not roll and unroll itself. On the other hand, the Jacobian-based methods can produce oscillating motion with discontinuities. Their biggest drawback however is the time needed to track the target; only under some circumstances, eg using fast C++ matrix libraries, can these kinds of methods reach the target of real-time application. Although Triangulation is a real-time methodology, it produces the most unrealistic poses for kinematic animations. Figure 4.8 presents the performance of each method on selected frames over time.

Table 4.3: *Average computational cost of the IK methods.*

	Time per Iteration (in msec)	Iterations per second
FABRIK	0.8	1164
CCD	8.8	213
Jacobian Transpose	9.9	101
Jacobian DLS	10.5	95
Jacobian SVD-DLS	11.5	87
Triangulation	48.01	21

4.3 Making Kine more flexible

In this section we implemented the FABRIK algorithm within the Kine [1] application; Kine is a 2D real-time gaming application that initially has a kinematic chain with six joints. Kine allows you to interact with the IK solver; you click on the screen and the snake (the kinematic chain is drawn as a snake) solves the IK problem. There is also an option where you click and drag on the screen and the snake attempts to reach and track your mouse. Kine also offers the option to toggle the damping sector, applying and/or adjusting DoF restrictions. You are also able to add more links, optimise orientation, and modify the application to a 3D environment.

Figure 4.9 shows examples of FABRIK and CCD methods implemented within the Kine environment. Despite FABRIK being approximately 10 times faster than CCD, both methods can solve the IK problem in real-time. Nevertheless, the most important observation is that Kine verifies that FABRIK out-performs CCD in producing more realistic poses. The environment presented in this section has been adopted from the work of Jeff Lander [1].

4.3.1 Multiple end effectors

Most real models, such as the hand, legged bodies etc, consist of multiple chains, each chain having at least one end effector. Hence, it is essential to test our methodology in cases where more than one end effector exists. To test FABRIK under these conditions, we implemented the 'Y-shaped' multibody pictured in figure 4.13 and a hand multibody presented in figure 4.14. The 'Y-shape' multibody has 10 joints with 2 end effectors. The target positions (the red balls in the figures) moved in sinusoidally varying curves in and out of reach of the multibody. The target positions moved in small increments and in each time step the joint positions were updated. The simulations were visually inspected for oscillations and tracking abilities. The end-effectors can successfully track the target positions when they are within reach, and remain pointing at the targets when these are out of reach. Figure 4.13 presents a simple example of how FABRIK performs with multiple end effectors; although it is hard to show in images, FABRIK can easily track both targets with a smooth motion and without oscillations, shaking or erratic discontinuities.

Figure 4.14 shows another example of implementing FABRIK into a multiple end effector model. This is a fully unconstrained hand example with 5 end effectors and 26 joints in total,

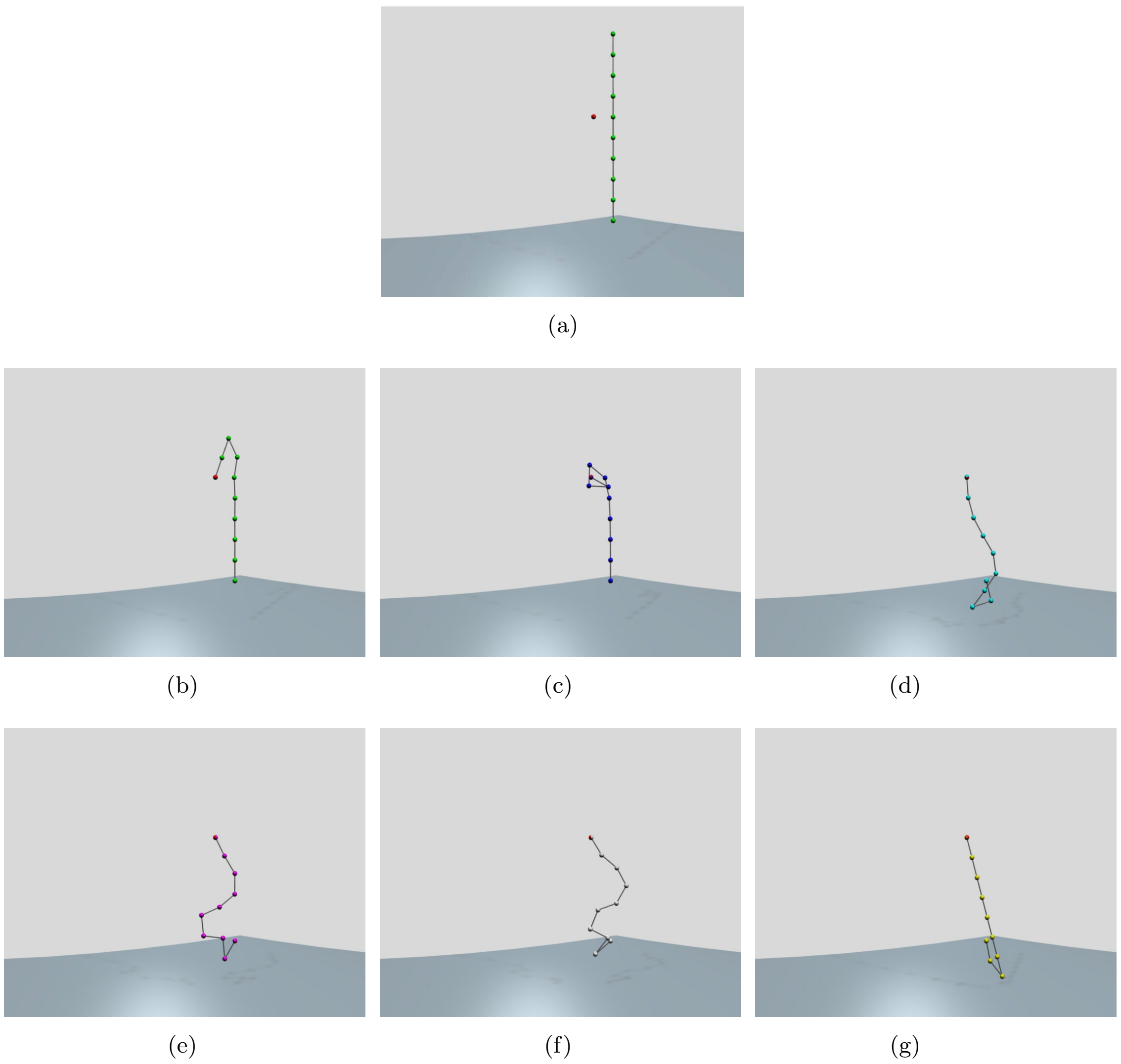


Figure 4.6: *Experimental solutions using some of the most popular IK methods. The kinematic chains consisted of 10 unconstrained joints, allowing 3 degrees of freedom on each joint. (a) Initial position, (b) FABRIK, (c) CCD, (d) J. Transpose, (e) J. DLS, (f) J. SVD-DLS, (g) Triangulation.*

allowing 3 DoF on each joint. Incorporating a highly constrained model, such as [37], and restricting the motion of each joint to a feasible set, the hand will have even more natural movement.

Figure 4.10 shows an example of an animated tracking of a humanoid with 13 joints and 4 end effectors. In this demonstration, the frame rate was low (2 frames per second); FABRIK can easily track the animated humanoid in real-time, producing very reasonable results. Figures 4.11 and 4.12 show the reconstruction quality of different methodologies on the same humanoid model. The differences between the methodologies are more obvious on shoulders, elbows and hips. FABRIK produces visually the most realistic postures, having the smaller reconstruction error compared to the original sequences. These animations have been obtained from an optical markered motion capture system and have not been filtered; thus, the algorithm is shown to be robust in noisy environments. Selected internal joints have been artificially deleted in order

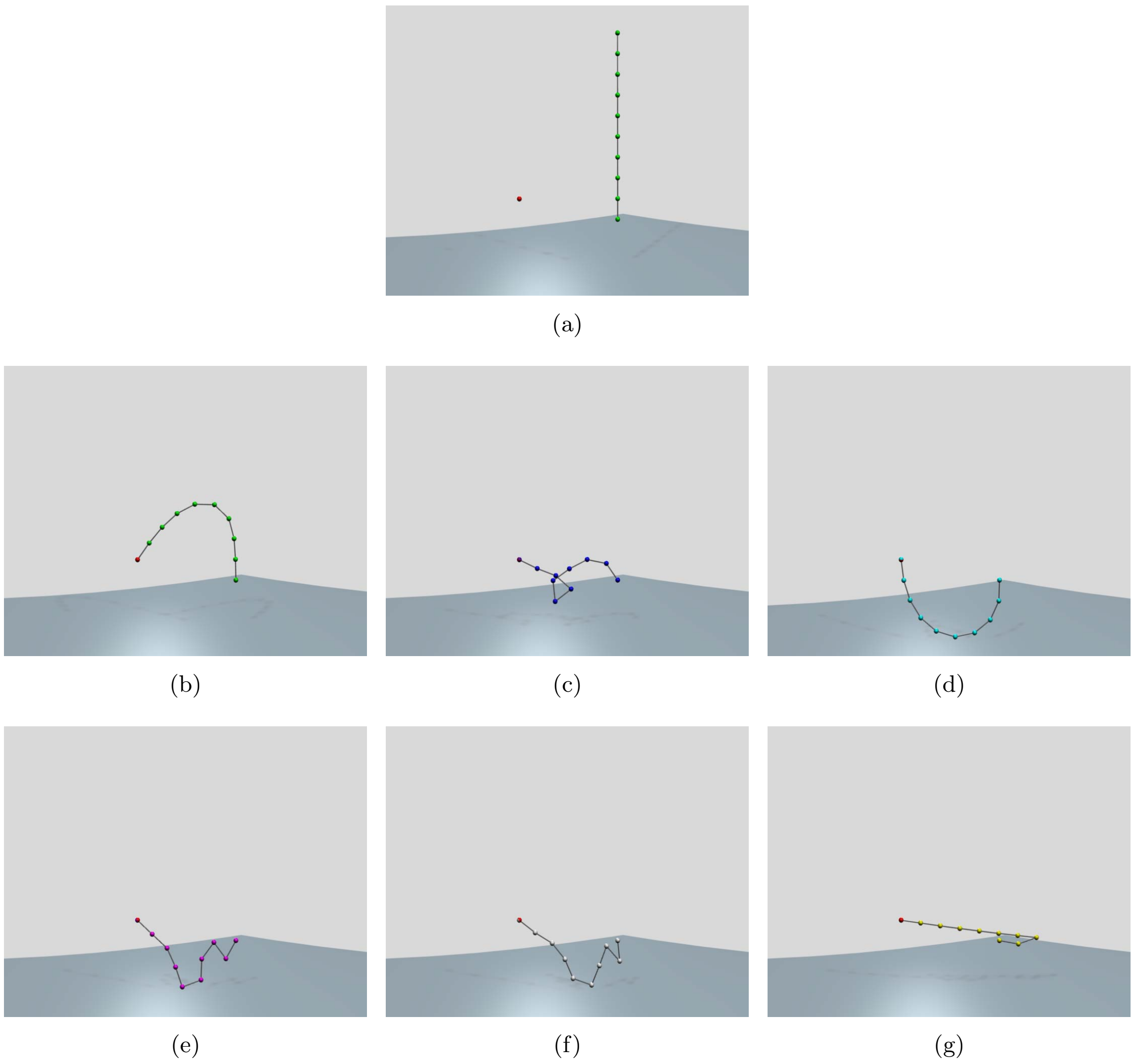


Figure 4.7: *Experimental solutions using some of the most popular IK methods. The kinematic chains consisted of 10 unconstrained joints, allowing 3 degrees of freedom on each joint. (a) Initial position, (b) FABRIK, (c) CCD, (d) J. Transpose, (e) J. DLS, (f) J. SVD-DLS, (g) Triangulation.*

to examine the reconstruction quality of each methodology. These humanoids do not have a mesh that defines their external shape, so self collisions are not considered within these reconstruction examples.

Table 4.4 shows the performance (over 20 runs) of each methodology for the case of a dancing human model. The computational cost and the reconstruction quality for tracking the animated model is also presented. FABRIK gives the best results with respect to computational cost and reconstruction quality; it requires the fewest iterations to achieve the desired posture and produces visually the smoothest and most natural poses. The average error presented in table 4.4 refers to the difference between the estimated joint positions and the true joint positions.

Table 4.4: *Reconstruction comparison. Average results (over 20 runs).*

	FABRIK	CCD	J.Transpose	J.DLS	J.SVD-DLS
Number of Iterations	65	67	1352	804	723
Average time[†] (msec)	1.6	20.5	1928	1533	1494
Time per iteration[†] (msec)	0.0246	0.3060	1.4334	1.9067	2.0664
Average Error (mm)	58.68	69.99	137.42	84.84	83.73

[†] This is a MATLAB executable time.

4.3.2 Applying restrictions

Most IK problems have rotational and orientational restrictions since most real world joints have limitations on their movements. Joint constraints can be easily added to our proposed methodology (see subsection 3.7.2). The experimental dataset used to test the reconstruction quality of the constrained FABRIK is made up of 10 joints, each having angle rotational restrictions allowing movements only within a range. The same humanoid model, as described in section 4.3.1, is used to examine the reconstruction quality of the proposed methodology with and without constraints.

FABRIK can be easily constrained producing visually realistic postures without oscillations and discontinuities. The constrained version is slightly slower than its unconstrained counterpart, requiring now almost 3.0ms to reach the target. Nevertheless, it is still much faster than other IK methods and approximately 10 times faster than the constrained CCD. The reconstruction quality is high, producing postures with an average error of just over 30mm, almost half the average error of the unconstrained version. On the other hand, while it is not difficult to apply manipulator constraints to CCD, the resulting animation often still has unnatural movements, especially when the target is at a significant distance from the end effector. The unconstrained version of CCD produces different joint poses compared to its constrained version, even if the latter is not violating the angle restrictions. It is interesting to note that there are instances where the constrained version of CCD needs fewer iterations and therefore performs slightly faster than its unconstrained version. This happens because the constraints prevent the chain from rolling and unrolling itself before reaching the target. Figure 4.15, shows examples of FABRIK and CCD implementations with and without joint restrictions. Figure 4.16 shows the reconstruction quality of the unconstrained and the constrained FABRIK version.

Table 4.5 shows the number of iterations and the time needed to reach the target for both unconstrained and constrained FABRIK and CCD approaches.

4.4 Applications

FABRIK has been successfully used for real-time marker prediction and centre of rotation (CoR) estimation [62]. The joint positions of the estimated markers are re-positioned assuming that the inter-joint distance is constant over time. Incorporating bone length constraints using

Table 4.5: *Average results when joint constraints are incorporated.*

	Number of Iterations	Matlab exe. time (*sec*)	Frames per second
FABRIK	15.461	0.01328	75.301
CCD	26.308	0.12359	8.091
FABRIK Constrained	17.142	0.03110	32.154
CCD Constrained	26.857	0.29281	3.415

FABRIK ensures that the model will have a more feasible motion. The proposed approach predicts the missing markers and estimates the joint positions reliably even if large sequences with occluded data exist, in which more than 1 marker is occluded on each limb, even if the limb rapidly changes direction. Figure 4.17 shows an example of FABRIK implementation for CoR estimation, maintaining the fixed inter-joint distance assumption.

FABRIK's performance has been also tested for hand tracking and reconstruction [63]. FABRIK captured the movements of the hand model in real-time, using the minimum possible number of markers. Needing only prior knowledge about the geometry of the hand, the hand model and the restrictions of each joint, it reproduces good estimates of the captured motion. Joint constraints are applied to ensure that the hand motion is within a feasible set, giving a visually natural motion of the hand. This method was effective and real-time implementable.

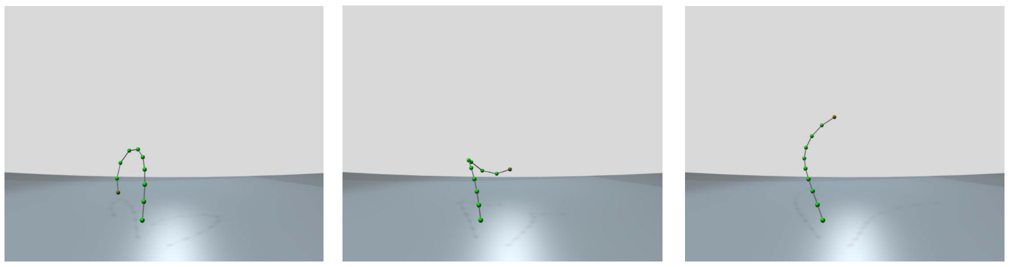
4.5 Conclusions

FABRIK is a novel methodology for solving the IK problem which does not suffer from singularity problems and which is fast and computationally efficient. Our experiments show that FABRIK requires on average the fewest iterations to reach the target, both with constrained and unconstrained kinematic chains. At the same time, it produces visually the most realistic postures, with and without constraints, reaching the desired position with the lowest computational cost. It has the best performance on tracking moving targets, producing the most natural movements in real-time, without shaking or erratic discontinuities. FABRIK can be also extended to a multiple end effector version supporting multiple kinematic chains.

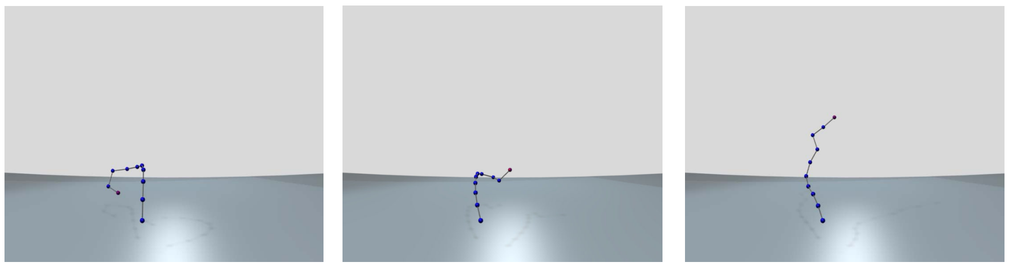
CCD is also a real-time IK solver. It is much faster than any Jacobian-based method but it is 10 times slower than FABRIK. The bigger drawback of CCD is the generation of unrealistic postures since it often rolls and unrolls itself before reaching the target. This rolling tends to overemphasise the movements of the joints closer to the end effector of the kinematic chain, thus producing unnatural movements. The CCD algorithm performs better when it is tracking a moving target (with small step-size) or when the distance between end effector and target is significantly small. In the case where the initial distance between target and end effector is large, CCD can produce unrealistic animation. Angle constraints can be easily added to the CCD methodology, controlling the movement of the manipulator. There are however, instances where the animation produced still has unnatural movements, even if manipulator constraints have been applied, especially when the target is at a significant distance from the end effector. Another limitation of CCD is that handling problems with multiple end effectors is not straightforward, since it is designed to solve problems with serial chains.

The Jacobian methods return reasonable results; the chain poses, at most times, are more realistic than CCD, especially when the target is positioned at a significant distance from the end effector. The biggest advantage of the Jacobian methods over all other methods is that, by default, they can treat problems with multiple end effectors very easily. Manipulator constraints can be incorporated within the Jacobian algorithm, but the way in which these restrictions are applied is not straightforward. The Jacobian-based methods also perform poorly when the target is moving in a sinusoidal trajectory and the end-effector must track its position. They can produce unrealistic movements and motion with oscillation, shaking and discontinuities. There are also instances where the Jacobian methods suffer from singularity problems, since matrix inversions need to be calculated. Their biggest drawback however is that they converge very slowly to their final solutions since they use a linear approximation with a small step; only under some circumstances, eg using fast C++ matrix libraries, can these kinds of methods reach the target of real-time application.

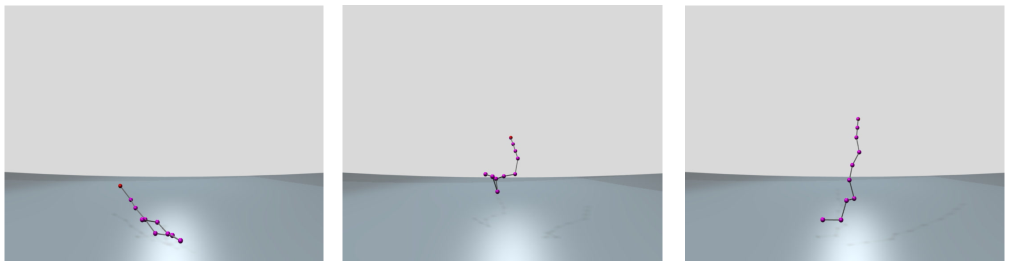
Triangulation returns the poorest results from the methods used within this report. The kinematic chain does not have a realistic shape; the joints close to the end-effector are usually in a straight line, emphasising the rotation of the joints neighbouring the root. Use of the Triangulation algorithm is limited to problems with a single end effector, making it unsuitable for complex character models with multiple end effectors. By definition, the Triangulation algorithm does not support manipulator restrictions. In this report, angle constraints have been incorporated confirming that the Triangulation algorithm often suffers from an inability to find a feasible solution, even if there is a solution, since each joint position is calculated independently without considering the restrictions of the next joint.



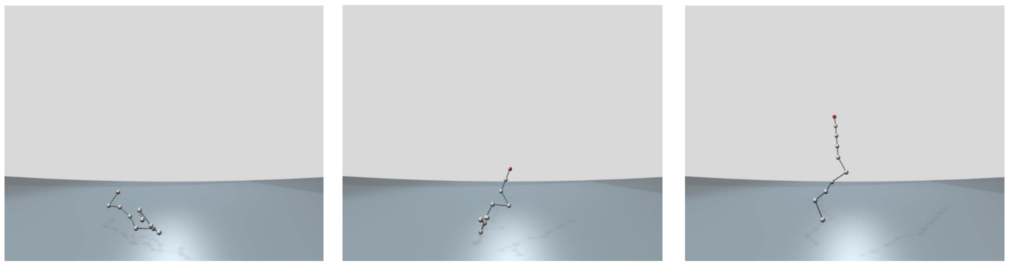
(a)



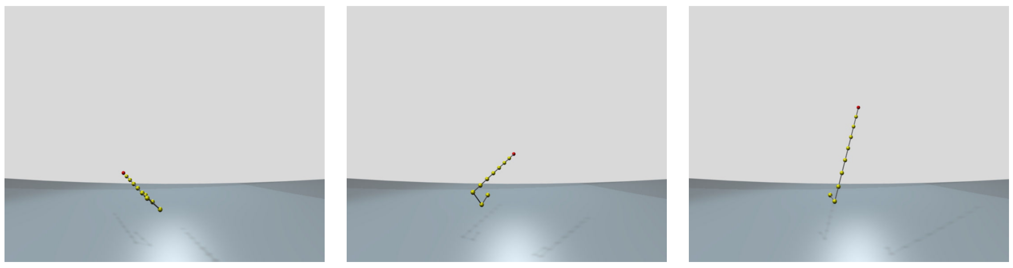
(b)



(c)

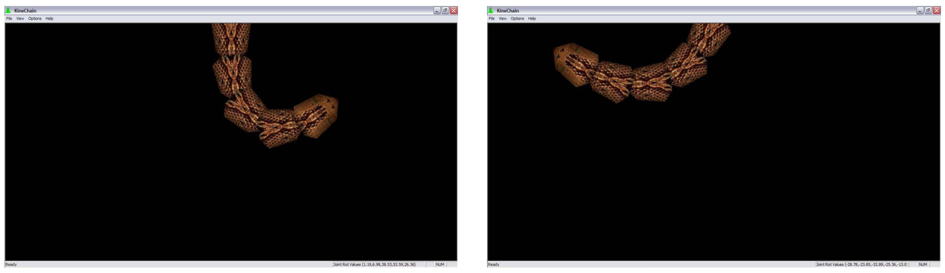


(d)

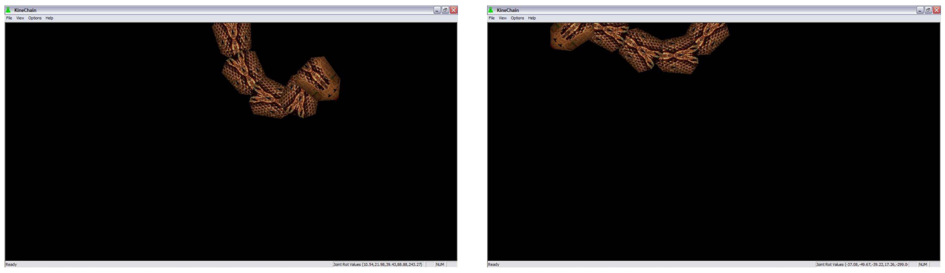


(e)

Figure 4.8: *An example of the target tracking using different methods. The frames presented here are the same for each methodology.* ***(a)*** *FABRIK,* ***(b)*** *CCD,* ***(c)*** *DLS,* ***(d)*** *SVD-DLS,* ***(e)*** *Triangulation*

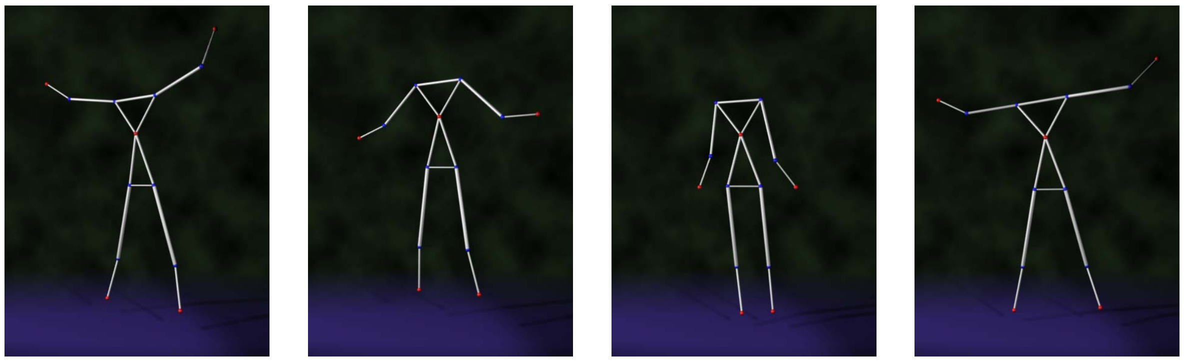


(a)

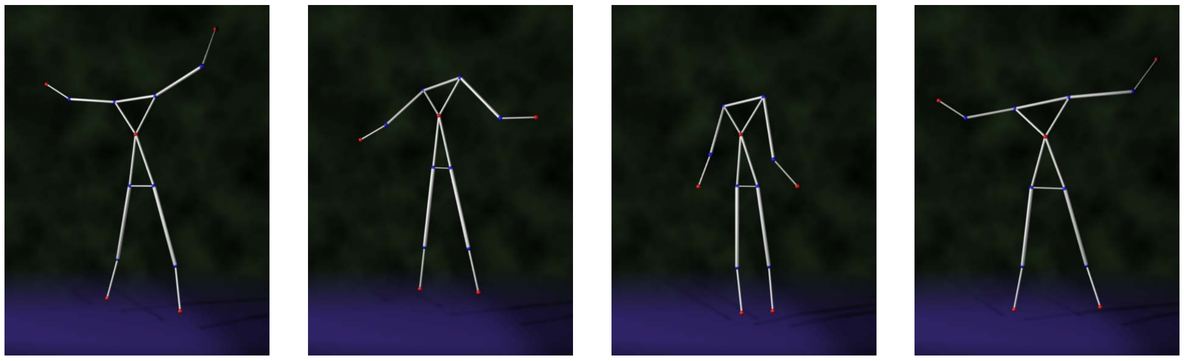


(b)

Figure 4.9: *FABRIK and CCD solution using the Kine application. (a) FABRIK solution, (b) CCD solution.*



(a)



(b)

Figure 4.10: *A low rate body tracking example. The joints in red are the known positions of the end effectors and those in blue are the estimated joint positions. (a) shows the true body poses and (b) the estimated poses using FABRIK.*

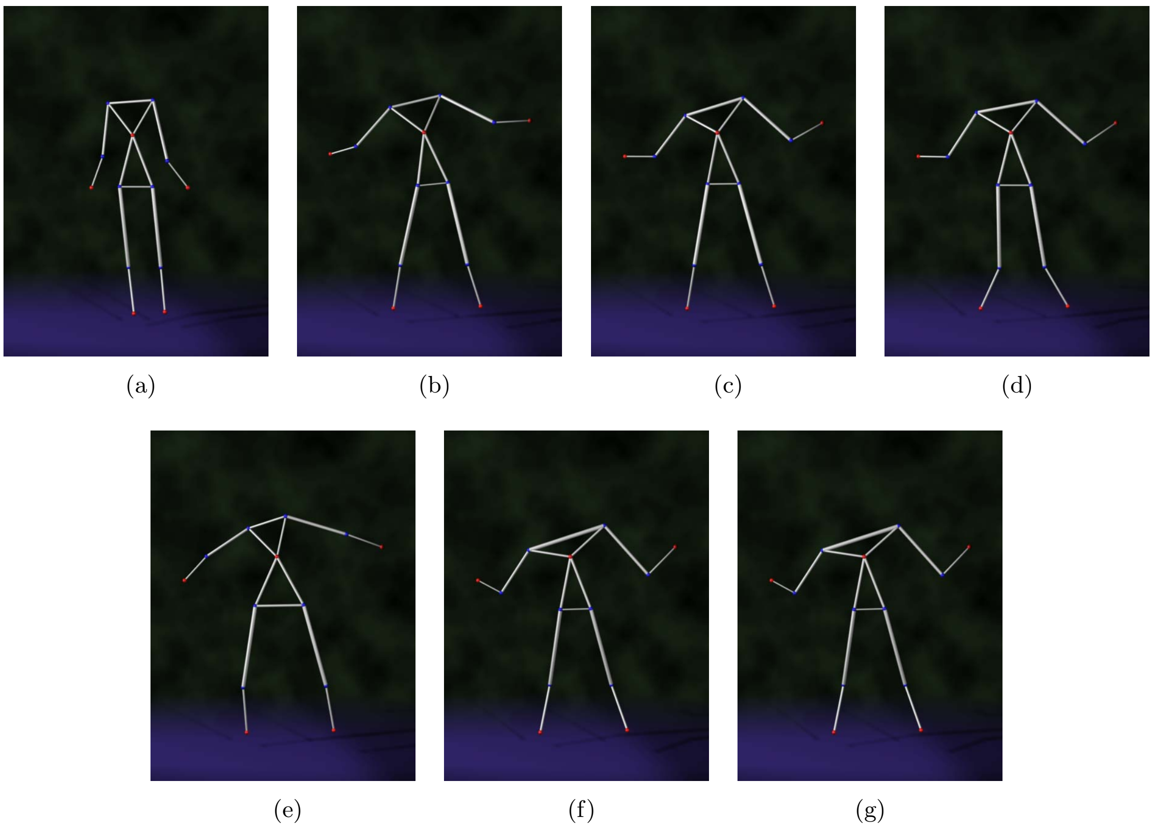


Figure 4.11: *Body reconstruction using different IK methodologies. The joints in red are the known positions of the end effectors and those in blue are the estimated joint positions. (a) shows the initial position and (b) the true final position. (c) shows the FABRIK solution, (d) the CCD solution, (e) the J. Transpose solution, (f) the J. DLS solution, (g) the J. SVD-DLS solution.*

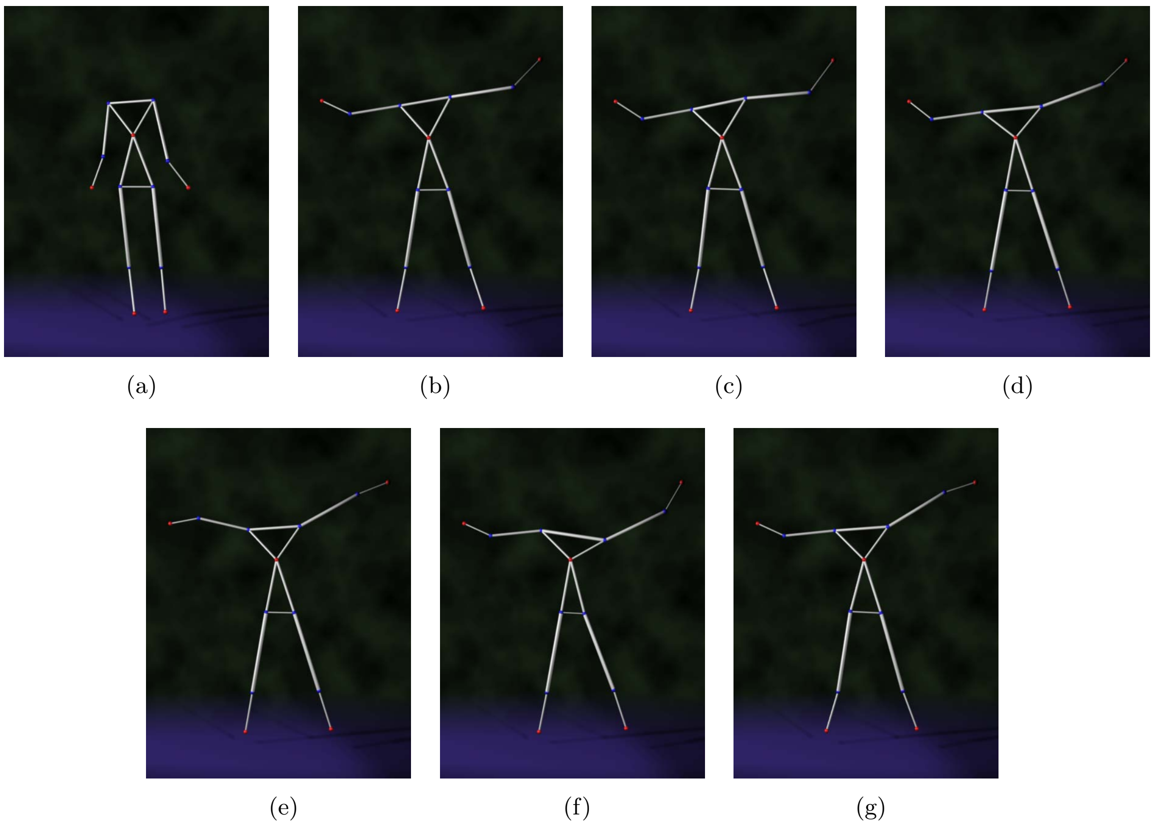


Figure 4.12: *Body reconstruction using different IK methodologies. The joints in red are the known positions of the end effectors and those in blue are the estimated joint positions. (a) shows the initial position and (b) the true final position. (c) shows the FABRIK solution, (d) the CCD solution, (e) the J. Transpose solution, (f) the J. DLS solution, (g) the J. SVD-DLS solution.*

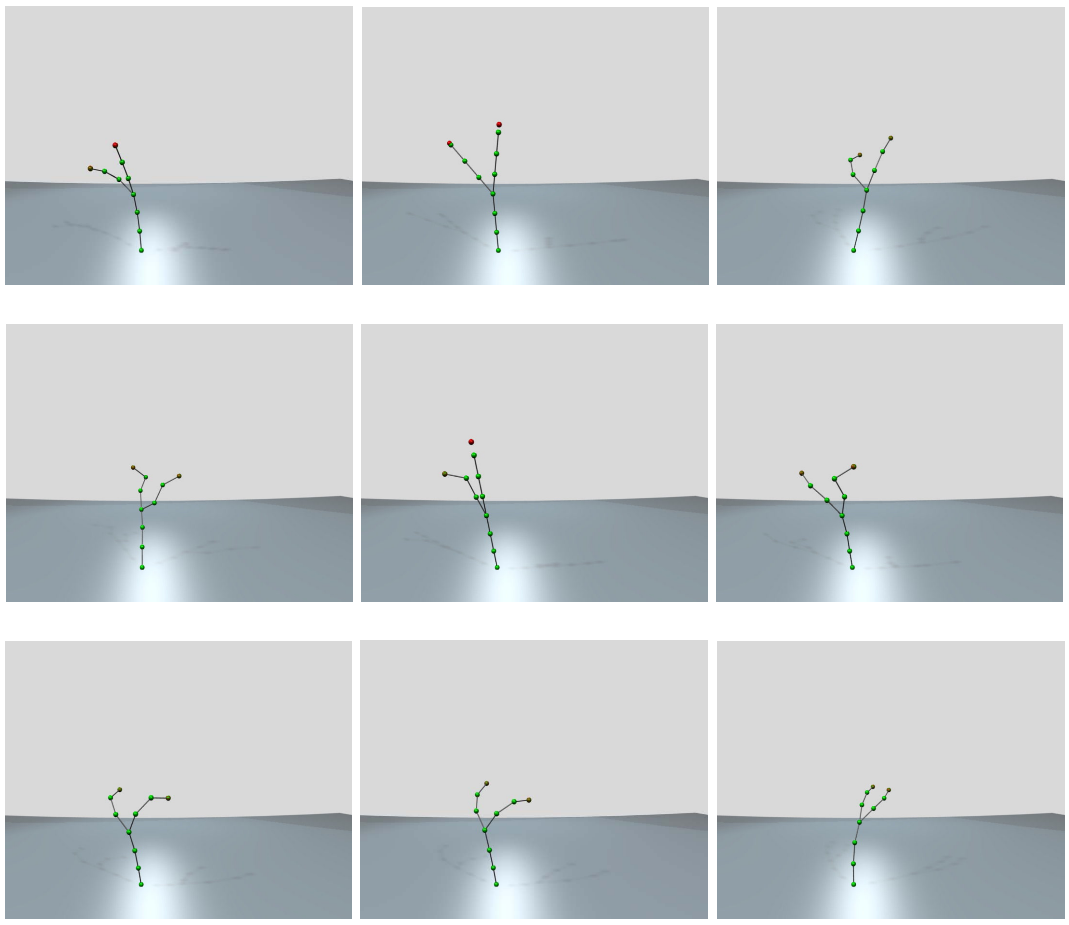


Figure 4.13: *Example of FABRIK implementation with multiple end effectors moving over time; a kinematic chain with 10 unconstrained joints, 2 end effectors and 2 targets.*

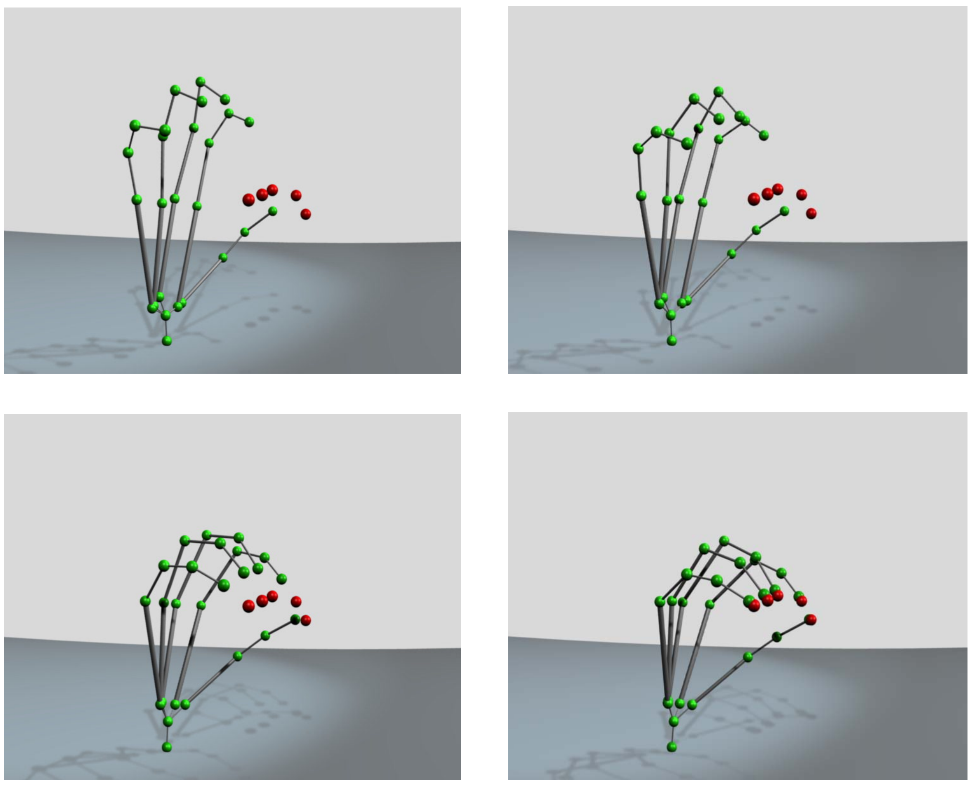


Figure 4.14: *Example of FABRIK implementation with multiple end effectors over time. This is a fully unconstrained hand example, allowing 3 DoF on each joint.*

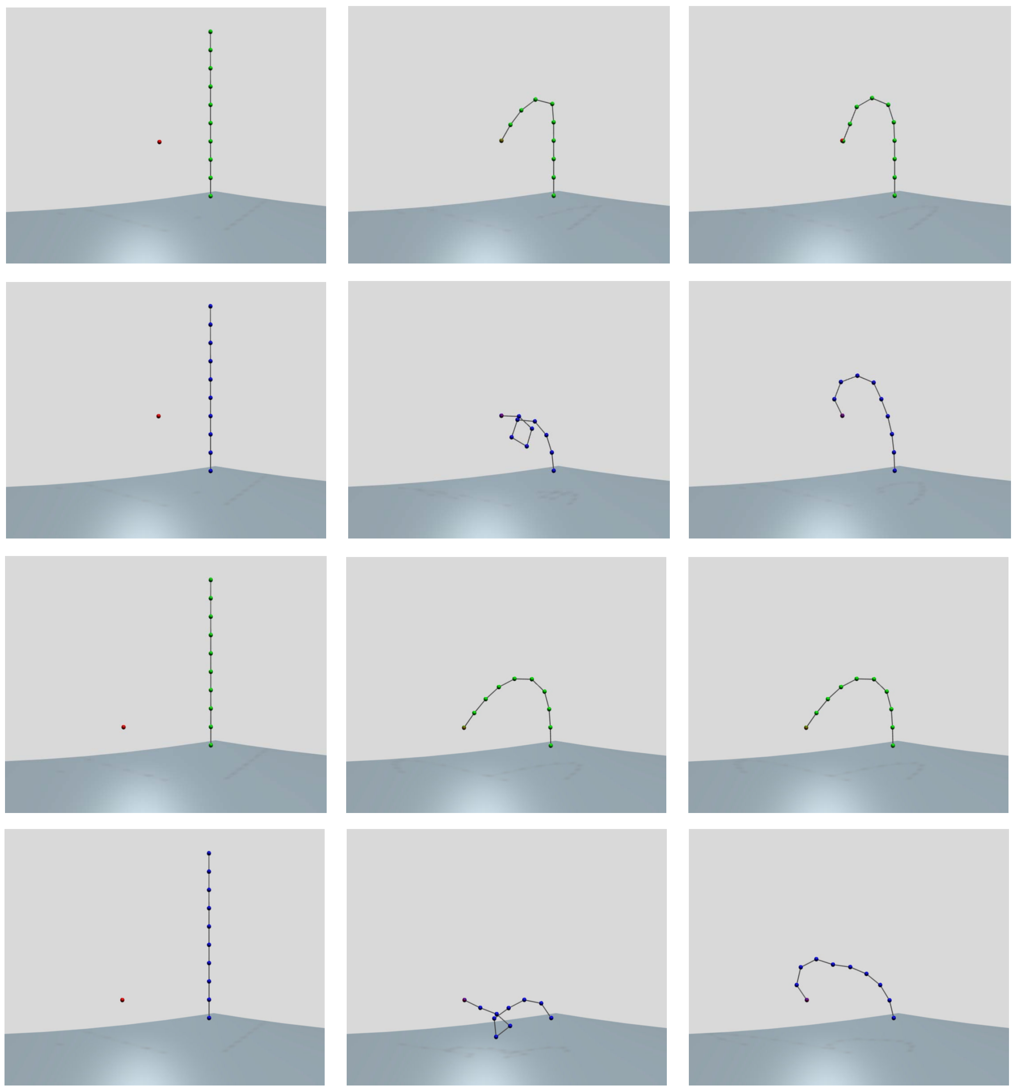


Figure 4.15: *An example of FABRIK and CCD implementations with and without incorporating constraints. Top and third lines show the FABRIK solution and second and last lines the CCD solution. (a) the initial position of the kinematic chain, (b) the unconstrained solution, (c) the constrained solution.*

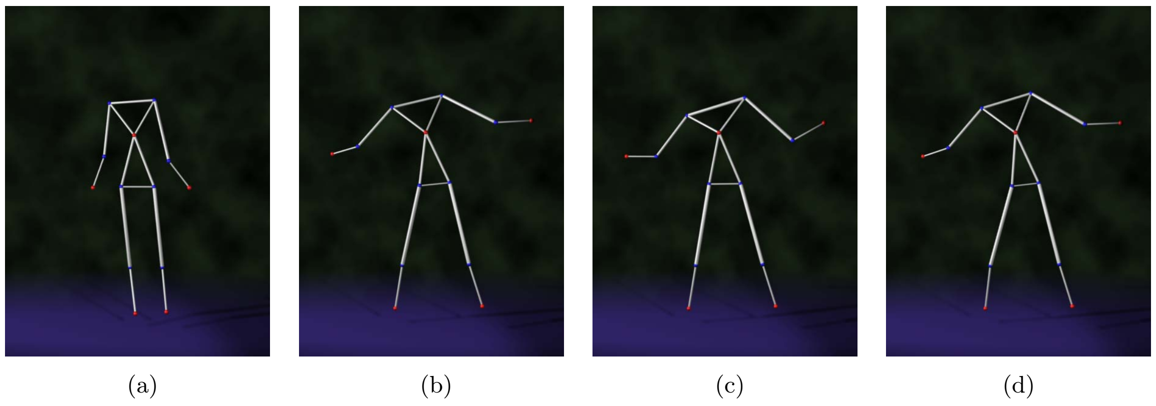


Figure 4.16: *An example of implementation. (a) The initial position, (b) the real posture, (c) the solution using unconstrained FABRIK, (d) the solution after incorporating joint restrictions.*

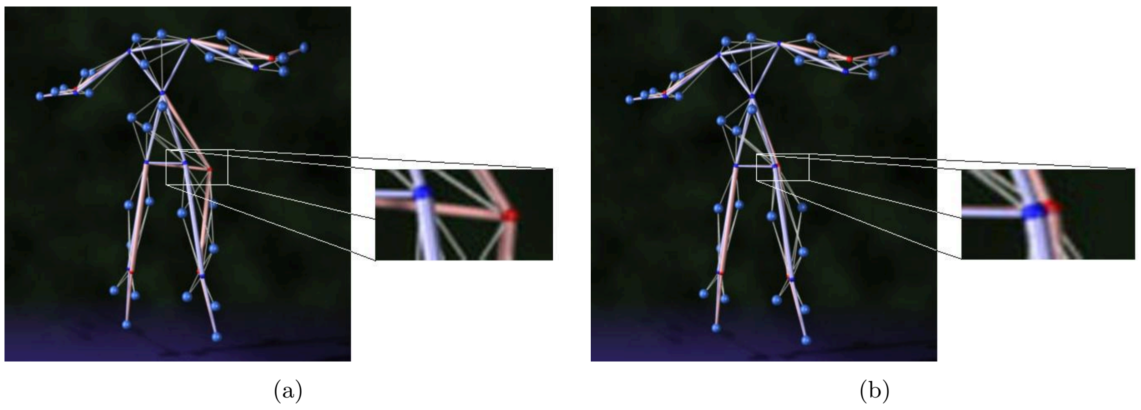


Figure 4.17: *An example of FABRIK implementation for CoR estimation under extreme cases with extended data occlusion. (a) shows results using an integrated UKF with a constant inter-marker constraint, (b) shows the results when FABRIK was applied in order to maintain the fixed inter-joint distance assumption. The true positions are coloured in blue and the predicted in red.*

5 Conclusions and Future Work

IK methods are used to control the postures of articulated bodies in frame animation production. IK finds applications in several areas such as robotics, computer animation, ergonomics and the computer games industry. However, most of the currently available methods suffer from high computational cost and production of unrealistic poses. This report presents a review of algorithms related to the Inverse Kinematics problem. It is mainly divided into three sections; the first section concerns the articulated body model describing the human joint types, a brief introduction to human models and to human motion. Section 2 considers numerical solutions to the IK problems; it describes the most popular IK solvers during the last decades such as the Jacobian family of methods (Pseudo-inverse, Transpose, DLS, SVD-DLS, SDLS), the Newton family of methods (e.g. BFGS), methods that solve the IK problem from a control prospective (FIK), sequential monte carlo methods and methods which learn the space of meaningful shapes from example meshes. CCD, a popular heuristic inverse kinematic algorithm, with its extension (IIK) is also presented. A new heuristic iterative methodology is also presented, called Forward And Backward Reaching Inverse Kinematics (FABRIK). FABRIK avoids the use of rotational angles or matrices, and instead finds each joint position via locating a point on a line. FABRIK is the first algorithm which uses an iterative method with points and lines to solve the IK problem. It divides the problem into 2 phases, a forward and backward reaching approach, and it supports (to the best of our knowledge) all the rotational joint limits and joint orientations by repositioning and re-orienting the target at each step. It does not suffer from singularity problems and it is fast and computationally efficient. No pre-recorded motion database is necessary, thereby avoiding the need for extra memory. This section also illustrates how joint constraints can be incorporated within the FABRIK methodology, restricting postures to a feasible set and how it can be expanded in order to support problems with multiple chains and multiple end effectors. In the last section, the experimental environment used within this report is described. Section 3 also presents, compares and discusses the differences between the most popular IK solvers. The IK methods are tested under several different conditions; with and without manipulator constraints, with single and multiple end effectors, how they perform on tracking moving targets and how they perform when the target has significant distance from the end effector. Several tests and comparisons have been implemented between the proposed algorithms in respect of their computational cost, processing time, conversion error, the number of iterations needed to reach the target as well as how realistic their resulting postures are.

The experimental results of this report show that the Jacobian methods return reasonable results; the chain poses, at most times, are more realistic than CCD, especially when the target is positioned at significant distance from the end effector. CCD, on the other hand, generates unrealistic postures since it often rolls and unrolls itself before reaching the target, overemphasising the movements of the joints closer to the end effector of the kinematic chain. However, CCD performs better on tracking a moving target avoiding the oscillations and motion discontinuities exhibited by the Jacobian methods. Jacobian methods have the advantage that, by default, they can treat problems with multiple end effectors very easily compared to CCD, which is designed to solve problems on serial chains. Joint restrictions can be easily added to the CCD methodology controlling the movement of the manipulator and producing more realistic motion. Jacobian methods, in contrast, support manipulator constraints, but the way in which these restrictions are applied is not straightforward. There are also instances where the Jacobian methods suffer from singularity problems, since matrix inversions need to be calculated. However, the main reason why CCD is very popular is its low computational cost, solving the IK problem in real-time. Conversely, the high computational cost is the biggest drawback of the Jacobian methods, making this family of solvers unsuitable for real-time applications.

FABRIK is a new methodology for solving the IK problem. Instead of using rotational angles to minimise the position and orientation errors, it finds each new joint position via locating a point on a line. FABRIK and CCD are both heuristic iterative methods, thus they do not suffer from singularity problems. It is experimentally proven that FABRIK requires on average fewer iterations to reach the target than any other IK method considered here, both with constrained and unconstrained kinematic chains. It produces the most realistic postures, with and without constraints, reaching the desired position with the lowest computational cost and executing in real-time. It has also the best performance on tracking moving targets, producing the most natural movement, without oscillations and discontinuities. FABRIK can be easily extended to solve problems with multiple end effectors supporting complex models with multiple kinematic chains.

Future work will see the introduction of the proposed algorithm within kinematic chain models in order to ensure more realistic chain movements. Analytically and anatomically correct models are necessary to control and constrain the movements of any legged body. A sophisticated hand model will be implemented which takes into account, not only the joint rotational and orientational restrictions (i.e. [63]), but also constraints related to the hand model, such as self-collisions, inertia, flexion etc. This extension will provide accurate results, ensuring that the hand will have more natural poses, without violating any biomechanical or model constraints.

List of Abbreviations

BFGS	-	Broyden, Fletcher, Goldfarb, Shanno method
CCD	-	Cyclic Coordinate Descent
CGA	-	Conformal Geometric Algebra
CoR	-	Centre of Rotation
DLS	-	Damped Least Squares
DoF	-	Degrees of Freedom
FABRIK	-	Forward And Backward Reaching Inverse Kinematics
FIK	-	Feedback Inverse Kinematics
FK	-	Forward Kinematics
FTL	-	Follow The Leader
GA	-	Geometric Algebra
HMM	-	Hidden Markov Model
IIK	-	Inductive Inverse Kinematics
IK	-	Inverse Kinematics
SDLS	-	Selectively Damped Least Squares
SIK	-	Sequential Inverse Kinematics
SMCM	-	Sequential Monte Carlo Methods
SVD	-	Singular Value Decomposition
UPM	-	Uniform Posture Map

Bibliography

[1] Jeff Lander. Making kine more flexible. *Game Developer*, 5(3):15–22, 1998.

[2] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics (TOG)*, 13(4):313–336, 1994.

[3] A. Balestrino, G. De Maria, and L. Sciavicco. Robust control of robotic manipulators. In *Proceedings of the 9th IFAC World Congress*, volume 5, pages 2435–2440, 1984.

[4] W. A. Wolovich and H. Elliott. A computational technique for inverse kinematics. *The 23rd IEEE Conference on Decision and Control*, 23:1359–1363, December 1984.

[5] J. Baillieul. Kinematic programming alternatives for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 722–728, March 1985.

[6] C. W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *Proceeding of the IEEE Transactions on Systems, Man and Cybernetics*, 16(1):93–101, 1986.

[7] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Trans. ASME, Journal of Dynamic Systems, Measurement, and Control*, 108(3):163–171, September 1986.

[8] Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, 2005.

[9] Alexandre N. Pechev. Inverse kinematics without matrix invertion. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 2005–2012, Pasadena, CA, USA, May 19-23 2008.

[10] R. Fletcher. *Practical methods of optimization; (2nd Ed.).* Wiley-Interscience, New York, NY, USA, 1987.

[11] Nicolas Courty and Elise Arnaud. Inverse kinematics using sequential monte carlo methods. In *Proceedings of the V Conference on Articulated Motion and Deformable Objects, AMDO'08*, volume 5098, pages 1–10, Mallorca, Spain, 2008. LNCS.

[12] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John Deweese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics (TOG)*, 27(3):1–11, 2008.

[13] Luis Unzueta, Manuel Peinado, Ronan Boulic, and Ángel Suescun. Full-body performance animation with sequential inverse kinematics. *Graph. Models*, 70(5):87–104, 2008.

[14] R Boulic, J Varona, L Unzueta, M Peinado, A Suescun, and F Perales. Evaluation of on-line analytic and numeric inverse kinematics approaches driven by partial vision input. *Virtual Reality*, 10(1):48–61, 2006.

[15] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *SIGGRAPH '04: ACM Transactions on Graphics*, pages 522–531, New York, NY, USA, August 2004. ACM.

[16] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *ACM Transactions of Graphics*, 24(3):488–495, 2005.

[17] Kevin G. Der, Robert W. Sumner, and Jovan Popović. Inverse kinematics for reduced deformable models. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1174–1179, New York, NY, USA, 2006. ACM.

[18] Li-Chun Tommy Wang and Chih Cheng Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, 1991.

[19] Chris Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. *Master Dissertation, Simon Fraser University, Department of Computer Science*, 1993.

[20] Adrian A. Canutescu and Roland L. Dunbrack. Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Science*, 12(5):963–972, May 2003.

[21] Joel Brown, Jean-Claude Latombe, and Kevin Montgomery. Real-time knot-tying simulation. *The Visual Computer: International J. of Computer Graphics*, 20(2):165–179, 2004.

[22] Martin Fêdor. Application of inverse kinematics for skeleton manipulation in real-time. In *Proceedings of the 19th spring conference on Computer graphics, SCCG '03*, pages 203–212, New York, NY, USA, 2003. ACM.

[23] Schubert Carvalho, Ronan Boulic, and Daniel Thalmann. Interactive low-dimensional human motion synthesis by combining motion models and pik. *Computer Animation and Virtual Worlds*, 18, 2007. Special Issue of Computer Animation and Social Agents (CASA2007).

[24] Jinxiang Chai and Jessica K. Hodgins. Constraint-based motion optimization using a statistical dynamic model. In *ACM Transactions on Graphics (TOG), ACM SIGGRAPH 2007*, page 8, New York, NY, USA, 2007. ACM.

[25] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Transactions on Graphics (TOG)*, 24(1):98–117, 2005.

[26] PhaseSpace Inc:. Optical motion capture systems, `http://www.phasespace.com`.

[27] Blender Foundation:. 3d content creation suite program, `http://www.blender.org`.

[28] N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, New York, Oxford, 1993.

[29] John J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley Longman Publishing Co., Inc., 1989.

[30] W. Maurel and D. Thalmann. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics*, 24(2):203–18, 2000.

[31] Xuguang Wang and Jean Pierre Verriest. A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation. *Journal of Visualization and Computer Animation*, 9(1):33–47, 1998.

[32] Manohar M. Panjabi Augustus A. White III. *Clinical biomechanics of the spine.* J.B. Lippincott Company, Second Edition, 1990.

[33] James Urey Korein. *A geometric investigation of reach.* MIT Press, Cambridge, MA, USA, 1985.

[34] G. Monheit and N.I. Badler. A kinematic model of the human spine and torso. *IEEE Computer Graphics and Applications*, 11(2):29–38, Mar 1991.

[35] Hans Rijpkema and Michael Girard. Computer animation of knowledge-based human grasping. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 339–348, New York, NY, USA, 1991. ACM.

[36] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Inc., Boca Raton, FL, USA, 1994.

[37] Paris Kaimakis and Joan Lasenby. Gradient-based hand tracking using silhouette data. In *Proceeding of the 3rd International Symposium on Visual Computing (ISVC)*, volume 1, pages 24–35, Lake Tahoe, NV/CA, USA, November 26-28 2007.

[38] Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. Anatomy-based modeling of the human musculature. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 163–172, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[39] Alan Watt and Mark Watt. *Advanced animation and rendering techniques.* Addison - Wisley, ACM Press, New York, NY, USA, 1991.

[40] Types of Joints in the Human Body. Microsoft encarta online encyclopedia 2008, ©1997-2008 Microsoft Corporation, `http://encarta.msn.com`.

[41] Jonathan Cameron and Joan Lasenby. A real-time sequential algorithm for human joint localization. In *ACM SIGGRAPH Posters*, page 107, New York, USA, 2005. ACM Press.

[42] Andreas Aristidou, Jonathan Cameron, and Joan Lasenby. Predicting missing markers to drive real-time centre of rotation estimation. In *Proceedings of the V Conference on Articulated Motion and Deformable Objects, AMDO'08*, volume 5098, pages 238–247, Mallorca, Spain, 2008. LNCS.

[43] D. Hestenes and G. Sobczyk. *Clifford Algebra to Geometric Calculus: A unified language for mathematics and physics.* D. Reidel, 1984.

[44] Chris Doran and Anthony Lasenby. *Geometric Algebra for Physicists.* Cambridge University Press, Cambridge UK, 2003.

[45] David E. Orin and William W. Schrader. Efficient computation of the jacobian for robot manipulators. *The International Journal of Robotics Research*, 3(4):66–75, 1984.

[46] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man and Cybernetics*, 7(12):868–871, December 1977.

[47] Anthony A. Maciejewski and Charles A. Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 4(3):109–117, 1985.

[48] Michael Meredith and Steve Maddock. Adapting motion capture data using weighted real-time inverse kinematics. *Computers in Entertainment (CIE)*, 3(1):1–15, 2005.

[49] Kwan W. Chin, B. R. von Konsky, and A. Marriott. Closed-form and generalized inverse kinematics solutions for the analysis of human motion. volume 5, pages 1911–1914, 1997.

[50] Damian Merrick and Tim Dwyer. Skeletal animation for the exploration of graphs. In Neville Churcher and Clare Churcher, editors, *Australasian Symposium on Information Visualisation, (invis.au'04)*, volume 35 of *CRPIT*, pages 61–70, Christchurch, New Zealand, 2004. ACS.

[51] Jin Ok Kim, Bum Ro Lee, Chin Hyun Chung, Jun Hwang, and Woongjae Lee. The inductive inverse kinematics algorithm to manipulate the posture of an articulated body. In *Proceedings of the International Conference on Computational Science, ICCS 2003*, volume 2657 of *Lecture Notes in Computer Science*, pages 305–313. Springer, June 2-4 2003.

[52] R. Müller-Cajar and R. Mukundan. Triangulation: A new algorithm for inverse kinematics. In *Proceedings of the Image and Vision Computing New Zealand 2007*, pages 181–186, New Zealand, December 2007.

[53] Andreas Aristidou and Joan Lasenby. FABRIK: a fast, iterative solver for the inverse kinematics problem. *Submitted to Graphical Models*, 2010.

[54] N. Klop̆car, M. Tomšič, and J. Lenarčič. A kinematic model of the shoulder complex to evaluate the arm-reachable workspace. *Journal of Biomechanics*, 40(1):86 – 91, 2007.

[55] L. Herda, R. Urtasun, A. Hanson, and P. Fua. Automatic Determination of Shoulder Joint Limits using Experimentally Determined Quaternion Field Boundaries. *International Journal of Robotics Research*, 22(6), 2003.

[56] Jonathan Blow. Inverse kinematics with quaternion joint limits. *Game Developer*, April 2002.

[57] Jane Wilhelms and Allen Van Gelder. Fast and easy reach-cone joint limits. *Journal of Graphic Tools*, 6(2):27–41, 2001.

[58] P. Baerlocher and B. Boulic. Parametrization and range of motion of the ball-and-socket joint. In *Deformable Avatars*, pages 180–190. Kluwer Academic Publishers, 2001.

[59] Sung Joon Ahn, Wolfgang Rauh, and Hans-Jrgen Warnecke. Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola. *Pattern Recognition*, 34(12):2283 – 2303, 2001.

[60] Ming C. Lin and Stefan Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA Conf. on Mathematics of Surfaces*, pages 37–56, 1998.

[61] The Mathworks MATLAB and Simulink for Technical Computing, `http://www.mathworks.com`.

[62] Andreas Aristidou and Joan Lasenby. Real-time marker prediction and CoR estimation in optical motion capture. *Submitted to Image and Vision Computing*, 2010.

[63] Andreas Aristidou and Joan Lasenby. Motion capture with constrained inverse kinematics for real-time hand tracking. In *IEEE Proceedings of the 4th International Symposium on Communications, Control and Signal Processing (ISCCSP 2010)*, Limassol, Cyprus, March 3-5 2010.