University of Cambridge

Tracking and Modelling Motion for Biomechanical Analysis

Andreas Aristidou
Hughes Hall

Cambridge University Engineering Department
Signal Processing and Communication Group

October 5, 2010

*This dissertation is submitted
for the degree of Doctor of Philosophy*

Typeset by the author with the $\LaTeX 2_\varepsilon$ documentation system

Submitted Version, July, 2010.

Resubmitted with minor corrections, October, 2010.

Signal Processing and Communication Group
Department of Engineering
University of Cambridge
Trumpington Street, CB2 1PZ

Declaration

I, Andreas Aristidou of Hughes Hall hereby declare that, except where specifically indicated in the text, the work submitted herein is my own original work and includes nothing which is the outcome of work done in collaboration. This dissertation is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other university. This dissertation is the result of my own work.

I also declare that the length of this dissertation is less than 65,000 words and that the number of figures is less than 150.

Signature: ....

Andreas Aristidou

Cambridge

October 5, 2010

Abstract

This thesis focuses on the problem of determining appropriate skeletal configurations for which a virtual animated character moves 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 smooth and natural solutions to the Inverse Kinematics (IK) problem. However, many of the currently available methods suffer from high computational cost and production of unrealistic poses. In this study, a novel heuristic method, called Forward And Backward Reaching Inverse Kinematics (FABRIK), is proposed, which returns visually natural poses in real-time, equally comparable with highly sophisticated approaches. It is capable of supporting constraints for most of the known joint types and it can be extended to solve problems with multiple end effectors, multiple targets and closed loops. FABRIK was compared against the most popular IK approaches and evaluated in terms of its robustness and performance limitations. This thesis also includes a robust methodology for marker prediction under multiple marker occlusion for extended time periods, in order to drive real-time centre of rotation (CoR) estimations. Inferred information from neighbouring markers has been utilised, assuming that the inter-marker distances remain constant over time. This is the first time where the useful information about the missing markers positions which are partially visible to a single camera is deployed. Experiments demonstrate that the proposed methodology can effectively track the occluded markers with high accuracy, even if the occlusion persists for extended periods of time, recovering in real-time good estimates of the true joint positions. In addition, the predicted positions of the joints were further improved by employing FABRIK to relocate their positions and ensure a fixed bone length over time. Our methodology is tested against some of the most popular methods for marker prediction and the results confirm that our approach outperforms these methods in estimating both marker and CoR positions. Finally, an efficient model for real-time hand tracking and reconstruction that requires a minimum number of available markers, one on each finger, is presented. The proposed hand model is highly constrained with joint rotational and orientational constraints, restricting the fingers and palm movements to an appropriate feasible set. FABRIK is then incorporated to estimate the remaining joint positions and to fit them to the hand model. Physiological constraints, such as inertia, abduction, flexion etc, are also incorporated to correct the final hand posture. A mesh deformation algorithm is then applied to visualise the movements of the underlying hand skeleton for comparison with the true hand poses. The mathematical framework used for describing and implementing the techniques discussed within this thesis is Conformal Geometric Algebra (CGA).

Keywords
The following keywords may be used for indexing purposes:
Centre of Rotation Estimation, Computer Vision, Conformal Geometric Algebra, Filtering, Forward And Backward Reaching Inverse Kinematics (FABRIK), Hand Reconstruction, Hand Tracking, Human Animation, Inverse Kinematics, Joint Configuration, Unscented Kalman Filter, Marker prediction, Motion Capture, Physiological constraints, Variable Turn Model.

Acknowledgements

This is a great opportunity to thank those who supported me through my doctorate experience in Cambridge. Foremost, I am heartily thankful to my supervisor, Joan Lasenby, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. Her perpetual energy and enthusiasm in research had motivated me as well as her regular supervision which made this study possible.

It was an honor for me to have Prof William Fitzgerald and Prof Adrian Hilton as the examiners for my thesis. Their comments were constructive, enable a considerable improvement of both the content and the presentation of my thesis. I would like to thank them for agreeing to be my examiners and for making my viva a unique experience.

I am indebted to many of my colleagues in Signal Processing Group for welcoming and supporting me over this period. More specifically, I owe an enormous dept to Jonathan for his continued help and for introducing me to Geometric Algebra and quaternions and to Paris for his useful comments and feedback. Also, I would like to thank Richard for introducing me to Blender, for his fruitful discussions and help with capturing data and producing video examples. I am also very grateful to Charles for his help with video processing and to Adam for allowing us to use his voice as background to some videos.

Starting my PhD, I thought that getting the degree and graduating was the most important achievement. However, along the way I realised that what matters is not the destination but the journey itself and what you learn and achieve through this journey. Here in Cambridge, I had the opportunity to meet many distinguished scientists, to work in a highly professional environment and equally coordinate with them. These are opportunities of a lifetime. It was an honor for me to be member of this community and learn so much from them. Beside the academic part of my life here, I also had the chance to make new friends. There are so many that it is difficult to mention all of them without leaving somebody disgruntled. I owe a huge thanks to the Hellenic Society and Real Madra members for welcoming and helping me to easily adapt to the Cambridge life style. Special thanks to Zoe for her fertile discussions and to Themis for his invaluable help and for introducing me to Dr Lasenby. Finally, a deep thankyou to two special friends, Kantas and my housemate Bamies.

I am also very grateful to my parents and sister, for their love and support, both psychologically and financially, throughout my life and for giving me the opportunity to follow my ambitions. I owe my deepest gratitude to my godmother Nike and my uncle Tasos, as well as my cousins Peny and Marina, for their hospitality and support, for more than a month, at the time of my knee operation in Athens. Lastly, I offer my regards to all of those who believed on me, supported and inspired me in any respect during the completion of my studies.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

THIS thesis addresses the problem of manipulating articulated figures in an interactive and intuitive fashion for the design and control of their posture. It finds applications in the areas of robotics, computer animation, ergonomics and the computer games industry. A novel heuristic method, called FABRIK, is proposed, which solves the Inverse Kinematics problem in real-time and returns visually natural poses comparable with more complex and highly sophisticated approaches. We also derive a robust methodology for marker prediction under multiple marker occlusion for extended time periods, in order to drive real-time centre of rotation (CoR) estimations. Finally, an efficient real-time hand pose tracker is presented, using optical motion capture data. Physiological constraints are incorporated allowing motion within an anatomically feasible set.

1.1. Introduction and Motivation

In computer graphics, articulated figures are a convenient model for humans, animals or other virtual creatures found commonly in 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.

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 a large number of degrees of freedom exists.

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 graphics 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 solutions.

In this work, the most popular Inverse Kinematics 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 is easy to implement and 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 used 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 the sake of comparison 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 movement, reconstruction quality, conversion criteria and number of iterations needed.

Inverse Kinematics solvers are very popular in the robotics, computer graphics and animation communities; they are extensively used to control and animate legged figures or articulated robots. The most popular method used to extract accurate data and animate such models is motion capture. Optical motion capture is a technology used to turn the observations of a moving subject (taken from a number of cameras) into 3D position and orientation information about that subject. Such information can be used for the following applications: to better analyse techniques in sports training and performance (e.g. posture, velocities, accelerations, angles, trajectories) [HNT+06, Gol]; to study human movements for medical reasons; to observe asymmetries and abnormalities in rehabilitation medicine (e.g. gait in stroke or prosthetic patients) [HDSB05, PTP+05, BSR07]; in the generation of virtual characters for films or computer games [Men99] and for human-computer interactions (HCI), including interaction with game consoles. IK has recently also been adopted in protein science for protein structure prediction and reconstruction [CD03]. There are two basic approaches used to capture such data, markered and markerless. Throughout this work, markered motion capture is used. The problem of establishing the motion of interest is simplified by attaching markers of some type to the subject being recorded. These markers can then be easily located in an image and their movement can be used to infer the complete movement of the person to whom they are attached.

In general, to achieve accurate skeletal reconstruction of any legged body, three markers must be available on each limb segment at all times. However, even with many cameras, there are instances where occlusion of markers by elements of the scene leads to missing data. In order to establish its position without ambiguity, each marker must be visible to at least two cameras in each frame. Although many methods have been developed to handle the missing marker problem, most of them are not real-time, are usually limited to short time period occlusions and often require manual intervention.

In this thesis, we investigate methodologies for real-time marker prediction, under multiple cases of occlusion, to drive centre of rotation (CoR) estimates and then to automatically establish the skeleton model. A real-time integrated framework is presented, which predicts the occluded marker positions using a variable turn model within an Unscented Kalman filter. The previous marker positions are used within the framework in addition to information related to the missing markers of the current frame, inferred from an approximate rigid body assumption. The predicted marker positions are then used to locate the joints. Without assuming any skeleton model, we take advantage of the fact that for markers on a given limb segment, the inter-marker distance is approximately constant. The proposed marker constraint methodology is simple, real-time implementable and very efficient. It can deal with all the cases of marker occlusion within a limb, resulting in accurate predictions even when all markers on a limb segment are missing for an extended period of time. The proposed approach also takes advantage of the special, but common, case where missing markers are still visible to one camera. With a continuous stream of accurate labelled 3D data, we can perform real-time CoR estimation; the CoR position is thereafter corrected via a real-time Inverse Kinematic technique (FABRIK) which ensures that the inter-joint pairwise distances remain constant over time. A skeletal reconstruction is thereby achieved, producing information which can be used for visual performance feedback. Experiments demonstrate that our methodology effectively recovers good estimates of the true positions of the missing markers and CoRs, even if all the markers on a limb are occluded for a long period of time. The resulting motion is natural and smooth, and can be achieved in real-time.

In recent years there has been a growing demand for reliable hand motion tracking systems, a technology used to turn the observations of a moving hand into 3D position and orientation information. However, building a fast and effective hand pose tracker remains challenging; the high dimensionality of the pose space, the ambiguities due to self-occlusions and the significant appearance variations due to shading make efficient tracking difficult. Marker-based motion capture has been demonstrated in several interactive systems (including but not limited to hand interaction); the results are highly accurate and easy to configure. There are, however, instances where we do not have many markers available or it is impossible to attach three markers on each limb segment; the large number of markers needed is often prohibitive. It may therefore be infeasible to track the object and reconstruct its skeletal model (i.e. the hand model). A new way of capturing the movement of these articulated models is therefore required, using the minimum possible number of markers. Instead of attaching three markers on each limb segment, we investigate a system in which a single marker is attached and captured on each finger (end effector), one marker at the chain base (root) and two markers at strategic positions to help us define the hand orientation. The markers' positions are tracked using an optical motion capture system, such as [Pha]. However, prior knowledge about the geometry of the hand, the hand model and the restrictions of each joint are needed. Joint constraints are applied to ensure that finger motion is within a feasible set, giving a visually natural motion of the hand. An Inverse Kinematics solver (FABRIK) is incorporated to estimate the remaining joint positions and to fit them to the hand model. Physiological constraints related to the hand anatomy are then enforced to restrict the motion only to natural possible poses, without violating any model constraint. Finally, a mesh deformation algorithm has been applied to drive the animation of the underlying hand skeleton using a set of per-bone weights. The implemented mesh videos are compared with their true hand motions and the results verify that the suggested method is effective; the method is real-time implementable and tracks the hand motion smoothly, without oscillations, even with a low capture frame rate.

1.2. Literature Review

The area of inverse kinematics has been extensively studied during the last decades. In this literature review we present the most popular methods which solve the IK problem and compute the poses of a manipulator. A joint localisation review is also given in addition to methods for marker prediction and centre of rotation estimation. We conclude this section with a review of recent hand pose tracking methods. Note that this section is not meant as an exhaustive literature review but rather a brief outline of the background literature. A more detailed review can be found separately in each chapter, dealing with the relevant subject.

Inverse Kinematics review: The production of realistic and plausible motions posed a problem for scholars for many years in the field of robotics technology and computer graphics. During recent decades, several models have been implemented for solving the IK problem from many different areas of study. [ZB94] treats 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 [BDMS84, WE84, Bai85, Wam86, NH86, BK05]. 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 [Pec08] 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, see [Fle87] for a detailed review. However, the Newton methods are complex, difficult to implement and have high computational cost per iteration.

A very popular IK method is the Cyclic Coordinate Descent (CCD) algorithm, which was first introduced by [WC91] and then biomechanically constrained by [Wel93]. CCD has been extensively used in the computer games industry [Lan98] and has recently been adapted for protein structure prediction [CD03]. 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. [UPBS08] describes a Sequential IK solver (SIK), which is a direct extension of [BVU$^+$06], in that its inputs 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 (CCD), in different parts of the body, in a specific order. [KM05] also adopted the CCD kinematic algorithm and solved its crucial problem of resulting unnatural poses. The proposed extension in [KM05] is able to solve problems with humanoid hierarchy, dividing the whole body into groups of joints near an end effector (typically head, trunk, arms and legs). In order to satisfy the desired centre of mass, the lightest group moves first, adjusting its centre of mass by changing the length of the limb and rotating it (assuming it is a rigid body).

Recently, [CA08] and [HRE$^+$08] proposed a Sequential Monte Carlo Method (SMCM) and particle filtering approach respectively. The proposed particle IK solver treats the character skeleton as a set of 3 degrees of freedom (DoF) particles having inter-length constraints. An iterative constrainer, with various pre-conditions and parameters, is then applied over the particles, tuning its behavior both statically and dynamically. The final particle positions and the length constraints are then used to reconstruct the resulting DoF of the body. Neither method suffers from matrix singularity problems and both perform reasonably well. However, these statistical methods have high computational cost. [GMHP04] 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 correlated with similar poses in the training data. In [SZGP05, DSP06], the authors used mesh-based IK 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, and satisfy vertex constraints imposed by the user. However, these methods require an off-line training procedure, their results are highly dependent on the training data and limited only to those models and movements on which the system has been trained. A detailed explanation of all these methods is given in chapter 3.2.

[BLM04] present a real-time method which uses a 'Follow-the-Leader' (FTL) non-iterative technique which is similar to each individual iteration of FABRIK. FTL was specifically designed for rope simulation and has neither been used for IK solutions nor extended so that it can function as an IK solver. FTL does not use points and lines to estimate joint positions, does not work in an iterative fashion and manipulates the kinematic chain (ball-and-socket joints connected by rigid links) taking into account only that the end effector should move to a desired position. Although similar to FABRIK in its basic structure, the FTL algorithm has not been extended to support joint constraints and orientations (these are largely superfluous in rope simulation), nor has it been applied to cases where multiple end effectors exist.

Inverse Kinematics is a method for manipulating articulated figures in an interactive and intuitive fashion for the design and control of their postures. However, in order to apply IK techniques for motion and body reconstruction, a joint localisation approach is needed to generate the CoRs and automatically establish legged skeletons from optical motion capture data.

Joint Localisation review: Many papers have focused on methods for localisation of the CoR. *Sphere fitting* approaches are the most commonly used methods for calculating the CoR. This group of methods assumes that all markers remain a constant distance from the centre of rotation. In [SPB$^+$98], the Levenberg-Marquardt method is used to optimise the CoR location and the radii of the marker spheres, via a cost function which sums a per marker cost over all markers and all frames. Halvorsen et al. describe a closed form solution using the geometric properties of the sphere [HLL99]. In [GL02], Gamage and Lasenby also introduce a closed form solution, using a cost function of the squared differences in the squared distance from the CoR to a marker and the radius of the sphere associated with that marker. An alternative approach provided by Halvorsen, [Hal03], gives a Bayesian analysis of the algorithm of [GL02], providing a first order approximation of the effect of isotropic Gaussian noise upon the algorithm.

Another group of methods is that termed *Transformational techniques*; they assume that markers are rigidly attached to limb segments. Such an approach was implemented in [Hol91], [OBBH00] and [ETDH06], where the limb orientation was obtained from sets of optical markers. In [CL05], a sequential algorithm was presented to locate the rotation centres of a human skeleton from marker data assuming that all markers are attached to a rigid body. The method is closed form, thus enabling real-time implementation.

Marker prediction and CoR estimation review: Whilst several methods to estimate the location of missing markers have been proposed, the performance of most is unsatisfactory in the presence of unusual motions or of many contiguous occlusion-affected frames. Indeed, there are instances, even with expensive motion capture systems, where occlusion of markers by elements of the scene leads to missing data. Several methods have been proposed to predict the occluded markers in order to drive CoR estimation and skeletal reconstruction. Interpolation of the data using linear or non-linear approaches is commonly used [WH97, RCB98]; this can produce accurate results, but it is a post-processing technique requiring data prior to and after the occlusion. Recently, [PLH$^+$09] presented an extrapolation algorithm which assumes that the most common motion behaviors are circular or linear movements; however, this method can produce reliable predictions only for a limited number of occluded frames. While the discarding of frames with missing markers is another common technique, the omission of specific data could lead to the loss of useful information. Long-running occlusions leading to a large sequence of missing data can also cause complete failure of the system.

In [RM06], Rhijn and Mulder proposed a model-based optical tracking and model estimation system for composite interaction devices; however, it is an off-line procedure unsuitable for real-time applications. Dorfmüller in [DU03] used an extended Kalman filter (EKF) to predict the missing markers using previously available marker information, while Welch et al. in [WBV$^+$99] used an EKF to resolve occlusions based on the skeletal model of the tracked person. Tak and Ko, [TK05], employed an Unscented Kalman Filter to ensure motion capture data remains in a feasible set. These methods require manual intervention or become ineffective in cases where markers are missing for an extended period of time. In our earlier work in [ACL08], we presented an EKF method using a constant velocity (CV) model with marker constraints from neighbouring[1] markers. However, the CV model (2nd order Kalman Filter (KF)) limits its use to problems with constant marker velocity. These methods also do not take into consideration the fact that bones are rigid, thus the inter-joint pairwise distances remain constant over time.

Herda et al., in [HFP$^+$00] and [HFP$^+$01], used a post-processing approach to increase the robustness of a motion capture system by using a sophisticated human model. The neighbouring markers that share kinematic relations with the occluded markers were used to help the estimation of the missing markers. However, the skeleton information must be known a priori in order to apply this method. [HSD05] also takes advantage of the fact that the markers on a limb have fixed inter-marker pairwise distances. This approach may become ineffective when all or a significant number of markers are missing so that no information on that limb can be inferred from the available neighbouring markers. Ringer and Lasenby, [RL02], also present an automatic method to identify indistinguishable markers based on cliques[2]. However, this requires an off-line procedure in order to determine marker cliques and parameters of the skeletal structure.

In [GMHP04], a style-based inverse kinematic method has been developed where a Gaussian

[1] *Neighbours* are the markers belonging to the same limb segment.
[2] Markers in a *clique* have constant distances between each other.

Process Latent Variable Model (GPLVM) was used along with a pre-specified kinematic model. Although it is a real-time processing method, it requires knowledge of skeleton information, which severely restricts its use. Chai and Hodgins, [CH05], present a method that uses the neighbouring markers to estimate the missing marker in the current frame. They propose a local linear model from these neighbours and then reconstruct the full pose of the frame by conducting an optimisation in the space constrained by the model. Yu et al., in [YLD07], proposed an online motion capture labelling approach which also recovers missing markers. They cluster the markers into a number of rigid bodies based on the standard deviations of the marker-pair distances. If their fitting-rigid-bodies algorithm did not classify all the markers into rigid bodies, a missing marker auto-recovery method is applied assuming that the inter-marker distances are fixed over time. However, a training session is needed for the fitting-rigid-bodies algorithm, the auto-recovery method for marker estimation does not take into account the limb segment rotation, no information about markers visible to a single camera is considered and the CoR estimation is not investigated under marker occlusions.

Recently, Liu et al., in [LZWM06, LM06], presented a piecewise linear approach for estimating human motions from a pre-selected set of informative markers (principal markers). A pre-trained classifier identifies an appropriate local linear model for each frame. Missing markers are then recovered using available marker positions and the principal components of the associated model. However, this data-driven family of methods requires an off-line training procedure and the results are highly dependent on training data and limited to those models and movements the system has been trained on. An extended literature review of these methods is also given in 4.3.

Hand tracking and reconstruction review: There are many approaches for tracking and configuring the hand model. The hand gesture identification algorithms can be classified into 2 major classes: glove-based and vision-based methods. In general, glove-based methods are real-time, however they are expensive (e.g. P5 Data glove) and only detect limited finger movements with low accuracy. Wang and Popovic [WP09] and Fredriksson et al [FRF08] proposed methods for hand tracking using a single camera and an ordinary cloth glove which was imprinted with a custom pattern; while this offers a simple, computationally cheap and promising solution, it is still not as reliable as the optical mocap systems. The vision-based methods, on the other hand, are more accurate but they have problems with occlusions, noise and spurious data. Lien and Huang [LH98] proposed a hand model together with a closed-form Inverse Kinematics solution for the finger fitting process. The 3D positions of the markers were obtained using colour markers and stereo vision, and the finger poses were chosen using a search method which finds the best solution amongst all possible positions. While this method is implementable in real-time, it is complex and can fail when different size models are used. De la Gorce et al [GPF08] also proposed a 3D hand tracking approach from monocular video. Stenger et al [SMC01, STTC06] proposed statistical methods, such as an Unscented Kalman Filter and a Hierarchical Bayesian Filter, to track hand motion. Such methods are still far from real-time thus limiting their use. Kaimakis and Lasenby [KL07] used a set of pre-calibrated cameras to extract the hand's silhouette as a visual cue. The 2D silhouette data is then modelled as a conic field and physiological constraints are imposed to improve the reliability of the hand tracking [KL09]. A more detailed literature review of hand tracking and reconstruction methods is given in section 5.2.

1.3. Outline of the Thesis

The body of this thesis may be broadly divided into 4 main parts, each of which is described in chapters 2 to 5. The first part describes the mathematical and experimental framework. Within this first part, chapter 2 gives a brief introduction to Geometric Algebra (GA) and to those aspects of GA that have been used within this thesis. We briefly outline Conformal Geometric Algebra (CGA) and examine how the Conformal Model can represent geometric primitives such as points, lines, circles, planes and spheres.

The second part introduces the Inverse Kinematics (IK) problem and discusses the most popular recent IK techniques. It briefly describes the advantages and disadvantages of each family of methods including Inverse Jacobian methods, Newton methods, Sequential Monte Carlo Methods and heuristic methods. Many of the currently available methods suffer from high computational cost and production of unrealistic poses. Chapter 3 presents a novel heuristic approach, called Forward And Backward Reaching Inverse Kinematics (FABRIK), which solves the IK problem in an iterative fashion. 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 and has low computational cost while producing visually realistic poses. FABRIK has been compared against the most popular manipulator solvers under several conditions, 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 needed.

The third part concerns the problem of fitting skeletal models to marker-based optical motion capture data. Section 4.4, studies the problem of estimating the centres of rotation (CoR) between every pair of limb segments and identifying the optimal skeleton in real-time. We present an efficient and accurate method for finding the rotors between 2 sets of vectors, and then we calculate the CoR by taking advantage of the approximation that all markers on a segment are attached to a rigid body. Part 3 also presents an integrated framework which predicts the occluded marker position and thereby maintains a continuous flow of data. Marker occlusion is a common phenomenon in motion capture systems due to camera system failure or marker occlusion by other limbs. In detail, section 4.2 describes the experimental environment including the cameras used for the experiments, the association between the markers and the limbs, and the clustering of markers into groups corresponding to the limb segments to which they are attached. Section 4.5 presents a real-time predicting approach using a variable turn model within an Unscented Kalman filter, in combination with inferred information from neighbouring markers in order to fill in the missing data when markers are occluded. This approach takes advantage of the fact that markers located on the same limb of an articulated body have constant inter-marker distance and presents marker prediction solutions under 5 different scenarios; the proposed marker constraints are efficient, simple and real-time implementable. This work also imposes the common case that missing markers are often visible to only a single camera, resulting in more accurate predictions. Section 4.7 describes how the FABRIK Inverse Kinematics solver can be incorporated into the marker prediction and centre of rotation problem, re-positioning the joint locations in such a manner that the inter-joint distances remain constant over time. Finally, section 4.8 shows comparisons of the proposed methodology against some of the most popular methods for marker prediction and the results confirm that our approach outperforms these methods in estimating both marker and CoR positions in terms of absolute error and computational cost.

The last part of the thesis describes a sophisticated hand model approach for real-time tracking and reconstruction. It presents a simple and efficient methodology for tracking and reconstructing 3D hand poses using a markered optical motion capture system. It is an experimental use of the IK solver presented in Chapter 3, and the marker prediction technique described in Chapter 4. Markers were positioned at strategic points, and FABRIK was incorporated to fit the rest of the joints to the hand model. The model is highly constrained with rotational and orientational constraints, allowing only natural finger motion. The hand model movements are also restricted by physiological constraints, allowing motion only to positions within an anatomically feasible set. In addition, a marker prediction system, similar to the one presented in Chapter 4, is applied to deal with cases where the markers are not visible to the motion capture cameras. The results were visualised using a mesh deformation algorithm, driving the animation of the hand according to the underlying hand skeleton. The method is real-time implementable and tracks the hand motion smoothly and without oscillations, even with a low capture frame rate.

In Chapter 6 we conclude with some final remarks and propose directions for future work. Finally, in the Appendices we give the Geometric Algebra derivations for most of the implemented methods.

2 Mathematical Background

Throughout this chapter, we present a brief overview of Geometric Algebra (GA) in order to provide explanations of those elements used within this thesis. GA provides a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. The Conformal model of GA give us the ability to describe algorithms in a geometrically intuitive and compact manner since basic entities, such as spheres, lines, planes and circles, are simply represented by algebraic objects. GA is also more numerically stable and more efficient than rotation matrices making it popular for applications in computer graphics and robotics. More detailed treatment of geometric algebra can be found in [DL03].

2.1. Geometric Algebra

Classical vector algebra has a number of problems when we move from three dimensional space (Euclidean) to higher dimensional space. Hence, Hermann Grassmann (1809-77) and William Clifford (1845-79), attempted to create an 'algebra of vectors' in order to generalise conventional vector algebra to higher dimensions. The modern day extension of this work is now known as 'Geometric Algebra'.

2.1.1. The products

The geometric product is the most fundamental product of Geometric Algebra. However, it is often useful first to define the *inner* and *outer products* for vectors and then to introduce the geometric product of vectors.

The inner product

A vector space alone is insufficient for describing Euclidean geometry as it lacks the concepts of distance and angles. Distances and angles are important in order to define entities like circles or perpendicular lines. Both can be defined through the introduction of a scalar product between vectors. This is known as the *inner product*, is written as $a \cdot b$ and returns a scalar. In Euclidean space the scalar product is always positive,

$$a^2 = a \cdot a > 0 \qquad \forall \;\; a \neq 0 \tag{2.1}$$

Thus, the length of a vector $|a|$ can be defined as:

$$|a| = \sqrt{(a \cdot a)} \tag{2.2}$$

Hence, the inner product between a and b is defined via

$$a \cdot b = |a||b| \cos(\theta) \tag{2.3}$$

where θ is the angle between the vectors. The inner product can be also defined for higher dimensional generalisations of the vector (multivectors).

The outer product

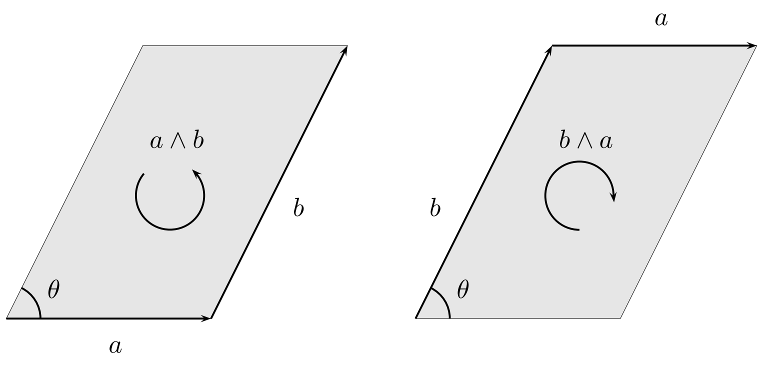


Figure 2.1.: *The outer product. The outer or wedge product of a and b returns a directed area element of area $|a||b|\sin(\theta)$.*

The major failure of the cross product is that exists only in 3 dimensions. In 2D there is nowhere else to go, whereas in more than 3 dimensions this direction is not uniquely defined. The solution of this problem was solved by Grassmann ([Gra62]) encoding a plane geometrically, without relying on the notion of a vector perpendicular to it. Grassmann introduced a key feature of GA namely the *outer* or *exterior product*, which is written as $a \wedge b$. Unlike the cross product, which results in a perpendicular vector, the outer product of a pair of vectors results in a directed area, as shown in figure 2.1. This is the directed area swept out by a and

b and can be visualised as the parallelogram obtained by sweeping one vector along the other. Changing the order of the vectors reverses the orientation of the plane. The plane has area $|a||b|\sin(\theta)$, which is defined to be the magnitude of $a \wedge b$.

The outer product has the following key properties:

- The outer product of vectors is antisymmetric

$$a \wedge b = -b \wedge a \tag{2.4}$$

  It follows that $a \wedge a = 0$.

- Also the outer product is distributive over addition

$$a \wedge (b + c) = a \wedge b + a \wedge c \tag{2.5}$$

- and associative

$$a \wedge (b \wedge c) = (a \wedge b) \wedge c = a \wedge b \wedge c \tag{2.6}$$

- The outer product of a vector and a bivector defines a trivector that is an oriented volume.

- Although the formation of a bivector is often illustrated as the result of sweeping one vector along a second to form a parallelogram, the use of any particular shape is misleading as it is easy to show that the outer product of many different pairs of vectors will result in the same bivector.

The geometric product

William Clifford (1845-1879) made the next step and, investigating the work of Grassmann, he turned GA to a useful algebra. Clifford introduced the key feature of GA, the *geometric product*, in order to define a product that could identify the roles of the terms in a complex product. The geometric product can be expressed in terms of the inner and outer products, and it is defined as:

$$ab = a \cdot b + a \wedge b \tag{2.7}$$

The result of the geometric product seems strange, having two components lying in two different spaces, a scalar $(a \cdot b)$ and a bivector $(a \wedge b)$. This combination is referred to as a *multivector* and is analogous to complex numbers where real and imaginary numbers are combined to form the complex number.

Since,

$$ba = b \cdot a + b \wedge a = a \cdot b - a \wedge b \tag{2.8}$$

we can define the inner and outer product in terms of the symmetric and antisymmetric parts of the geometric product. Thus, for vectors a and b

$$\begin{aligned} a \cdot b &= \frac{1}{2}(ab + ba) \\ a \wedge b &= \frac{1}{2}(ab - ba) \end{aligned} \tag{2.9}$$

Then, the geometric product can be extended to an arbitrary number of vectors with the following properties:

- The geometric product is associative

$$(ab)c = a(bc) = abc \tag{2.10}$$

- The geometric product is distributive over addition

$$a(b + c) = ab + ac \tag{2.11}$$

- The symmetric part of the geometric product of two vectors is a scalar
- The geometric product has a unique inverse

$$ab\left(\frac{b}{b^2}\right) = a \tag{2.12}$$

  hence, multiplying by $\frac{b}{b^2}$ is the inverse of multiplying by b.

The geometric product can also be used to define the inner and outer product of elements with single grades, A_r and B_s, where r and s denote the grade of the vector. Such elements are called blades[1].

$$A_r \cdot B_s = \langle A_r B_s \rangle_{|r-s|} \qquad \text{if } r, s > 0 \tag{2.13}$$

$$A_r \cdot B_s = 0 \qquad \text{if } r = 0 \text{ or } s = 0 \tag{2.14}$$

$$A_r \wedge B_s = \langle A_r B_s \rangle_{r+s} \tag{2.15}$$

where $\langle\rangle_t$ denotes the grade extraction operator, which sets all multivector components of grade other than t to zero. For the extraction of the scalar part of a multivector, the subscript 0 is usually dropped, and it is formulated as $\langle\rangle$ instead of $\langle\rangle_0$.

2.1.2. Operators

Geometric Algebra also has some other useful operators and elements. The most important of these are the *reverse* of a multivector, the *dual* and the *pseudoscalar*.

[1]The term *blade* in GA is used to refer to quantities that can be written as the outer product of r-vectors, where r is termed the *grade*. For example, an r-blade can always be written as $A_1 \wedge A_2 \wedge \ldots \wedge A_r$. More information about grades and the new orthonormal basis for computation can be found in section 2.1.3.

Reversion

Reversion substantially refers to the reversing of the order of any set of vectors that can be used to define a multivector, and it is symbolised as $\tilde{}$. A general example of reversion for the case of a 3D GA is given below:

$$\begin{aligned} M &= \alpha + a + B + \beta I \\ \tilde{M} &= \alpha + a - B - \beta I \end{aligned} \tag{2.16}$$

where M is multivector, α is a scalar, a is a vector, B is a bivector and βI is a trivector. Using the fact that $a \wedge b = -b \wedge a$, the reversion operator can be expressed by changing the sign of some particular grades of elements (bivectors and trivectors in 3D) within the multivector.

The pseudoscalar

The pseudoscalar refers to the highest grade basis element and it is symbolised as I_n, where n is the dimension of the space. For instance, in 3D GA the pseudoscalar can be written as:

$$I_3 = e_1 e_2 e_3 \tag{2.17}$$

where e_1, e_2, e_3 are the orthonormal basis vectors. The reversion of I_3 is equal to

$$\tilde{I}_3 = e_3 e_2 e_1 = -e_1 e_2 e_3 = -I_3 \tag{2.18}$$

The pseudoscalar is also denoted as a simple I, because its grade is always the highest and it is unnecessary to repeat it.

Since the pseudoscalar I is the unique right-handed unit trivector in the algebra, it gives us a number of new products. When we take the product of I with the vector e_i, we produce a bivector, eg.

$$Ie_1 = e_1 e_2 e_3 e_1 = -e_1 e_2 e_1 e_3 = e_2 e_3 \tag{2.19}$$

$$e_1 I = e_1 e_1 e_2 e_3 = e_2 e_3 = I e_1 \tag{2.20}$$

Vectors therefore commute with the 3D pseudoscalar,

$$Ia = aI, \qquad \text{for all } a \tag{2.21}$$

Hence, all the basis bivectors can be expressed as the product of the pseudoscalar and a *dual* vector.

$$\begin{aligned} Ie_1 &= e_1 e_2 e_3 e_1 = e_2 e_3 \\ Ie_2 &= e_1 e_2 e_3 e_2 = -e_1 e_3 = e_3 e_1 \\ Ie_3 &= e_1 e_2 e_3 e_3 = e_1 e_2 \end{aligned}$$

The square of the pseudoscalar is equal to -1:

$$I^2 = e_1e_2e_3e_1e_2e_3 = e_1e_2e_3e_3e_1e_2 = e_1e_2e_1e_2 = -1 \tag{2.22}$$

Duality

Duality is another frequently used concept of Geometric Algebra. The dual transformation is denoted as $[\,]^*$ and we map into the dual space by multiplying with the pseudoscalar as follows

$$[M]^* = MI \tag{2.23}$$

2.1.3. The orthonormal basis for computation

Although many results in GA can be reached without resorting to a basis, it is helpful to have a basis specifying the numerical multivectors for the 3D GA. Hence, the multivector basis can be defined via outer products of the three orthonormal basis vectors e_1, e_2 and e_3. The outer product is usually visualised geometrically as the movement of one vector along the other to form a 'directed area'. This is a new object, neither a vector nor a scalar. It is termed a *bivector*. Similarly, the result of the outer product of this bivector and another vector is a *trivector* (as already seen in section 2.1.2). Generally, an n-volume is termed an n-vector. Table 2.1 presents the multivector orthonormal basis.

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:	$e_{123} = e_1 \wedge e_2 \wedge e_3$		

We can say that a scalar is grade 0, a vector is grade 1, a bivector is grade 2, as it is formed from 2 vectors, etc. Generally, an n-vector has grade n. Figure 2.2 illustrates an example of a trivector (sweeping $a \wedge b$ along c).

2.1.4. Rotation using rotors

Consider any three orthonormal basis vectors of $\mathbb{R}^3$, $\{e_1, e_2, e_3\}$, then the new three basis bivectors generated are $B_1 = e_2e_3$, $B_2 = e_3e_1$ and $B_3 = e_1e_2$. The basis bivectors all square to -1, and all anticommute as given below:

$$B^2 = e_ie_je_ie_j = -e_ie_je_je_i = -1 \quad \text{for } i \neq j$$

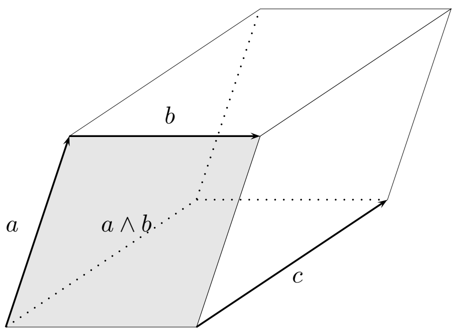


Figure 2.2.: *The trivector or directed volume is the result of sweeping $a \wedge b$ along c.*

An important feature of the bivectors is their effect on vectors. For example, the effect of the bivector B_3 on the vectors e_1 and $e_1 + e_2$ is to rotate them counter-clockwise by 90 degrees. This example is illustrated in figure 2.3.

$$\begin{aligned} e_1 B_3 &= e_1 e_1 e_2 = e_2 \\ e_2 B_3 &= e_2 e_1 e_2 = -e_2 e_2 e_1 = -e_1 \\ (e_1 + e_2) B_3 &= e_1 B_3 + e_2 B_3 = e_2 - e_1 \end{aligned}$$

This is applicable for every bivector $e_i e_j$ over the plane defined by e_i and e_j. It is important here to mention that this method also works in higher-dimension spaces.

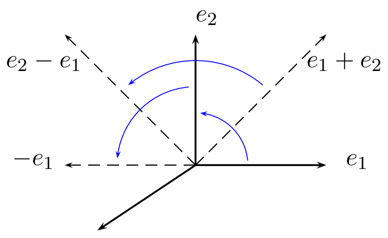


Figure 2.3.: *The rotation effect of bivector B_3 over vectors e_1 and $e_1 + e_2$.*

Thus, any vector a in the plane defined by e_1 and e_2 can be represented using the formula:

$$\begin{aligned} a &= r (e_1 \cos\theta + e_2 \sin\theta) \\ &= e_1 r (\cos\theta + B_3 \sin\theta) \end{aligned} \tag{2.24}$$

where r is the distance of the point a from the origin and θ is the angle between a and e_1. This can be generalised if, instead of B_3, the unit bivector $\hat{B} = \frac{a \wedge b}{|a \wedge b|}$ is used. Hence, an operator which performs rotation in the plane described by $\hat{B}$ by an angle 2θ, can be expressed as:

$$\begin{aligned} R &= \exp\left(\theta \hat{B}\right) \\ &= 1 + \frac{\theta \hat{B}}{1!} + \frac{\theta^2 \hat{B}^2}{2!} + \frac{\theta^3 \hat{B}^3}{3!} + \dots \\ &= \cos\theta + \hat{B} \sin\theta \end{aligned} \tag{2.25}$$

Then, any vector a which lies in the plane of the bivector $\hat{B}$ can be represented by:

$$a = e_1 r \exp\left(\theta \hat{B}\right) \tag{2.26}$$

Using this, it can be shown that a rotation by ϕ radians in the plane of $\hat{B}$ is accomplished by:

$$a \mapsto a' = a \exp\left(\phi \hat{B}\right) = a\left(\cos(\phi) + \hat{B}\sin(\phi)\right) \tag{2.27}$$

It is important to mention that here the vector to be rotated a lies in the plane of rotation. In a different case, the vector should be decomposed into a component that lies in the plane, $a_{\|}$, and one normal to the plane, $a_{\perp}$.

$$a = a_{\|} + a_{\perp} \tag{2.28}$$

Now, assuming that $R = \exp\left(-\frac{\phi}{2}\hat{B}\right)$ and $\tilde{R} = \exp\left(\frac{\phi}{2}\hat{B}\right)$, consider the following operation:

$$\begin{aligned} a' &= R\left(a_{\|} + a_{\perp}\right)\tilde{R} \\ &= \left(\cos\left(\frac{\phi}{2}\right) - \hat{B}\sin\left(\frac{\phi}{2}\right)\right)\left(a_{\|} + a_{\perp}\right)\left(\cos\left(\frac{\phi}{2}\right) + \hat{B}\sin\left(\frac{\phi}{2}\right)\right) \\ &= a_{\perp} + \left(\cos^2\left(\frac{\phi}{2}\right) - \sin^2\left(\frac{\phi}{2}\right)\right)a_{\|} + 2\cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\phi}{2}\right)\hat{B}a_{\|} \\ &= a_{\perp} + a_{\|}\left(\cos(\phi) + \hat{B}\sin(\phi)\right) \end{aligned} \tag{2.29}$$

Equation 2.29 relies on $\hat{B}a_{\perp} = a_{\perp}\hat{B}$ and $\hat{B}a_{\|} = -a_{\|}\hat{B}$, which states that a bivector commutes with a perpendicular vector and anticommutes with a parallel vector.

Thus, the component of the vector that lies in the plane is rotated around an axis normal to the plane without affecting the component that is normal to the plane. This leads to a general method of rotation in any plane. Therefore, a given rotation ϕ in a plane specified by $\hat{B}$ can be performed as:

$$a \mapsto Ra\tilde{R} \tag{2.30}$$

using the element $R = \exp\left(-\hat{B}\frac{\phi}{2}\right)$. R is referred to as a *rotor*.

The most important properties of rotors are that $R\tilde{R} = 1$ and thus $\tilde{R} = R^{-1}$, and $Ra\tilde{R} = (-R)a(-\tilde{R})$, which means that there is a double covering of the space of rotations.

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.

2.1.5. Reflection

Consider a reflection of the vector a in the plane orthogonal to a unit 3-vector m, where $m^2 = 1$.

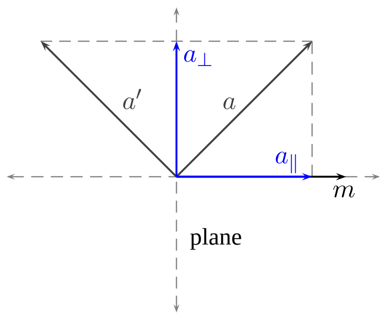


Figure 2.4.: *A reflection of the vector a in a plane perpendicular to m. It is obvious that* $a = a_\perp + a_{||}$ *and* $a' = a_\perp - a_{||}$

The component of a that is parallel to the plane m_r changes sign, whereas the perpendicular components remains unaffected, as shown in figure 2.4. Hence, the parallel component is the projection onto m and the perpendicular component is the remainder.

$$a_{||} = (a \cdot m)m \tag{2.31}$$

$$a_\perp = a - a \cdot mm = (am - a \cdot m)m = (a \wedge m)m \tag{2.32}$$

The result of the reflection is therefore:

$$\begin{aligned} a' &= a_\perp - a_{||} = -a \cdot mm + a \wedge mm \\ &= -(m \cdot a + m \wedge a)m = -mam \end{aligned} \tag{2.33}$$

This can be extended to higher dimensions as given in [HS84].

2.2. Conformal Geometric Algebra

This section is an introduction to the Conformal Model of Geometric Algebra (CGA) as first introduced by Hestenes and Sobczyk in 1984 [HS84]. CGA is a mathematical framework that offers a compact and geometrically intuitive formulation of algorithms and an easy and immediate computation of rotors; it is thus suitable for applications in engineering, computer vision and robotics.

CGA extends 3D Euclidean space to 5 dimensions by adding two extra basis vectors and thereby providing tools with which to represent and manipulate geometry. The additional basis vectors e and $\bar{e}$ are added to the 3 existing basis vectors e_1, e_2 and e_3 of the 3D GA. e and $\bar{e}$ have opposite signatures;

$$e^2 = +1 \qquad\qquad \bar{e}^2 = -1 \tag{2.34}$$

The new basis vectors thus provide a space that allows *null vectors*, n and $\bar{n}$, which are defined as:

$$n = e + \bar{e} \qquad\qquad \bar{n} = e - \bar{e} \tag{2.35}$$

where n is associated with the point at infinity and $\bar{n}$ with the origin. It is very easy to prove that these vectors are null since:

$$\begin{aligned} n^2 &= (e + \bar{e}) \cdot (e + \bar{e}) \\ &= e^2 + 2(e \cdot \bar{e}) + \bar{e}^2 \\ &= 1 + 0 - 1 = 0 \end{aligned} \tag{2.36}$$

and

$$\begin{aligned} \bar{n}^2 &= (e - \bar{e}) \cdot (e - \bar{e}) \\ &= e^2 - 2(e \cdot \bar{e}) + \bar{e}^2 \\ &= 1 - 0 - 1 = 0 \end{aligned} \tag{2.37}$$

Two basic identities for n and $\bar{n}$ are:

$$n \cdot \bar{n} = (e + \bar{e}) \cdot (e - \bar{e}) = e^2 - \bar{e}^2 = 2 \tag{2.38}$$

$$x \cdot n = x \cdot \bar{n} = 0 \tag{2.39}$$

where $x \in \mathbb{R}^3$. Another observation is that the squared bivector E^2, where $E = n \wedge \bar{n}$ is equal to 4;

$$\begin{aligned} E^2 &= (n \wedge \bar{n}) \cdot (n \wedge \bar{n}) \\ &= (n \cdot \bar{n})(n \cdot \bar{n}) - n^2\bar{n}^2 \\ &= 4 \end{aligned} \tag{2.40}$$

since $n \cdot \bar{n} = 2$ and $n^2 = \bar{n}^2 = 0$ and using the identity $(a \wedge b) \cdot (c \wedge d) = -(a \cdot b)(b \cdot d) + (a \cdot d)(b \cdot c)$.

The Hestenes' mapping

The mapping used in Hestenes' illustration ([HS84], page 302) is used to take a spatial (conventional 3D) vector to the equivalent CGA representation. Thus,

$$H(x) = \frac{1}{2}(x - e)\, n\, (x - e) \tag{2.41}$$

where x is a 3D Euclidean vector. Substituting for $n = e + \bar{e}$ and using the fact that $\bar{e} \cdot x = \bar{e} \cdot e = n \cdot x = 0$, it is possible to rewrite the equation in terms of null vectors as:

$$H(x) = \frac{1}{2}\left(x^2 n + 2x - \bar{n}\right) \tag{2.42}$$

We can show that $H(x)$ is always a null vector by evaluating:

$$\begin{aligned}[H(x)]^2 &= \frac{1}{4}(x^2n + 2x - \bar{n}) \cdot (x^2n + 2x - \bar{n}) \\ &= -\frac{1}{2}x^2 n \cdot \bar{n} + x^2 \\ &= -x^2 + x^2 = 0 \end{aligned} \tag{2.43}$$

as $\bar{e} \cdot x = \bar{e} \cdot e = n \cdot x = 0$. Another interesting observation is that, for any vector A and $B \in \mathbb{R}^5$ which represent the points a and b in $\mathbb{R}^4$, $A \cdot B$ is related to the Euclidean distance between the points a and b, as proved in equation 2.44.

$$\begin{aligned}A \cdot B &= H(a) \cdot H(b) \\ &= \frac{1}{4}(a^2n + 2a - \bar{n}) \cdot (b^2n + 2b - \bar{n}) \\ &= -\frac{1}{2}a^2 + a \cdot b - \frac{1}{2}b^2 \\ &= -\frac{1}{2}(a - b)^2 \end{aligned} \tag{2.44}$$

Hence, the Euclidean distance between the two points a and b can be defined as:

$$d\,(A, B) = \sqrt{-2\,(A \cdot B)} \tag{2.45}$$

More information about *distance geometry* can be found in [Hes01], [HS84] and [DH93].

2.2.1. Rotation

In Conformal Geometric Algebra rotations are also performed with the rotor elements as in usual Geometric Algebra (see equation 2.25). Thus, given that $R\bar{n}\tilde{R} = \bar{n}$ and $Rn\tilde{R} = n$, $x \mapsto Rx\tilde{R}$ can be replaced by $H(x) \mapsto H(Rx\tilde{R})$, as is shown below.

$$\begin{aligned}RH(x)\tilde{R} &= R\frac{1}{2}\left(x^2n + 2x - \bar{n}\right)\tilde{R} \\ &= \frac{1}{2}\left(x^2Rn\tilde{R} + 2Rx\tilde{R} - R\bar{n}\tilde{R}\right) \\ &= \frac{1}{2}\left(x^2n + 2Rx\tilde{R} - \bar{n}\right) \\ &= H(Rx\tilde{R}) \end{aligned} \tag{2.46}$$

2.2.2. Translation

The translation along a vector a is defined as the mapping $x \mapsto x + a$ for x. Here we show that this can be performed by applying a rotor $R = T_a = \exp(\frac{na}{2})$. Using the power series expansion of the exponential, the rotor can be simplified to:

$$T = \exp\left(\frac{na}{2}\right) = 1 + \frac{na}{2} + \frac{1}{2}\left(\frac{na}{2}\right)^2 + \ldots = 1 + \frac{na}{2} \tag{2.47}$$

since n is null, $an = -na$ and therefore the higher order terms are all zero.

The rotor T acts over the vectors n, $\bar{n}$ and x as below:

$$\begin{aligned} Tn\tilde{T} &= \left(1+\frac{na}{2}\right) n \left(1+\frac{an}{2}\right) \\ &= n + nan + \frac{nanan}{4} = n \end{aligned} \tag{2.48}$$

Similarly,

$$\begin{aligned} T\bar{n}\tilde{T} &= \left(1+\frac{na}{2}\right) \bar{n} \left(1+\frac{an}{2}\right) \\ &= \bar{n} + \frac{na}{2}\bar{n} + \bar{n}\frac{an}{2} + \frac{na}{2}\bar{n}\frac{an}{2} \\ &= \bar{n} - 2a - a^2 n \end{aligned} \tag{2.49}$$

and

$$\begin{aligned} Tx\tilde{T} &= \left(1+\frac{na}{2}\right) x \left(1+\frac{an}{2}\right) \\ &= x + n(a \cdot x) \end{aligned} \tag{2.50}$$

Therefore, the translation over the null vector $H(x)$ will be:

$$\begin{aligned} TH(x)\tilde{T} &= \left(1+\frac{na}{2}\right) \frac{1}{2}\left(x^2 n + 2x - \bar{n}\right) \left(1+\frac{an}{2}\right) \\ &= \frac{1}{2}\left(x^2 n + 2(x + n(a \cdot x)) - (\bar{n} - 2a - a^2 n)\right) \\ &= \frac{1}{2}\left((x+a)^2 n + 2(x+a) - \bar{n}\right) \\ &= H(x+a) \end{aligned} \tag{2.51}$$

To summarise, the $x \mapsto x + a$ can be replaced by $H(x) \mapsto TH(x)\tilde{T} = H(x + a)$.

2.2.3. Dilation

A dilation by a factor of α can be represented by the mapping $x \mapsto \alpha x$. This can be achieved by considering the rotor $R = D_\alpha = \exp\left(\frac{\alpha}{2}e\bar{e}\right)$. Since $-e\bar{e}n = ne\bar{e} = n$ and $-\bar{n}e\bar{e} = e\bar{e}\bar{n} = \bar{n}$ and by expanding $\exp\left(\frac{\alpha}{2}e\bar{e}\right)$, the dilation of the vector $H(x)$ by a factor of $\exp(-\alpha)$ about the origin returns:

$$\begin{aligned} DH(x)\tilde{D} &= \exp\left(\frac{\alpha}{2}e\bar{e}\right) \frac{1}{2}\left(x^2 n + 2x - \bar{n}\right) \exp\left(-\frac{\alpha}{2}e\bar{e}\right) \\ &= \frac{1}{2}\left(x^2 \exp\left(\frac{\alpha}{2}e\bar{e}\right) n + 2x - \exp\left(\frac{\alpha}{2}e\bar{e}\right)\bar{n}\right) \\ &= \frac{1}{2}\left(x^2 \exp\left(-\alpha\right) n + 2x - \exp\left(\alpha\right)\bar{n}\right) \\ &= \exp(\alpha)\frac{1}{2}\left(\exp(-2\alpha)x^2 n + 2\exp(-\alpha)x - \bar{n}\right) \\ &= H(\exp(-\alpha)x) \end{aligned} \tag{2.52}$$

Hence, the dilation $x \mapsto \exp(-\alpha)x$ can be represented by

$$H(x) \mapsto DH(x)\tilde{D} = \exp(\alpha)H(\exp(-\alpha)x) \tag{2.53}$$

2.2.4. Inversion

Inversion in the origin corresponds to the mapping $x \mapsto \frac{x}{x^2}$ or for non-singular vectors $x \mapsto x^{-1}$. This can be performed by reflecting in e. The reflections in e of the vectors n, $\bar{n}$ and x are given by:

$$\begin{aligned}
-ene &= -ee\bar{n} = -\bar{n} \\
-e\bar{n}e &= -een = -n \\
-exe &= x
\end{aligned}$$

The inversion of $H(x)$ under the reflection in e then gives

$$\begin{aligned}
-eH(x)e &= -e\frac{1}{2}\left(x^2 n + 2x - \bar{n}\right)e \\
&= \frac{1}{2}\left(-x^2\bar{n} + 2x + n\right) \\
&= x^2\frac{1}{2}\left(\frac{1}{x^2}n + 2\frac{x}{x^2} - \bar{n}\right) \\
&= x^2 H\left(\frac{x}{x^2}\right)
\end{aligned} \tag{2.54}$$

Thus, the inversion $x \mapsto \frac{x}{x^2}$ is replaced by $H(x) \mapsto -\frac{eH(x)e}{x^2} = H(\frac{x}{x^2})$.

2.2.5. Blades in CGA

We have already seen that the term *blade* in GA is used to refer to quantities that can be written as the wedge product of vectors. For example, an r-blade can always be written as $A_1 \wedge A_2 \wedge \ldots \wedge A_r$, which differs from an r-vector that may be any linear combination of r-blades. Here we assumed that all null-vectors X are defined such that $X \cdot n = -1$, unless otherwise stated.

Vectors and 2-blades

A 2-blade is formed from the outer product $A \wedge B$ of two null vectors A and B. Considering the differentiation between signs of null vectors in CGA, many different separations are possible;

$$A \wedge B = -(B \wedge A) = -(B) \wedge A = B \wedge (-A) \tag{2.55}$$

The outer product $A \wedge B$ can be separated into a pair of individual null-vectors that are unique up to a scale, as shown below.

Extracting vector A and vector B from $A \wedge B$

We often need to extract the two vectors A and B from the bivector $A \wedge B$. This paragraph presents a solution of this problem using *projectors*. Assume that A and B are normalised so that $A \cdot n = -1$ and $B \cdot n = -1$. Let the 2-blade $T = A \wedge B$ and form

$$F = \frac{1}{\beta} A \wedge B \tag{2.56}$$

where $\beta > 0$ and $\beta^2 = T^2$, so that $F^2 = 1$ if $\beta^2 \neq 0$. Thus, two projector operators can be defined as:

$$\begin{aligned} P &= \frac{1}{2}(1 + F) \\ \tilde{P} &= \frac{1}{2}(1 - F) \end{aligned} \tag{2.57}$$

where $\tilde{P}$ denotes the normal reversion operation applied to P. Note that $PP = P$, which can be verified as follows

$$\begin{aligned} PP &= \frac{1}{4}(1 + F)(1 + F) \\ &= \frac{1}{4}(1 + 2F + 1) \\ &= \frac{1}{2}(1 + F) \end{aligned} \tag{2.58}$$

Similarly, it can be shown that $\tilde{P}\tilde{P} = \tilde{P}$. An equally important property is that $\tilde{P}P = P\tilde{P} = 0$ which, again, is easy to show

$$P\tilde{P} = \frac{1}{4}(1 + F)(1 - F) = \frac{1}{4}\left(1 - F^2\right) = \frac{1}{4}(1 - 1) = 0 \tag{2.59}$$

In the same way, it can be shown that $\tilde{P}P = 0$.

It is important here to show what effect these projectors have on A and B. Hence, P and $\tilde{P}$ acting on A and B will return the following results:

$$PA = 0, \qquad PB = B, \qquad \tilde{P}A = A, \qquad \tilde{P}B = 0,$$

This can be easily verified as below:

$$\begin{aligned} PA &= \frac{1}{2}\left[1 + \frac{1}{\beta} A \wedge B\right] A \\ &= \frac{1}{2}\left[A + \frac{1}{\beta}(A \wedge B) A\right] \\ &= \frac{1}{2}\left[A + \frac{1}{\beta}(A \cdot B) A\right] \\ &= \frac{1}{2}(A - A) = 0 \end{aligned} \tag{2.60}$$

since

$$\begin{aligned}(A \wedge B)\, A &= (A \wedge B) \cdot A \\ &= -A^2 B + (A \cdot B)\, A \\ &= (A \cdot B)\, A \text{ , because } A^2 = 0\end{aligned}$$

and $A \cdot B = -\beta$. This follows from equation 2.45, where $A \cdot B$ must be negative, the facts that $A^2 = B^2 = 0$ and also from:

$$\begin{aligned}\beta^2 &= (A \wedge B) \cdot (A \wedge B) \\ &= -A^2 B^2 + (A \cdot B)^2 \end{aligned} \tag{2.61}$$

Using similar manipulations, it can be shown that $PB = B$, $\tilde{P}A = A$ and $\tilde{P}B = B$. The next step is to consider the vector obtained by dotting $A \wedge B$ with n.

$$\begin{aligned}(A \wedge B) \cdot n &= -n \cdot (A \wedge B) \\ &= -\,(n \cdot A)\, B + (n \cdot B)\, A \\ &= (B - A)\end{aligned} \tag{2.62}$$

using the fact that A and B are normalised points such that $A \cdot n = B \cdot n = -1$. Thus, it follows that:

$$P\,[(A \wedge B) \cdot n] = P\,(B - A) = B \tag{2.63}$$

$$-\tilde{P}\,[(A \wedge B) \cdot n] = -\tilde{P}\,(B - A) = A \tag{2.64}$$

It is also noted that since $AP = \tilde{P}A = A$ it follows that $\tilde{P}AP = \tilde{P}\tilde{P}A = \tilde{P}A$. In that way we have:

$$\begin{aligned}&\tilde{P}AP = \tilde{P}A \\ &PA\tilde{P} = 0 \\ &PB\tilde{P} = PB \\ &\tilde{P}BP = 0\end{aligned}$$

which mean that the projectors can be written as two-sided operations. Hence, the two points A and B can be extracted from a 2-blade $A \wedge B$ as:

$$A = -\tilde{P}\,[(A \wedge B) \cdot n] \equiv -\tilde{P}\,[(A \wedge B) \cdot n]\, P = -\tilde{P}\,[\langle (A \wedge B)\, n \rangle_1] \tag{2.65}$$

$$B = P\,[(A \wedge B) \cdot n] \equiv P\,[(A \wedge B) \cdot n]\, \tilde{P} = P\,[\langle (A \wedge B)\, n \rangle_1] \tag{2.66}$$

The use of projectors ensures that one projector, P, will always return the first point, A, and the second projector, $\tilde{P}$, will return the second point, B. As a result, an orientation may

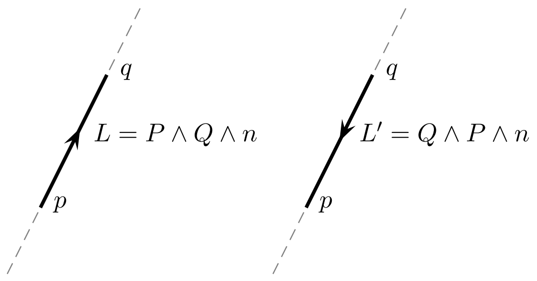


Figure 2.5.: *A line $L = P \wedge Q \wedge n$ passing through the null vectors P and Q and the 'opposite' line, $L' = Q \wedge P \wedge n$.*

be inferred for this point pair, hence one can be considered to occur 'before' the other. Note that, it is not possible to extract the null vector from a bivector having one of its points as n (e.g. $B = A \wedge n$) using the 2-blade projector-based separation method. One approach to get round this problem was proposed in [LLW04] and in [Cam07] with minor corrections. A bivector B may be in the form $A \wedge n$ or $n \wedge A$, where A is the null vector representing the spatial point a. Hence,

$$a = \frac{1}{4}\left(B \wedge \bar{n}\right).N \quad \Leftrightarrow n \text{ is the first component} \tag{2.67}$$

$$a = \frac{1}{4}\left(\bar{n} \wedge B\right).N \quad \Leftrightarrow n \text{ is the second component} \tag{2.68}$$

where $N = n \wedge \bar{n}$.

The usefulness of these results will become palpable later when intersections are considered (section 2.2.6).

3-Vectors

In this section we study trivectors. Consider the null vectors P, Q and R in the 5D space representing the points p, q and r respectively in 3D space .

Lines as trivectors In order to define a line with direction p to q, the trivector L needs to be introduced. The trivector L is formed as:

$$\begin{aligned} L &= P \wedge Q \wedge n \\ &= Q \wedge n \wedge P \\ &= n \wedge P \wedge Q \end{aligned} \tag{2.69}$$

and corresponds to a line that is passing through the points represented by the null vectors P and Q. It can be shown algebraically that these 3 lines return the same blade. However, the line $L' = Q \wedge P \wedge n = -L$ will have the opposite direction, differing only in sign, due to the anti-commuting nature of the outer product of the vectors. An example of the lines L and L' is given in figure 2.5.

Circles as trivectors A circle can be defined as a trivector C, where the point at infinity, n, is replaced by a third point on the circle, R. Hence, the equation of a circle in CGA is formed as:

$$C = P \wedge Q \wedge R \tag{2.70}$$

where C corresponds to a circle that is passing through the points represented by the null vectors P, Q and R, as shown in figure 2.6.

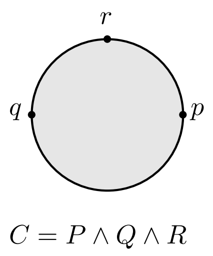


Figure 2.6.: *A circle $C = P \wedge Q \wedge R$ passing through the null vectors P, Q and R.*

The unit circle (figure 2.7) in the plane is formed as the circle passing through the three key points, e_1, $-e_1$ and e_2. Hence, for any unit length vector x, we know that $H(x) = \frac{1}{2}(n + 2x - \bar{n}) = (x + \bar{e})$. Thus, we have:

$$H(e_1) \wedge H(e_2) \wedge H(-e_1) = 2e_1e_1\bar{e}$$

and hence the trivector $C = 2e_1e_1\bar{e}$ represents the unit circle. The dual of the unit circle C^* is equal to

$$C^* = CI_4 = 2e = (n + \bar{n}) \tag{2.71}$$

where I_4 is the pseudoscalar given by $I_4 = e_1e_2e\bar{e}$.

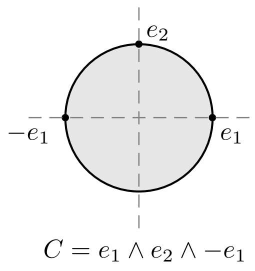


Figure 2.7.: *The unit circle.*

If X lies on the circle, we know that $X \wedge C = 0$. Since C^* is the dual of a trivector in a 4D space, it is a vector. Hence, an alternative but useful representation of the circle is given by

$$X \cdot C^* = 0$$

In the equation 2.44 we proved that for any two normalised point representations A and B,

$A \cdot B = -\frac{1}{2}(a-b)^2$. Thus, consider a point on circle, X, and the circle centre, B, then the radius ρ of the circle can be calculated as:

$$X \cdot B = -\frac{1}{2}(x-b)^2 \equiv -\frac{1}{2}\rho^2 \tag{2.72}$$

For a normalised point representation X this implies that

$$X \cdot \left(B - \frac{1}{2}\rho^2 n\right) = 0 \tag{2.73}$$

since $X \cdot n = -1$. Comparing this with $X \cdot C^*$ we see that our normalised C^* is

$$C^* = B - \frac{1}{2}\rho^2 n \tag{2.74}$$

Therefore, the vector C^* encodes in a neat fashion the centre and the radius of the circle in the plane. We also note that the radius of the circle can be easily calculated by squaring C^*

$$\begin{aligned}(C^*)^2 &= \left(B - \frac{1}{2}\rho^2 n\right)^2 \\ &= -\rho^2 B \cdot n = \rho^2\end{aligned} \tag{2.75}$$

since $B^2 = 0$, $n^2 = 0$ and $B \cdot n = -1$. Using equation 2.75 it is easy to show that:

$$B = C^* + \frac{1}{2}(C^*)^2 n \tag{2.76}$$

Note that the above relations assume that C^* is normalised such that $C^* \cdot n = -1$ since $C^* \cdot n = B \cdot n = -1$, as it is assumed that B is a normalised null vector.

However, there is a more elegant way of calculating the centre of a circle in 3D, as proved in [LLW04]. The centre of a circle, C, is also given by reflecting the point at infinity, n, in the circle as:

$$B = CnC \tag{2.77}$$

4-Vectors

This section studies 4-vectors. 4-vectors in 5D space reprsent 3-planes, Φ, and spheres, Σ.

Planes as 4-vectors A 3-plane, Φ, passing through the three points defined by null vectors P, Q and R is given by:

$$\Phi = P \wedge Q \wedge R \wedge n \tag{2.78}$$

Figure 2.8 illustrates an example of a 3-plane passing through the points p, q and r.

The physical quantities we might want to extract from such a 4-vector are clearly the normal to the plane and the perpendicular distance of the plane from the origin. This can be achieved

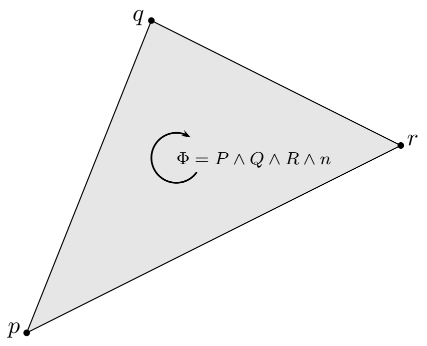


Figure 2.8.: *The 3-plane* $\Phi = P \wedge Q \wedge R \wedge n$.

using the dual of the plane. Consider the plane $z = d$, which is parallel to the xy-plane and distance d from it. The plane Φ can be represented as

$$\begin{aligned}\Phi &= F(de_3) \wedge F(e_1 + de_3) \wedge F(e_2 + de_3) \wedge n \\ &= \frac{1}{8}\{(2de_3 - \bar{n}) \wedge (2[e_1 + de_3] - \bar{n}) \wedge (2[e_2 + de_3] - \bar{n}) \wedge n\} \\ &= de_1 \wedge e_2 \wedge e_3 \wedge n - \frac{1}{2} e_1 \wedge e_2 \wedge \bar{n} \wedge n \\ &= de_1 \wedge e_2 \wedge e_3 \wedge n - e_1 \wedge e_2 \wedge e \wedge \bar{e} \qquad (2.79)\end{aligned}$$

Then it is simple to show that the dual of Φ is given by

$$\Phi^* = \Phi I = dn + e_3 \qquad (2.80)$$

This holds for any d. If the plane is rotated by R, such that $Re_3\tilde{R} = \hat{n}$, the general equation for the dual of the new plane $R\Phi^*\tilde{R}$ is given by

$$\Phi^* = dn + \hat{n} \qquad (2.81)$$

Note that the above assumes that the dual is normalised such that $[\Phi^*]^2 = 1$; this can be ensured by normalising the plane such that $\Phi^2 = 1$. Therefore, given 3 points on the plane, P, Q and R, we form the normalised plane Φ and its dual Φ^*, and we can then extract $\hat{n}$ and d as follows

$$d = \frac{1}{2}\Phi^* \cdot \bar{n} \qquad (2.82)$$

$$\hat{n} = \Phi^* - \frac{1}{2}\left(\Phi^* \cdot \bar{n}\right) n \qquad (2.83)$$

Spheres as 4-vectors Spheres can be defined as a 4-vector where the point at the infinity, n, is replaced by a point on the sphere. Hence, given any 4 points p, q, r and s whose 5D representations are the null vectors P, Q, R and S respectively, the sphere passing through those points is defined by the 4-vector $\Sigma = P \wedge Q \wedge R \wedge S$.

We know that $X \wedge \Sigma = 0$ for any X lying on the sphere. This can be rewritten as

$$X \cdot \Sigma^* = 0 \tag{2.84}$$

where Σ^* is the dual to Σ, hence is a vector. As for the circle solution, we can show that the dual representation of the sphere naturally encodes the centre and the radius of the sphere. Thus

$$X \cdot C = -\frac{1}{2}(x-c)^2 \equiv -\frac{1}{2}\rho^2 \tag{2.85}$$

where X is a point on a sphere, C is the centre of the sphere and ρ is the radius of the sphere. For a normalised point X, where $X \cdot n = -1$ this means that

$$X \cdot \left(C - \frac{1}{2}\rho^2 n\right) = 0 \tag{2.86}$$

Comparing equation 2.86 with equation 2.84, we can find that

$$\Sigma^* = C - \frac{1}{2}\rho^2 n \tag{2.87}$$

Thus, the dual vector Σ^* encodes the centre and radius of the sphere. The radius and the centre can immediately be calculated by squaring Σ^*

$$(\Sigma^*)^2 = \left(C - \frac{1}{2}\rho^2 n\right)^2 = \rho^2$$

since $C^2 = 0$, $n^2 = 0$ and $C \cdot n = -1$. Using this equation, we can easily show that:

$$C = \Sigma^* + \frac{1}{2}(\Sigma^*)^2 n \tag{2.88}$$

As for the circle case, the centre, C, of a sphere, Σ, is also given by reflecting the point at infinity, n, in the sphere:

$$C = \Sigma n \Sigma \tag{2.89}$$

Note that we have used the idea of reflection, developed in section 2.1.5, to reflect geometric objects in other geometric objects [LLW04].

5-Vectors

5-vectors, such as the pseudoscalar I_5, have two orientations and these correspond to Stolfi's concept [Sto91] of an oriented universe. The pseudoscalar is defined as $I_5 = e_1 \wedge e_2 \wedge e_3 \wedge e \wedge \bar{e} = e_{12345}$, and it satisfies $I_5^2 = -1$.

2.2.6. Intersections

This section outlines the various ways of intersecting objects within the conformal geometric algebra model. In this report, an operator termed *the meet* is used [HS84], denoted by the symbol $\vee$, which given two objects A and B returns their intersection. For instance, the meet of 2 entities defined by blades D_m and E_s of grades m and s, lying within an embedding entity of grade p is given by

$$D \vee E = \left[\langle DE \rangle_{2p-m-s}\right]^* \tag{2.90}$$

Intersecting lines with lines

Here we consider the intersection of two lines. Let the lines be L_1 and L_2. The meet of these two lines is given by

$$X = L_1 \vee L_2 = [\langle L_1 L_2 \rangle_{2n-r-s}]^* \tag{2.91}$$

where n denotes the dimension of the embedding space, r is the grade of the first line and s the grade of the second. Thus, $2n - r - s = 10 - 3 - 3 = 4$, so the dual object has grade 1. However, if the lines intersect at a point, the meet, X, will not return this intersection point. Instead, if the lines intersect, $L_1 \vee L_2 = 0$ and if the lines do not intersect, $L_1 \vee L_1 \propto n$. In order to find the intersection point, we use the following procedure. Assume that the lines L_1 and L_2 intersect at point P. Reflect the line L_1 in line L_2 and get $L_1' = L_2 L_1 L_2$. Then, find the line which is perpendicular to L_2, passing through the point P, which can be shown to be equal to $L_2^P = L_1 - L_2 L_1 L_2$ [LLW04]. Take any arbitrary point representation Y and reflect in L_2^P via $Y' = L_2^P Y L_2^P$. Then, take the midpoint of Y and Y', $Y'' = \frac{1}{2}(Y + Y')$, which must lie on the line L_2^P. Thereafter, reflect the point Y'' in line L_2 to give $Y''' = L_2 Y'' L_2$ and again take the midpoint $P' = \frac{1}{2}(Y'' + Y''')$. The representation of the intersection point can then be extracted via equation 2.92; figure 2.9 illustrates the above.

$$P = \frac{-(P' n P')}{2(P' \cdot n)^2} \tag{2.92}$$

Note, Y'' and P' as given here are not null vectors but are the null vectors representing the midpoints plus some multiples of n. In equation 2.92 these multiples of n are eliminated.

Intersecting circles with circles and lines

We now consider the intersection of two circles or a circle and a line. Firstly, we will focus on the meet of two circles, C_1 and C_2.

$$X = C_1 \vee C_2 = [\langle C_1 C_2 \rangle_{2n-r-s}]^* \tag{2.93}$$

where $2n-r-s = 10-3-3 = 4$, so that the dual object has grade 1. However, the intersection of two circles mostly returns two intersection points (when they lie in the same plane), and the

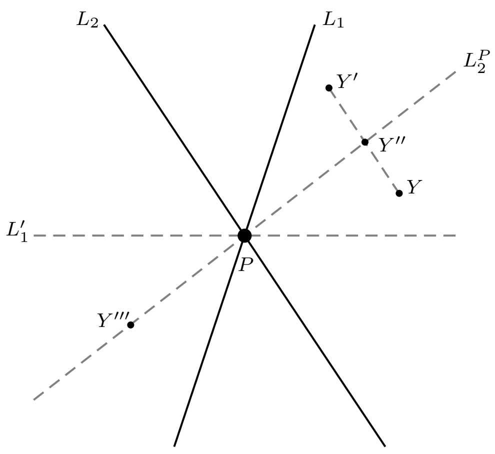


Figure 2.9.: *The solution for line to line intersection.*

1-grade object X can not give us the two points. In that case, the meet does not return the intersection points but just an object that helps us to conclude whether the circles intersect at two points, a point or if they do not intersect at all. Hence,

$$\begin{aligned}
&\text{if circles have two intersections,} \quad C_1 \vee C_2 = 0,\\
&\text{if circles have one intersection,} \quad C_1 \vee C_2 = X, \text{ where } X^2 = 0\\
&\text{if circles have no intersection,} \quad C_1 \vee C_2 = X, \text{ where } X^2 \neq 0
\end{aligned}$$

If $C_1 \vee C_2 = X$ and $X^2 = 0$, then it is obvious that the intersection point is represented by X. If the meet gives zero, there are two intersections, and these can easily be found by intersecting the plane of one of the circles with the other circle as:

$$B = C_1 \vee (C_2 \wedge n) = [\langle C_1(C_2 \wedge n)\rangle_{2n-r-s}]^* \tag{2.94}$$

where $2n - r - s = 10 - 3 - 4 = 3$, so that the dual object has grade 2. Thus, the two points can be extracted from the bivector B using the projectors given in equation 2.65.

Now, consider the intersection of a circle C_1 and a line L_1. The meet is again a grade 1 object, and it similarly holds that:

$$\begin{aligned}
&\text{if circle and line have two intersections,} \quad C_1 \vee L_1 = 0,\\
&\text{if circle and line have one intersection,} \quad C_1 \vee L_1 = X, \text{ where } X^2 = 0\\
&\text{if circle and line have no intersection,} \quad C_1 \vee L_1 = X, \text{ where } X^2 \neq 0
\end{aligned}$$

Equally, if $C_1 \vee L_1 = X$ and $X^2 = 0$, then the intersection point is represented by X and if the meet gives zero, the two intersections can be found by intersecting the plane of the circle with the line. It is also important to mention that, in the case where $C_1 \vee L_1 = X$ and $X^2 \neq 0$, which means that no intersection exist, the sign of X^2 tell us whether the line passes through the circle ($X^2 < 0$) or does not pass through the circle ($X^2 > 0$).

Intersecting planes with planes, circles and lines

This section studies the intersection between planes and planes, planes and circles, and planes and lines, Firstly, the plane to plane intersection will be studied. Consider two planes Φ_1 and Φ_2. The meet between two planes gives:

$$L = \Phi_1 \vee \Phi_2 = \left[\langle \Phi_1 \Phi_2 \rangle_{2n-r-s}\right]^* \tag{2.95}$$

where $2n - r - s = 10 - 4 - 4 = 2$. Therefore, the dual object has grade $5 - 2 = 3$, and represents a line. The sign of L^2 indicates whether the planes intersect. If $L^2 > 0$, then the planes intersect in the line L. On the other hand, when $L^2 = 0$, the intersection does not exist, which means that the two planes are parallel.

The intersection of a plane Φ_1 and a circle C_1, when this exists, is a pair of points, or a single point and it is formulated as:

$$B = \Phi_1 \vee C_1 = \left[\langle \Phi_1 C_1 \rangle_{2n-r-s}\right]^* \tag{2.96}$$

where $2n - r - s = 10 - 4 - 3 = 3$, and so the dual object has grade 2. Looking at the sign of the resulting 2-blade, B, it is possible to tell if the intersection exists and whether the intersection is a single point or a pair of points. Therefore, if $B^2 > 0$, then the meet between Φ_1 and C_1 gives two points. Given the bivector, B, the two points of the intersection can be extracted via the projectors given in equation 2.65. In the case where $B^2 = 0$, the intersection is a single point, X. It is then trivial to find the representation of that point using the formula $X = BnB$. Finally, when $B^2 < 0$ holds, the intersection between the plane and the circle does not exist.

Replacing the circle C_1 with a line L_1, the meet will still give us a 2-blade, B. However, a line and a plane intersect at most in one single position. Thus, the bivector will be of the form $B = X \wedge n$, where X is the representation of the point of the intersection. If $B^2 > 0$ the line and plane intersect in a point, if $B^2 = 0$ the line and plane are parallel and never intersect and if $B = 0$ the line lies in the plane.

Intersecting spheres with circles or lines

Now we consider, the intersection of a sphere, Σ_1 (4-blade), with a circle, C_1 (3-blade) or a single line, is the subject of this section. The intersection, where it exists, could be a single point or a pair of points (or the circle C_1 for the case where the circle is exactly on the outline of the sphere).

According to the meet formulation, the intersection of a sphere, Σ_1, with a circle, C_1 can be expressed as:

$$B = \Sigma_1 \vee C_1 = \left[\langle \Sigma_1 C1 \rangle_{2n-r-s}\right]^* \tag{2.97}$$

$2n - r - s = 10 - 4 - 3 = 3$, hence, the dual quantity will have grade $5 - 3 = 2$, which is a bivector, and represents the 2 intersecting points. Once again, the sign of the resulting squared 2-blade, B^2, gives us information on whether the intersection exists and if the sphere and circle intersect in a single point or a pair of points. In the case of two intersections the points can be extracted from B using the projectors, as before, and in the case of tangency, the single point of contact is obtained by taking BnB.

Similarly, substituting the circle C_1 with a line L_1 and intersecting with a sphere Σ_1, the meet again returns a 2-vector whose square denotes whether there are two, one or no intersection. The intersection points can be obtained easily in the same way as for the circle case.

Intersecting spheres with spheres or planes

Here we deal with the intersection of two spheres Σ_1 and Σ_2 or the intersection of a sphere Σ_1 and a plane Φ_1. This intersection, where it exists, is a circle (or a single point).

Firstly, we will consider the intersection between two spheres. Spheres do not intersect if the distance from their centres is less than the sum of their radii $d > (\rho_0 + \rho_1)$. Also, if $d < |\rho_1 - \rho_2|$, one of the two spheres is completely contained in the other, hence no intersection exists. If $d < |\rho_1 - \rho_2|$ and $\rho_1 < \rho_2$, then the first sphere is contained in the second, otherwise the second sphere is contained in the first. In the case where $d = |\rho_1 - \rho_2| = 0$, then the two spheres are identical and the distance between them is trivially 0.

The two spheres are intersecting only when $|\rho_1 - \rho_2| \leq d \leq \rho_1 + \rho_2$. The intersection will be a circle (or a point, if the two spheres merely touch one another) with normal plane Φ. This circle can be calculated using the formula for the meet ([LLW04]):

$$C = \Sigma_1 \vee \Sigma_2 = \left[\langle \Sigma_1 \Sigma_2 \rangle_{2n-r-s}\right]^* \tag{2.98}$$

$2n - r - s = 2 \cdot 5 - 4 - 4 = 2$, so that the dual quantity will have grade $5 - 2 = 3$, which is a trivector, and generally represents the circle of intersection. The value of C can tell us whether the result is a circle $\left(C^2 > 0\right)$, a single point $\left(C^2 = 0\right)$ or there is no intersection $\left(C^2 < 0\right)$. In the case where $\left(C^2 > 0\right)$, the centre and radius of the circle can be extracted according to section 2.2.5. Similarly, using the same extracting formula from C for the case where $\left(C^2 = 0\right)$, we will find that the circle has zero radius and its centre will be the point of tangency of the two spheres. In the same way, an attempt to extract the radius and the centre for the case where $\left(C^2 < 0\right)$ leads to an imaginary value for the radius, and a centre (because no intersection exist) which lies on the shortest line joining the surface of the spheres (i.e. that joining the centres). If the two spheres have the same radii, it is the midway point on this line.

In the same way, instead of having a second sphere, Σ_2, we can have an intersection with a plane, Φ_1. The result of the meet between sphere and plane will also be a trivector, C,

$$C = \Sigma_1 \vee \Phi_1 = \left[\langle \Sigma_1 \Phi_2 \rangle_2\right]^* \tag{2.99}$$

and the sign of the square of this trivector, C^2 indicates whether the two objects are tangent, intersect in a circle or do not intersect at all.

2.3. Conclusions

This chapter introduced Geometric Algebra and examined how the Conformal Model can be used to represent geometric primitives such as points pairs, lines, circles, planes and spheres. We also looked at how rotors can be applied to those objects within the Conformal Model. CGA is a mathematical framework that offers a compact and geometrically intuitive formulation of algorithms and an easy and immediate computation of rotors; it is thus suitable for applications in engineering, computer vision and robotics. The model described in this chapter will be the mathematical basis on which this thesis is based.

3 A Novel Inverse Kinematics Solver

In this chapter, we address 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 (DoF), thus, it is often difficult to produce a realistic character animation.

3.1. 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 a 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 graphics 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 [CD03]. However, the production of real-time IK solvers that are able to return realistic postures, without erratic discontinuities and singularities, posed a problem for researchers for many years; many algorithms have been implemented for computing skeletal poses but most suffer from unnatural poses, have difficulties in dealing with complex figures and are computationally expensive.

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, and 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.

In this chapter, vectors will be designated in bold font to distinguish them from other symbols and avoid any confusion.

3.1.1. The articulated body model

This section provides a brief introduction to the human skeleton and joint modelling. 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 and various possible restrictions. In fact, the human body consists of more than 200 bones and joints.

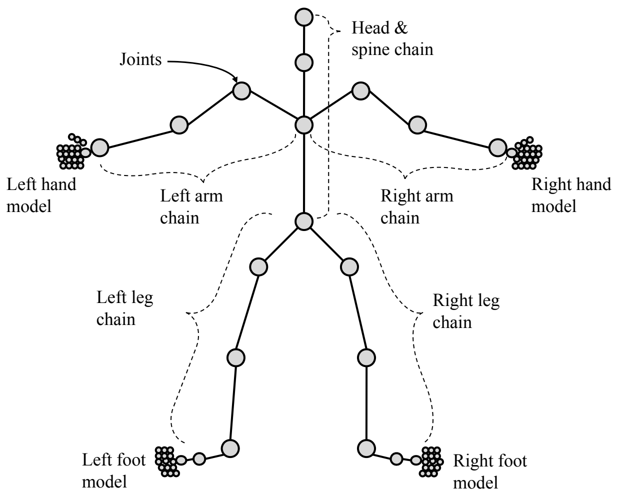


Figure 3.1.: *An example of a skeletal structure of a human.*

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 3.1). 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 3.2 shows examples of a model of a human body and human legs taken by a motion capture system (PhaseSpace Impulse System [Pha]) and graphically processed in Blender [Ble]. 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 (the start of the chain) is reached. By definition, in the IK problem, the root joint is assumed fixed but methods can cope easily with translation of the root.

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

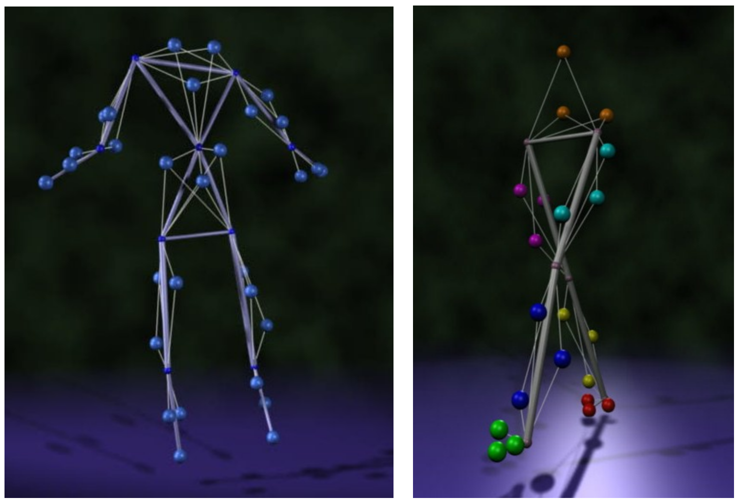


Figure 3.2.: *Part of a skeletal animation presenting joints of a human body (left) and human legs (right).*

types. The main human joint types are enumerated below (see also figure 3.3):

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.

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 [BPW93, Cra89]. Because of their complex nature, most of the

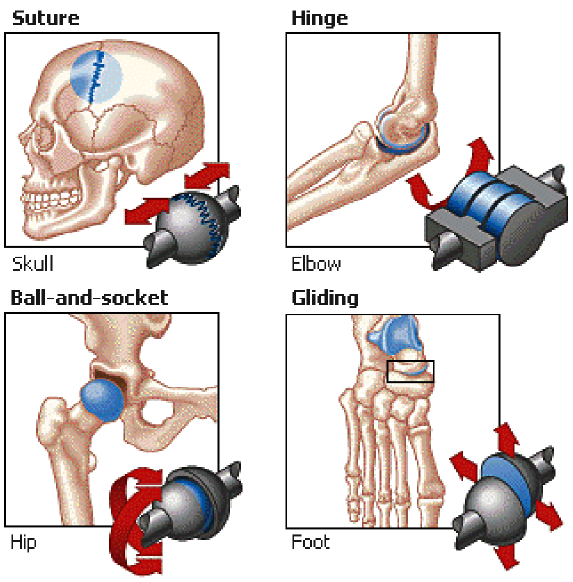


Figure 3.3.: *Human joints with their available movements. The images have been taken from the Microsoft Encarta Online Encyclopedia 2008 [Enc].*

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 [MT00, WV98, KTL07, HUHF03]; *the spine model*, a complex arrangement of 24 vertebrae (usually, for simplicity, the spine is modelled as a simple chain of joints [BPW93, IP90, Kor85, MB91]); *the hand model*, this is the most versatile part of the body comprising a large number of joints [RG91, MSZ94, KL07]; *the strength model*, which takes account of the forces applied from the skeletal muscles to the bones [BPW93].

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 human animation with skin, face, clothes etc [SPCM97].

Figure 3.3 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 [BPW93, Cra89, MT00, Kor85, MSZ94, WW91].

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.

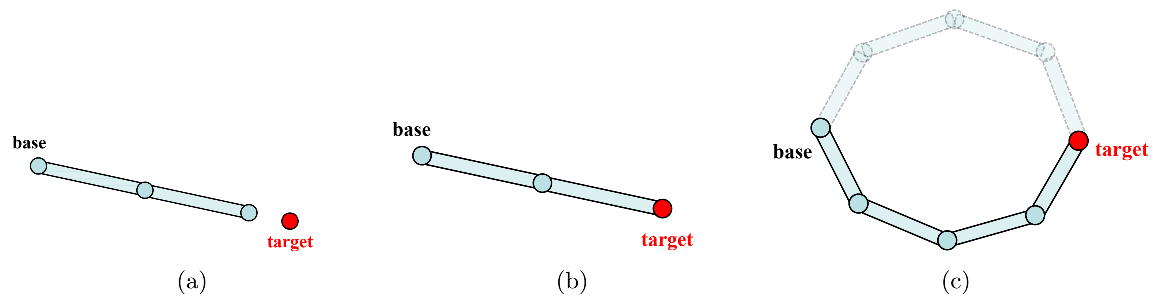


Figure 3.4.: *Possible solutions of the IK problem: (a) The target is unreachable; in many cases it is impossible for the linked structure to touch the target, (b) One solution: there are instances where there is only one solution to the problem, (c) Many solutions: most often, the IK problem has more than a single solution.*

- *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.

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 3.4). 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.

3.2. Related Work

The production of realistic and plausible motions remains an open challenge within the robotics and animation communities. Several algorithms have been implemented for computing the poses of a skeletal structure; the most popular techniques for solving the IK problem are presented in this section.

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 configurations may not always be in terms of angles), 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. The most popular numerical approach is to use the Jacobian matrix to find a linear approximation to the IK problem.

3.2.1. 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 3.5); 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

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

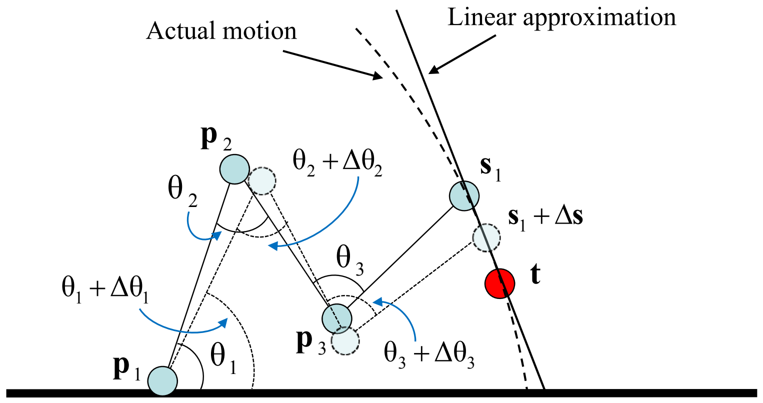


Figure 3.5.: *The Jacobian solution is a linear approximation of the actual motion of the kinematic chain.*

where $i = 1, ..., k$ and $j = 1, ..., n$. Orin and Schrader in [OS84] 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 ($m = 3k$).

Equation 3.1 for forward dynamics can now be written as

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

where the dot notation specifies the first derivative with respect to time. Using the current values $\boldsymbol{\theta}$, $\vec{\mathbf{s}}$ and $\vec{\mathbf{t}}$, 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

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

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

Thus, the FK problem can be expressed as $\vec{\mathbf{e}} = J\Delta\boldsymbol{\theta}$ and the IK problem can be rewritten as $\Delta\boldsymbol{\theta} = J^{-1}\vec{\mathbf{e}}$. 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 section.

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

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

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 [Lie77, MK85]. A more sophisticated nullspace method, the *Extended Jacobian* method, was introduced by Baillieul [Bai85]; 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.

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

Jacobian transpose

The Jacobian transpose method was first used for inverse kinematics in [BDMS84, WE84]. 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, [BDMS84, WE84] 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

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

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. The product DD^T is the $m \times m$ matrix 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

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

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

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

Damped Least Squares

The Damped Least Squares method (DLS) was first used for inverse kinematics by [Wam86, NH86]. 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

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

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

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

Now $J^T J$ is an $n \times n$ matrix, where n is the number of degrees of freedom. It is easily proven that $\left(J^T J + \lambda^2 I\right)^{-1} J^T = J^T \left(J J^T + \lambda^2 I\right)^{-1}$; 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,

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

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.

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 simple version away from singularities, but smooths out the performance of the simple pseudo-inverse method in areas close to singularities.

Selectively Damped Least Squares

The Selectively Damped Least Squares (SDLS) method was presented by Buss and Kim in [BK05] 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.

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 [Wel93]. 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 [F03], 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 [MM05]. 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 $\mathbf{p}_w = W\mathbf{p}$.

Feedback Inverse Kinematics

The Feedback Inverse Kinematics (FIK) method [Pec08] 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. [Pec08] 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.

3.2.2. 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 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 [Fle87, CKM97].

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 [ZB94]. 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.

3.2.3. IK using Sequential Monte Carlo Methods

Sequential Monte Carlo Methods (SMCM) have been recently introduced for solving IK problems. Courty and Arnaud in [CA08] 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 [HRE$^+$08] which uses a body pose goals set and attempts to satisfy the goals by forming a system of constraints over the linked character bodies.

3.2.4. Style or mesh-based Inverse Kinematics

[GMHP04] 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 [SZGP05], 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. [DSP06] 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 dependent on the training data and limited only to those models and movements the system has been trained on.

3.2.5. Heuristic Inverse Kinematics algorithms

Cyclic Coordinate Descent

Cyclic Coordinate Descent (CCD), which was first introduced by [WC91] and then biomechanically constrained by [Wel93], is an iterative heuristic 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

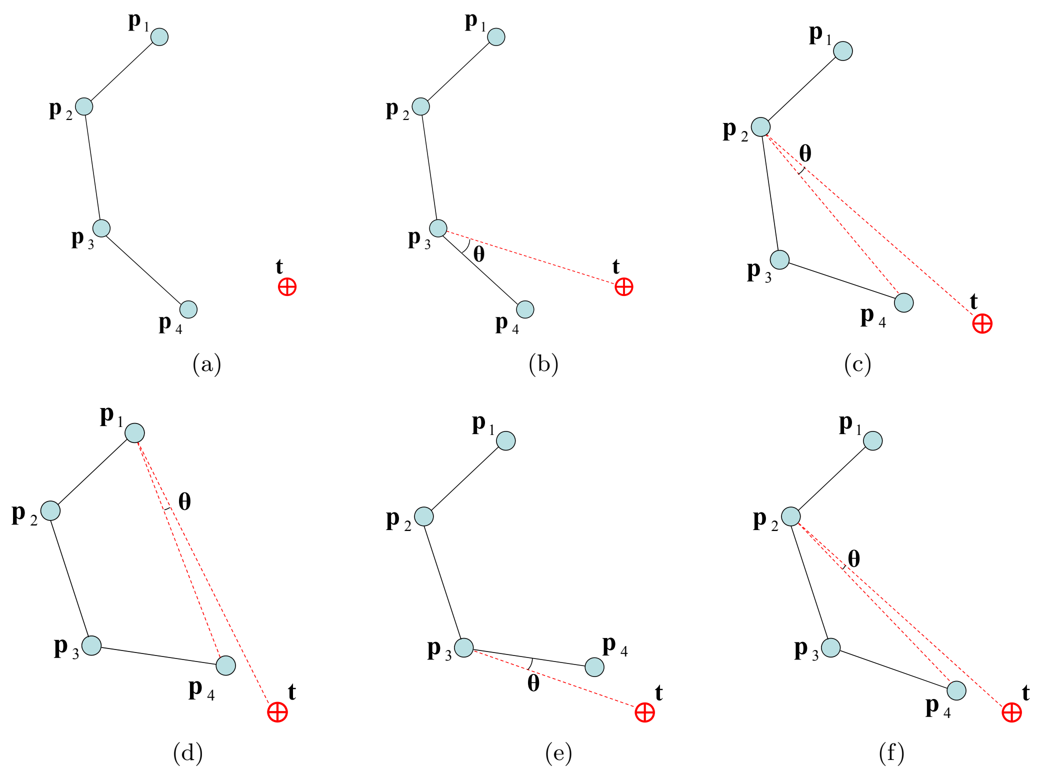


Figure 3.6.: *An example of visual solution of the IK problem using the CCD algorithm. (a) The initial position of the manipulator and the target, (b) find the angle θ between the end effector, joint $\mathbf{p}_3$ and the target and rotate the joint $\mathbf{p}_4$ by this angle, (c) find the angle θ 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.*

used for solving the inverse kinematic problem in the computer games industry (e.g. [Lan98]). CCD has also been effectively used in protein science for protein structure prediction and/or structure determination [CD03].

CCD 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. 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. Figure 3.6 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 these many resulting postures. Therefore, manipulator constraints must be incorporated to restrict motions to a feasible set. In CCD it is easy to apply local constraints but it is more difficult to implement global manipulation restrictions.

CCD is a very quick method but it is not free from problems; it 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. CCD is designed to handle serial chains; however multiple goals are necessary for most graphics and robotics applications. It is, however, non-straightforward to extend it to problems with multiple targets and end effectors. [MD04] 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 into smaller serial chains and treats each chain independently. [KM05] also adopted the CCD kinematic algorithm and solved its crucial problem of resulting unnatural poses. The proposed extension in [KM05] is able to solve problems with humanoid hierarchy, dividing the whole body into groups of joints near an end effector (typically head, trunk, arms and legs). In order to satisfy the desired centre of mass, the lightest group moves first, adjusting its centre of mass by changing the length of the limb and rotating it (assuming it as a rigid body).

Inductive Inverse Kinematics algorithm: The Inductive Inverse Kinematics (IIK) algorithm [KLC+03] 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 organised 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.

Triangulation Inverse Kinematics

Another method which does not use an iterative approach is presented in [MCM07]. 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. An improved version is given in [Muk09] where the n-link IK problem is reduced into a *two*-link problem, making sure that each link is rotated at most once in an attempt to reach the target position.

Sequential Inverse Kinematics

Sequential Inverse Kinematics (SIK), which is presented in [UPBS08], is a direct extension of [BVU+06]. 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 (for instance CCD), in different parts of the body, in a specific order. The SIK, according to [UPBS08], outperforms many IK methods regarding the joint average position error, the joint average orientation error and the median processing time of each methodology.

3.3. FABRIK: A New Heuristic IK Methodology

In this section, a new heuristic method for solving the IK problem, FABRIK [AL10d], 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 $\mathbf{b}$. FABRIK is illustrated in pseudo-code in Algorithm 1 and a graphical representation of its full iteration with a single target and 4 joints is presented and explained in figure 3.7.

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 $\mathbf{p}_{n-2}$ and

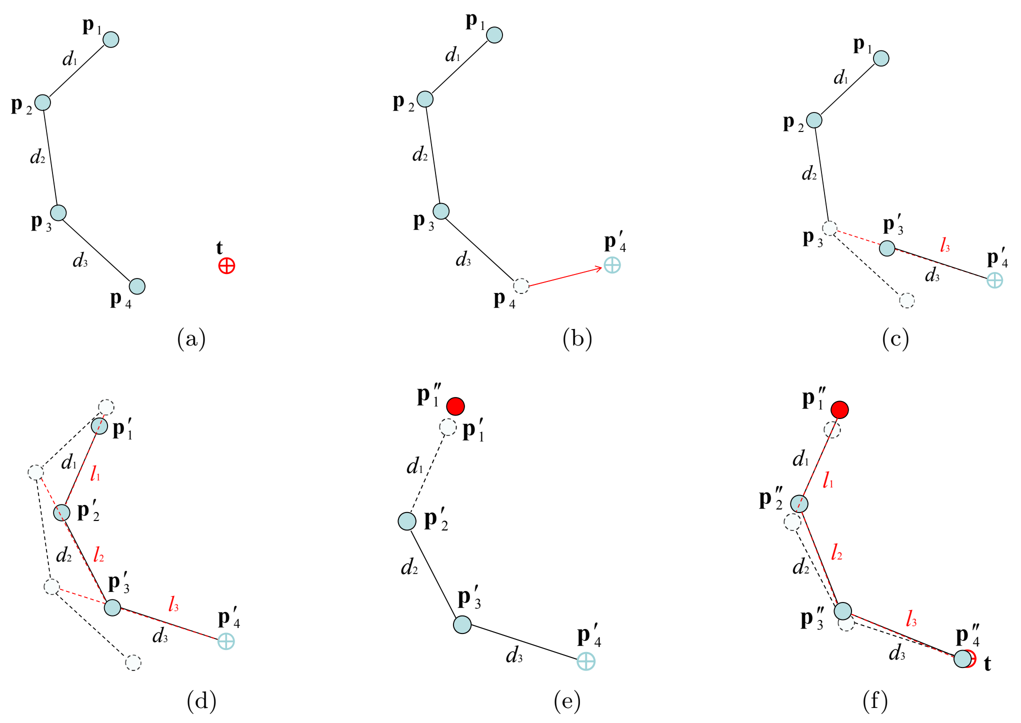


Figure 3.7.: *An example of a full iteration of FABRIK for the case of a single target and 4 manipulator joints. (a) The initial position of the manipulator and the target, (b) move the end effector $\mathbf{p}_4$ to the target, (c) find the joint $\mathbf{p}'_3$ which lies on the line l_3 that passes 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.*

$\mathbf{p}'_{n-1}$, 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 indicated 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 (CGA) [HS84, DL03] to produce faster results and to converge to the final answer in fewer iterations; CGA 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; the new joint position will be taken as the point on the circle (created by the intersection of the 2 spheres) nearest to the previous joint position. Another simple optimisation is the direct construction of a line pointing towards the target, when the latter is unreachable. A similar solution of the algorithm using CGA is given in Appendix A.5 and [AL10a].

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 value decomposition), it does not suffer from singularity problems and returns smooth motion without erratic discontinuities.

A singularity problem might occur when the chain is completely straight and the target is located on that alignment but between two joints (on the bone). In such an instance, FABRIK does not converge to a solution but enters to an infinite loop (a similar problem is encountered in the CCD algorithm). The solution is to choose a very small angle in the beginning of the backward stage of the algorithm, in order to allow the chain to bend into a direction that satisfies the user constraints.

3.3.1. 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 to the parent sub-base. This will produce as many different positions of the sub-base

[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.

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
```

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 to the manipulator root. If there are more intermediate sub-bases, 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 is reached; if more sub-bases exist, the same process is applied. The method is repeated until all end effectors reach the targets or there is no significant change between their previous and their

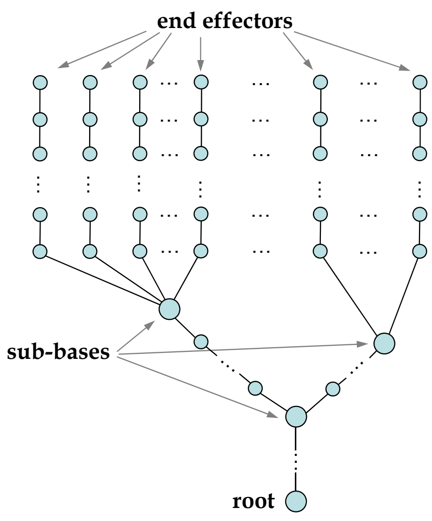


Figure 3.8.: *An example of a model figure with multiple end effectors and multiple sub-bases.*

new positions. An example of a model figure having multiple end effectors and sub-bases is presented in figure 3.8.

More sophisticated (and complex) models can be also tackled. Extending the proposed algorithm to take into account the figure's shape, constraints and properties, will reduce the number of iterations needed to reach the target and will return more feasible postures. For example, FABRIK has been successfully applied to real-time hand tracking and reconstruction in motion capture, as it is presented in Chapter 5 and published in [AL10a, AL10b].

3.3.2. FABRIK within closed loops

FABRIK can also cope with cases where the "end effector" is not positioned at the end of the chain (i.e. it is a leaf) in the same way as for the sub-bases described in section 3.3.1. The whole model could be divided into groups of joints near the end effectors (such as head, trunk, arms and legs) and then sequentially adapt the body postures in a specific order, similarly to [UPBS08] and [KM05]. Obviously, the adaption hierarchy varies between models. An example where FABRIK has been successfully adjusted within closed loops of a humanoid, achieving real-time centre of rotation correction in motion capture, under marker occlusions is presented in Chapter 4 and been published in [AL10c].

3.4. Applying Joint 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, IK techniques and tracking in motion capture systems to reduce visually unrealistic movements.

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

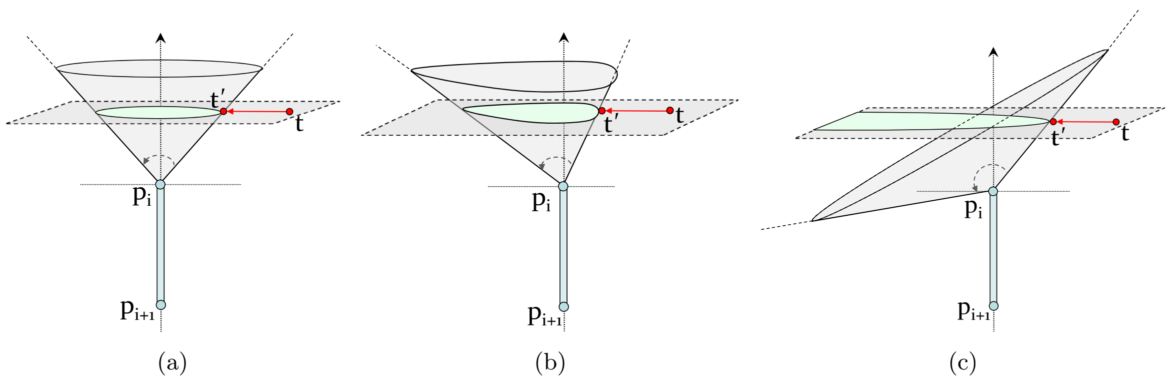


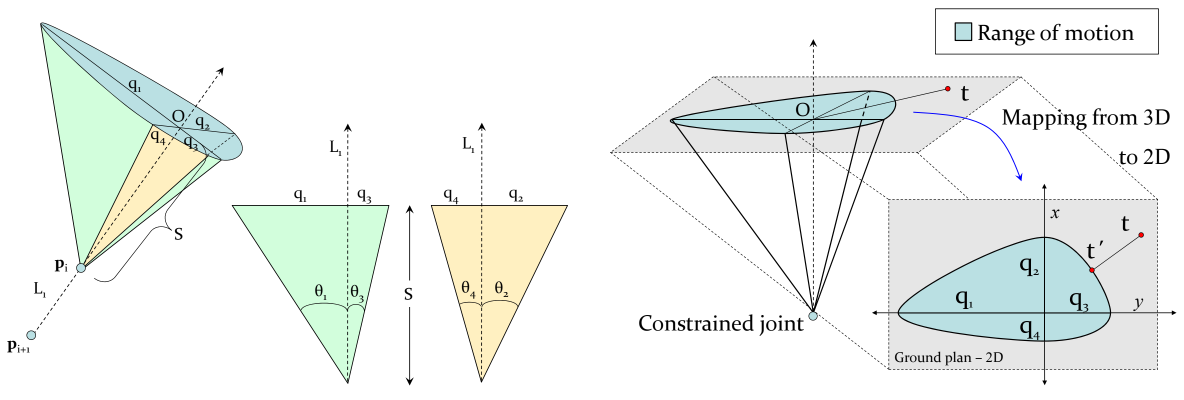
Figure 3.9.: *The target is re-positioned within the allowed range of motion which is defined by the conic section. There are 3 types of joint restriction, as described by the angles $\theta_1, ..., \theta_4$: (a) a circle, (b) an ellipsoidal shape and (c) a parabolic shape.*

here as *rotational* (2 DoF), that moves the bone to its final direction vector, and another which we call *orientational* (1 DoF), 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 [Blo02] 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 [WG01] present a 3D "reach cone" methodology using planes, treating the joint limits in the same way as [Blo02]. [Kor85, BB01] 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.

3.4.1. Restricting the motion to the allowed bounds

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 ability to converge to 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. In contrast to most existing techniques for joint constraints, the proposed methodology simplifies the 3D problem into a 2D problem, meaning that the complexity and the required processing time is reduced. In this chapter, a joint restriction



(a) (b)

Figure 3.10.: *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.*

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
```

methodology is presented for the general case of a ball and socket joint; this example should be considered an illustration of how joint or model constraints can be incorporated within FABRIK.

Assume we have a ball-and-socket joint with orientational limits described by the rotor $\mathbf{R}$ and rotational limits described by the angles $\theta_1, ..., \theta_4$. A graphical representation of a joint limit boundary could be an irregular cone which is defined by these angles. The rotational limits are enforced by re-positioning the target point as the nearest point on a conic section from the target position; this procedure is described in detail later. There are 3 possible conic sections, according to the angles defining the irregular cone: if all θs are equal, the conic section is a circle; if all θs are greater or smaller than 90^o and are not equal, the conic section has an ellipsoidal shape; finally, if there are θs both greater and smaller than 90^o, then the joint boundary limits are defined by a parabolic shape, as illustrated in figure 3.9. In the subsequent analysis shown here, the joint limits are assumed to be defined by an ellipsoidal shape, since this is the most common case, but similar procedures apply for different conic sections. Figure 3.10 gives a graphical representation of the implemented constraints and the irregular cone describing the rotational motion bounds for the case of an ellipsoidal shape.

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

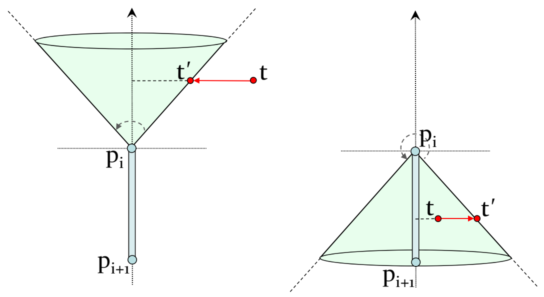
Algorithm 3: The rotational constraints.

```
     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 what conic section describes the allowed range of motion
3.7  Find the conic section which is associated with that quadrant using the distances $q_j = S \tan \theta_j$,
     where $j = 1, .., 4$
3.8  Check whether the target is within the conic section or not
3.9  if within the conic section then
3.10     use the true target position $\mathbf{t}$
3.11 else
3.12     Find the nearest point on that conic section from the target
3.13     Map (rotate and translate) that point on the conic section via reverse of 3.4 and use that point
         as the new target position
3.14 end
```

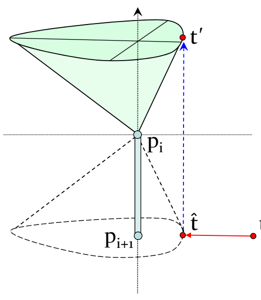
the orientation frames at joints $\mathbf{p}'_i$ and $\mathbf{p}_{i-1}$ and if this rotor represents a rotation greater than a limit, reorient the joint $\mathbf{p}_{i-1}$ in such a way that the rotation will be within the limits. Repeat the procedure for all the joints on both stages of the algorithm. 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 3.10. 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 3.10(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 example, in figure 3.10(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 [ARW01]. Obviously, it is not necessary to calculate all the ellipses which define the composite ellipsoidal shape of figure 3.10(b), but only the ellipse related to the quadrant in which the target is located. It is important here to recall that, if the constraints which define the allowed range of motion are described by a different conic section (circle or parabola), the target should be re-positioned as the nearest point on that conic section, similarly to the ellipsoidal case. 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 and a demonstration of the process is given in figure 3.12.

This is a versatile, and easily visualisable, method of restricting where the bone can go.



(a)



(b)

Figure 3.11.: *Solution for special joint restriction cases: (a) the original case and when the allowed range of motion is greater than 180 degrees, (b) when the target is located on a different hemisphere than the irregular cone.*

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 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 to be 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 $\mathbf{p}'_i$, as is illustrated in figure 3.12.

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.

The proposed restriction methodology can be easily extended to manage joints limits greater than 180 degrees. For instance, when the angle which defines the allowed range of motion is greater than 180 degrees, the associated irregular cone will define the area which is outside

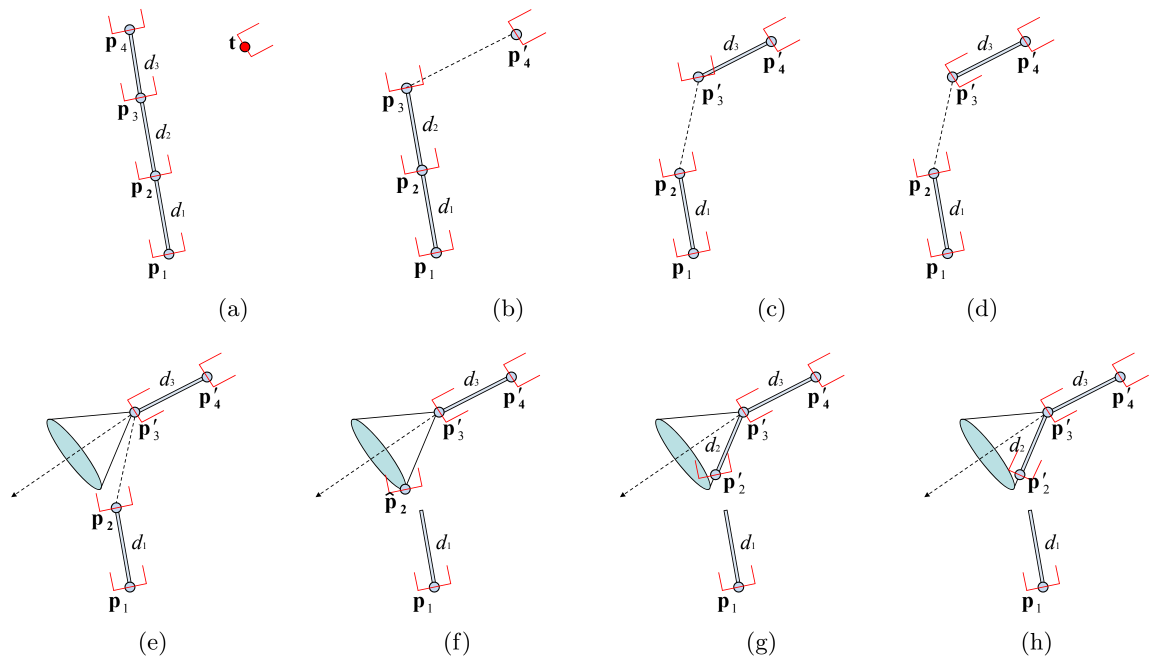


Figure 3.12.: *Incorporating rotational and orientational constraints within FABRIK. (a) The initial configuration of the manipulator and the target, (b) relocate and reorient joint $\mathbf{p}_4$ to target $\mathbf{t}$, (c) move joint $\mathbf{p}_3$ to $\mathbf{p}'_3$, which lies on the line that passes 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.*

the limits. In that case, the joint restriction methodology will work in a reverse fashion; if the target is within the irregular cone area, meaning it is outside the limits, it will be projected to the cone surface, as is demonstrated in figure 3.11(a). Another special case of joint restriction occurs when the target is located in such a position that is not in the same hemisphere (in figure 3.11(b), the upper hemisphere) as the irregular cone. The limits of motion are defined as the irregular cone in the upper hemisphere and a reflection of the cone in the lower hemisphere; the target in the lower hemisphere is projected onto the limit boundary by first projecting its position onto the reflected cone and taking the associated point on the regular cone, as shown in figure 3.11(b). Obviously, the algorithm works in a similar way for different conic sections.

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 [Blo02, WG01]; 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 it is desirable to retrieve the joint angles, all necessary information is of

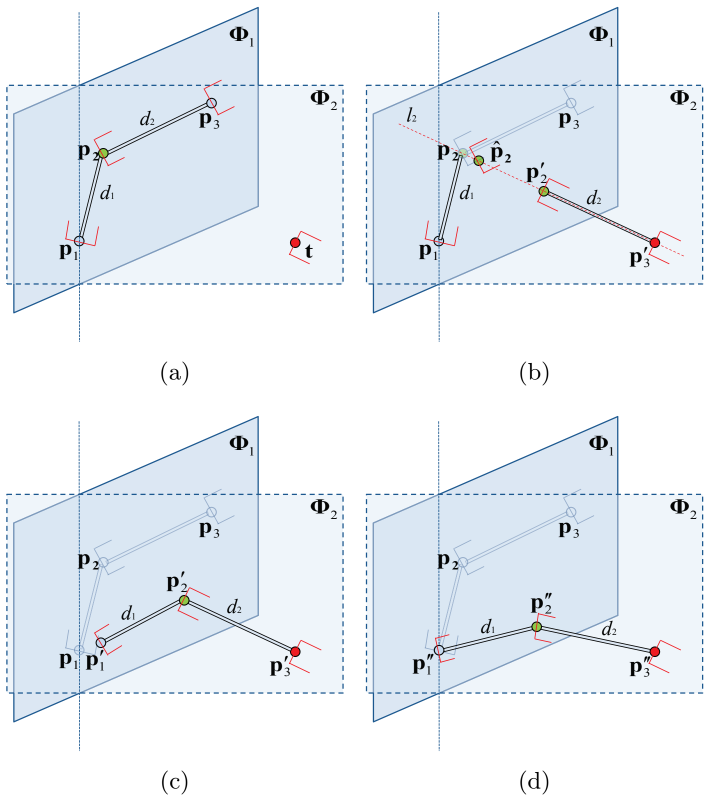


Figure 3.13.: *Incorporating constraints for a hinge joint. (a) The initial configuration of the manipulator and the target. Since $\mathbf{p}_2$ is a hinge joint with 2 DoF, all joints lie in the plane $\mathbf{\Phi}_1$. The root and the target, which is oriented, also define the plane $\mathbf{\Phi}_2$. (b) Relocate and reorient joint $\mathbf{p}'_3$ to target $\mathbf{t}$. Then, project $\mathbf{p}_2$ onto the plane $\mathbf{\Phi}_2$ to give a new point $\hat{\mathbf{p}}_2$, and find the point $\mathbf{p}'_2$ on line l_2 that passes from the joint position $\mathbf{p}'_3$ and the projected joint position $\hat{\mathbf{p}}_2$ and has distance d_2 from $\mathbf{p}'_3$. Reorient the new joint using the orientation constraints. (c) move and reorient joint $\mathbf{p}_1$ to $\mathbf{p}'_1$, which lies on the line that passes through the points $\mathbf{p}'_2$ and $\mathbf{p}_1$ and has distance d_3 from $\mathbf{p}'_4$, (d) The problem is now again a 2D problem and all joints lie on plane $\mathbf{\Phi}_2$. Thus, the prototype version of FABRIK can be applied to all the remaining joints in a forward and backward fashion.*

course avialable (position and orientation of each joint).

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, sliding or shifting joints (joint types more usually discussed in robotics) are not supported. Self-collisions can be handled using existing techniques, such as [LG98]; but more work is needed to ascertain if the FABRIK framework gives any advantages when dealing with self occlusions.

Having in mind that the target should be reoriented and repositioned in such a way as it satisfy the user or the model constraints, different joint models can be formulated using

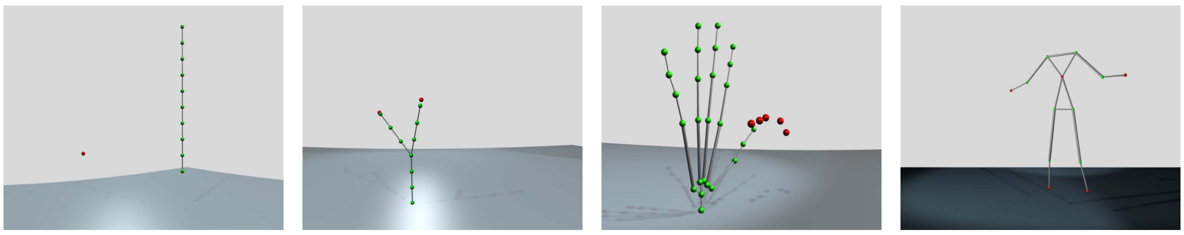


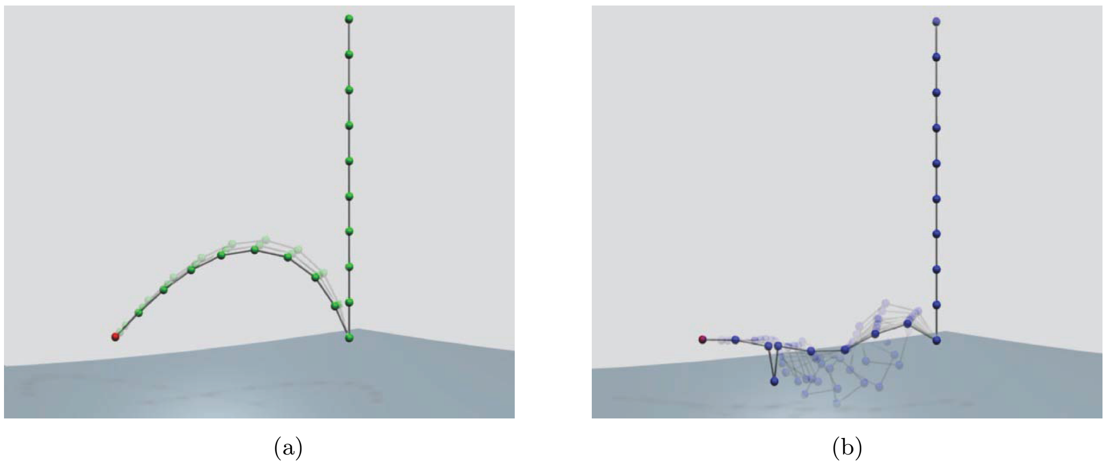
Figure 3.14.: *The structure of the models used in our experimental examples. (a) A kinematic chain consisting of 10 joints and 1 end effector. There are 2 kinematic chain models, an unconstrained and a constrained version, (b) a kinematic model with 10 unconstrained joints and 2 end effectors, (c) a 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.*

similar techniques. For instance, the joint limitations for simple 2D joint models, such as the *hinge* joint, can be simplified using alternative approaches. Since FABRIK operates on the joint coordinates by adjusting the positions in an iterative fashion, the 2D restrictions can be enforced by projecting the joint onto the plane of orientation. That plane is defined by the root and the (oriented) target position. An illustration showing how restrictions can be enforced for a hinge manipulation is given in figure 3.13. Similar techniques can be applied to incorporate constraints for different types of joint, in a variety of motions.

3.5. 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 each joint and 4 end effectors. Figure 3.14 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.



(a) (b)

Figure 3.15.: *The stages illustrated in order that the end effector reaches the target. (a) The FABRIK solution and (b) the CCD solution.*

3.6. Results

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 Jacobian and DLS parameter values used in our experiments are the parameter values suggested by [BK05]; 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. All experiments were run using MATLAB [MAT] 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 [Ble].

3.6.1. 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 3.19.

FABRIK produces results significantly faster than all IK methods tested. It is approximately 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 3.15 shows an example of an IK solution using FABRIK and CCD; the figure presents

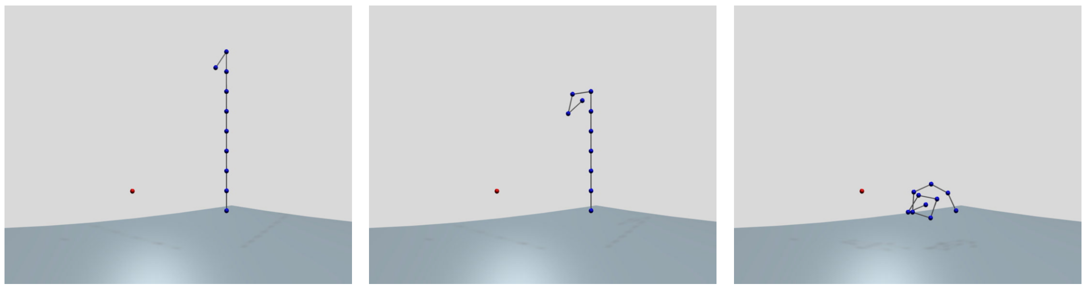


Figure 3.16.: *Unnatural joint angles exhibited by CCD; the kinematic chain rolls itself before reaching the target, producing unrealistic poses.*

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.

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 3.15 and 3.16). 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 is designed to handle problems with serial chains; it has to be modified in order to solve problems with multiple end effectors [KM05].

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 easily. Constraints can be applied within the Jacobian algorithms, but the way in which these restrictions are incorporated is 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 3.17, where the number of iterations needed to reduce the distance between target and end effector as this changes

Table 3.1.: *Average results (over 20 runs) for a single kinematic chain with 10 joints.*

	Reachable Target				Unreachable Target	
	Number of Iterations	Matlab exe. time (*sec*)	Time per iteration (in *msec*)	Iterations per second	Number of Iterations	Matlab exe. time (*sec*)
FABRIK	15.461	0.01328	0.8	1164	67.564	0.06207
CCD	26.308	0.12356	8.8	213	390.135	3.92869
Jacobian Transpose	1311.190	12.98947	9.9	101	6549.000	33.90473
Jacobian DLS	998.648	10.48051	10.5	95	2881.667	14.87918
Jacobian SVD-DLS	808.797	9.29652	11.5	87	2808.452	15.97591

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 3.1 presents the average runtimes of each of the methods, as well as the number of iterations needed to reach the target, for both cases of a reachable and an unreachable target. It also indicates on average the time needed per iteration for each method and how many iterations per second each methodology can support for the case of a single chain with 10 joints and 1 end effector. 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, where the Jacobian Transpose, DLS and SCD-DLS methods need 9.9ms, 10.5ms and 11.5ms per iteration respectively.

Figure 3.19 compares the performance of each algorithm for solving inverse kinematic problems; it shows 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 3.18 plots the convergence of each method, meaning the time taken to achieve the solution with the requested degree of accuracy. It is clearly observed that FABRIK converges to the target faster than any other

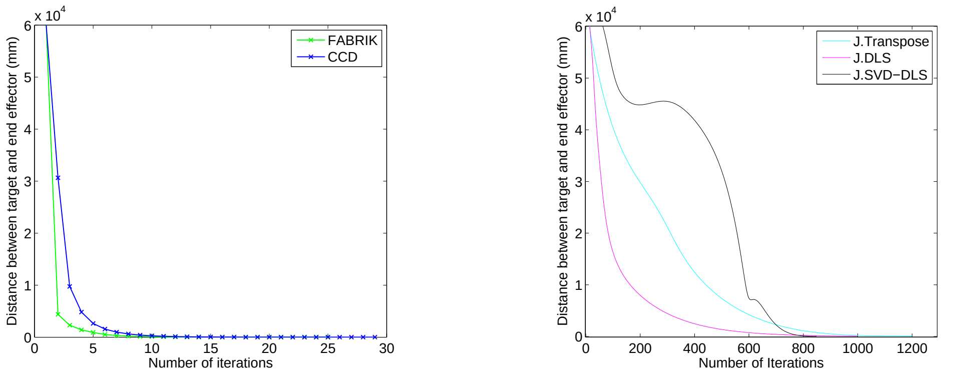


Figure 3.17.: *The iterations needed to reach the target against the distance between target and end effector.*

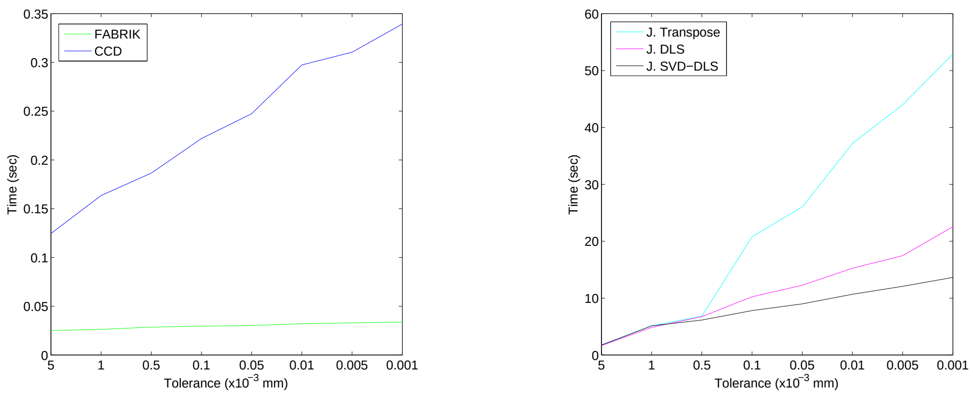


Figure 3.18.: *An example presenting the time needed for each methodology to achieve the solution with the degree of accuracy requested.*

implemented methodology. An iteration of FABRIK has the lowest computational cost and figure 3.18 verifies that FABRIK always converges to the target, if the latter is reachable.

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 it is within reach, and keeping the end effector pointing at the target when it is unreachable. The accuracy of the tracking was measured over a period of a thousand simulation steps. FABRIK tracks the target in real-time producing 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 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 still 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 the Jacobian-based 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 3.20 presents the performance of each method on selected frames over time.

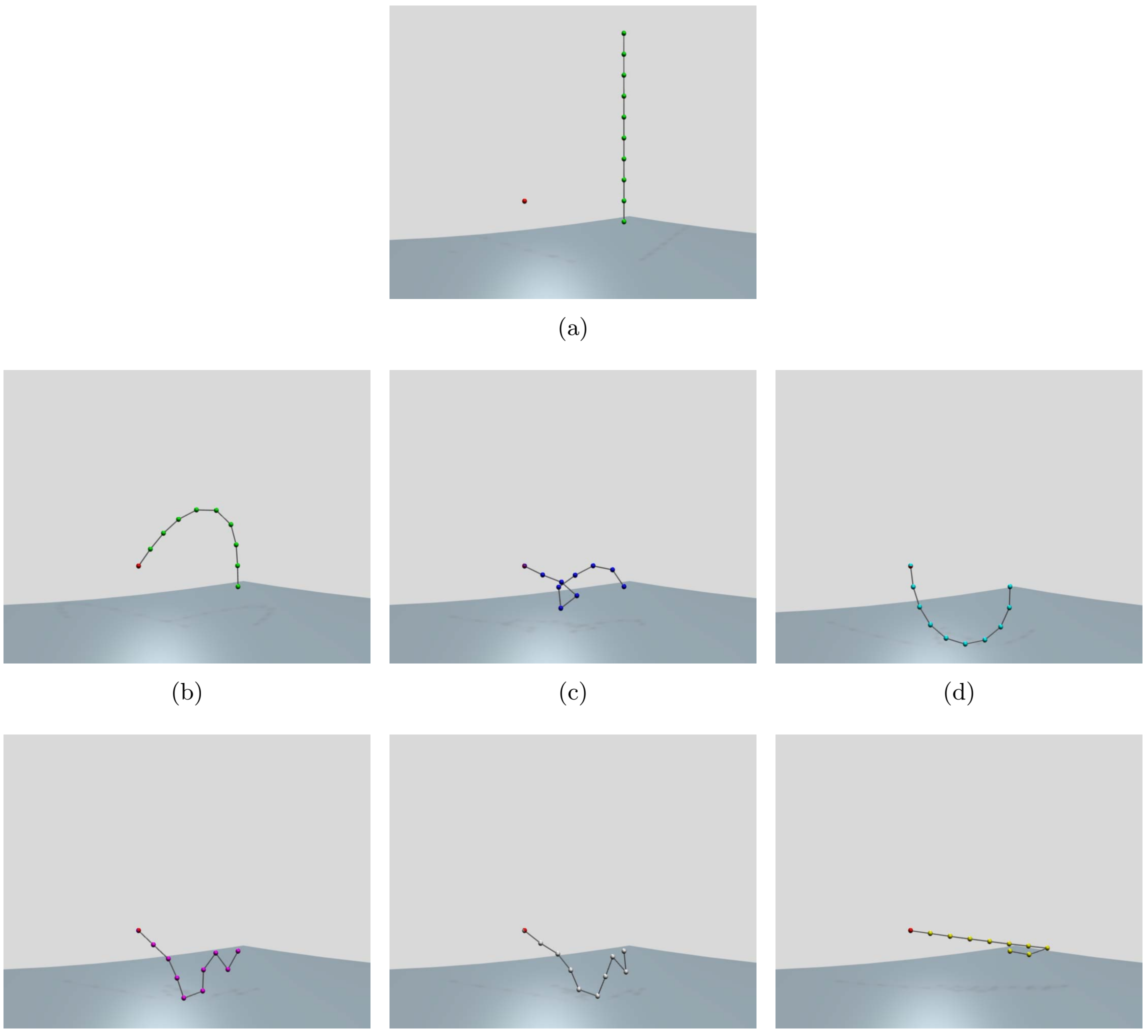
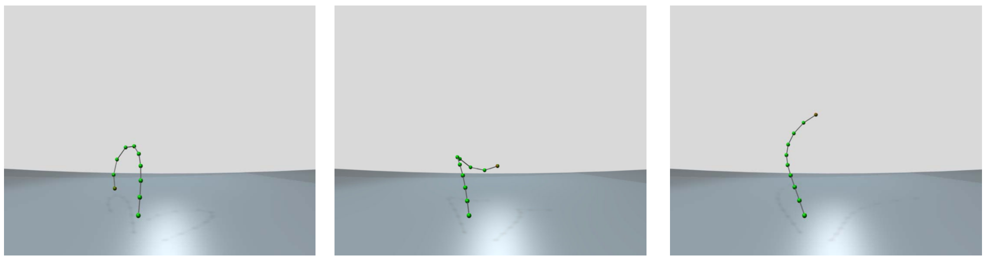

(a)
(b)
(c)
(d)
(e)
(f)
(g)

Figure 3.19.: *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.*

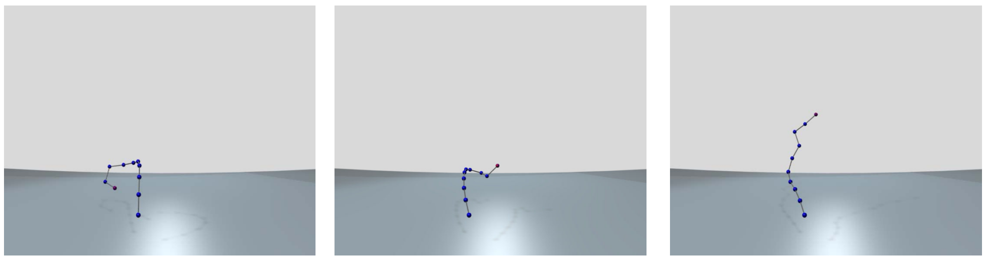
Making Kine more flexible

In this section we implemented the FABRIK algorithm within the Kine [Lan98] 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 track your mouse. You are also able to add more links, optimise orientation, and modify the application to a 3D environment.

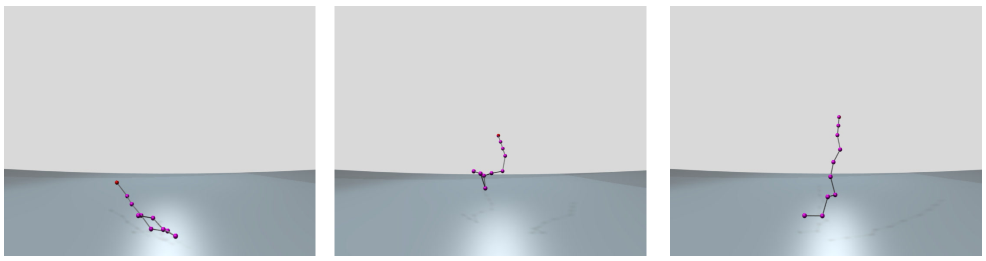
Figure 3.21 shows examples of the FABRIK and CCD methods implemented within the Kine environment when the end effector moves through large distances. It is clearly observed that FABRIK out-performs CCD in producing smoother poses. The environment presented in this section has been adopted from the work of Jeff Lander; we would like to express our enormous thanks to Jeff for giving us permission to use his code and application. The Kine application



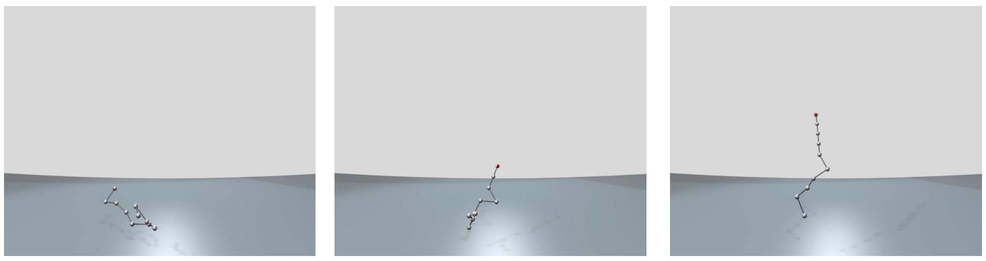
(a)



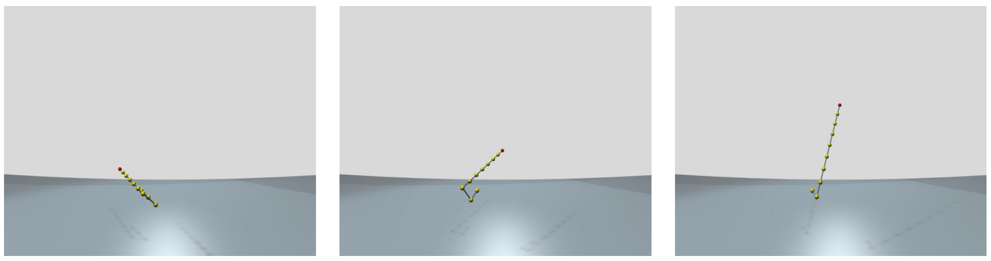
(b)



(c)



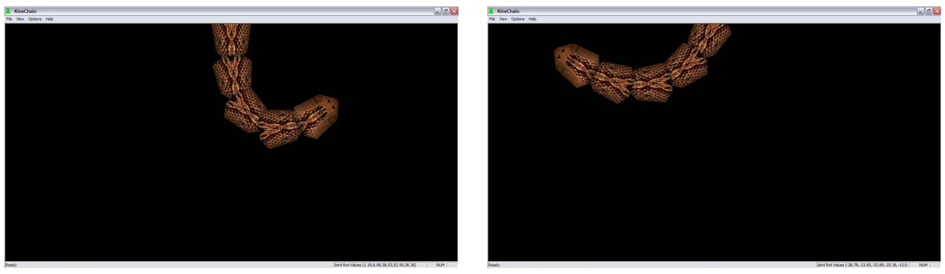
(d)



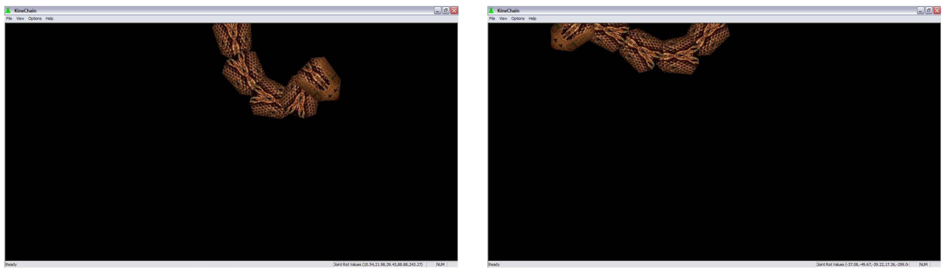
(e)

Figure 3.20.: *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*

is included in the supplementary materials, offering the opportunity to interact and evaluate the results in real-time.



(a)



(b)

Figure 3.21.: *FABRIK and CCD solution using the Kine application. (a) FABRIK, (b) CCD.*

3.6.2. 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 3.22, also used in [BK05], and a hand multibody presented in figure 3.23. 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 3.22 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 3.23 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, allowing 3 DoF on each joint. Incorporating a highly constrained model, such as [KL07], and restricting the motion of each joint to a feasible set, the hand will have even more natural movement. Such a model has been implemented and presented in Chapter 5.

Figure 3.24 shows an example of a tracking animation of a humanoid with 13 joints, 5 of which are treated as end effectors. In this demonstration, the frame rate was low (3 frames per second); the 3Hz frame rate selection increases the distance between target and end effector, thus the performance of each method is more obvious. FABRIK can easily track the animated

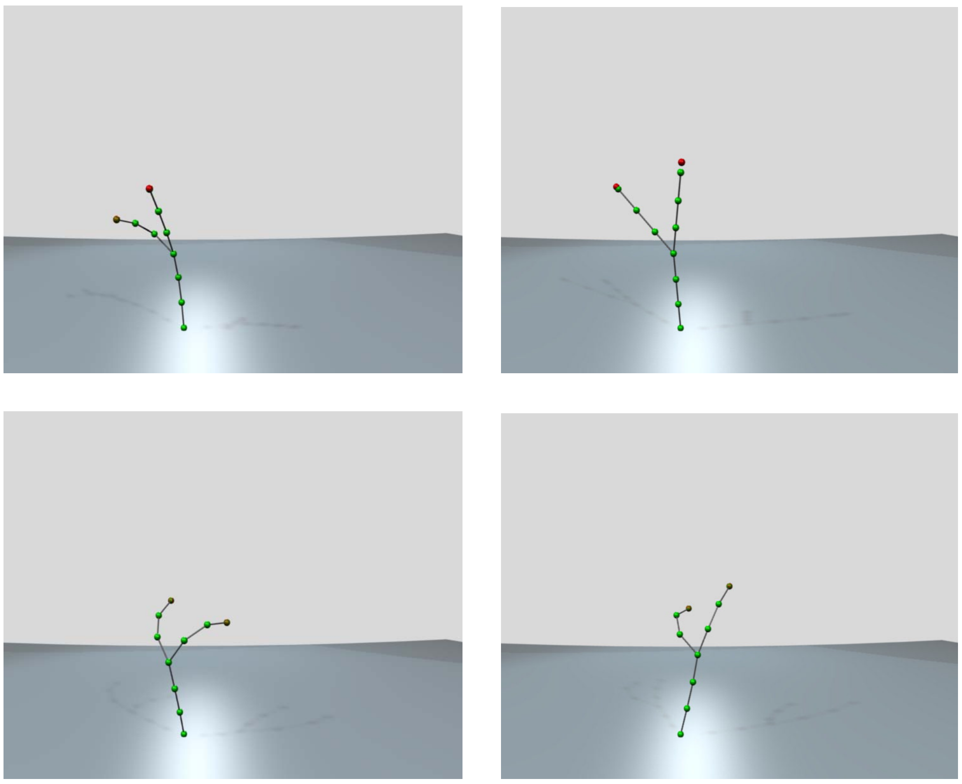


Figure 3.22.: *Example of FABRIK implementation with multiple end effectors moving over time; a kinematic chain with 10 unconstrained joints, 2 end effectors and 2 targets.*

Table 3.2.: *Reconstruction comparison. Average results (over 20 runs).*

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

[†] This is a MATLAB executable time.

humanoid in real-time, producing very reasonable results. Figure 3.25 shows the reconstruction quality of different methodologies over the same humanoid model. The differences between the implemented methodologies, on these unconstrained humanoid examples, are more obvious on shoulders, elbows and hips. FABRIK produces visually natural 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 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 3.2 shows the performance (over 20 runs) of each methodology for the case of a dancing humanoid 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

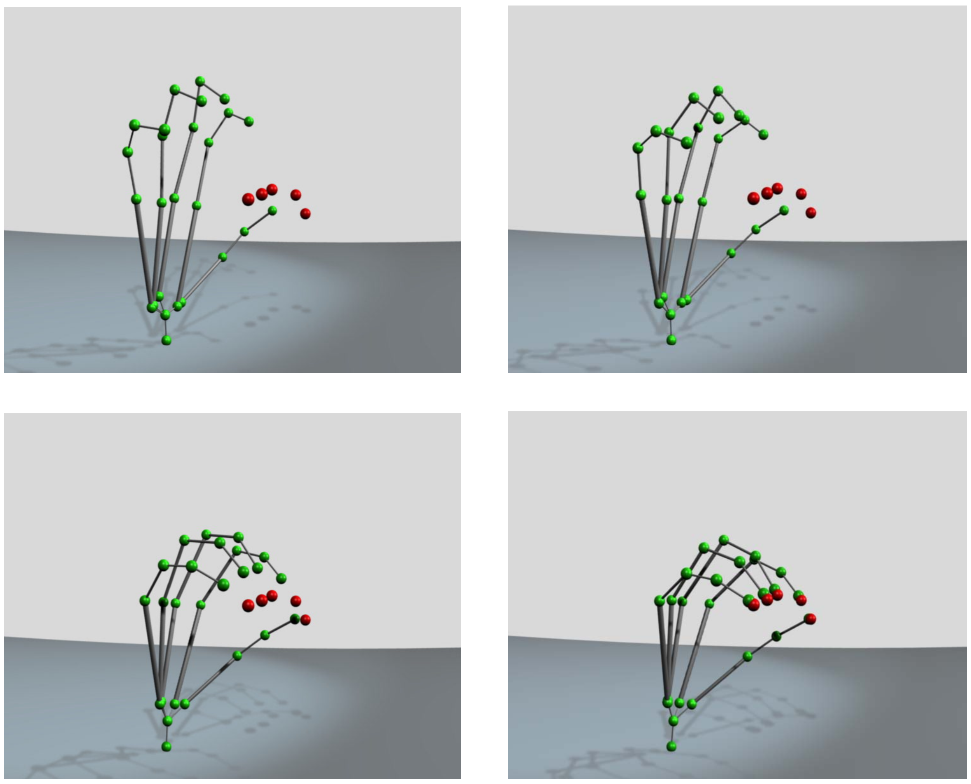


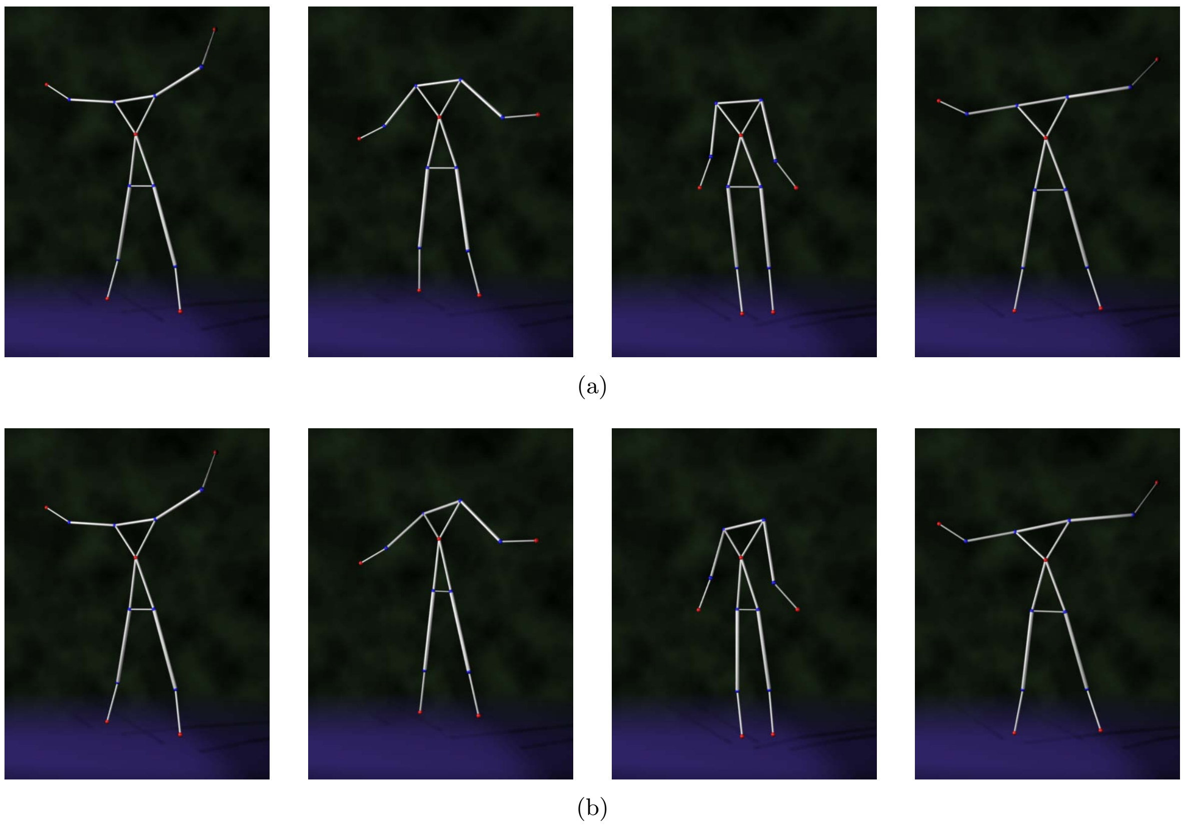
Figure 3.23.: *Example of FABRIK implementation with multiple end effectors over time. This is a fully unconstrained hand example, allowing 3 DoF on each joint.*

cost and reconstruction quality; it requires the fewest iterations to achieve the desired posture and produces visually the smoothest poses. The median error presented in table 3.2 refers to the difference between the estimated joint positions and the true joint positions. A video included in the supplementary materials demonstrates FABRIK and compares its performance against other state of the art methods.

3.6.3. 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 3.4.1). 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 3.6.2, 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 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



(a)

(b)

Figure 3.24.: *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.*

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 3.26, shows examples of FABRIK and CCD implementations with and without joint restrictions. On this example, rotational limits have been applied restricting the allowed bending of each joint angle to a maximum value. Figure 3.27 shows the reconstruction improvement between the unconstrained and constrained versions of FABRIK applied to the humanoid model; rotational and orientational constraints have been employed on each joint limiting the angle and the twist between limbs to a feasible set. Finally, table 3.3 shows the number of iterations and the time needed to reach the target for both the unconstrained and constrained FABRIK and CCD approaches.

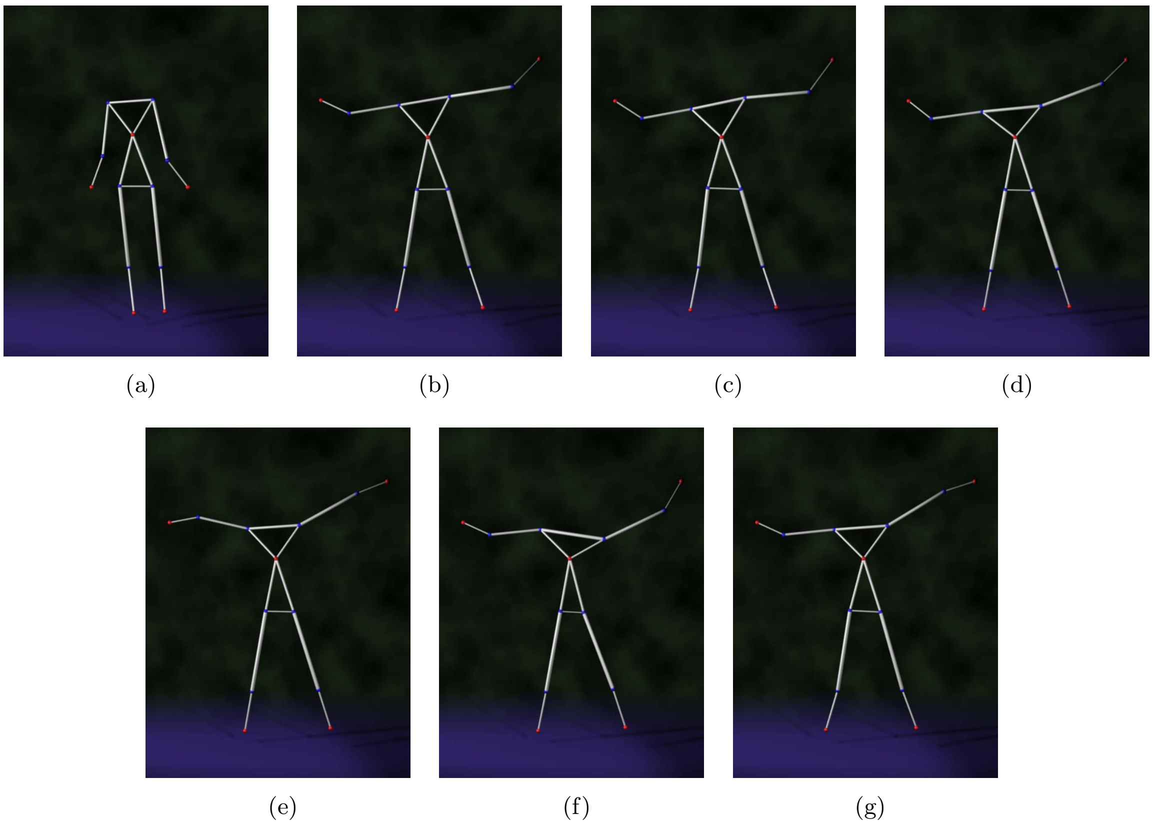


Figure 3.25.: *Body reconstruction using different IK methodologies. The joints in red are the known positions of the end effectors and those in blue are the estimated joint positions. (a) shows the initial position and (b) the true final position. (c) shows the FABRIK solution, (d) the CCD solution, (e) the J. Transpose solution, (f) the J. DLS solution, (g) the J. SVD-DLS solution.*

3.7. Applications

FABRIK has been successfully used for real-time marker prediction and centre of rotation (CoR) estimation [AL10c]. 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 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.

FABRIK's performance has been also tested for hand tracking and reconstruction [AL10b]. FABRIK captures 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 is effective and real-time implementable.

Table 3.3.: *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

3.8. Conclusions

IK methods are used to control the postures of articulated bodies in frame animation production. However, most of the currently available methods suffer from high computational cost and production of unrealistic poses. In this report, FABRIK, a simple, fast and reliable IK solver is presented. This is the first algorithm to use 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. Also, a reliable methodology for applying joint restrictions, which supports and utilises all the advantages of FABRIK, is presented. 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 fewer iterations to reach the target than any other IK method tested, both with constrained and unconstrained kinematic chains. At the same time, it produces visually smooth postures, with and without constraints, reaching the desired position with very low computational cost. FABRIK can also be 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

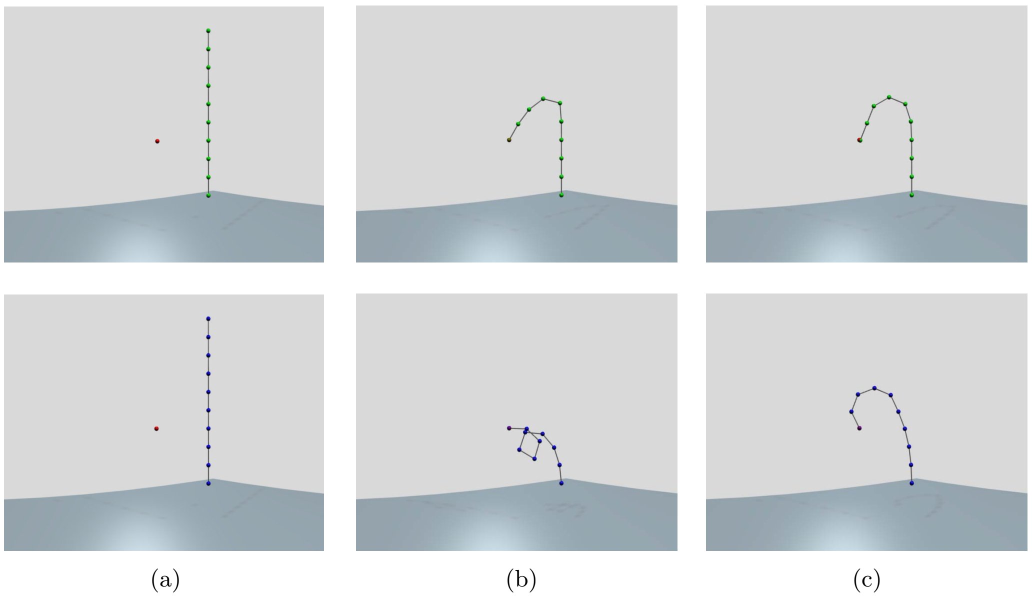
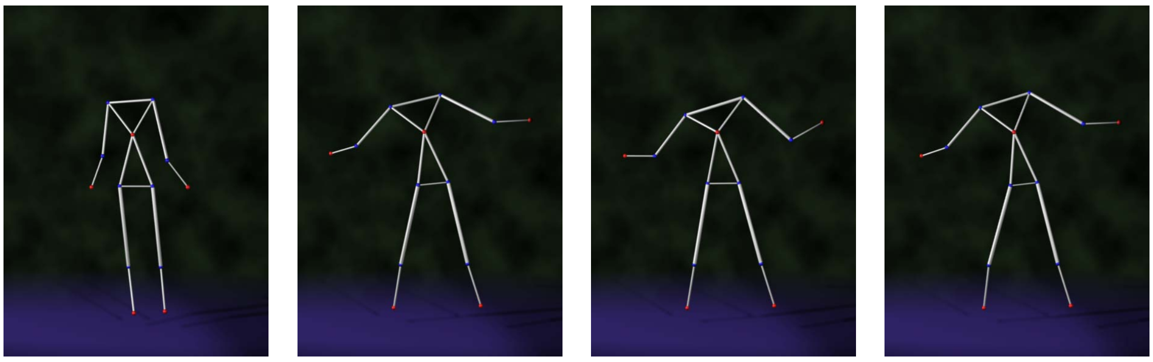


Figure 3.26.: *An example of FABRIK and CCD implementations with and without incorporating constraints. The top line shows the FABRIK solution and the bottom line the CCD solution. (a) the initial position of the kinematic chain, (b) the unconstrained solution, (c) the constrained solution.*

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.

The FABRIK algorithm is a powerful tool that can be used to provide reliable solutions to the IK problem both in robotics and computer vision. Chapters 4 and 5 demonstrate examples of



(a) (b) (c) (d)

Figure 3.27.: *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.*

implementing FABRIK in motion capture problems for real-time marker prediction and centre of rotation estimation as well as hand tracking and reconstruction.

4
Motion Capture Solutions under Marker Occlusions

THIS chapter addresses the problem of real-time joint localisation of legged skeletons in the presence of missing data. The data is assumed to be labelled 3D marker positions from a motion capture system. An integrated framework is presented which predicts the occluded marker positions using a variable turn model within an Unscented Kalman filter. Inferred information from neighbouring markers is used as observation states; these constraints are efficient, simple and real-time implementable. This work also takes advantage of the common case that missing markers are often still visible to a single camera, resulting in more accurate predictions. An Inverse Kinematics technique is then applied ensuring that the bone lengths remain constant over time; the system can thereby maintain a continuous data-flow. The marker and Centre of Rotation (CoR) positions can be calculated with high accuracy even in cases where markers are occluded for a long period of time. Our methodology is tested against some of the most popular methods for marker prediction and the results confirm that our approach outperforms these methods in estimating both marker and CoR positions.

4.1. Introduction

Optical motion capture is a technology used to turn the observations of a moving subject (taken from a number of cameras) into 3D position and orientation information about that subject. It is commonly used to better analyse techniques for sports training [HNT$^+$06]; to observe asymmetries and abnormalities in rehabilitation medicine (clinical analysis) [HDSB05, PTP$^+$05, BSR07]; for biomechanics (prosthetics, ergonomics); to study the person's movements for medical reasons or sport performance [Gol]; for gait labs; or for visualisation of virtual characters for films and computer games [Men99]. In general, the problem of automatic skeleton generation can be separated into three stages. First is the marker clustering, then the problem of finding the joint location and finally, the identification of the full body. In order to achieve accurate skeletal reconstruction of any legged body using optical motion capture systems, 3 markers must be available on each limb segment at all times. However, even with many cameras, there are instances where occlusion of markers by elements of the scene leads to missing data. In order to unambiguously establish its position, each marker must be visible to at least two cameras in each frame. Although many methods have been developed to handle the missing marker problem, most of them are not applicable in real-time, are usually limited to short time period occlusions and often require manual intervention.

This chapter investigates methodologies for real-time marker prediction, under multiple cases of occlusion, to drive CoR estimates and then to automatically establish the skeleton model. A real-time integrated framework is presented, which predicts the occluded marker positions using a variable turn model within an Unscented Kalman filter. The previous marker positions are used within the framework in addition to information related to the missing markers of the current frame, inferred from an approximate rigid body assumption. The predicted marker positions are then used to locate the joints. Without assuming any skeleton model, we take advantage of the fact that for markers on a given limb segment, the inter-marker distance is approximately constant. The proposed marker constraint methodology is simple, real-time implementable and very efficient. Our method is automatic and scalable, without requiring any parameters to be set by the user. It considers all the cases of marker occlusion within a limb resulting in accurate predictions even in cases where all markers on a limb segment are missing for an extended period of time. The proposed approach also takes advantage of the special, but common, case where missing markers are still visible to one camera. With a continuous stream of accurate labelled 3D data, we can perform real-time CoR estimation; the CoR position is thereafter corrected via a real-time Inverse Kinematic technique which ensures that the inter-joint pairwise distances remain constant over time. A skeletal reconstruction is thereby achieved, producing information which can be used for visual performance feedback. Experiments demonstrate that our methodology effectively recovers good estimates of the true positions of the missing markers and CoRs, agreeing with human intuition, even if all the markers on a limb are occluded for a long period of time. The movements produced are smooth, without abnormalities or oscillations, resulting in natural motion.

4.2. Obtaining 3D Marker Positions

Motion capture hardware, such as that provided by PhaseSpace [Pha], CodaMotion [Cod] and Vicon [Vic], is under constant development, providing real-time acquisition of labelled

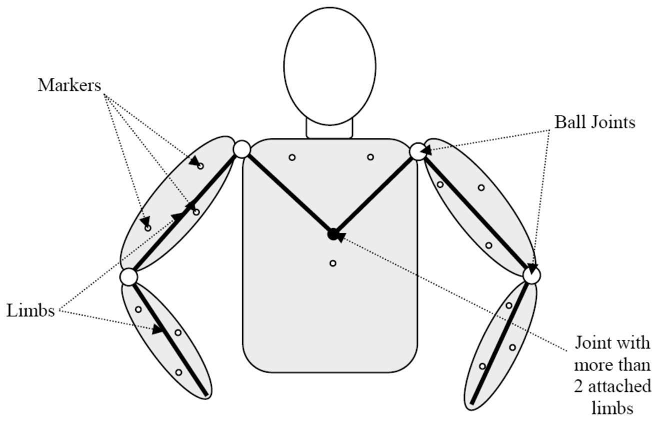


Figure 4.1.: *Marker-to-limb association example with three markers on each limb. This example shows the upper body with the marker-to-limb association.*

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.

In order to overcome the tracking problem, we used a 16 camera and 480 *frames per second* motion capture system using modulated LEDs, provided by PhaseSpace [Pha]. These cameras contain a pair of linear scanner arrays operating at high frequency each of which can capture the position of any number of bright spots of light as generated by the LEDs. The markers consist of a control circuit and bright red LEDs which are wired to a synchronisation box.

For reconstruction it is necessary for each marker to be visible by at least two cameras in each frame. The system offers a fast rate of capture (480Hz) and allows the individual markers to be identified by combining the information from several frames and hence identifying the marker from its unique modulation. The markers are placed at strategic points on the articulated body so that these points can be easily and accurately located by the cameras. It is desirable to allocate the markers such that there is the same number of markers on each limb, if possible, in order to obtain the same quality of data for all parts of the body we are interested in. The subject moves in a specified space that can be tracked by the cameras and the markers attached to its body are tracked over time and used to reconstruct the three-dimensional pose of the subject at each instant of time.

4.2.1. Marker-to-limb association

Since the three-dimensional trajectory of each marker is available we are able to determine which markers are attached to which limb on the body. This is done in two steps:

1. Firstly, markers attached to the same limb are grouped. This is done by finding which markers maintain the same distance from each other throughout the motion. Since the data is noisy we expect that the distance between the markers does not remain constant. Therefore, the variances of the distances between markers are calculated and the markers are clustered as belonging to the same limb if the variance of the distance between them is less than a certain threshold. [JMF99] gives a survey on clustering techniques.

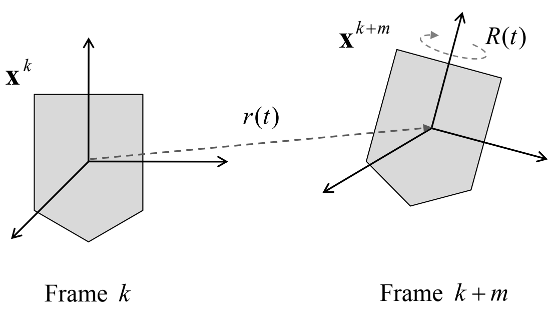


Figure 4.2.: *Description of the rigid body. The vector $r(t)$ specifies the position of the centre of mass, relative to the origin. The rotor $R(t)$ defines the orientation of the body, relative to a fixed copy placed at the origin. The $\mathbf{x}^k$ is a vector in a reference body (in this example in frame k), and $\mathbf{x}^{k+m}$ is a vector in space of the equivalent point of the moving body (in this example after m frames).*

2. Afterwards, association of the groups of markers to specific limbs of the body is necessary. This is done either manually or automatically by first calculating the distance between the centroid of each group of markers in a given time frame and then, according to the model used, the limbs are identified. It is assumed that the skeleton consists of rigid limbs connected with ball joints.

An example of associated markers in an upper body model is presented in figure 4.1.

4.2.2. Rigid body dynamics

A rigid body can be viewed as a system of particles moving subject to the constraint that all inter-particle distances are fixed. The final body position can be expressed in terms of a rotation and translation from a fixed “reference” body on to the body in space (figure 4.2). We let $r(t)$ denotes the position of the centre of mass and $\mathbf{x}^{k+m}$ denote the position in space of a point in the body. These are related by:

$$\mathbf{x}^{k+m} = R\mathbf{x}^k\tilde{R} + r(t) \tag{4.1}$$

where $\mathbf{x}^k$ is a fixed constant vector in the reference copy of the body. In this manner we have placed all of the rotational motion in the time-dependent rotor $R(t)$; rotors will be regularly used later on this chapter.

4.3. Related Work

Several papers have focused on methods for localisation of the centre of rotation (CoR) using MoCap data. These methods can be separated into two major groups, the *sphere fitting* methods and the *transformational* methods. Both families of methods assume that at least 3 markers are available on each limb segment; there are, although, instances where marker data are unavailable due to marker occlusions.

Sphere fitting: Sphere fitting methods are the most commonly used methods for calculating the CoR. This group of methods assumes that all markers remain a constant distance from the centre of rotation. A first step for solving this problem is to find a rigid body transformation moving the problem into a reference frame (e.g. *frame 1*) so that the limb segment on one side of the joint is viewed as stationary and then solve a simpler one-sided problem. Dorst, in [Dor05], presents a first order approximation which also illustrates the dependence of the attitude estimation error on the distribution of the point cloud (in this case the marker locations on the limb segments).

In [SPB+98] the Levenberg-Marquardt method was implemented in order to optimise the location of the centre of rotation and the radii of the marker spheres. A cost function $S = (|x_m^k - C| - r_m)^2$ was introduced, where x_m^k is the marker position of the marker m at frame k, C is the centre of the sphere and r_m is the radius of the sphere associated with the marker m. The overall cost is the sum of the individual costs over all markers and all frames. This algorithm requires a series of weights to be set, such as perceived accuracies of markers and a spatial reweighting of data points via a voxel grid. This method also has disadvantages; the inaccuracy of the heuristics used to set the weight parameters and the non-linear nature of the solver, making it susceptible to problems with local minima.

Halvorsen et al., [HLL99], describe a closed form solution using the geometric properties of the sphere. They used the fact that the perpendicular bisectors of chords of a sphere intersect at the sphere origin, and every pair of frames provide an approximate perpendicular bisector for each marker. However, this method is dependent on which data points are used to form the chords, affecting the accuracy and effectiveness of the algorithm.

In [GL02], Gamage and Lasenby also introduced a closed form solution, using a cost function of the squared differences in the squared distance from the CoR C, to a marker m, at position x_m^k in frame k, and the radius of the sphere associated with the marker r_m. That is

$$\left((C - x_m^k)^2 - (r_m)^2\right)^2 \tag{4.2}$$

An alternative approach provided by Halvorsen, [Hal03], gives a Bayesian analysis of the algorithm of [GL02], providing a first order approximation of the effect of isotropic Gaussian noise upon the algorithm. An extension to [Hal03] can be achieved by assuming that the measured points are the result of Gaussian noise only in the axial direction[1].

Transformational: The transformational method assumes that all positions of the markers are rigidly attached to limb segments. Such an approach was implemented in [Hol91]. A technique for using magnetic motion capture data to determine the joint parameters of an articulated hierarchy was described in [OBBH00]. This technique makes it possible to determine limb lengths, joint locations, and sensor placement for a human subject without external measurements, just from the motion data acquired during the capture session. The parameters are computed by performing a linear least squares fit of a rotary joint model to the input data.

[1]The axial direction is that parallel to a line from the centre of the sphere to the true marker location.

In many recent papers, such as [ETDH06], the same techniques are applied but the orientation is obtained from sets of optical markers. In [CL05] a sequential algorithm was presented to locate the rotation centres of a human skeleton from marker data assuming that all markers on a body segment are attached to a rigid body. This method does not suffer from optimisation steps with computational requirements that grow with the amount of data supplied ([GL02], [KOF05]) and no user feedback to set marker weights, as in [SPB$^+$98], are needed. The method is closed form, thus enabling real-time implementation.

Whilst several methods to estimate the location of centre of rotation have been proposed, the performance of most is unsatisfactory in the presence of unusual motions or of many contiguous occlusion-affected frames. Indeed, there are instances, even with expensive motion capture systems, where occlusion of markers by elements of the scene leads to missing data. Many of these methods behave suboptimally with diverse motions, a high percentage of missing markers and/or external occlusions. Several methods have been proposed to deal with the problem of occluded markers in order to drive CoR estimation and skeletal reconstruction, but these do not generally run in real-time and often require manual intervention. Interpolation of the missing data using linear or non-linear approaches is commonly used [WH97, RCB98, Neb99a, AW00, Neb99b]; this can produce accurate results, but it is a post-processing technique requiring data prior to and after the occlusion. Some motion capture (MoCap) systems also provide missing markers recovery solutions using interpolation techniques in combination with kinematic information, but they do not reliably work in real-time. Recently, [PLH$^+$09] presented an extrapolation algorithm which assumes that the most common motion behaviors are circular or linear movements; however, this method can produce reliable predictions only for a limited number of occluded frames. While the discarding of frames with missing markers is another common technique, the omission of specific data could lead to the loss of useful information. Long-running occlusions leading to a large sequence of missing data can also cause complete failure of the system.

Rhijn and Mulder, [RM06], proposed a model-based optical tracking and model estimation system for composite interaction devices. The proposed system automatically constructs the geometric skeleton structure, degrees of freedom (DoF) relations, and DoF constraints between segments, and thus pre-defined models are not required. The system supports segments with only a single marker, so that interaction devices can be small with a low number of markers. However, it is an off-line procedure unsuitable for real-time applications. Dorfmüller in [DU03] used an extended Kalman filter (EKF) to predict the missing markers using previously available marker information, while Welch et al. in [WBV$^+$99] used an EKF to resolve occlusions based on the skeletal model of the tracked person. Tak and Ko, [TK05], employed an Unscented Kalman Filter to ensure motion capture data remains in a feasible set. Again, these methods require manual intervention or become ineffective in cases where markers are missing for an extended period of time. In our earlier work, [ACL08], we presented an EKF method using a constant velocity (CV) model with marker constraints from neighbouring[2] markers. However, the CV model (2$^{\text{nd}}$ order Kalman Filter (KF)) limits its use to problems with constant marker

[2] *Neighbours* are the markers belonging to the same limb segment.

velocity. These methods also do not take into consideration the fact that bones are rigid, thus the inter-joint pairwise distances remain constant over time. Li et al [LMPF09] propose DynaMMo, an approach that uses a Linear Dynamical System (LDS) to model motion capture data under sequences with missing values; in [LMPF10], the same authors introduce BoLeRo, a similar technique which also takes into consideration bone length constraints. The suggested algorithm has a 'hard' and 'soft' version of bone constraints assuming rigidity limitations of the distance between markers on a given segment limb. Although their algorithm results in smooth motion, the method is complex and expensive in terms of computational cost.

Herda et al., in [HFP$^+$00] and [HFP$^+$01], used a post-processing approach to increase the robustness of a motion capture system by using a sophisticated human model. They can predict the 3D location and visibility of the markers increasing the robustness of the marker tracking and reducing the need for human intervention during the reconstruction process. The neighbouring markers that share kinematic relations with the occluded markers were used to help the estimation of the isolated markers. However, the skeleton information must be known a priori in order to apply this method. [HSD05] also takes advantage of the fact that the markers on a limb have fixed inter-marker pairwise distances. Thus, in the case where a marker is missing, its position can be recovered through the distance constraints imposed by the markers of the same limb. This approach may become ineffective when all or a significant number of markers are missing so that no information on that limb can be inferred from the available neighbouring markers. Ringer and Lasenby, [RL02], also present an automatic method to identify indistinguishable markers based on cliques[3]. However, this requires an off-line procedure in order to determine marker cliques and parameters of the skeletal structure. Zordan and Van Der Horst in [ZVDH03] mapped 3D marker position data to joint trajectories for a fixed limb-length skeleton, by attaching virtual springs and controllers, to follow the Cartesian-based marker data. In general, these skeleton methods could work well for short time occlusions but fail to track the missing markers for large occluded sequences.

In [GMHP04], a style-based inverse kinematic method has been developed where a Gaussian Process Latent Variable Model (GPLVM) was used along with a pre-specified kinematic model. Wang et al, [WFH08], presented a Gaussian process dynamical model (GPDM) in order to learn the human pose and motion models. They observed the motion using a chain of latent variables and nonlinear mapping from the latent space; the proposed learned model was also able to cope with marker occlusions. Although these are real-time processing methods, they require knowledge of skeleton information which severely restricts their use. Chai and Hodgins, [CH05], present a method that uses the neighbouring markers to estimate the missing marker in the current frame. They propose a local linear model from these neighbours and then reconstruct the full pose of the frame by conducting an optimisation in the space constrained by a pre-recorded database. Yu et al., [YLD07], proposed an online motion capture labelling approach which also recovers missing markers. They cluster the markers into a number of rigid bodies based on the standard deviations of the marker-pair distances and if their fitting-rigid-bodies algorithm did not classify all the markers into rigid bodies, a missing marker auto-recovery

[3]Markers in a *clique* have constant distances between each other.

method is applied assuming that the inter-marker distances are fixed over time. However, a training session is needed, the auto-recovery method for marker estimation does not take into account the limb segment rotation, no information about markers visible to a single camera is considered and the CoR estimation is not investigated under marker occlusions.

Park and Hodgins, [PH06], fill the missing data by learning a statistical model of the spatial relationship between each marker and its neighbours; they use a Principal Component Analysis (PCA) on each marker position and its neighbours throughout the entire motion. In [THR07], the authors modelled the human motion and filled the gaps in the data using a Conditional Restricted Boltzmann Machine (CRMB) with discrete hidden states. Their approach was trained using non-linear binary representations, conditioned on previous frames; at the same time, they took into consideration the correlation between joint angles, to produce more accurate results. Liu et al., [LZWM06, LM06], presented a piecewise linear approach for estimating human motions from a pre-selected set of informative markers (principal markers). A pre-trained classifier identifies an appropriate local linear model for each frame. Missing markers are then recovered using available marker positions and the principal components of the associated model. In [HGP04], the data were mapped onto a target motion by searching over patterns in existing databases. However, this data-driven family of methods requires an off-line training procedure and the results are highly dependent on training data and limited to those models and movements the system has been trained on. So far, no existing method operates inter-joint constraints to maintain a constant bone length over time. Also, no methodology currently exploits the useful information we can extract from the special, but frequent, cases where markers are visible to only a single camera.

4.4. Calculating the Centre of Rotation

Locating the CoRs is important in both computer graphics and rehabilitation medicine. During capture, markers must be carefully placed on the body in order to obtain good results. Results using markers placed too close to the CoR are more susceptible to errors since a small error may cause large deviations in the estimated rotation, leading to erroneous calculation of the model parameters. The data discussed here are labelled from an active marker system (PhaseSpace [Pha]) where no tracking is necessary. In general, 3 markers per bone segment are required to estimate the CoR for joints with 3 DoF; for simpler problems having fewer DoF, such as knees and elbows, the CoR can be calculated with fewer markers [CP07].In this report, we consider the general case of joints with 3 DoF since no prior knowledge of the model or joint-type is assumed to be known.

4.4.1. Finding the rotor between 2 sets of vectors

The CoR estimation is a crucial step in acquiring a skeleton from raw motion capture data. To calculate the joints between two sets of markers, it is helpful to have the rotation of a limb at any given time. We can estimate the orientation of a limb at time k relative to a reference frame using the *Procrustes* formulation, as described by Horn in [Hor87].

Assume a set of n-labelled points x_i and the same set of points after an unknown rotation $R(t)$, w_i; the problem of finding a least square solution of the unknown rotor (unit quaternion) R can be formulated as,

$$R = \arg\max \sum_{i=1}^{n} \left(R x_i \tilde{R} \right) \cdot w_i \tag{4.3}$$

In order to avoid introducing quaternion terminology and methods, we can express the solution in terms of Geometric Algebra. To clarify the following equations, the marker index i will temporally be suppressed. Let:

$$R = a + B_1 e_{23} + B_2 e_{13} + B_3 e_{12} \tag{4.4}$$

$$x_i = x_1 e_1 + x_2 e_2 + x_3 e_3 \qquad\qquad w_i = w_1 e_1 + w_2 e_2 + w_3 e_3 \tag{4.5}$$

where a is a scalar, e_1, e_2, e_3 are orthogonal vectors in the 3-D space, and e_{23}, e_{31} and e_{12} are the unit bivectors, as described in chapter 2. Each of these encodes a distinct plane, and there are 3 of them to match the 3 independent orthogonal planes in the 3-D space. Then:

$$\begin{aligned}(R x_i \tilde{R}) \cdot w_i = {} & a^2 (x \cdot w) + 2aB_1(x_3 w_2 - x_2 w_3) + 2aB_2(x_3 w_1 - x_1 w_3) \\ & + 2aB_3(2x_2 w_1 - 2x_1 w_2) + B_1^2(x_1 w_1 - x_2 w_2 - x_3 w_3) \\ & + 2B_1 B_2(-x_2 w_1 - x_1 w_2) + 2B_1 B_3(x_3 w_1 + x_1 w_3) \\ & + B_2^2(-x_1 w_1 + x_2 w_2 - x_3 w_3) + 2B_2 B_3(-x_3 w_2 + x_2 w_3) \\ & + B_3^2(-x_1 w_1 - x_2 w_2 + x_3 w_3)\end{aligned}$$

Eq. 4.3 can now be expressed as follows:

$$R = \arg\max \sum_{i=1}^{n} \left(R_v^T N R_v \right) \tag{4.6}$$

where:

$$R_v = \begin{bmatrix} a \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

$$N = \begin{bmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 & x_3 w_2 - x_2 w_3 & x_3 w_1 - x_1 w_3 & x_2 w_1 - x_1 w_2 \\ x_3 w_2 - x_2 w_3 & x_1 w_1 - x_2 w_2 - x_3 w_3 & -(x_2 w_1 + x_1 w_2) & x_3 w_1 + x_1 w_3 \\ x_3 w_1 - x_1 w_3 & -(x_2 w_1 + x_1 w_2) & -x_1 w_1 + x_2 w_2 - x_3 w_3 & x_3 w_2 - x_2 w_3 \\ x_2 w_1 - x_1 w_2 & x_3 w_1 + x_1 w_3 & x_3 w_2 + x_2 w_3 & -x_1 w_1 - x_2 w_2 + x_3 w_3 \end{bmatrix}$$

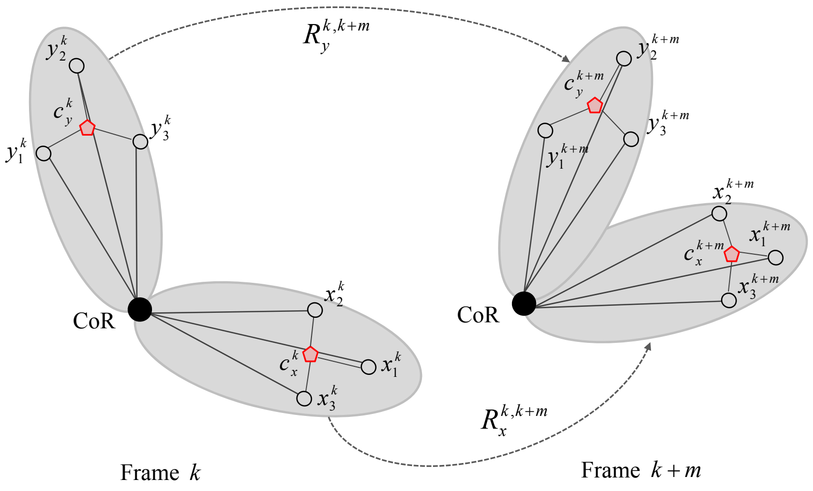


Figure 4.3.: *The motion of two limbs with a time-difference of m frames.*

In terms of limb motion, it is noted that x_1, x_2, x_3 and w_1, w_2, w_3 correspond to values given below, with x_i^k, y_i^k, c_x^k and c_y^k be the centre of mass of the 3 markers, as shown in figure 4.3.

$$\begin{aligned} x_1 &= x_1^k - c_x^k & \qquad w_1 &= x_1^{k+m} - c_x^{k+m} \\ x_2 &= x_2^k - c_x^k & w_2 &= x_2^{k+m} - c_x^{k+m} \\ x_3 &= x_3^k - c_x^k & w_3 &= x_3^{k+m} - c_x^{k+m} \end{aligned} \tag{4.7}$$

$$c_x^k = \frac{x_1^k + x_2^k + x_3^k}{3} \qquad c_x^{k+m} = \frac{x_1^{k+m} + x_2^{k+m} + x_3^{k+m}}{3} \tag{4.8}$$

As a valid rotor must always obey the property $R\tilde{R} = 1$ it must be the case that $R_\upsilon^T R_\upsilon = 1$. If this condition is relaxed and normalisation is introduced into the above matrix expression we have:

$$R \quad = \quad \arg\max \left(\frac{R_\upsilon^T \sum_{i=1}^{n} (N) \, R_\upsilon}{R_\upsilon^T R_\upsilon} \right) \tag{4.9}$$

This expression is now in the form of the Rayleight Quotient; $\sum_{i=1}^{n} (N)$ is clearly Hermitian and the value of R at the maximum value of the expression to be maximised. Thus, provides a least squares estimate of the required rotor, where the eigenvector associated with the greatest eigenvalue maximises the matrix product. Therefore the rotor R_υ which corresponds to the rotation is equal to that eigenvector.

Hereafter, in this thesis, vectors will be symbolised in bold font to distinguish them from other symbols and avoid any confusion.

4.4.2. Joint localisation

After extracting the rotation of two limbs between the current and a reference frame, the location of the joints can be calculated using the approach in [CL05]. This takes advantage

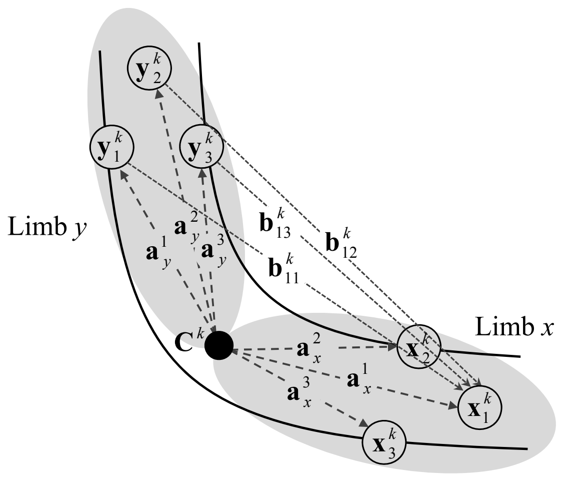


Figure 4.4.: *A typical marker placement with relevant markers shown.*

of the approximation that all markers on a segment are attached to a rigid body. Suppose the markers are placed on two segments (x and y) joined by a CoR. Let the CoR location in frame k be $\mathbf{C}_k$. The vectors from the CoR to markers in the reference frame are denoted by $\mathbf{a}_x^i$ and $\mathbf{a}_y^j$ for limbs x and y respectively, where i and j are marker labels. Figure 4.4 presents a typical marker placement with the $\mathbf{b}_{ij}^k$, $\mathbf{a}_x^i$ and $\mathbf{a}_y^j$ as shown. The positions of the markers in frame k are given by:

$$\mathbf{x}_i^k = \mathbf{C}^k + R_x^k \mathbf{a}_x^i \tilde{R}_x^k \qquad \mathbf{y}_j^k = \mathbf{C}^k + R_y^k \mathbf{a}_y^j \tilde{R}_y^k \tag{4.10}$$

where R_x and R_y are the rotors expressing the rotation of the joint limbs x and y respectively. $\tilde{R}$ is the reverse of R. Let $\mathbf{b}_{ij}^k$ be the vector from $\mathbf{x}_i^k$ to $\mathbf{y}_j^k$, that is:

$$\mathbf{b}_{ij}^k = \mathbf{x}_i^k - \mathbf{y}_j^k = R_x^k \mathbf{a}_x^i \tilde{R}_x^k - R_y^k \mathbf{a}_y^j \tilde{R}_y^k \tag{4.11}$$

A cost function S can be constructed that has a global minimum at the correct values of $\mathbf{a}_x^i$ and $\mathbf{a}_y^j$ if the data is noise free, and returns a good estimate in the presence of moderate noise.

$$S = \sum_{k=1}^{m} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left[\mathbf{b}_{ij}^k - \left(R_x^k \mathbf{a}_x^i \tilde{R}_x^k - R_y^k \mathbf{a}_y^j \tilde{R}_y^k \right) \right]^2 \tag{4.12}$$

where n_x, n_y are the number of markers on limbs x and y respectively, and m is the number of frames used for the calculations. The minimum is given by the solution of the simultaneous linear equations, obtainable by differentiation [CL05]:

$$\mathbf{a}_x^i = \frac{1}{m} \sum_{k=1}^{m} \tilde{R}_x^k \bar{\mathbf{b}}^k R_x^k + \frac{1}{m} \sum_{k=1}^{m} \tilde{R}_x^k R_y^k \bar{\mathbf{a}}_y \tilde{R}_y^k R_x^k \tag{4.13}$$

$$\mathbf{a}_y^j = \frac{1}{m} \sum_{k=1}^{m} \tilde{R}_y^k \bar{\mathbf{b}}^k R_y^k + \frac{1}{m} \sum_{k=1}^{m} \tilde{R}_y^k R_x^k \bar{\mathbf{a}}_x \tilde{R}_x^k R_y^k \tag{4.14}$$

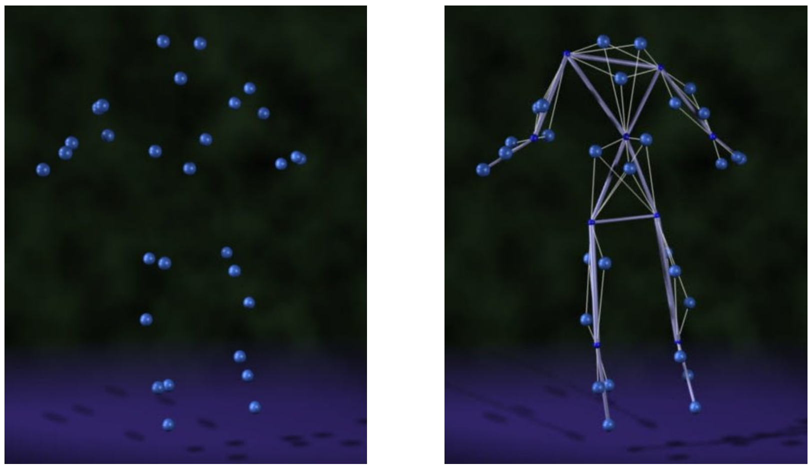

(a) (b)

Figure 4.5.: *An example of CoR estimation of a human body model using [CL05]. (a) The marker positions as returned by the motion capture system, (b) the calculated CoRs and the skeletal reconstruction.*

where

$$\bar{\mathbf{b}}^k = \frac{1}{n_x n_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \mathbf{b}_{ij}^k \qquad \bar{\mathbf{a}}_w = \frac{1}{n_w} \sum_{i=1}^{n_w} \mathbf{a}_w^i \qquad w = \{x, y\}$$

Having calculated the R_w^k and $\bar{\mathbf{a}}_w$, we can locate the CoR. Figure 4.5 demonstrates an example of CoR estimation and skeletal reconstruction in real-time using the above method. However, due to occlusions, there are instances where not all marker positions are available. If all markers are available on one limb segment, w, the CoR may be estimated using only the current R_w^k and $\bar{\mathbf{a}}_w$ as estimated in the previous frame, when all markers were visible, via (4.10). If there are markers occluded on both limb segments, a marker prediction methodology is needed.

4.5. Marker Prediction

The marker position estimates can by predicted using filtering, with each single marker tracked individually and incorporating constraints from the neighbouring markers. Most tracking problems require a dynamic model for accurate estimation of the trajectory of a maneuverable object. The key for an efficient target tracking algorithm is being able to extract information about the target's state from the observations. Hence, a model that takes account of velocity and acceleration changes of the target state over time is needed.

During the last decades various mathematical models of target motions have been developed. Singer ([Sin70, SB71]) proposes a model which assumes that the target acceleration is a zero-mean first-order stationary Markov process. Based on *Singer's* assumption, many papers have proposed a constant or variable acceleration (e.g [LJ03]).

The general state-space model can be generally divided into two models, the state transition and the state measurement model

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad p(\mathbf{y}_t|\mathbf{x}_t) \tag{4.15}$$

where $\mathbf{x}_t \in \mathbb{R}^{n_x}$ denotes the state of the system at time t and $\mathbf{y}_t \in \mathbb{R}^{n_y}$ the observations. The states follow a first order Markov process and the observations are assumed to be independent given the states. Therefore, the dynamic model can be expressed as follows

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) \qquad \mathbf{y}_t = \mathbf{h}(\mathbf{u}_t, \mathbf{x}_t, \mathbf{n}_t) \tag{4.16}$$

where, $\mathbf{y}_t \in \mathbb{R}^{n_y}$ is the output observation, $\mathbf{u}_t \in \mathbb{R}^{n_u}$ is the input observation, $\mathbf{x}_t \in \mathbb{R}^{n_x}$ is the state of the system, $\mathbf{v}_t \in \mathbb{R}^{n_v}$ is the process noise and $\mathbf{n}_t \in \mathbb{R}^{n_\mathbf{n}}$ the measurement noise. The mapping $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \mapsto \mathbb{R}^{n_x}$ and $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\mathbf{n}} \mapsto \mathbb{R}^{n_y}$ represent the deterministic process and measurement models.

A simple and popular model used to track the motion of a target is the *nearly constant turn* (NCT) model [LJ03], which assumes that the target moves with nearly constant speed and turn rate. The altitude changes are most often modeled independently by a nearly *constant velocity* (CV) model or a random walk model along the $\mathbf{z}$ direction, leading to an acceptable accuracy in practice.

4.5.1. Variable Turn Model

The NCT and CV models were based on a constant-speed condition and constant turn-rate assumption which restricts the variety of possible supported maneuvers. The Variable Turn Model (VTM), [LJ03], is a more sophisticated model which assumes that a target's velocity and acceleration are not constant over time. Figure 4.6 compares target prediction with variable acceleration using the VTM model against a linear prediction.

It is easily seen that a constant speed (i.e $\dot{\nu} = 0$) motion corresponds to $\mathbf{a} \cdot \mathbf{v} = 0$, where $\mathbf{a}$ and $\mathbf{v}$ are the target acceleration and velocity vectors. This relationship is described by

$$\mathbf{a} = \boldsymbol{\Omega} \times \mathbf{v} \tag{4.17}$$

where $\boldsymbol{\Omega}$ is the angular velocity vector of the target and is equal[4] to

$$\boldsymbol{\Omega} = \frac{\mathbf{v} \times \mathbf{a}}{\nu^2} \tag{4.18}$$

where $\boldsymbol{\Omega} \cdot \mathbf{v} = 0$ and $\nu = \|\mathbf{v}\|$ is the target speed.

Asseo and Ardila in [AA82] describe a general motion of a rigid body in space. They show that, under the orthogonal velocity condition $\boldsymbol{\Omega} \perp \mathbf{v}$, the target acceleration can be expressed as a second-order Markov process with state dependence

$$\dot{\mathbf{a}} = -2\alpha\mathbf{a} - (2\alpha^2 + \omega^2)\mathbf{v} + \mathbf{w} \qquad \mathbf{w} = \ddot{\mathbf{v}}^{\mathcal{B}} + \dot{\boldsymbol{\Omega}} \times \mathbf{v} \tag{4.19}$$

[4] $\mathbf{v} \times \mathbf{a} = \mathbf{v} \times \dot{\mathbf{v}} = \mathbf{v} \times (\boldsymbol{\Omega} \times \mathbf{v}) = (\mathbf{v} \cdot \mathbf{v})\boldsymbol{\Omega} - (\boldsymbol{\Omega} \cdot \mathbf{v})\mathbf{v} = \nu^2\boldsymbol{\Omega} - (\boldsymbol{\Omega} \cdot \mathbf{v})\mathbf{v}$, if and only if $\boldsymbol{\Omega} \cdot \mathbf{v} = 0$, that is $\boldsymbol{\Omega} \perp \mathbf{v}$.

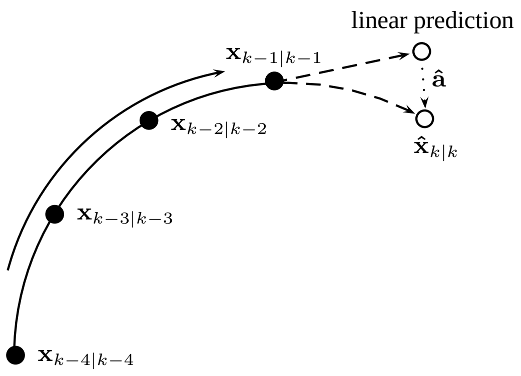


Figure 4.6.: *Target prediction with variable acceleration using the VTM model compared to the linear prediction.*

where if $\mathbf{v} = \nu^i e_i$, then $\mathbf{v}^{\mathcal{B}} = \dot{\nu}^i e_i$ (sum over i assumed) and

$$\omega \triangleq \|\mathbf{\Omega}\| = \frac{\|\mathbf{v} \times \mathbf{a}\|}{\nu^2} \qquad \alpha = -\frac{\mathbf{v} \cdot \mathbf{a}}{\nu^2} \tag{4.20}$$

Thus, the target equations of a 3D discrete motion of the VTM with state $\mathbf{x}_k = [x_k, \dot{x}_k, \ddot{x}_k, y_k, \dot{y}_k, \ddot{y}_k, z_k, \dot{z}_k, \ddot{z}_k]^T$ are now represented by

$$\mathbf{x}_k = \operatorname{diag}\left[F_x(\omega_{c,x}, \alpha_x), F_y(\omega_{c,y}, \alpha_y), F_z(\omega_{c,z}, \alpha_z)\right] \mathbf{x}_{k-1} + B\mathbf{u}_{k-1} + \mathbf{w}_{k-1}$$

where $F_i(\omega_{c,i}, \alpha_i)$ for each component $i = x, y, z$ is given by:

$$f^i_{11} = 1; f^i_{21} = f^i_{31} = 0$$

$$f^i_{12} = \frac{A - B}{\omega_{c,i}(\alpha_i^2 + \omega_{c,i}^2)}$$

where $A = 2\alpha_i \omega_{c,i}$ and

$$B = e^{-\alpha_i T}(2\alpha_i \omega_{c,i} \cos(\omega_{c,i} T) + (\alpha_i^2 - \omega_{c,i}^2)\sin(\omega_{c,i} T))$$

$$f^i_{13} = \frac{\omega_{c,i} - e^{-\alpha_i T}(\omega_{c,i}\cos(\omega_{c,i} T) + \alpha_i \sin(\omega_{c,i} T))}{\omega_{c,i}(\alpha_i^2 + \omega_{c,i}^2)}$$

$$f^i_{22} = \frac{e^{-\alpha_i T}(\omega_{c,i}\cos(\omega_{c,i} T) + \alpha_i \sin(\omega_{c,i} T))}{\omega_{c,i}}$$

$$f^i_{23} = \frac{e^{-\alpha_i T}\sin(\omega_{c,i} T)}{\omega_{c,i}}$$

$$f^i_{32} = -\frac{(\alpha_i^2 + \omega_{c,i}^2)e^{-\alpha_i T}\sin(\omega_{c,i} T)}{\omega_{c,i}}$$

$$f^i_{33} = \frac{e^{-\alpha_i T}(\omega_{c,i}\cos(\omega_{c,i} T) + \alpha_i \sin(\omega_{c,i} T))}{\omega_{c,i}}$$

and

$$\begin{aligned} Q &= \operatorname{cov}(\mathbf{w}_{k-1}) \\ &= \operatorname{diag}\left[S_x Q_x(\omega_{c,x}, \alpha_x), S_y Q_y(\omega_{c,y}, \alpha_y), S_z Q_z(\omega_{c,z}, \alpha_z)\right] \end{aligned}$$

where $\omega_{c,i}^2 = \alpha_i^2 + \omega_i^2$, S_x, S_y and S_z are the power spectral densities of the white noise $\mathbf{w}$ of each component x, y, z. $Q_i(\omega_{c,i}, \alpha_i)$ for $i = x, y, z$ is given by:

$$q_{11}^i = \frac{A + B + C}{4\alpha_i \omega_{c,i}^2 (\alpha_i^2 + \omega_{c,i}^2)^3}$$

$$A = e^{-2\alpha_i T}[(\alpha_i^2 - 3\omega_{c,i}^2)c + (\omega_{c,i}^2 - 3\alpha_i^2)s - (\alpha_i^2 + \omega_{c,i}^2)^2]$$

$$B = 8e^{-\alpha_i T}\alpha_i \omega_{c,i}[2\alpha_i \omega_{c,i} c_0 + (\alpha_i^2 - \omega_{c,i}^2)s_0]$$

$$C = \alpha_i^2 \omega_{c,i}^2 (4\alpha_i T - 11) + \omega_{c,i}^4 (1 + 4\alpha_i T)$$

$$q_{12}^i = \frac{e^{-2\alpha_i T}(\omega_{c,i} c_0 + \alpha_i s_0 - e^{-\alpha_i T}\omega_{c,i})^2}{2\omega_{c,i}^2 (\alpha_i^2 + \omega_{c,i}^2)^2}$$

$$q_{13}^i = \frac{e^{-2\alpha_i T}(c - s - \alpha_i + \omega_{c,i}^2) + 4e^{-\alpha_i T}\alpha_i \omega_{c,i} s_0 - \omega_{c,i}^2}{4\alpha_i \omega_{c,i}^2 (\alpha_i^2 + \omega_{c,i}^2)^2}$$

$$q_{22}^i = \frac{e^{-2\alpha_i T}(c - s - \alpha_i - \omega_{c,i}^2) + \omega_{c,i}^2}{4\alpha_i \omega_{c,i}^2 (\alpha_i^2 + \omega_{c,i}^2)^2}$$

$$q_{23}^i = \frac{e^{-2\alpha_i T} s_0^2}{2\omega_{c,i}^2}$$

$$q_{33}^i = \frac{e^{-2\alpha_i T}(c + s - \alpha_i - \omega_{c,i}^2) + \omega_{c,i}^2}{4\alpha_i \omega_{c,i}^2}$$

$$q_{21}^i = q_{31}^i = q_{32}^i = 0$$

$$c = \alpha_i^2 \cos(2\omega_{c,i} T), \qquad s = \alpha_i \omega_{c,i} \sin(2\omega_{c,i} T)$$

$$c_0 = \cos(\omega_{c,i} T), \qquad s_0 = \sin(\omega_{c,i} T)$$

4.5.2. Unscented Kalman Filter

Kalman filtering has been extensively used for real-time estimation of linear dynamic systems. However, the traditional Kalman filter [Kal60] is not suitable for use with non-linear dynamical systems, even if Gaussian approximations to the joint distribution of state $\mathbf{x}$ and measurement $\mathbf{y}$ are made. The Extended Kalman Filter (EKF) [Jaz70] is a minimum mean-square-error (MMSE) estimator which extends the scope of the Kalman filter to nonlinear optimal filtering problems. It forms a Gaussian approximation to the joint distribution of state and measurement using a Taylor series-based transformation. Nevertheless, EKF implementation is complex (Jacobian and Hessian matrices with second order filters are required), difficult to tune, and only reliable for systems that are almost linear on the timescale of the updates.

The Unscented Kalman Filter (UKF), [JU97], propagates mean and covariance information through nonlinear transformations providing more accurate results than the EKF, for a similar computational cost. Consider propagating an n_x-dimensional random variable $\mathbf{x}$ and assume $\mathbf{x}$ has mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_x$. First, a set of $2n_x + 1$ weighted samples or *sigma points* $\mathcal{S}_i = \{W_i, \mathcal{X}_i\}$ are deterministically chosen so that the true mean and covariance of the random variable $\mathbf{x}$ can be completely recovered from them. A set of scaled sigma points $\mathcal{S} = \{\mathbf{W}, \mathcal{X}\}$ can be calculated by setting:

$$\lambda = \alpha^2(n_x + \kappa) - n_x \tag{4.21}$$

and selecting the sigma point set by:

$$\begin{aligned} W_0^{(m)} &= \lambda / (n_x + \lambda) && i = 0 \\ W_0^{(c)} &= \lambda / (n_x + \lambda) + (1 - \alpha^2 + \beta) && i = 0 \\ W_i^{(m)} &= W_i^{(c)} = 1 / \{2(n_x + \lambda)\} && i = 1, ..., 2n_x \end{aligned}$$

where α is a positive scaling parameter which controls the size of the sigma point distribution. $\kappa \geq 0$ is also a scaling parameter and W_i is the weight associated with the i^{th} point such that $\sum_{i=0}^{2n_x} W_i = 1$. [MDFW00] proposed $\kappa = 0$, to guarantee positive semidefiniteness of the covariance matrix and $0 \leq \alpha \leq 1$ to avoid sampling non-local effects when the nonlinearities are strong. $\beta \geq 0$ is a weighting term which incorporates knowledge of the higher order moments of the distribution. $\beta = 2$ is the optimal choice for a Gaussian prior.

Let the original state and noise variables at time k be $\mathbf{x}_k^\alpha = [\mathbf{x}_k^T, \mathbf{v}_k^T, \mathbf{n}_k^T]$. The sigma point selection scheme is applied to this augmented state Random Variable (RV) to calculate the corresponding sigma matrix, $\mathcal{X}_k^\alpha$. The mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}$ of the Gaussian approximation is updated to the posterior distribution of the states as follows:

$$\begin{aligned} \bar{\mathbf{x}}_0 &= E[\mathbf{x}_0] \\ \mathbf{P}_0 &= E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T] \\ \bar{\mathbf{x}}_0^\alpha &= E[\mathbf{x}^\alpha] = [\mathbf{x}_0^T \; 0 \; 0]^T \end{aligned} \tag{4.22}$$

$$\mathbf{P}_0^\alpha = E[(\mathbf{x}_0^\alpha - \bar{\mathbf{x}}_0^\alpha)(\mathbf{x}_0^\alpha - \bar{\mathbf{x}}_0^\alpha)^T] = \begin{bmatrix} \mathbf{P}_0 & 0 & 0 \\ 1 & \mathbf{Q} & 0 \\ 1 & 0 & \mathbf{R} \end{bmatrix}$$

For $k \in \{1, ..., \infty\}$ the sigma points are equal to:

$$\mathcal{X}_{k-1}^a = \left[\bar{\mathbf{x}}_{k-1}^\alpha \; \bar{\mathbf{x}}_{k-1}^\alpha \pm \sqrt{(n_\alpha + \lambda)\mathbf{P}_{k-1}^\alpha}\right] \tag{4.23}$$

and the time update is given by:

$$\begin{aligned}
\mathcal{X}^x_{k|k-1} &= \mathbf{f}\left(\mathcal{X}^x_{k-1}, \mathcal{X}^\nu_{k-1}\right) \\
\bar{\mathbf{x}}_{k|k-1} &= \sum_{i=0}^{2n_a} W_i^{(m)} \mathcal{X}^x_{i,k|k-1} \\
\mathbf{P}_{k|k-1} &= \sum_{i=0}^{2n_a} W_i^{(c)} [\mathcal{X}^x_{i,k|k-1} - \bar{\mathbf{x}}_{k|k-1}][\mathcal{X}^x_{i,k|k-1} - \bar{\mathbf{x}}_{k|k-1}]^T \\
\mathcal{Z}_{k|k-1} &= \mathbf{h}\left(\mathcal{X}^x_{k|k-1}, \mathcal{X}^n_{k-1}\right) \\
\bar{\mathbf{z}}_{k|k-1} &= \sum_{i=0}^{2n_a} W_i^{(m)} \mathcal{Z}^x_{i,k|k-1}
\end{aligned}$$

where $\mathbf{f}(.)$ is the transition and $\mathbf{h}(.)$ the observation function. The measurement update equations are:

$$\begin{aligned}
\mathbf{P}_{\tilde{z}_k\tilde{z}_k} &= \sum_{i=0}^{2n_a} W_i^{(c)} [\mathcal{Z}_{i,k|k-1} - \bar{\mathbf{z}}_{k|k-1}][\mathcal{Z}_{i,k|k-1} - \bar{\mathbf{z}}_{k|k-1}]^T \\
\mathbf{P}_{x_k z_k} &= \sum_{i=0}^{2n_a} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \bar{\mathbf{x}}_{k|k-1}][\mathcal{Z}_{i,k|k-1} - \bar{\mathbf{z}}_{k|k-1}]^T \\
\mathbf{K}_k &= \mathbf{P}_{x_k y_k} \mathbf{P}^{-1}_{\tilde{y}_k\tilde{y}_k} \\
\bar{\mathbf{x}}_k &= \bar{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{Z}_k - \bar{\mathbf{z}}_{k|k-1}) \\
\mathbf{P}_k &= \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{P}_{\tilde{z}_k\tilde{z}_k} \mathbf{K}_k^T
\end{aligned}$$

where, $\mathbf{x}^\alpha = [\mathbf{x}^T\ \mathbf{v}^T\ \mathbf{n}^T]^T$, $\mathcal{X}^\alpha = [(\mathcal{X}^x)^T\ (\mathcal{X}^\nu)^T\ (\mathcal{X}^n)^T]^T$, λ is the composite scaling parameter, $n_\alpha = n_x + n_\nu + n_n$, $\mathbf{Q}$ is the process and $\mathbf{R}$ the measurement noise covariance, $\mathbf{K}$ is the Kalman gain, W_i are the Unscented Transform weights and $\mathbf{Z}_k$ is the observation vector.

The transition function $\mathbf{f}(.)$ and the observation function $\mathbf{h}(.)$ are very important for implementing efficient UKF filtering. In this model the transition function is taken to be a variable turn model (VTM) with target velocity and acceleration the velocity and acceleration of the relevant marker. However, a detailed look at real marker data indicates that estimated velocities are significantly noisy. Many factors contribute to marker position noise, such as optical measurement noise, miscalibration of the optical systems, reflections, motion of markers relative to the skin and motion of the skin relative to the rigid body (underlying bone). As a result, the target acceleration is mostly noisy. One method of measuring accelerations would be to attach accelerometers to the markers placed on the limb segments. Such a system, which synchronises measurements from the accelerometer and active markers was trialled and described in [Bac09]. While the system shows some promise, we are faced with dealing with both the noise on the accelerometer readings and the fact that the reference frame of those readings must be determined. Given these difficulties, we propose to use estimates of accelerations obtained from the position data. A real-time median filter has been applied to the target velocity (more details in [ALC09]). Having a smoothed velocity, an adequate acceleration can be calculated and the proposed model can then be used straightforwardly.

On the other hand, for the observation function we use a simple model which assumes that the rotation between two consecutive frames is constant. Thus, the time update of the observation state is equal to

$$\mathcal{Z}_{k|k-1} = R^{k-2,k-1}\mathcal{X}^{x}_{k|k-1}\tilde{R}^{k-2,k-1} \tag{4.24}$$

where $R^{p,q}$ is the rotor for the rotation between frames p and q, assuming that the rotation of the markers between two consecutive frames remains constant. It is important at this point to recall that the rotation between the previous and the current frame is re-calculated every frame just after the marker prediction using the estimated marker positions.

Early numerical implementations using Particle Filtering [DDFG01] showed that, although it could improve the performance of our state estimates, this improvement was marginal compared to the UKF-VTM and therefore not worth the added computational effort. Particle filtering has high computational cost thus ruling out its use for real-time applications.

4.6. Applying Marker Constraints

The observation vector, $\mathbf{Z}^j_k$, gives the true (observed by the mocap system) position of the tracked marker j when available, otherwise it represents estimated position. The state vector represents true position and velocity as given above. To cope with cases where markers are missing for long periods of time, a tracker that uses information from both the previous frames and the current positions of neighbouring visible markers has been implemented, taking into account the rigidity of the body. Assuming that there are three markers on each limb and the inter-marker distance is constant over time, the observation vector can be updated as given below for 5 different scenarios. The proposed marker constraints methodology, from inferred neighbouring markers, was first published in [ACL08]; the approach is simple, real-time implementable and does not assume prior knowledge other than fixed inter-marker distance. This assumption is true in a noise free environment, however since limb segments rapidly change direction, there is noise on marker motion relative to the skin and motion of the skin relative to the rigid body (underlying bone). The solutions to the mathematical problems discussed in this section are given in details, using CGA, in the Appendix A.

4.6.1. All markers are visible on a given limb

Where all markers are visible on a given limb, then:

$$\mathbf{Z}^j_k = H\mathbf{x}^k_j + \mathbf{v}^k_j \tag{4.25}$$

where $\mathbf{x}^k_j$ is the current state of a tracked marker j on the limb, H is the *observation model* (in this case the identity) and $\mathbf{v}^k_j$ is the observation noise. $\mathbf{v}$ is assumed to be zero mean multivariate normal with covariance $\mathbf{R}$.

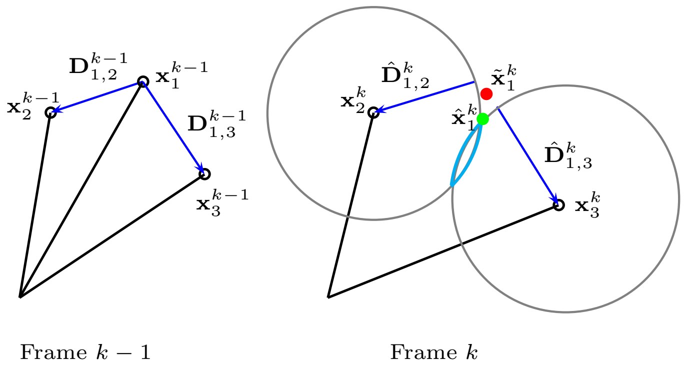


Figure 4.7.: *The observation vector in the case of 2 visible markers. The red dot,* $\tilde{\mathbf{x}}_1^k$*, represents the average value as given in equation 4.29. The green dot,* $\hat{\mathbf{x}}_1^k$*, is the point on the intersection of the 2 spheres which is closest to* $\tilde{\mathbf{x}}_1^k$*.*

4.6.2. One missing marker on a limb segment

In the case where two markers are visible on the limb,

$$\mathbf{Z}_k^1 = H\hat{\mathbf{x}}_1^k + \mathbf{v}_1^k \tag{4.26}$$

where $\hat{\mathbf{x}}_1^k$ is the estimated position of the occluded marker m_1 in frame k (assuming, in this example, that m_1 is the missing marker). $\hat{\mathbf{x}}_1^k$ can be calculated as given below. Firstly we calculate $\mathbf{D}_{1,2}^{k-1}$ and $\mathbf{D}_{1,3}^{k-1}$ which correspond to the vectors between marker m_1 and markers m_2, m_3 in frame $k-1$ respectively. These vectors are given by:

$$\mathbf{D}_{i,j}^{k-1} = \mathbf{x}_j^{k-1} - \mathbf{x}_i^{k-1} \tag{4.27}$$

Thereafter, these vectors are rotated as

$$\hat{\mathbf{D}}_{i,j}^{k} = R^{k-2,k-1}\mathbf{D}_{i,j}^{k-1}\tilde{R}^{k-2,k-1} \tag{4.28}$$

assuming that the rotation between the current and the previous frame is the same as that between the previous 2 frames. Predicting the current rotation using previous rotations offers marginal improvement to the system performance, and such a small improvement means that it is not worth the additional computational cost [ALC09]. One obvious way to proceed is to calculate the point $\tilde{\mathbf{x}}_1^k$ which is an average of the estimated positions in frame k using the $\hat{\mathbf{D}}$ vectors;

$$\tilde{\mathbf{x}}_1^k = \frac{\left(\mathbf{x}_2^k - \hat{\mathbf{D}}_{1,2}^k\right) + \left(\mathbf{x}_3^k - \hat{\mathbf{D}}_{1,3}^k\right)}{2} \tag{4.29}$$

where $\mathbf{x}_i^k$ is the position of marker i in frame k. We now improve on this estimate by finding the solution of the intersection of the two spheres in frame k with centres $\mathbf{x}_2^k$, $\mathbf{x}_3^k$ and radii $|\hat{\mathbf{D}}_{1,2}^k|$ and $|\hat{\mathbf{D}}_{1,3}^k|$ respectively. $\hat{\mathbf{x}}_1^k$ is assigned as the closest point on the circle of intersection to $\tilde{\mathbf{x}}_1^k$. Figure 4.7 illustrates this process.

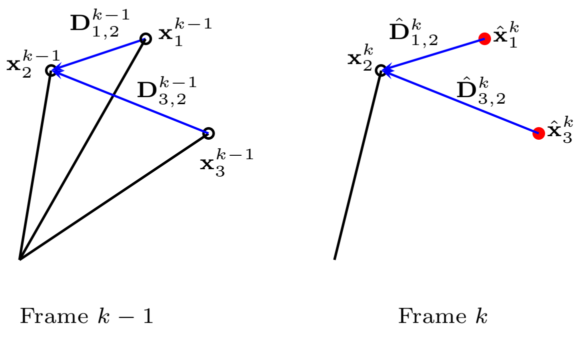


Figure 4.8.: *The observation vector in the case of only one visible marker. The red dots, $\hat{\mathbf{x}}_j^k$, where $j = \{1, 3\}$ represent the estimated position of the missing marker m_j as given in equation 4.31.*

4.6.3. Two missing markers on a limb segment

In the case of only one marker visible (for example m_2) on a given limb, the observation vector is given as:

$$\mathbf{Z}_k^j = H\hat{\mathbf{x}}_j^k + \mathbf{v}_j^k \tag{4.30}$$

where $\hat{\mathbf{x}}_j^k$ is the estimated position of the occluded marker m_j $(j = 1, 3)$ in frame k. $\hat{\mathbf{x}}_j^k$ is given by:

$$\hat{\mathbf{x}}_j^k = \mathbf{x}_2^k - \hat{\mathbf{D}}_{j,2}^k \tag{4.31}$$

where $\mathbf{x}_2^k$ is the position of the visible marker m_2 on the limb in the current frame and $\hat{\mathbf{D}}_{j,2}^k$ is as described above. In that case, it is assumed that the rotation around the axis joining the two remaining markers on the limb does not exist. Figure 4.8 demonstrates this process.

4.6.4. All markers on a limb segment are missing

When all markers on a limb are occluded, we consider two possible subcases; the case where the other limb segment has some markers visible and the case where both limb segments have all of their markers occluded. If some markers on the other limb segment are visible (assume the y limb), the missing marker positions can be calculated using the CoR estimate, $\hat{\mathbf{C}}_k = \mathbf{y}_i^k - R_y^k \bar{\mathbf{a}}_y \tilde{R}_y^k$ where $i = \{1, 2, 3\}$. Our methodology takes advantage of the approach described in section 4.4.2 for CoR estimation and provides better approximations of the CoR using information from the adjacent limbs. The observation vector of the Unscented Kalman filter is then updated as:

$$\mathbf{Z}_k^j = H\hat{\mathbf{x}}_j^k + \mathbf{v}_j^k \tag{4.32}$$

where $\hat{\mathbf{x}}_j^k$ is the estimated position of the occluded marker m_j $(j = 1, 2, 3)$ in frame k. $\hat{\mathbf{x}}_j^k$ is given by;

$$\hat{\mathbf{x}}_j^k = \hat{\mathbf{C}}_k + \hat{\mathbf{D}}_{j,c}^k \tag{4.33}$$

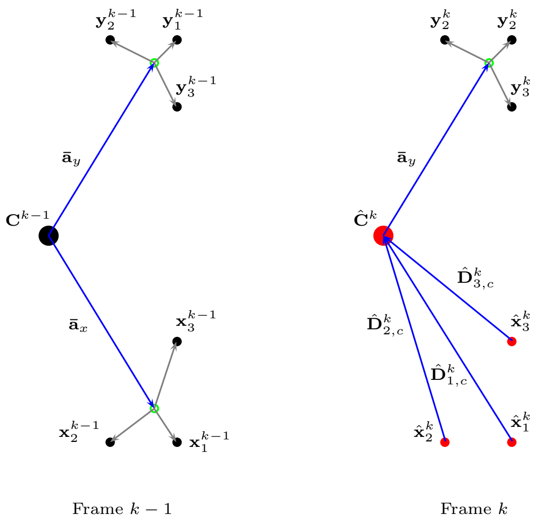


Figure 4.9.: *The estimation procedure when all markers on a single limb segment are occluded. The red dots represents the estimated position of the CoR, $\hat{\mathbf{C}}^k = \mathbf{y}_i^k - R_y^k \bar{\mathbf{a}}_y \tilde{R}_y^k$ where $i = \{1,2,3\}$, and the estimated marker positions on limb segment x, $\hat{\mathbf{x}}_j^k = \hat{\mathbf{C}}_k + \hat{\mathbf{D}}_{j,c}^k$, where $j = \{1,2,3\}$. $\bar{\mathbf{a}}_x$ and $\bar{\mathbf{a}}_y$ are updated using the predicted marker positions in the current frame.*

where $\hat{\mathbf{D}}_{j,c}^k$ is an estimate of the vector between marker m_j and the CoR. This approach takes advantage of the fact that the distance between markers and the CoR is constant. This vector is approximated by $\hat{\mathbf{D}}_{j,c}^k = R^{k-2,k-1}\mathbf{D}_{j,c}^{k-1}\tilde{R}^{k-2,k-1}$ where $\mathbf{D}_{j,c}^{k-1} = \mathbf{x}_j^{k-1} - \mathbf{C}_{k-1}$. This assumes that the rotation of the markers between two consecutive frames remains constant. Figure 4.9 illustrates this procedure.

If both limb segments have all markers occluded, only information from previous frames can be used. The observation vector, $\mathbf{Z}_k^j$, in this instance is calculated using a rotor based method. This method also assumes that the segment rotation between two consecutive frames is constant. The observation vector can now be expressed as

$$\mathbf{Z}_k^j = H\hat{\mathbf{x}}_j^k + \mathbf{v}_j^k \tag{4.34}$$

where $\hat{\mathbf{x}}_1^k$ is equal to $\hat{\mathbf{x}}_1^k = R^{k-2,k-1}\mathbf{x}_1^{k-1}\tilde{R}^{k-2,k-1}$. This method performs better than a simple variable velocity model since it ensures that all markers on a limb do not move independently [ALC09].

4.6.5. Markers visible in only one camera

Each marker can be reconstructed by the motion capture system if it is visible in at least two cameras. Indeed, looking at numerous real datasets, we have observed a high probability that the missing markers are not entirely occluded; information about position is often returned by a single camera. This information identifies a line, L_1, starting from the camera and passing through the position of the missing marker. By relaxing the constraints that the inter-marker distance is constant and accepting that the real position of the marker is on that line, we may be able to obtain a more accurate estimate of the position of the relevant marker. This

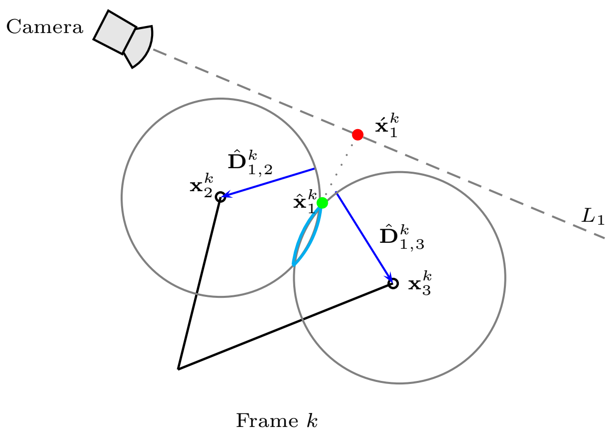


Figure 4.10.: *The observation vector in the case of 2 visible markers and one marker visible only by one camera. The red dot, $\acute{\mathbf{x}}_1^k$, is now used for the calculation of the observation vector, $\mathbf{Z}_k^1 = H\acute{\mathbf{x}}_1^k + \mathbf{v}_1^k$.*

position, $\acute{\mathbf{x}}_1^k$, corresponds to the projection from the point $\hat{\mathbf{x}}_1^k$ onto the line L_1, as in figure 4.10. This is applicable for the cases in which the motion capture system fully reconstructs one or two marker locations and another marker is visible in just one camera. If a limb segment has only one known and one partially visible marker, the system is more reliable when it first predicts the partially visible marker and then the entirely occluded marker.

4.7. Bone Length Constraints using Inverse Kinematics

The UKF-VTM model for marker prediction returns good estimates of the marker positions, even if the markers are occluded for an extended period of time. It is, however, easily observable (see fig. 4.11) that there are instances where the reconstructed markers may break the rigid body assumption (limbs may not have constant lengths over time), resulting in a violation of the model's structure. This is more obvious in extreme cases where many markers from the same limb segment are occluded for an extended time period (see fig. 4.19); the limb segment lengths drift (as the inter-joint distances were not fixed to be constant) even if the estimated marker positions do not significantly differ from their true positions. Therefore, it seems sensible to constrain the bone lengths to be constant over time, taking into account the rigidity if the body, as the skeleton methods does; this extension does not presuppose any knowledge of the model. For such unconstrained models we can preserve bone lengths using Inverse Kinematics (IK) techniques (see Chapter 3). Ishigaki et al. [IWZL09] implement a real-time IK control interface for character animation which translates the performance into corresponding actions; that was achieved by integrating prerecorded motions with online performance and dynamic simulation. If the input motion does not match the conditions, a kinematic process is applied to match users' motion with the example interactions. However, the proposed inter-joint constraint approach requires offline training, meaning that the results depend on the training data, and prior knowledge of the model is required. To the best of our knowledge, this is the first time that a real-time IK technique has been used for CoR correction under multiple marker occlusion in optical motion capture.

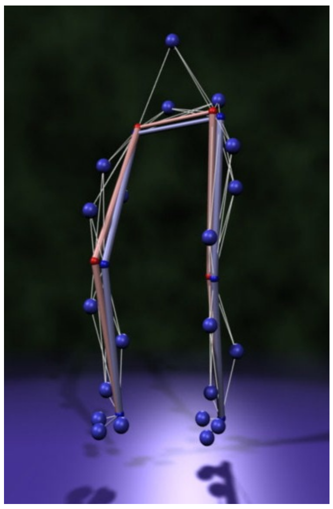


Figure 4.11.: *The results produced using the UKF-VTM model. It is seen that the distances between hips and knees change, since it is not guaranteed that these distances are constant. The observed positions are coloured blue, while the predicted positions are red.*

4.7.1. Adjusting FABRIK for CoR correction in optical motion capture

In order to achieve a real-time framework, the proposed IK solver must process a large number of frames per second and must return natural poses which satisfy the user constraints. FABRIK, as described in Chapter 3 and [AL10d], is a real-time IK solver which returns smooth postures in an iterative fashion. FABRIK simply repositions the CoRs in order to ensure the constant inter-joint distance over time, without considering the limb segment rotation; the limb segment orientation is known since there are 3 markers attached to each limb segment, thus time and computational cost can be saved. Nevertheless, FABRIK can also treat more complex cases where it is desirable to estimated the bone rotation.

Problems with Serial Chain models

FABRIK uses the positions of the joints that have been estimated using predicted marker positions to find updates that meet the fixed inter-joint distance assumption. In the most general case, where the estimated CoRs are not positioned at the end of the chain, the solution can be achieved using a forward and backward iterative mode, very similar to the basic FABRIK algorithm. The modified method starts from a joint of the chain which was calculated using observed marker positions and works forwards, adjusting each estimated joint along the way until the next joint which was calculated using observed data. Thereafter, it works backward in the same way, in order to complete a full iteration.

A graphical representation of the algorithm in action is given in figure 4.12. Assume $\mathbf{p}_1, ..., \mathbf{p}_n$ are the joint positions of a chain, where $\mathbf{p}_1$ and $\mathbf{p}_n$ are joint positions calculated using observed data, and the joints inbetween are joints estimated using predicted marker positions. Set the distances between each joint to be $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|_{\text{avg}}$, for $i = 1, ..., n-1$, $\mathbf{b} = \mathbf{p}_1$ and $\mathbf{t} = \mathbf{p}_n$; these distances can be established by averaging over time from the frames where the markers, and the joints, are available. Then, check whether the target is reachable or not; find the

Algorithm 4: A full iteration of FABRIK for CoR correction.

```
Input: The joint positions $\mathbf{p}_i$ for $i = 1, ..., n$, and the average distances between each joint
       $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|_{\text{avg}}$ where $n$ is the number of joints of the chain.
Output: The new joint positions $\mathbf{p}_i$ for $i = 1, ..., n$.
3.1  $\mathbf{b} = \mathbf{p}_1$; $\mathbf{t} = \mathbf{p}_n$;
3.2  % Initialisation of the $dif_A$ value.
3.3  $dif_A = 1$
3.4  while $dif_A > tol$ do
3.5      % STAGE 1: FORWARD REACHING
3.6      % Set $\mathbf{p}_1$ to its initial position.
3.7      $\mathbf{p}_1 = \mathbf{b}$
3.8      for $i = 1, ..., n-1$ do
3.9          % Find the distance $r_i$ between the estimated joint position $\mathbf{p}_{i+1}$ and the joint $\mathbf{p}_i$
3.10         $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
3.11         $\lambda_i = d_i / r_i$
3.12         % Find the new joint positions $\mathbf{p}_i$.
3.13         $\mathbf{p}_{i+1} = (1 - \lambda_i)\,\mathbf{p}_i + \lambda_i \mathbf{p}_{i+1}$
3.14     end
3.15     % STAGE 2: BACKWARD REACHING
3.16     % Set $\mathbf{p}_n$ to its initial position.
3.17     $\mathbf{p}_n = \mathbf{t}$
3.18     for $i = n-1, ..., 1$ do
3.19         % Find the distance $r_i$ between the joint position $\mathbf{p}_{i+1}$ and the estimated joint $\mathbf{p}_i$
3.20         $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
3.21         $\lambda_i = d_i / r_i$
3.22         % Find the new joint positions $\mathbf{p}_i$.
3.23         $\mathbf{p}_i = (1 - \lambda_i)\,\mathbf{p}_{i+1} + \lambda_i \mathbf{p}_i$
3.24     end
3.25     $dif_A = |\mathbf{p}_1 - \mathbf{b}|$
3.26 end
```

distance between $\mathbf{p}_1$ and $\mathbf{p}_n$, $dist$, 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 performed in two stages. In the first stage, the algorithm estimates each joint position starting from $\mathbf{p}_1$, moving forward to $\mathbf{p}_n$. Thus, initialise $\mathbf{p}_1 = \mathbf{b}$ and find the line, l_1, which passes through the joint positions $\mathbf{p}_1$ and $\mathbf{p}_2$. The new position of the 2^{nd} joint, $\mathbf{p}'_2$, lies on that line with distance d_1 from $\mathbf{p}_1$. Similarly, the new position of the 3^{rd} joint, $\mathbf{p}'_3$, can be calculated using the line l_2, which passes through the $\mathbf{p}_3$ and $\mathbf{p}'_2$, and has distance d_2 from $\mathbf{p}'_2$. The algorithm continues until all new joint positions are calculated, including $\mathbf{p}'_n$.

Having in mind that initially $\mathbf{p}_n$ was the observed position of the n^{th} joint, 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 observed position of the n^{th} joint and moving backwards to the 1^{st} joint. Therefore, let the new position for the n^{th} joint, $\mathbf{p}''_n$, be its initial position $\mathbf{t}$. Then, using the line l_{n-1} that passes through the points $\mathbf{p}''_n$ and $\mathbf{p}'_{n-1}$, we define the new position of the joint $\mathbf{p}''_{n-1}$ as the point on that line with distance d_{n-1} from $\mathbf{p}''_n$. This procedure is repeated for all the remaining joints, including $\mathbf{p}_1$. The procedure is then repeated, for as many iterations as needed, until $\mathbf{p}_1$ and $\mathbf{p}_n$ are close enough (to be defined) to their initial, observed positions. FABRIK always converges to any given chains/goal positions, when this is possible. If there are constraints which do not allow the chain to bend enough or if the

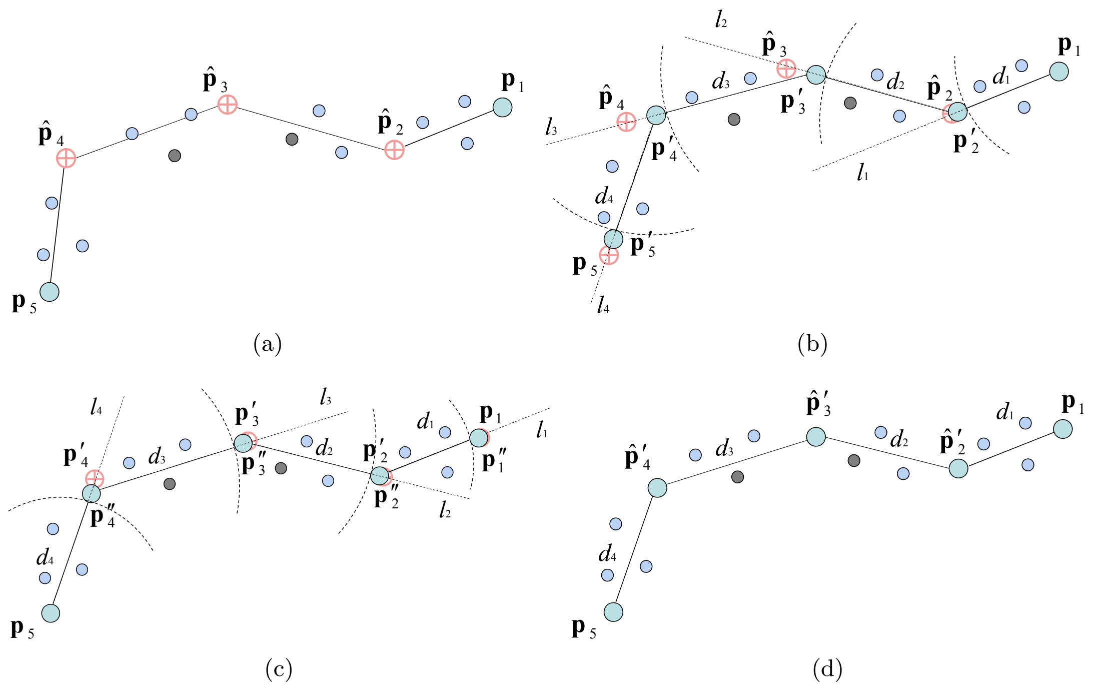


Figure 4.12.: *A simple example of FABRIK implementation for the case where the estimated joints are positioned inbetween 2 observed joint positions. (a) The initial position of the chain, where $\{\mathbf{p}_i\}$ are observed and $\{\hat{\mathbf{p}}_i\}$ are estimated joint positions. (b) The first phase of the algorithm; the joint $\mathbf{p}'_i$ has been adjusted as the point on line l_{i-1} with distance d_{i-1} from $\mathbf{p}_{i-1}$. (c) The second phase of the algorithm: let the new position of $\mathbf{p}'_5$ be its initial position $\mathbf{p}_5$; repeat the procedure starting this time from the other side of the chain. (d) The final posture; the algorithm is repeated for as many iterations as needed until the difference between the observed and the returned positions of the joints $\mathbf{p}_1$ and $\mathbf{p}_5$ is less than a given tolerance.*

target is not within the reachable area, there is a termination condition which compares the previous and the current position of the joint $\mathbf{p}_1$, and if this distance is less than a specified tolerance, FABRIK terminates its operation. Similarly to the prototype FABRIK algorithm, several optimisations can be achieved using CGA to produce faster results and to converge to the final answer in fewer iterations. Another simple optimisation is the direct construction of a line pointing towards $\mathbf{p}_n$, when the target is unreachable, adjusting the distances in such a way that each distance has changed uniformly. The adjusted FABRIK for CoR correction is illustrated in pseudo-code in Algorithm 4.

There are, indeed, some special instances where FABRIK does not necessarily work in an iterative fashion. Such a special case is when the joints, which have been estimated using predicted markers, are located at the end of the chain. This is a simplified case of the general solution, since the answer is given directly in one single iteration. In that case, it is ensured that the inter-joint distance remains constant and the bone lengths are not stretched out. The new simplified algorithm is given here: assume $\mathbf{p}_i$ is a true joint position. The joint update $\mathbf{p}'_{i+1}$ is set as the point on line l_i that passes through $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and has d_i distance from $\mathbf{p}_i$. This procedure is repeated for all the remaining estimated joints until the end of the chain. A graphical representation of an implemented example is given in figure 4.13.

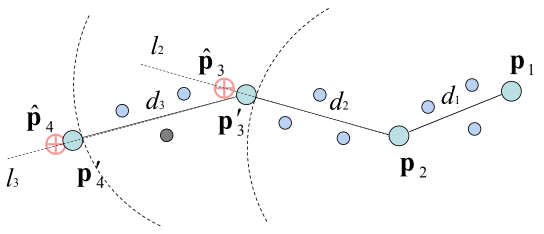


Figure 4.13.: *A simple case where the estimated joints are located at the end of the chain. In this example the serial chain has 4 joints where $\mathbf{p}_1$ and $\mathbf{p}_2$ have been calculated using observed marker positions and the $\hat{\mathbf{p}}_3$ and $\hat{\mathbf{p}}_4$ using predicted markers positions. Thus, set $\mathbf{p}'_3$ to be the point on line l_2 which has distance d_2 from $\mathbf{p}_2$ and $\mathbf{p}'_4$ the point on line l_3 that has d_3 distance from joint $\mathbf{p}'_3$.*

Problems with tree models

In reality, most of the legged models are comprised of several kinematic chains in a tree fashion. The proposed algorithm can be easily extended to process tree models with multiple serial chains. The solution is similar to the original version of FABRIK for multiple end effectors, given in Chapter 3. Hence, if the sub-base[3] joint is an observed position, FABRIK is used on each chain with estimated CoR positions individually, starting from the sub-base, in a forward and backward iterative fashion. If the sub-base is an estimate, the same procedure is applied but this time starting from the observed joints of the connected chains and moving towards to the sub-base. This will produce as many different positions of the sub-base as the number of the connected chains. The new position of the sub-base will then be the centroid of all these positions. In the second stage, the normal algorithm is applied separately for each chain, starting now from the sub-base and moving outwards to the starting joints. The method is repeated until all observed joints have no significant change between their initial and updated positions.

Applying Constraints

The main aim of the work in this Chapter is to describe a general solution, where each single joint is treated as ball joint, allowing 3 degrees of freedom. In this way, we can ensure the generality of the proposed method, producing solutions without prior knowledge of the model. Nevertheless, in cases where it is desirable to incorporate joint restrictions, FABRIK can be easily adapted to support joint limitations by readjusting the target position and orientation, on each step, to satisfy the joint biomechanical limits, as described in Chapter 3, [AL10d] and [AL09].

Obviously, the more information available regarding the model's structure and joint constraints, the more accurate and efficient will be the results. This information can help to give a visually more realistic motion within a feasible set; however, it will limit universal use of the proposed methodology and fails to satisfy our aim of no prior knowledge of the model.

The Inverse Kinematic system ensures that, even if the marker prediction system fails to track the marker positions, the derived CoRs will be good estimates of their true positions.

[3]A sub-base joint is a joint which connects 2 or more chains.

Table 4.1.: *Average results (over 20 runs) under multiple cases of occlusion (large occlusions of 2500 frames in total).*

Method	Missing markers on a limb segment	Marker's Error (cm)	Worst Case scenario (cm)	CoR's Error (cm)	Worst Case scenario (cm)	Proc. (limbs per second)
UKF-VTM	One marker	1.2958	3.5677	1.0820	1.7888	315
	Two markers	3.4737	7.6584	1.9867	2.7845	
	All markers	8.4012	12.8749	7.8591	13.1183	
Interpolation	One marker	9.4687	15.2552	7.8777	6.1189	900
	Two markers	10.5241	16.8874	9.3340	8.5874	
	All markers	11.0010	17.2282	12.2514	20.2111	
Extrapolation	One marker	1.8574	4.8811	1.5874	3.1147	270
	Two markers	4.5214	9.0444	2.5147	3.5558	
	All markers	10.5961	13.8541	11.5824	15.2588	
EKF	One marker	1.3982	3.8971	1.2356	2.1496	370
	Two markers	3.5331	8.0145	2.0302	3.0098	
	All markers	8.4205	13.0014	7.9055	13.1198	

This happens since the IK procedure restricts the limb segment (bone) length to a constant value over time. The length of each limb segment is calculated when all joints have been estimated using observed marker positions in previous frames. The inter-joint distances in reality are not constant since marker positions are noisy, violating the assumption of a fixed inter-marker pairwise distance. The effectiveness of the proposed method is strongly related to the stability of the inter-joint distance. The proposed bone length constraint is independent and can be easily adapted to most marker prediction methods for CoR estimation.

Self-Collision Determination: Collision detection has been a fundamental problem in computer animation, physically-based modeling, geometric modeling, and robotics. Since the data used in these examples are captured from a markered optical motion capture system and since the 3D animated humanoids do not have a mesh that defines their external shape, self-collisions are not considered. Nevertheless, self-collisions can be handled using existing techniques, such as [LG98].

4.8. Results and Discussion

Experiments were carried out using a 24 camera PhaseSpace motion capture system capable of capturing data at 480Hz [Pha]. The algorithm described in this chapter can process up to 300 *limb segment pairs per second* (using MATLAB). Our datasets comprise real data (i.e. captured data with natural occlusions or occlusions generated by artificial deletion) with more than 10000 *frames* in each. Our methodology has been tested on various motions including dancing (14 segment body datasets), fast walking (7 segment leg datasets) and boxing (5 segment arm datasets). Figure 4.14 shows 2 of the models that have been used in our experiments. Using real data with occlusions generated by artificial deletion, we are able to calculate the error of the proposed methodologies; the error is measured as the average distance between the observed and the estimated position (for marker and the CoRs) after artificial deletions. The error varies between different instances of marker occlusion. As more markers become available, more information relative to the limb segment is available and thus, a higher accuracy is achieved.

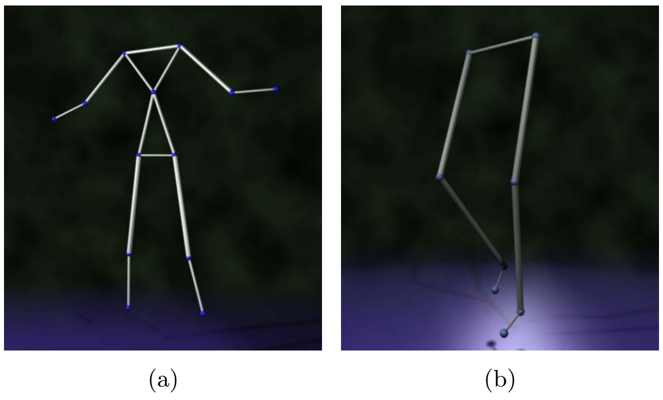


Figure 4.14.: *Two of the models used in our experiments. (a) a 14 segment human body, (b) a 7 segment lower human body.*

The magnitude of the error reported in the results is given in terms of real world distances. Within this work, our methodology (referred as UKF-VTM) was tested and compared against some of the most popular marker prediction approaches: an EKF model, such as [ACL08], using similar marker constraints as the ones proposed here, a cubic spline interpolation and a real-time extrapolation method with marker position constraints, as reported in [PLH+09].

Using the VTM we took into consideration the velocity, direction and acceleration changes of the markers' trajectories over time. Certain drawbacks of the simple EKF were overcome by using a more evolved and sophisticated method; the use of a variable turn model gives significant improvements in cases where the trajectories of the markers are variable and have abrupt fluctuations in speed and direction. The proposed system (UKF-VTM), without applying bone constraints, returns an average error (over 20 runs) of 1.296cm in the case of one missing marker, 3.474cm when 2 markers are missing and 8.401cm when all markers are occluded. The corresponding CoR estimation error is 1.082cm in the case of one missing marker, 1.9867cm in the case of two markers and 7.859cm in the case where all markers are not visible to the cameras. Table 4.1 lists and compares the results between each method implemented here under several occlusion scenarios. In general, if the methods are not constrained with the proposed inter-marker constant distance assumption, they fail to track the missing marker paths when the occlusion lasts longer than a time threshold. Table 4.1 also tabulates the worst case scenario (distance between the observed and predicted positions) of marker prediction and CoR estimation. The worst case scenarios usually appear on abrupt changes in velocity and direction of the missing marker during the occlusion period, where the UKF-VTM model requires some time to efficiently track the target. Obviously, the interpolation method has the lowest computation cost, however it is an off-line application. The UKF-VTM framework increased the processing time by 20% compared to the simple EKF, processing 315 *limb segment pairs per second* in MATLAB; which does, however, still allow real-time implementation. The UKF-VTM method performs better than the other methods looked at here, resulting in the most accurate results; our methodology efficiently recovers good estimates of the missing markers and accurate real-time CoR estimation. A further error reduction was observed when

Table 4.2.: *Average results (over 20 runs) on real data with occlusions generated by deletions. Case of one missing marker on each limb segment for more than 1500 frames.*

	Entirely occluded (Error cm)	Partially visible (Error cm)	Change
Marker position	1.3458	0.2554	-81.02 %
CoR position†	2.2247	0.5628	-74.70 %

† *when* $\bar{\mathbf{a}}_w$ *is updated using the predicted data.*

the missing markers were partially visible to one camera. The error is significantly decreased, by 80% for marker prediction and 75% for CoR estimation, compared to the case where this information was ignored. Table 4.2 states the prediction error for the case of one missing marker on each limb segment when the missing markers are both entirely occluded and visible in just one camera.

The difference between true and estimated positions is further reduced when the fixed inter-joint pairwise assumption was taken into consideration. The CoR error, in our methodology, is decreased on average (over 30 runs) by 11.9% in total, 2.96% when the estimated joints are located at the end of the chain and 12.7% when they are positioned between 2 observed positions. FABRIK ensures that the inter-joint pairwise distances are constant over time, thus eliminating the error in the CoR estimation due to unnatural bone extensions. The error reduction is larger in cases where the estimated joints are between 2 observed CoR positions, since both observed joints, in an iterative fashion, constrain the inter-joint distance from the estimated joints. In the case where the estimated joints are positioned at the end of the chain, only one observed joint contributes in the final solution. Table 4.3 presents the average error and achieved error reduction after applying IK to the case of a humanoid model with artificial deletions on 6 markers (over 30 markers in total) on more than 2500 frames out of 8000. At the same time, the processing time was increased by only 5.27%, processing now approximately 300 *limb segment pairs per second.* FABRIK has been applied to all methodologies ensuring that the inter-joint distances will remain unchanged over time; the average CoR error was reduced on average by 12.92% for the EKF, 45.33% for the extrapolation method and 62.8% for the interpolation approach.

Figure 4.15(a) shows zoomed examples of the true and predicted x-positions of an occluded marker and the CoR, after incorporating FABRIK, for the case of a single occlusion and a marker visible to just one camera, respectively. It is clear that the occluded marker can be tracked with high accuracy when it is visible in at least one camera and its CoR position can be reconstructed efficiently even if the occlusion period exceeds 1500 frames. Figure 4.15(b) also shows an example of the error variation of all methodologies due to occlusion for the case of 1 missing marker on each limb segment. Clearly, UKF-VTM has the lowest error both on average and in the worst case scenario; this error is further decreased when the marker is visible to just one camera.

Figure 4.15(c) reinforces the above showing the cumulative distribution function (CDF) of the estimation error for the case of one missing marker on each limb segment. The $x-$axis

Table 4.3.: *Error reduction using Inverse Kinematics.*

Method		Overall Results		Joints are located at the end of the chain		Joints are located in between 2 true joint positions	
		Error (cm)	Change	Error (cm)	Change	Error (cm)	Change
UKF-VTM	CoR[†]	2.0268		2.1013		2.0131	
	CoR[‡]	1.7856	-11.90%	2.0391	-2.96%	1.7574	-12.70%
Interpolation	CoR[†]	7.7149		8.3149		7.2650	
	CoR[‡]	2.8698	-62.80%	3.7950	-54.35%	2.2937	-68.42%
Extrapolation	CoR[†]	4.2437		4.9334		3.7263	
	CoR[‡]	2.3200	-45.33%	2.8495	-42.25%	1.9712	-47.10%
EKF	CoR[†]	2.5077		2.8684		2.3509	
	CoR[‡]	2.1836	-12.92%	2.5474	-11.19%	2.0212	-14.02%

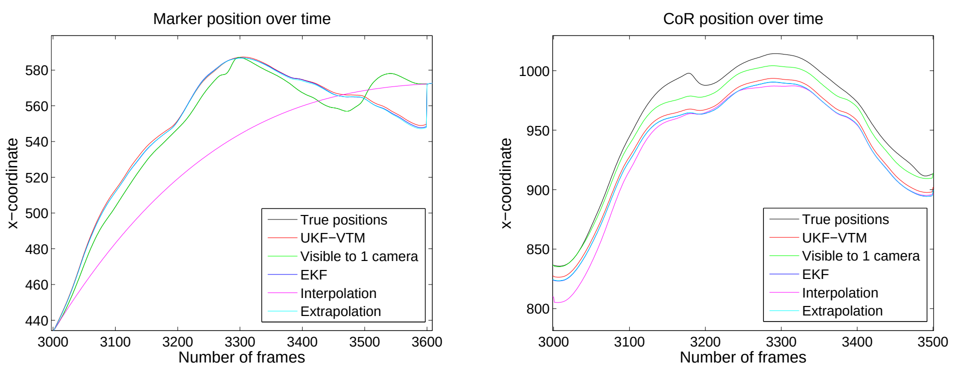
[†] *The CoR was calculated using only predicted marker positions.*
[‡] *The CoR was corrected using FABRIK, assuming that bones have fixed length over time.*

shows the estimation error and the $y-$axis shows the probability, for $y = a$, of having an error less than or equal to a. Hence, for example, the probability the estimation error is less than or equal to 1.2cm is approximately 0.57 for the extrapolation and less than 0.4 for interpolation, while it is 0.62 for the case of the UKF-VTM. The median estimation error of the markers using the EKF model is approximately 1.12cm (0.83cm for the CoR), where the corresponding median error for the case of UKF-VTM is approximately 1.09cm (0.75cm for the CoR and 0.12cm when markers are visible to just one camera).

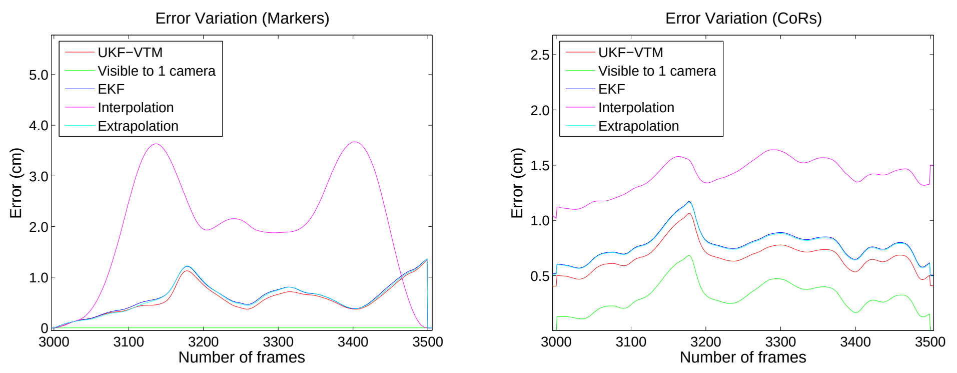
Obviously, interpolation produces smooth results since it uses previous and future positions. Our method shows small oscillations or discontinuities between the last predicted position and the first frame when the marker become available. This is expected since we do not use future positions and as the markers become available, we use the observed positions. This phenomenon can be avoided if, instead of using the observed marker position as the final result (after the occlusion), we continue using the UKF framework, having observation states as the true marker positions. In that way, although the average errors of the marker and CoR estimates will show a marginal increase, we will obtain smooth results without oscillations.

Extrapolation and interpolation return useful predictions for short-time occlusions but fail to track the marker positions when the occlusion is maintained for extended time periods, especially if markers change rapidly in direction and velocity. In particular, the path followed by the interpolation method does not reflect the actual state of the missing marker, but only connects the available positions using a cubic spline shape. In contrast, the UKF-VTM performs fairly well even if markers are missing for long time periods.

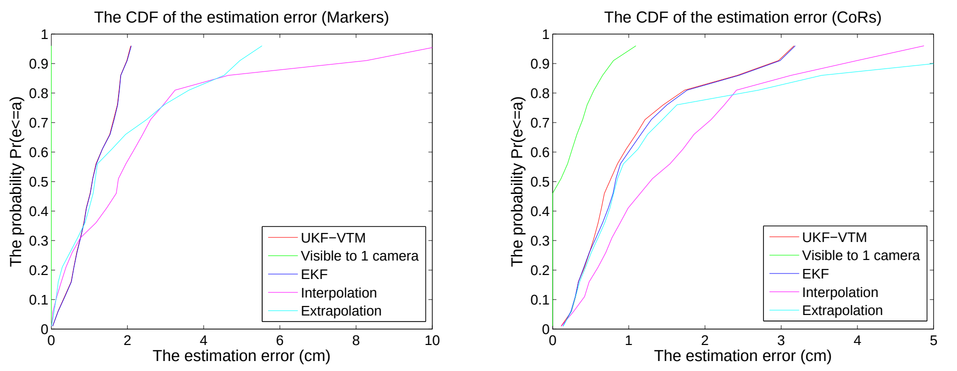
Figure 4.16 and 4.17 show examples of using our algorithm on real data; in blue are the true (observed by the mocap system) positions of the markers and CoRs and in red the predicted positions. Figure 4.18 is another example that compares the results on the leg model; the left picture presents the results when only the integrated UKF-VTM model is used, and the right picture shows the results when bone length control was incorporated using FABRIK. Note the bone length violation, which is more obvious on the hips.



(a)



(b)



(c)

Figure 4.15.: *Example showing comparison results for each methodology for the case of 1 missing marker on each limb segment; occlusions were introduced for extended time periods of up to 1500 consecutive frames. The left column shows the results for marker predictions and the right column for CoR estimates, after applying FABRIK. (a) the zoomed x-coordinate path over time (in this example note that the black and green curves for the markers are exactly overlayed), (b) the zoomed error variation over time, (c) the CDF of the estimation error (over 10 runs).*

Figure 4.19 shows another example of implementation under an extreme case with extended marker occlusions; the top picture shows the results when the fixed inter-joint distance was not imposed and the bottom picture when FABRIK was applied. Obviously, in the first case the skeletal structure was violated since the bones were not restricted to their original lengths; in the second case the results have been improved to a visually more natural shape.

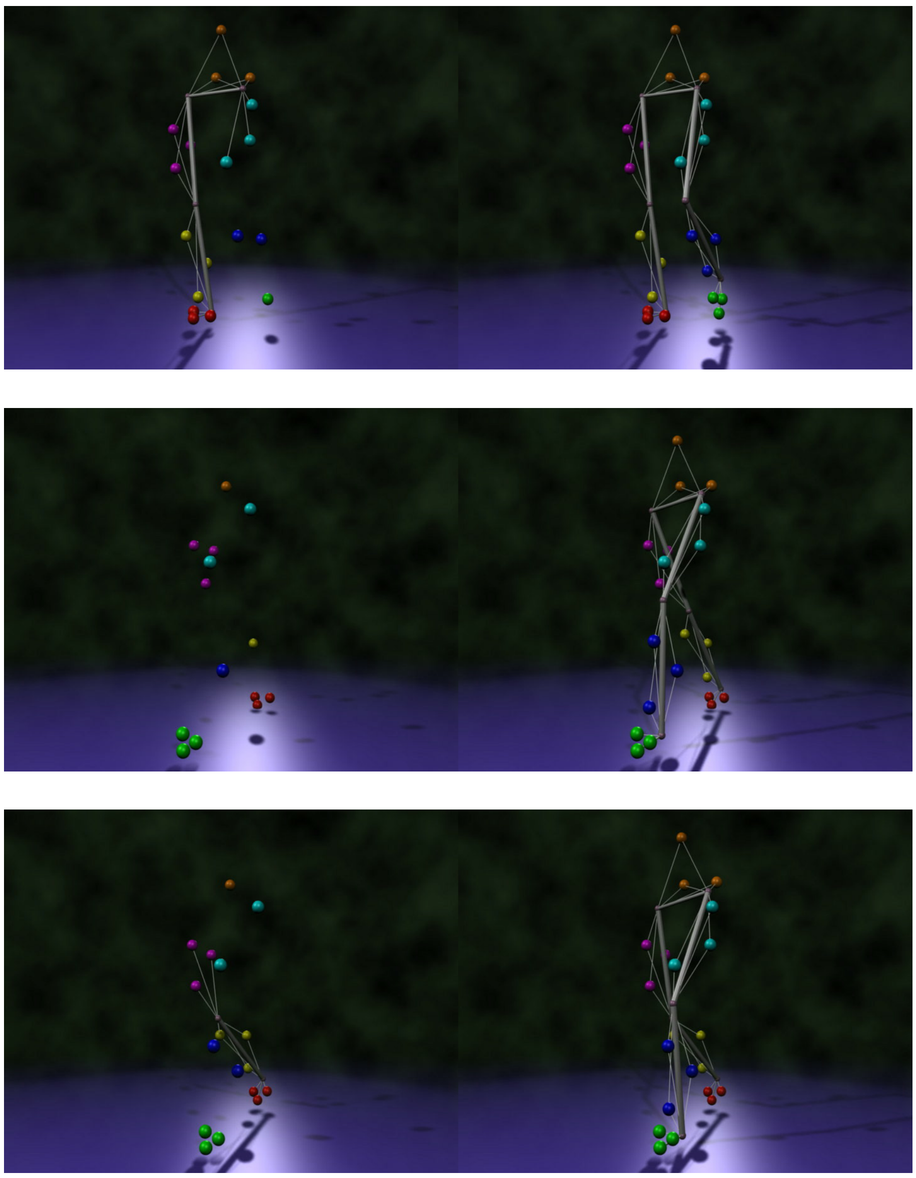
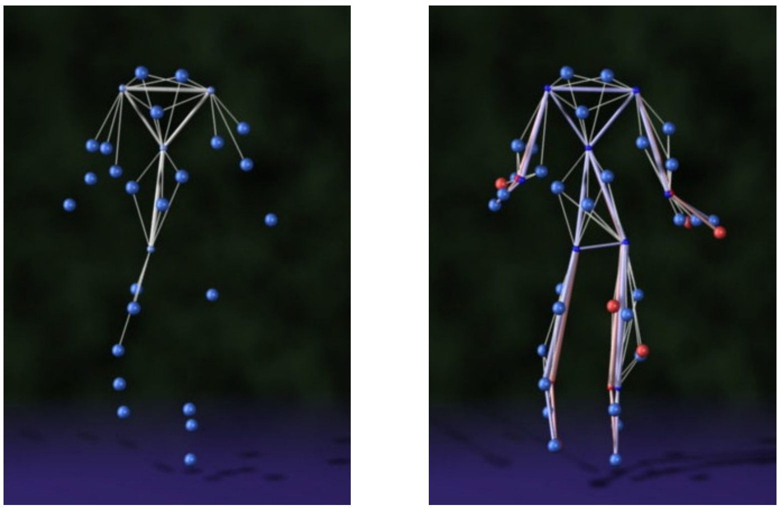


Figure 4.16.: *Examples of implementation on real data (Lower body). On the left side of the picture there are marker occlusions; on the right side of the image these markers have been correctly estimated. The markers, on each limb segment, are clustered and coloured using different colours.*

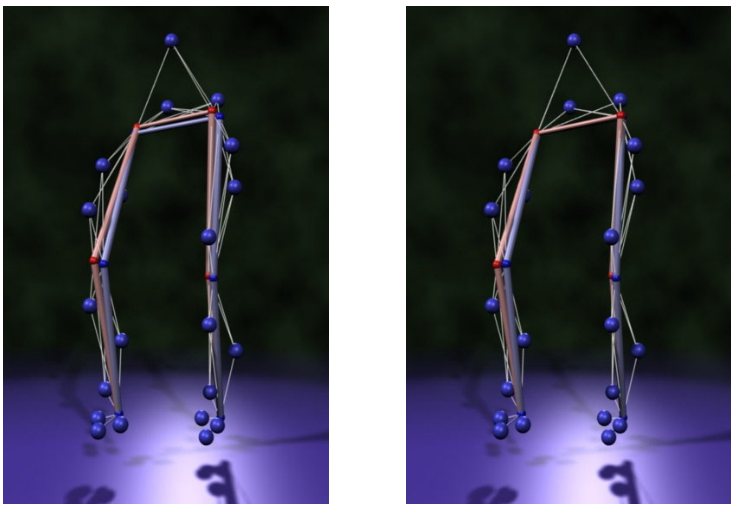
Experiments demonstrate that the proposed methodology can effectively track the occluded markers with high accuracy, even if they are occluded for extended periods of time, recovering in real-time good estimates of the true joint positions. FABRIK controls and corrects the CoR estimates decreasing the error between the estimated and true positions, thereby enabling real-time skeletal reconstruction. The resulting motion is natural and smooth, without oscillations and discontinuities, resembling that of true human movements.

A video included in the supplementary materials demonstrates our methodology and shows its performance with and without incorporating the FABRIK algorithm.



(a) (b)

Figure 4.17.: *An example of marker prediction and CoR estimation using the proposed methodology. The observed positions are coloured blue and the predicted positions red. (a) The artificially deleted data, (b) the predicted data.*



(a) (b)

Figure 4.18.: *An example of FABRIK implementation for CoR correction. Blue represents the observed positions and red the predicted positions: (a) using only the integrated UKF-VTM framework, (b) using FABRIK for bone length control.*

4.9. Conclusion

This chapter describes a methodology related to the problem of using marker-based optical motion capture data to automatically establish a skeleton model to which the markers are attached. It presents a real-time prediction method that estimates the missing markers and reconstructs the skeletal motion. An Unscented Kalman Filter framework with a variable turn model has been deployed for marker tracking. Information about the missing markers in the current frame inferred from an approximate rigid body assumption has been used for the obser-

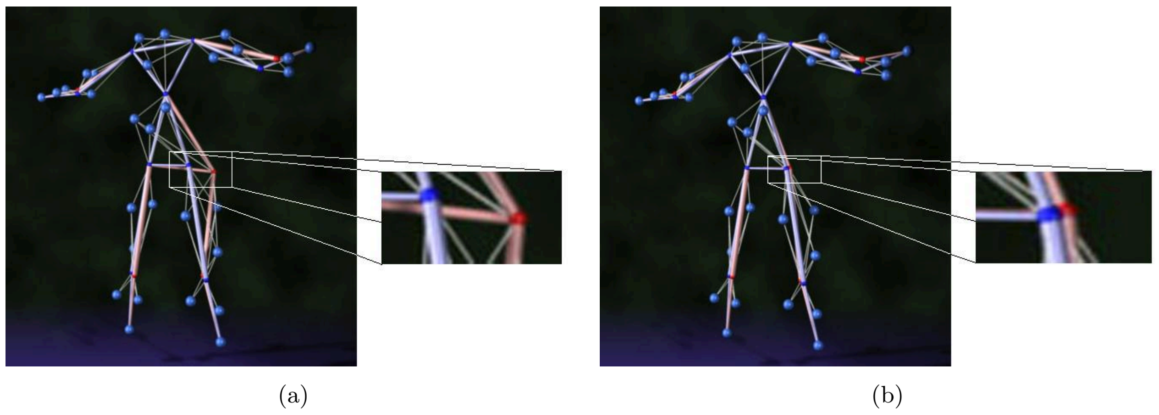


Figure 4.19.: *An example of FABRIK implementation under extreme cases with extended data occlusion. (a) shows results using the integrated UKF-VTM, (b) shows the results when FABRIK was applied in order to maintain the fixed inter-joint distance assumption. The observed positions are coloured in blue and the predicted in red.*

vation states. The proposed marker constraint model is simple and real-time implementable. At the same time, the system takes advantage of the information returned by each single camera, regarding the position of the missing markers which are visible to just one camera. The predicted data is then used for real-time joint localisation. Thereafter, the joint positions are re-positioned using Inverse Kinematics taking into account the fact that the inter-joint distance is constant over time. FABRIK is a real-time iterative IK solver which ensures that the bones lengths remain constant over time, resulting in more feasible motion. Our methodology outperforms in accuracy the methods used for comparison in this work. It is able to maintain real-time marker predictions, thus enabling good estimates of the CoR, even in the presence of several marker occlusions on each limb segment. It is reliable even if the limb rapidly changes direction and the occlusions exist for large sequences. The proposed technique could also be used to isolate bad inputs from single cameras; it is a common phenomenon that even one error in large marker inputs can result in several errors per second, pulling the marker position out of the expected path. Thus, this method can be utilised to eliminate pops and jumps from camera switching and errors as they fall outside the predicted positions.

Future work will introduce biomechanical constraints to restrict motions to those from a feasible set; however, prior knowledge of the model and joints will then be needed. Also, a hybrid system with low cost inertial measurement units could be used to validate the method proposed here.

5 Hand Pose Tracker

ARTICULATED hand-tracking systems have been widely used in virtual reality systems and the computer games industry, but due to their complexity and high computational cost, accurately tracking hands remains challenging. In this chapter, we focus on real-time hand tracking and reconstruction using optical motion capture technology. The IK system developed in Chapter 3 is adapted to control the postures of the hand, subject to physiological constraints that restrict the allowed movements to a feasible and natural set.

5.1. Introduction

In recent years, there has been a growing demand for reliable hand motion tracking systems, a technology used to turn the observations of a moving hand into 3D position and orientation information. Such information can be used for hand gesture recognition; to generate virtual figures for films or computer games; and for human-computer interaction (HCI) including interaction with game consoles. However, building a fast and effective hand pose tracker remains challenging; the high dimensionality of the pose space, the ambiguities due to self-occlusions and the significant appearance variations due to shading, make efficient tracking difficult.

Marker-based motion capture has been demonstrated in several interactive systems (including but not limited to hand interaction) producing results which are highly accurate and easily configurable. There are, however, instances where we do not have many markers available or it

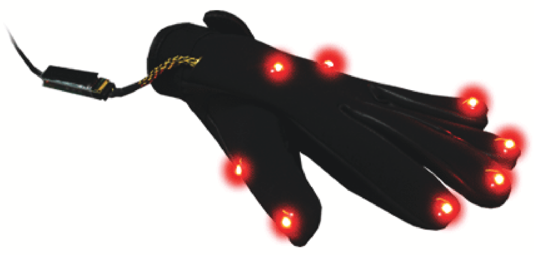


Figure 5.1.: *The impulse glove designed by PhaseSpace for real-time finger-tracking solutions.*

is impossible to attach 3 markers on each limb segment; the large number of markers needed is often prohibitive. It may therefore be infeasible to track the object and reconstruct its skeleton model (i.e. the hand model). Hence, it would be useful to find a new way of capturing the movement of these articulated models using the minimum possible number of markers. Such a method is presented in this chapter, reproducing good estimates of real captured hand motion. Instead of attaching 3 markers on each limb segment, a single marker is attached and captured on each finger (end effector), 1 marker at the chain base (root) and 2 markers at strategic positions to help us define the hand orientation. The markers' positions are tracked using an optical motion capture system, such as [Pha]. Physiological joint constraints are applied to find the hand palm and to ensure that the hand motion is within a feasible set, giving a smooth and visually natural motion of the hand. An Inverse Kinematics solver (FABRIK) is then incorporated to estimate the remaining joint positions and to fit them to the hand model. In addition, a marker prediction system, similar to the one presented in chapter 4, is applied to deal with cases where the markers are not visible to the motion capture cameras. The results were visualised using a mesh deformation algorithm, driving the animation of the hand according to the underlying hand skeleton. The skeletal fitting and the incorporation of constraints are real-time implementable and the hand motion is tracked smoothly and without oscillations, even with a low frame rate.

5.2. Related Work

There are many approaches for tracking and configuring the hand model. The hand gesture identification algorithms can be classified into two major classes: glove-based and vision-based methods. In general, glove-based methods are real-time, they are, however, expensive (e.g. P5 Data glove) and detect only a limited set of finger movements with low accuracy. Wang and Popovic [WP09] and Fredriksson et al [FRF08] proposed methods for hand tracking using a single camera and an ordinary cloth glove which was imprinted with a custom pattern; the pose corresponding to the nearest database match was then retrieved. Although this offers a simple, computationally cheap and promising solution, the resulting poses are highly correlated with the training data making these methods less reliable than optical mocap systems.

The vision-based methods, on the other hand, are more accurate, but they have problems with occlusions, noise and spurious data. Lien and Huang [LH98] proposed a hand model together with a closed-form Inverse Kinematics solution for the finger fitting process. The 3D positions were obtained using colour markers and stereo vision, and the finger poses were chosen using a search method which finds the best solution amongst all possible positions. While this method is implementable in real-time, it is complex and can fail when different size models are used. Park and Yoon, [PY06], also used a marker-based motion capture system; a LED-glove has been employed to produce interactions in multi-model display environments where the gestures were recognised and classified using hidden Markov chains. De la Gorce et al [GPF08] proposed a 3D hand tracking approach from monocular video. A formulation for exploiting both shading and texture is presented, which is able to handle the problem of self-occlusions. However, due to the high dimensionality of the human hand, this method might suffer from high computational complexity and singularities.

Several papers [RK94, SMC01, MTHC03, STTC06] focus on statistical methods, such as an Unscented Kalman Filter and a Hierarchical Bayesian Filter, to track the hand motion. These statistical methods approximate the posterior by a single Gaussian and update these approximations via a linearisation of the measurement process. [STW07] employed a mean shift embedded particle filter for visual tracking; they incorporate the mean shift optimisation into particle filtering to move the particles to local peaks in the likelihood. However, such methods are still far from real-time, therefore limiting their use.

In [SMFW04, DDHF06], a bare-hand tracking approach is implemented using edge data detection and silhouettes to identify the pose of the hand. Sudderth et al, [SMFW04], used a nonparametric belief propagation (NBP) algorithm for tracking the hand poses and kinematic constraints; in that way, they were able to handle cases with self-interactions and self-collisions. Nevertheless, these methods have high computational cost making it difficult to use them for real-time interactions in control applications. In contrast, [LGPW98, DMR06, SK07] achieve interactive speeds using bare-hand tracking systems at the cost of resolution and scope. Cerveri et al, [CDML$^+$07], utilised a kinematic model using a multi-camera system and markers, which consists of a hierarchical chain and rigid body segments. Limitations relative to the joint rotational and orientational constraints were taken into consideration to restrict the motion to natural postures. In [FAT05] and [CFAT07], vision-based hand shape estimation methods were introduced using shape features acquiring from camera images. The extracted features were then used to approximate the hand's state and the local state of each finger was estimated using Inverse Kinematics and physical hand constraints. Finally, Kaimakis and Lasenby [KL07] used a set of pre-calibrated cameras to extract the hand's silhouette as a visual cue. The 2D silhouette data is then modelled as a conic field and physiological constraints are imposed to improve the reliability of the hand tracking [KL09].

A prototype version of the hand model methodology proposed in this work was published in [AL10b]. That was a first step towards an effective real-time hand motion tracking; however, most of the currently implemented physiological constraints were ignored, reducing the final reconstruction quality. In addition, the hand was not visualised using a mesh deformation algorithm, thus it was difficult to observe the differences in shape as well as the error on the resulting pose.

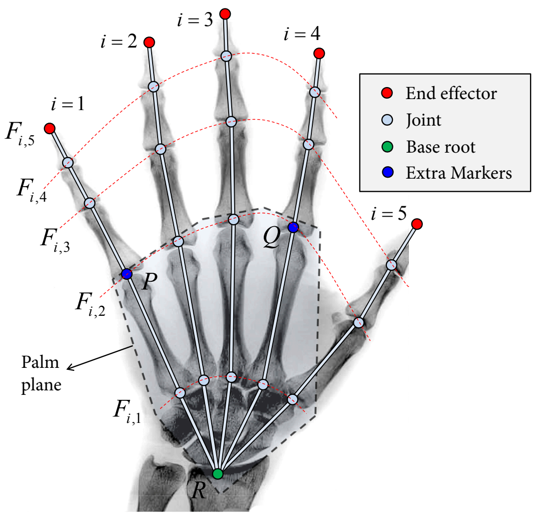


Figure 5.2.: *The hand's model geometry used in our implementation.*

PhaseSpace, [Pha], have designed an *impulse* glove with attached markers (see figure 5.1) for real-time finger-tracking solutions. This is similar to the configuration we will consider in this chapter.

5.3. Articulated Hand Model

Human motion is typically represented as a series of different configurations of a rigid multibody mechanism consisting of a set of segments connected by joints. These joints are hierarchically ordered and have one or more degrees of freedom (DoF). The DoFs describe the rotations relative to their parent joints up to the root joint, for which the position and orientation are represented with respect to a reference coordinate system. Most motion capture systems reconstruct figures by tracking several markers placed over the body of the performer. Hence, in order to configure a human pose, it is important to locate the end effectors in strategic positions as they are more easily specified by an animator and tracked by mocap systems.

It is assumed that the hand geometry, meaning the initial joint configuration of the hand, is known a priori. An example of a hand model is graphically represented in figure 5.2. The proposed hand model consists of 25 joints and has in total 25 DoFs. The end effector positions are captured using an optical motion capture system, such as PhaseSpace [Pha]. Using inverse kinematics, we then tracked and reconstructed the hand poses over time. The markers are identified (e.g. in PhaseSpace, each LED marker is pulsed at a different frequency) so that it is known a priori on which finger each marker is placed. It is also important to know the orientation of the hand in order to efficiently incorporate constraints. This can be achieved by attaching 2 extra markers at specific positions, p and q, on the back of the hand (reverse palm). Assuming that the palm is always flat, we can find the plane describing the orientation of the hand using p, q and the position of the base root, r, which also lies on the palm plane. For simplicity, markers p and q can be placed at the joint positions $F_{1,2}$ and $F_{4,2}$ respectively.

A factor which should also be considered is the noise on the marker data. The motion of the markers relative to the skin and the motion of the skin relative to the underlying bone constitute the most important causes of noise. In addition, noise due to miscalibration of the optical system and the optical measurement noise might affect the accuracy of the marker positions. The precision of the system is also highly related to the markers' positioning; the markers should be carefully placed at the end positions of the finger as well as at the root in order to fit to the hand geometry.

5.4. Inverse Kinematics

It is time-consuming for an animator to manually set all the DoFs of a virtual character. It is therefore more sensible to use a simulation mechanism, such as an Inverse Kinematics (IK) solver, to situate limbs according to their known end effector positions. The IK techniques require only positions and orientations of certain joints, usually named end effectors, to be specified by the animator and the remaining DoFs are automatically determined according to criteria that depend on the IK variant.

For the purpose of this study, FABRIK is chosen due to its efficiency, implementation simplicity and low computational cost. As we have seen in Chapter 3, FABRIK is a flexible IK solver which supports most of the existing joint types, is able to solve problems with multiple end effectors and targets, and can handle cases with closed loops.

5.4.1. The hand model

Before employing the IK solver, it is crucial to find the fingers' orientations, the chain roots and the end effectors for each chain; the target positions are assumed to be known since they are tracked by the motion capture system. The procedure is simple. Firstly, we estimate the hand orientation; thereafter, we calculate the palm joints and the finger orientations at each time step. When each finger orientation is known, the finger joints at the previous time step are translated and rotated in such a way that all joints belong to the current finger plane. Finally, a constrained version of FABRIK, with rotational limitations, is incorporated to fit the joints of each finger. This procedure is given in detail in the following paragraphs.

The first step is to find the hand orientation; hence, by accepting that the hand plane Φ_x is similar to the palm plane and that the markers p, q and r are lying on that plane, the hand orientation, meaning the plane Φ_x, can be estimated. Therefore,

$$\begin{aligned} P &= \frac{1}{2}\left(p^2 n + 2p + -\tilde{n}\right) \\ Q &= \frac{1}{2}\left(q^2 n + 2q + -\tilde{n}\right) \\ R &= \frac{1}{2}\left(r^2 n + 2r + -\tilde{n}\right) \end{aligned} \tag{5.1}$$

where P, Q, and R are the $5d$ null vectors representing points p, q and r respectively, and n

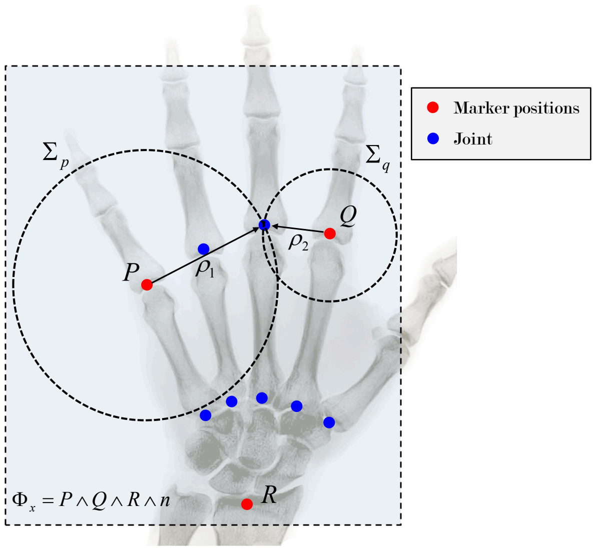


Figure 5.3.: *The palm plane constraints: the hand plane can be calculated using the marker positions P, Q and R, accepting that the markers lie on that plane and that the hand and palm planes are similar. The rest of the palm joints can be estimated, assuming that the inter-joint distances remain constant over time, by intersecting the spheres Σ_p and Σ_q with centres at the marker positions P and Q and radii of the distance between their centre and the joint position we are looking for.*

and $\bar{n}$ are the null vectors in CGA, as described in section 2.2. The plane Φ_x is equal to

$$\Phi_x = P \wedge Q \wedge R \wedge n = \left\langle \langle\langle PQ\rangle_2 R\rangle_3 n \right\rangle_4 \tag{5.2}$$

Note that the form given on the right hand side of 5.2, and other relevant equations, is useful for implementation purposes.

Calculating the palm joints

The next step is to incorporate constraints to obtain other palm joints. Thus, by assuming that the inter-joint distances (for the joints $F_{i,1}$ where $i = 1, \ldots, 5$ and $F_{j,2}$ where $j = 1, \ldots, 4$) are fixed over time and that all these joints lie on the palm plane, we can easily locate them using basic geometric entities such as planes, circles and spheres. An example of palm constraints is given in figure 5.3. For instance, the joint position we are working on can be estimated by intersecting the spheres with centres being the marker positions p and q and radii being the distance between the marker and that joint position (taken from the model). Therefore, find the sphere with its centre at the marker position P and radius equal to the distance between the marker P and the joint we are working on

$$\Sigma_p = \left(P - \frac{1}{2}\rho_1^2 n\right) I \tag{5.3}$$

where ρ is the sphere radii. Similarly, find the sphere with centre the marker position Q and radius equal to the distance between the marker Q and the joint we are working on

$$\Sigma_q = \left(Q - \frac{1}{2}\rho_2^2 n\right) I \tag{5.4}$$

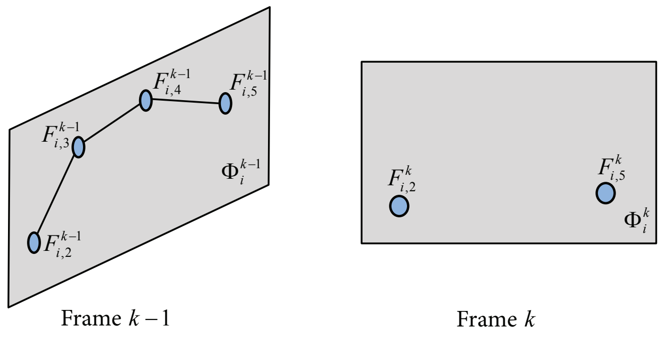


Figure 5.4.: *The joints' positions at times $k-1$ and k. Each finger joint at time $k-1$ needs to be rotated in such a way that all joints of that finger lie on the plane of the current frame k.*

The intersection of the two spheres gives a circle or a single point or no intersection. Thus, the meet between the two spheres is given by,

$$C = \Sigma_p \vee \Sigma_q = \left[\langle \Sigma_p \Sigma_q \rangle_2\right]^* \tag{5.5}$$

- if $C^2 > 0$, then C is a circle. In that case, the possible solutions are given by intersecting the circle C and the palm plane Φ_x

$$B = C \vee \Phi_x = [\langle C\Phi_x \rangle_3]^* \tag{5.6}$$

  - if $B^2 > 0$, the meet between C and Φ_x gives two points which can be extracted via projectors, as given in section 2.2.5. The new joint position is assigned as the point that is closer to the previous joint position (at time $k-1$).
  - if $B^2 = 0$, the intersection is a single point $X = BnB$.
  - if $B^2 < 0$, the intersection does not exist. For that instance, the new joint position is then taken as the nearest point on circle, C, from the previous joint position (at time $k-1$, see Appx A.1 for the solution).

- if $C^2 = 0$, the intersection is a single point $X = CnC$.
- if $C^2 < 0$, the two spheres do not intersect. In that case, the final joint position is given by averaging the distance between the two markers $x = (p+q)/2$.

Calculating the finger joints

In order to estimate the finger joints, we need to find the finger planes Φ_i, for $i = 1, \ldots, 4$. Each Φ_i can be calculated using the known joint positions $F_{i,2}$, the marker positions $F_{i,5}$ and by assuming that they are perpendicular to the palm plane Φ_x (note that this does not hold for the thumb plane Φ_5). Since both points from each finger are known (the motion capture system tracks the end effector positions $F_{i,5}$ and the finger roots $F_{i,2}$ lie on the palm plane with constant distance from the attached markers p and q, as explained in previous paragraphs), each finger plane can be estimated at the current time frame. The vector that is perpendicular

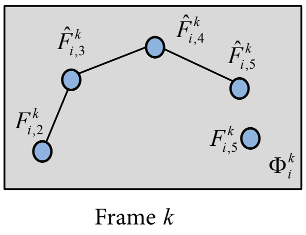


Figure 5.5.: *The current joint positions, after rotating them in order to lie on the current finger plane Φ^k_i. The problem of orientation is therefore solved and FABRIK can then be utilised assuming that the root of the chain is $F^k_{i,2}$, the end effector is the point $\hat{F}^k_{i,5}$ and the target is the current marker position $F^k_{i,5}$.*

to the hand plane Φ_x is given by

$$\hat{n} = \Phi^*_x - \frac{1}{2}\left(\Phi^*_x \cdot \bar{n}\right) n \tag{5.7}$$

as explained in section 2.2.5. The finger planes can then be calculated as

$$\Phi_i = F_{i,2} \wedge F_{i,5} \wedge \hat{n} \wedge n = \left\langle \left\langle \left\langle F_{i,2} F_{i,5} \right\rangle_2 \hat{n} \right\rangle_3 n \right\rangle_4 \quad \text{for} \quad i = 1, \ldots, 4 \tag{5.8}$$

The thumb orientation Φ_5 can be estimated using the marker position $F_{5,4}$, and the joint positions $F_{1,2}$ and $F_{5,2}$ that lie on the palm, assuming that when the thumb bends to the ventral side of the palm, it always points at the joint $F_{1,2}$ (approximately true in practice).

The next step is to estimate the rotation between the previous and the current frame of each finger plane. This can be done using CGA and rotors; the rotor R which expresses the rotation between the plane in the previous frame and the plane in the current frame, for each finger, can be found using either the *Procrustes* formulation [Hor87] or the GA equivalent given in [LFLD98]. Then each finger joint at time $k-1$ is translated and rotated in such a way that all joints of a given finger lie on the plane of the current frame k, as demonstrated in figures 5.4 and 5.5. Hence,

$$\hat{F}^k_{i,j} = R F^{k-1}_{i,j} \tilde{R} \tag{5.9}$$

where $i = 1, \ldots, 4$ and $j = 3, 4, 5$ (except for the thumb where $i = 5$ and $j = 2, 3, 4$).

All joints now lie on plane Φ^k_i. Lastly, FABRIK is applied to each finger chain, assuming that the root of the chain is $F^k_{i,2}$, the end effector is the rotated point $\hat{F}^k_{i,5}$ and the target is the current marker position $F^k_{i,5}$, as shown in figure 5.5. The inter-joint distances are constant over time, thus, for computational efficiency, they can be calculated and stored at the first frame.

The resulting posture can be further improved in accuracy and naturalness by incorporating constraints subject to the physiological model of the hand, taking into account the hand, fingers, muscle, skin and individual joint properties.

Table 5.1.: *Hand joint configuration*

		DoF	Rotational-x (degrees)		Rotational-y (degrees)		Orientational
			θ_1	θ_3	θ_2	θ_4	
$F_{i,1}$	$i = 1, ..., 4$	1	-	-	-	-	No twist
$F_{i,1}$	$i = 5$	2	20	20	30	40	No twist
$F_{i,2}$	$i = 1, ..., 4$	2	10	85	15	15	No twist
$F_{i,2}$	$i = 5$	2	5	30	10	10	No twist
$F_{i,3}$	$i = 1, ..., 5$	1	10	95	-	-	No twist
$F_{i,4}$	$i = 1, ..., 4$	1	10	90	-	-	No twist
Total		25					

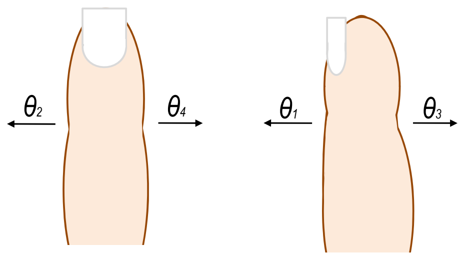


Figure 5.6.: *Graphical representation of the angles $\theta_1, ..., \theta_4$ which define the rotational constraints of each joint.*

5.5. Physiological Constraints

In addition to the basic rotational and orientational joint constraints, presented in [AL10b], it is important to incorporate motion limitations based on the model properties. Kaimakis and Lasenby, in [KL09], proposed a physiological model to restrict the hand poses according to the hand anatomy. That was the first paper which utilised physiological constraints taking into account the hand, finger and joint properties. In this study, the physiological hand constraints are defined using the same terminology as in [KL09], and each of these is outlined below in the order in which they appear in [KL09].

5.5.1. Inertia

The first physiological constraint discussed in [KL09] is *inertia*, a limitation correlated with the dynamics of the articulated structure. For implementation, it is assumed that all moving parts of the hand skeleton have similar velocity and acceleration at different time periods. Obviously, the kinematic movement can be divided into two classes, the spatial velocity of the hand's root giving the translation of the hand, and the local angular velocity of each bone.

The problem of inertia, in this study, is considered as solved since the marker positions are tracked by the optical motion capture system; however, this information can be useful in predicting the missing marker positions when the latter are occluded by elements of the scene. When the hand moves, the finger with the missing marker will have similar direction, velocity and acceleration to that of the hand. Thus, a better approximation of the filtering observation state can be achieved.

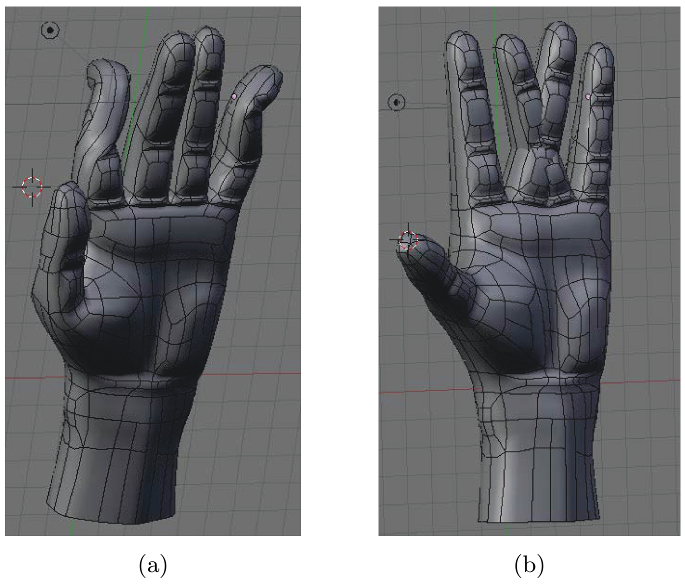


Figure 5.7.: *Examples of unnatural hand postures due to violation of the (a) flexion and abduction, (b) rigidity properties of the finger.*

5.5.2. Flexion

The design and physical restrictions of the human hand mean that fingers can move to the ventral side, but cannot move in any other direction. The movements that a hand can undergo are therefore restricted in terms of *flexion* and *extension*.

5.5.3. Abduction

Another family of limitations, caused by hand physiology, are the *abduction* and *adduction* constraints. These constraints control and limit the amount of sideways motion. In this study it is assumed that finger orientation is highly correlated with that of the hand palm.

The hands posture constraints, related to *flexion* and *abduction*, can be incorporated directly into FABRIK as rotational and orientational constraints, in the same way as described in section 3.4.1. For instance, a bone rotation can be limited by factorising it into two rotations: one "simple rotation" that moves the bone to its final direction vector and one that represents the twist around that final vector. The hand model studied here consists of 25 joints and has in total 25 DoFs. Table 5.1 lists the degrees of freedom for each joint as well as its rotational and orientational limits. Figure 5.6 presents the angles $\theta_1, ..., \theta_4$ which define the rotational limits of each joint $F_{i,j}$. Fingers do not twist, thus only rotational constraints are applied, locking the joint orientation to be identical to that of the palm (apart from the thumb).

FABRIK easily supports rotational and orientational constraints; the main idea is the re-positioning and re-orientation of the target to be within an allowed range bound. This can be accomplished by checking whether 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. The allowed range of motion is defined by the angles $\theta_1, ..., \theta_4$, which represent the minimum and maximum allowed

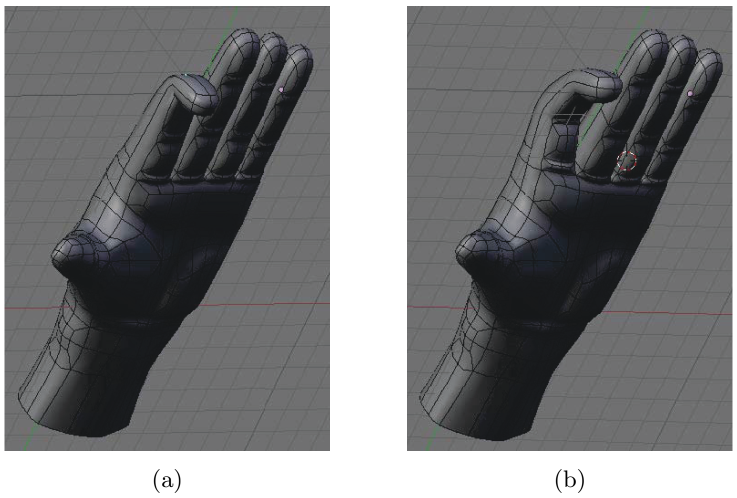


Figure 5.8.: *An example showing the intradigital correlation feature. (a) The intradigital correlation constraint is violated; even if the rotational and orientational constraints are satisfied, the posture of the hand is not natural since it is impossible to bend the distal phalanges without flexing the intermediate and proximal phalanges, (b) the correct posture of the hand when the intradigital correlation of the finger has been taken into account.*

rotation of each joint about the x and y-axes, respectively. More information on how FABRIK works, how joint limitations can be incorporated and how it can be extended to problems with multiple end effectors and targets, can be found in chapter 3 and [AL10d].

Figure 5.7(a) shows an example where the flexion and abduction constraints are not satisfied; the forefinger was erroneously rotated where the little finger was bent in an inappropriate direction and angle.

5.5.4. Intradigital correlation

As well as the limitations due to inertia, flexion and abduction, several posture restrictions are caused by the muscles of the hand. For instance, the phalangeal flexion in particular fingers is influenced by tendinous synapses with more than one phalanx of that finger. Therefore, it is clear that the muscle contraction and phalangeal flexion are not fully independent, but there is an inter-connection between them. [KL09] introduced the *intradigital correlation* constraint that is responsible for the inter-finger connections caused by certain tendons. In order to cope with these motion restrictions, we add an extra step; after the completion of the IK operation, a check mechanism is activated to identify whether the finger has a natural posture. If a violation of the transdigital correlation constraint is detected, the IK solver is repeated but this time with rotational constraints being applied to ensure that the flexion is uniformly distributed to all finger joints. However, in order to apply this constraint, the joint angles must be calculated, resulting in an increase in computational cost. Figure 5.8 shows an example where the intradigital correlation constraint is not satisfied; an unnatural pose of the hand is produced, even if the rotational and orientational constraints for individual joints are satisfied.

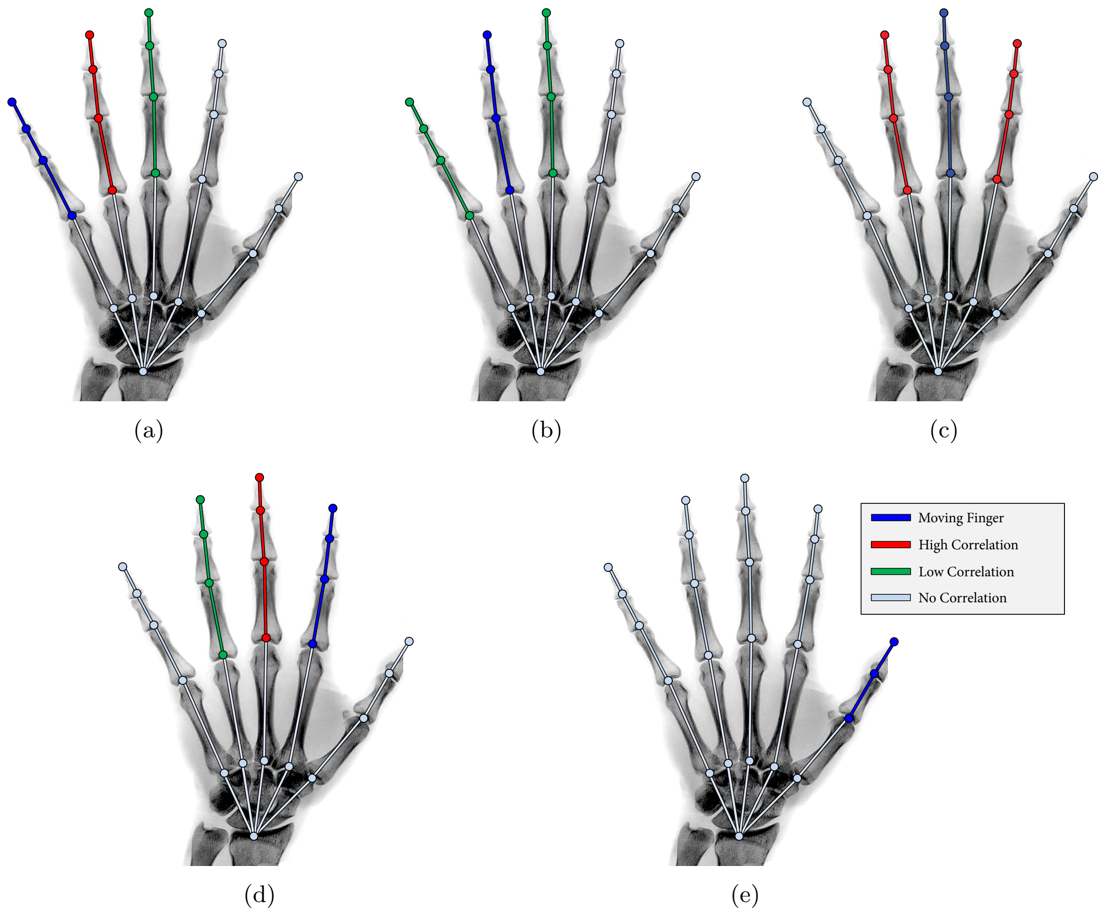


Figure 5.9.: *The linked pairs of bones which share a transdigital correlation movement. (a) the little finger and the affected neighbouring fingers, (b), (c) and (d) the ring, middle and index fingers, respectively, with the effect of their flexion on the neighbouring fingers, (e) the thumb. The thumb's movement is independent of the other fingers since it is directly connected to the trapezium. The moving fingers are highlighted in blue, the highly correlated fingers in red, the fingers having low correlation to the moving finger in green and finally, the fingers with no correlation are coloured in light gray.*

5.5.5. Transdigital correlation

Beyond the fingers' *intradigital correlation* constraint discussed in 5.5.4, the fingers also share *transdigital correlations*. In particular, [KL09] argues that certain ligaments and muscles interact to cause an amount of flexion to be transmitted across neighbouring fingers, as shown in figure 5.9. An exception to the transdigital correlation is the thumb, which moves independently of other fingers since it is directly connected to the trapezium. Figure 5.9 indicates the linked pairs of bones which share a correlation movement, subject to transdigital correlation.

An example showing violation of the transdigital correlation is given in figure 5.10(a) where even if both individual rotational and orientational constraints are satisfied, the resulting hand posture remains abnormal. Figure 5.10(b) shows the physiologically correct hand configuration for this specific pose.

The transdigital correlation constraint is assumed to be solved by the optical motion capture system, since each finger is tracked individually by labelled markers. However, this information

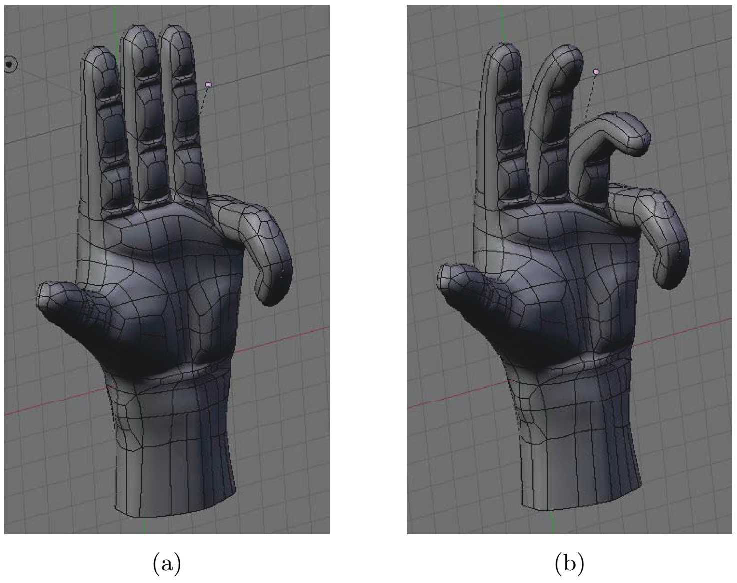


Figure 5.10.: *An example explaining the transdigital correlation of the hand. (a) the finger flexes without affecting its neighbouring fingers, breaching the transdigital correlation feature and producing an unnatural posture, (b) a realistic hand pose after taking into account the transdigital correlation between neighbouring fingers.*

finds applications in cases where the marker position is missing due to occlusions by elements of the scene, thus contributing to better estimates of the observation vectors.

5.5.6. Friction

Finally, [KL09] introduces *friction*, a hand property associated with the nature of the skin that restricts hand movements. For instance, when frictional forces are applied to a finger, they cause motion that is transmitted to other fingers. A clear example of the friction feature is given during the formation of the fist. Since, in this work, each finger position is tracked using a mocap system, this feature finds application only during the marker prediction algorithm returning better estimates of the missing marker positions.

5.6. Missing Marker Prediction

During motion capture, we encountered cases where markers were occluded or just not visible to the cameras. That is a common problem in optical motion capture since markers are self-occluded or occluded by other fingers or elements in the capture environment. To cope with the missing marker problem, a marker prediction mechanism is employed using a VTM-UKF model, similar to the one described in chapter 4 and [AL10c], where the observation vector is constrained subject to inferred information from the hand geometry and the available marker positions. Physiological hand constraints have been incorporated within the filter to give good estimates of possible positions of the observation state at each time period, such as those described in section 5.5. The VTM-UKF filter can also handle the negative effects of the

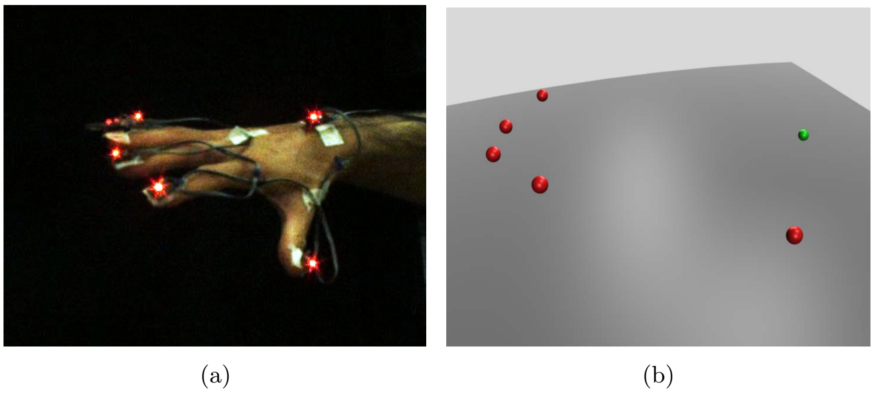


Figure 5.11.: *(a) The true hand pose, (b) the markers as seen from the motion capture system.*

marker data noise.

5.7. Experimental Results

Experiments were carried out using a 10 camera PhaseSpace motion capture system, capturing data at 100Hz [Pha]. The implemented methodology was able to process up to 70 frames per second. Runtimes were measured with custom MATLAB [MAT] code on a Pentium 2 Duo 2.2GHz. Our dataset comprises marker motion capture data. Data captured using colour video cameras are also used to compare the reconstruction quality between the estimated and the true hand postures. The reconstructed hand postures were visualised using a mesh deformation algorithm. Figure 5.11 shows an example of the hand with attached markers and how the markers are seen from the motion capture system.

5.7.1. Mesh deformation

A mesh deformation algorithm is employed to visualise the movements of the underlying hand skeleton in order to compare the resulting animations with the true hand poses. Animating an articulated 3D character requires manual rigging to specify its internal skeletal structure and to define how the input motion deforms its surface. [BP07] and [WL08] proposed mesh deformation algorithms driven by animation of an underlying skeleton, named bone-heat and bone-glow respectively. The articulated hand is automatically assigned a per-vertex and per-bone weighting given only by an underlying skeleton. In this work, we used the version of bone-heat implemented in Blender [Ble].

Figure 5.12 provides a representative illustration of implementing the mesh deformation algorithm; the mesh and armature representing the hand is automatically associated using the bone-heat algorithm.

5.7.2. Analysis of results

The proposed method is simple and has low computational cost, meaning it is real-time implementable. It requires 1.43msec per frame for tracking and fitting 25 joints, hence processing

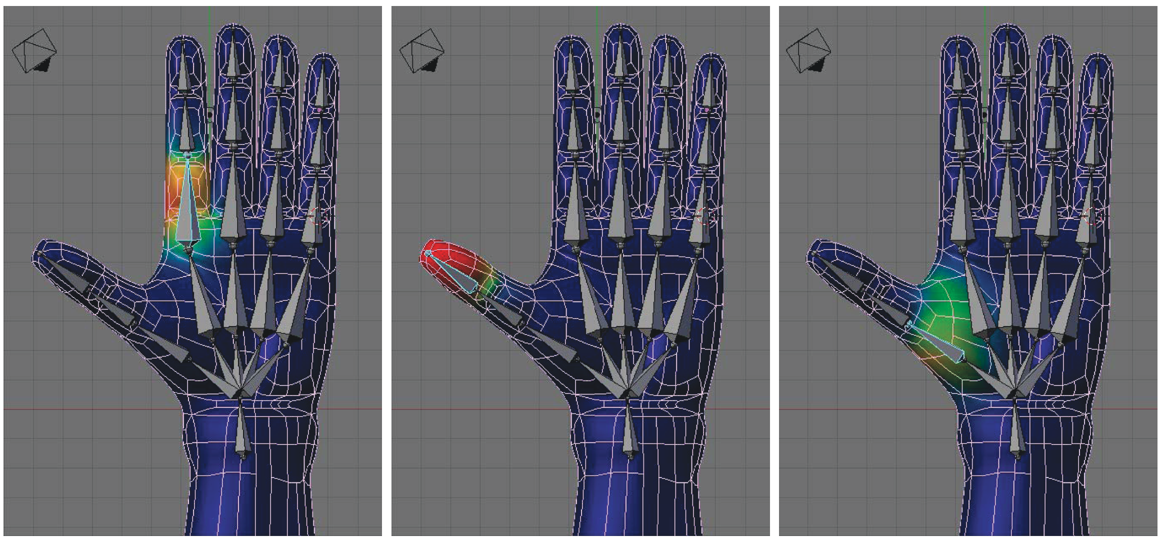


Figure 5.12.: *Example of simple linear blend skinning scheme applied using the weights from bone-heat. The weight assigned to each vertex has been indicated using a gradation from blue to red to indicate the range* $[0, 1]$.

Table 5.2.: *Average time needed at different frame rates.*

	Time per frame (msec)	Frames per second
Frame rate at 100Hz	1.4311	69.8
Frame rate at 10Hz	1.6251	61.5
Frame rate at 3Hz	2.2287	44.8

on average 70 frames per second when the data frame rate is high, and more than 40 frames per second for a low frame rate dataset. The rotational and orientational constraints ensure that each finger movement remains normal without showing asymmetries, or irregular bends and rotations. In addition, the physiological constraints restrict the results to anatomically correct postures, subject to the hand, muscle and skin properties.

The implemented system can smoothly track the hand movements, resulting in visually natural motion without abnormalities, oscillations and discontinuities. The reconstruction quality can be checked visually by comparing the generated 3D hand animations with the data captured using a colour video camera, as seen in figure 5.13; our system is precise, producing postures which meet the hand's physiological model restrictions and are very close to the true hand poses. Figure 5.14 shows an example of continuous hand pose tracking and reconstruction using a dataset captured at a frame rate of 10Hz. In this example, the hand flexes to its ventral side, to form a fist. It is difficult to illustrate the reconstruction quality in still images, but the resulting motion does not suffer from oscillation or discontinuities and each finger smoothly moves to the target.

We also investigate the performance of the system under different frame rates; clearly, the reconstruction quality is better when the frame rate becomes higher. However, even at a low frame rate (3 frames per second) the reconstruction quality of our methodology remains

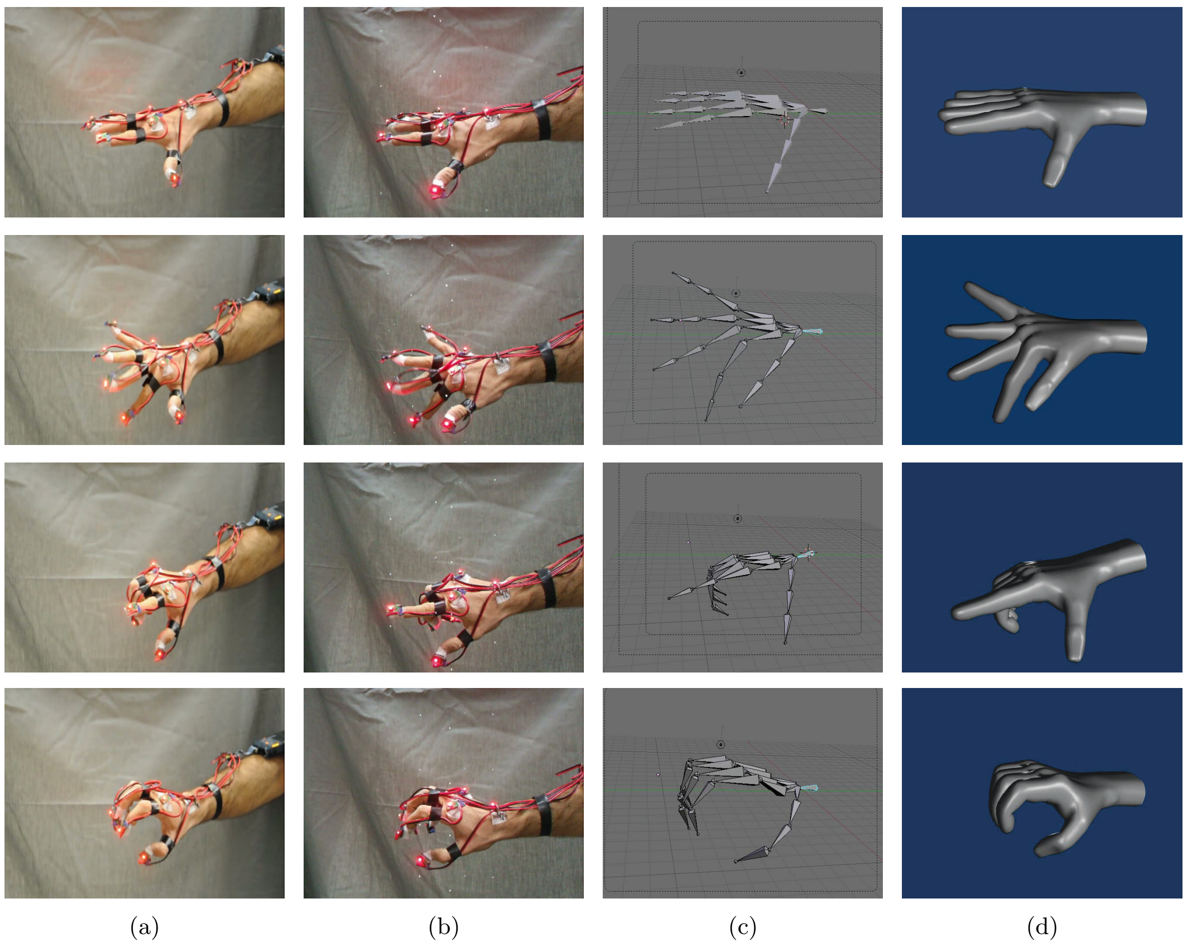


Figure 5.13.: *An example of hand reconstruction using our methodology at a frame rate of 100Hz frame. (a) View of the hand from RGB camera 1, (b) a different view of the hand from RGB camera 2, (c) the reconstructed skeleton and (d) the final visualised posture. The resulting poses are visually natural and biomechanically correct.*

sufficient, delivering smooth and visually natural results. The time needed for the IK solver to fit the joints to the model also varies for data captured at different frame rates; by reducing the frame rate, the distance between the target and end effectors is increased, thus more computational time is required from FABRIK to track the target positions. Table 5.2 lists the average time needed per frame to track the hand at different frame rates.

Despite the apparent accuracy in performance, the resulting postures of our approach are not unique; several possible poses could result from the 3D articulated hand tracking. However, the advantages of this method are its efficiency and ability to return natural and feasible motion, which meets the user constraints, with low computational cost. It is also important to note here that FABRIK results in poses which are closely related to previous states. Therefore, the final joint configuration might be different when the IK problem is solved with the end effectors in different initial positions but with similar final states. Nevertheless, these differences are minimal causing only a small decrease in performance. A video showing examples of implementation using different frame rates is included in the supplementary materials.

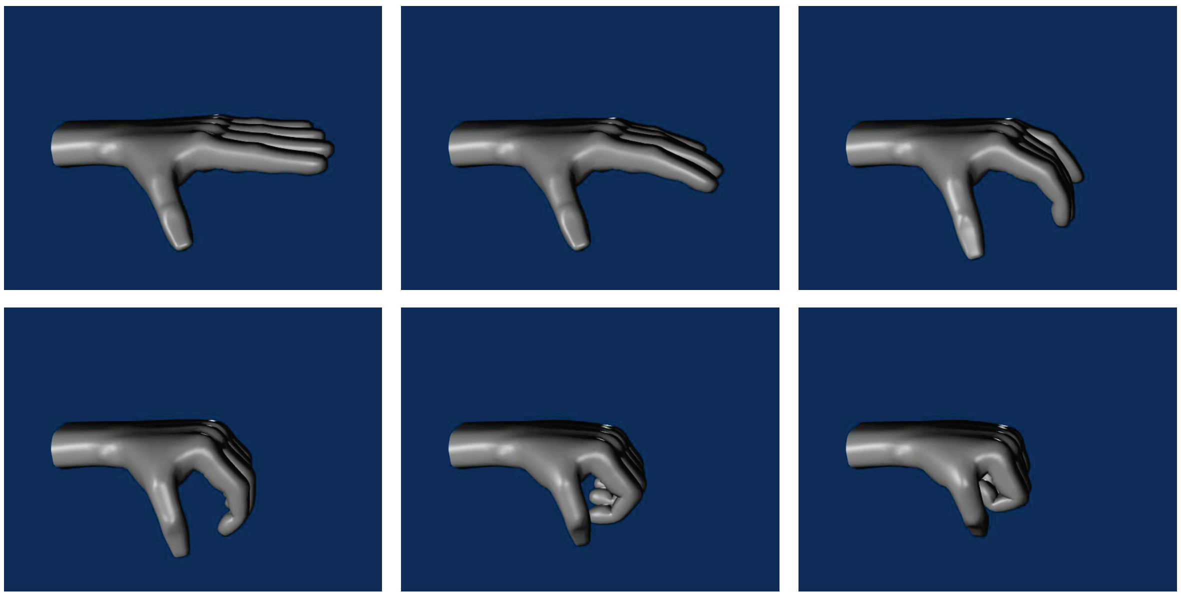


Figure 5.14.: *An example of continuous hand pose tracking, at a frame rate of 10Hz; in this example the hand flexes to its ventral side to form a fist.*

5.8. Conclusions and Future Work

In this chapter we have presented a system that can track a human hand of 25 DoFs relying on optical motion capture data; one labelled optical marker is attached on each finger, treated as end effectors, and 3 more markers are placed at strategic positions on the hand reverse-palm to help us identify the root and orientation of the hand. Physiological constraints relevant to the hand model, skin and muscle properties are incorporated, ensuring that the resulting movements remain within a feasible set. FABRIK, a real-time inverse kinematics solver, is employed to fit the remaining joint positions subject to the proposed physiological restrictions. The problem of missing data, due to marker occlusions, is solved using a marker prediction methodology similar to the one presented in chapter 4, recovering the missing positions in real-time. Finally, bone-heat, a mesh deformation algorithm, is used to visualise the results for evaluation and comparison.

The proposed methodology produces smooth and natural hand postures over time; the required processing time remains low enabling an effective real-time hand motion tracking and reconstruction system. The results are precise, producing visually natural and biomechanically correct movements. The system can process up to 70 frames per second; even with a low capture frame rate, the proposed methodology tracks the hand motion smoothly, without oscillations or discontinuities and with high reconstruction quality.

In future work, a more sophisticated model will be implemented which takes into consideration, in addition to the joint rotational and orientational restrictions, skin related constraints and constraints related to the rigidity of the hand model. The *rigidity* feature of the hand was not investigated in this work since the data was captured from a markered optical motion capture system and the 3D animated hand does not automatically have a mesh that defines its external shape. As the hand is the most mobile part of the human body, we expect a considerable degree of interaction between fingers, despite the limitations already discussed. Rigidity concerns such interactions, where different fingers may self-intersect, thus causing unnatural postures. While some of the constraints already discussed will limit such self-intersections, there are instances where extra movement restrictions must be applied. This problem is also known as self-collision and has been tackled in several papers, such as [KL09] and [LG98]. Figure 5.7(b) shows an example where two fingers collide disproportionately; by taking into account the rigidity of each finger, problems similar to hand collisions can be avoided.

6 Conclusions and Future Work

Motivated by the need to develop effective real-time inverse kinematics solutions, this thesis intends to provide insights into existing computer vision techniques and presents a novel iterative IK solver. Also, we investigate real-time marker prediction and centre of rotation estimation techniques as well as a hand pose tracker for real-time visual interactions.

In this chapter, we summarise the findings from previous chapters, we describe the main contributions of this thesis and we propose future directions and applications.

6.1. Contributions

CGA is a mathematical framework that offers a compact and geometrically intuitive formulation of algorithms and an easy and immediate computation of rotors. Rotors are simpler to manipulate than Euler angles and avoid the problem of gimbal lock[1]. It also simplifies the mathematical model since basic entities, such as spheres, lines, planes and circles, are simply represented by algebraic objects. Thus, CGA gives us the ability to describe algorithms in a geometrically intuitive and compact manner, making it suitable for applications in engineering, computer vision and robotics.

[1]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.

Forward And Backward Reaching Inverse Kinematics: Inverse Kinematics 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. Chapter 3 presents a review of algorithms related to the IK problem; it also introduces a novel iterative IK solver, named Forward And Backward Reaching Inverse Kinematics (FABRIK). Firstly, this chapter concerns the articulated body model describing the human joint types, and gives a brief introduction to human models and motion; it also considers the most popular numerical solutions to the IK problems, 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. Heuristic iterative approaches, such as CCD and its extension (IIK), as well as IK solvers that solve the IK problem in a sequential fashion, are discussed. Lastly, the FABRIK algorithm is compared against the most popular IK solvers and under several different conditions: (a) with and without manipulator constraints, (b) with single and multiple end effectors, (c) tracking moving targets and (d) when the target has significant distance from the end effector. Several tests and comparisons have been implemented between the IK algorithms regarding their computational cost, processing time, conversion error, the number of iterations needed to reach the target as well as how visually realistic the resulting postures are.

FABRIK is the first algorithm that uses an iterative method with points and lines to solve the IK problem. Thus, no rotational angles or matrix calculations are needed. It divides the problem into 2 phases, a forward and backward reaching approach, and it supports most of the joint types by repositioning and re-orienting the target at each step. It does not suffer from singularity problems, it is easy to implement and computationally efficient. No pre-recorded motion database is necessary, thereby avoiding the need for extra memory. An efficient approach of how joint constraints can be embedded to the FABRIK algorithm is also illustrated, restricting the resulting poses to a naturally feasible set. FABRIK is able to solve problems with closed loops and can be easily expanded to treat problems with multiple end effectors, meaning that it is able to support complex models with multiple kinematic chains. It is experimentally shown 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 visually realistic postures, with and without constraints, reaching the desired position with the lowest computational cost. It executes in real-time and has the best performance on tracking a moving target and produces natural movements without oscillations and discontinuities. The FABRIK algorithm can be applied to a variety of problems, such as visual modelling, virtual creature animation, game consoles, rehabilitation medicine etc. Some examples of FABRIK applications are demonstrated in chapters 4 and 5.

Marker Prediction and Centre of Rotation Estimation: Chapter 4 includes algorithms related to the problem of using marker-based optical motion capture data in order to automatically establish a skeleton model to which the markers are attached. The experimental framework was described, including the cameras used for the experiments, the association between the markers and the limbs, and how the marker data was clustered into groups corresponding to the limb segments to which they were attached. We also investigated the problem of fitting skeletal models to marker-based optical motion capture data. Using the well known *Procrustes* formulation, we showed how to establish estimates of the relative orientation of a limb in each frame relative to a reference frame. Thereafter, a method for real-time estimation of the joint locations and identification of the optimal skeleton was implemented. That method takes advantage of the approximation that all markers on a segment were attached to a rigid body, acquiring the skeleton model from a stream of motion capture data. However, a common issue for motion capture, even when multiple cameras are used, is marker occlusion by elements of the scene, leading to missing data. In order to unambiguously establish the marker position, each marker must be visible to at least two cameras in each frame. Section 4.5 introduces a prediction method which copes with such instances. An Unscented Kalman Filter with a variable turn model has been deployed for marker tracking. Information on the missing markers in the current frame were inferred from an approximate rigid body assumption, which has been used for the observation states. With a continuous stream of accurate 3D data, we were able to perform real-time CoR estimation and produce skeletal information for visual performance feedback. Without assuming any skeleton model, we took advantage of the fact that, for markers on a given limb segment, the inter-marker distance is approximately constant. Also, the proposed system uses information about the missing markers which are visible to a single camera, reducing the marker position estimation error to a minimum. To the best of our knowledge, this is the first time that such information has been used for real-time marker prediction in optical motion capture.

The joint estimations were further improved by employing the FABRIK algorithm to ensure that the inter-joint distances remain constant over time. FABRIK simply re-positions the CoRs and guaranteeing that the bones lengths will not change during the marker prediction period. This was done with an added computational cost of only 5.27%. Experiments demonstrate that our methodology outperforms, in terms of accuracy, the methods used for comparison and effectively recovers good estimates of the true positions of the missing markers. It is able to maintain real-time marker predictions for extended time periods, even in the presence of several marker occlusions, thus enabling good estimates of the CoR. The method is reliable even if the limb rapidly changes direction and provides precise estimates of the markers and CoR locations even on data with large sequences of missing markers. The resulting motion is natural and smooth, without discontinuities or oscillations. This is the first work which uses an IK methodology to restrict, in real-time, the inter-joint distances in optical motion capture under marker occlusions.

Hand Pose Tracker: Analytically and anatomically correct models are necessary to control and constrain the movements of any legged body. Chapter 5 introduces a hand pose tracker that requires a minimum number of available markers, one on each finger; the labelled markers' positions are tracked using an optical motion capture system, such as [Pha]. The hand model is highly constrained using a sophisticated physiology model, which takes into account not only the joint rotational and orientational restrictions, but also constraints related to the hand anatomy, such as self-collisions, inertia, flexion etc. This extension provides more accurate results, ensuring that the hand adopts smoother and more natural poses. FABRIK was also employed to fit the other joint positions and manipulate the finger movements. Physiological constraints were used to ensure that the states of the articulated body conform to the data, minimise the misfits and do not violate the hand shape. Finally, a mesh deformation algorithm was applied to visualise the movements of the underlying hand skeleton and to compare the resulting animations with the true hand poses. Our experimental results demonstrate high precision and reliability in tracking, while also achieving real-time performance.

6.2. Publications

Several papers have been published based on the content of this thesis; these papers are enumerated in inverse chronological order below:

Andreas Aristidou and Joan Lasenby, "Real-Time Marker Prediction and CoR Estimation in Optical Motion Capture", *Submitted to the Image and Vision Computing*, February 2010.

Andreas Aristidou and Joan Lasenby, FABRIK: a fast, iterative solver for the Inverse Kinematics problem, *Submitted to Graphical Models*, February 2010.

Andreas Aristidou and Joan Lasenby, Inverse Kinematics solutions using Conformal Geometric Algebra, *In Proceedings of the 4th conference on Applied Geometric Algebras in Computer Science and Engineering (AGACSE'10)*, The Netherlands, June 14-16, 2010.

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'10)*, Cyprus, May 3-5, 2010.

Andreas Aristidou and Joan Lasenby, Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver, *Technical Report, CUEDF-INFENG, TR-632*, Cambridge, September 2009.

Andreas Aristidou, Joan Lasenby and Jonathan Cameron, Methods for Real-time Restoration and Estimation in Optical Motion Capture, *Technical Report, CUEDF-INFENG, TR-619*, Cambridge, January 2009.

Andreas Aristidou, Jonathan Cameron and Joan Lasenby, Predicting Missing Markers to Drive Real-Time Centre of Rotation Estimation, *In Proceedings of the 5th International Conference on Articulated Motion and Deformable Objects, AMDO'08*, pages 239-247, Spain, July 9-11, 2008.

Andreas Aristidou, Jonathan Cameron and Joan Lasenby, Real-Time Estimation of Missing Markers in Human Motion Capture, *In IEEE Proceedings of the 2nd International Conference on Bioinformatics and Biomedical Engineering, iCBBE'08*, pages 1343-1346, China, May 16-18, 2008.

6.3. Future Directions

Forward and Backward Reaching Inverse Kinematics: Future work will see the introduction of FABRIK within a larger variety of analytically and anatomically correct models. A further study of the collision problem, with simultaneous study of more sophisticated joint types, is also essential for the production of smoother and more natural movements. FABRIK could also find applications in markerless motion capture, e.g. problems with silhouette data, to solve the data-fitting optimisation problem.

Marker prediction and CoR estimation: The marker prediction and CoR accuracy could be further improved by incorporating biomechanical constraints to control the postures of animated characters, in optical motion capture under marker occlusion; however, prior knowledge of the model and joints is needed, reducing its generality to specific models. In addition, a hybrid system with low cost inertial measurement units could be employed allowing extra validation of the data.

Hand pose tracker: A hand model which takes into consideration constraints relevant to the skin properties and the hand's rigidity will be studied. The data in this thesis is captured from an optical motion capture system, thus there is no mesh to define the hand's shape; the mesh presented in the examples was driven by the underlying skeleton using bone-heat, a mesh deformation algorithm. Problems relevant to the mobility of the hand, such as self-collisions, were not investigated.

Appendices

A Trigonometric solutions using Geometric Algebra

Appendix A illustrates the CGA mathematical solutions described in this thesis. Most of these CGA solutions are applied to the marker constraint problem, as discussed in chapter 4.6. An alternative implementation of the FABRIK algorithm using CGA is also presented.

A.1. Nearest Point on a Circle from a Point in Space

This section considers the problem of constraining the estimated marker positions to give a better approximation of the observation vector used in the UKF filter (section 4.5.2). Using information from the rigid body, and also keeping in mind the fact that markers have a constant distance between each other, we can find estimates of the non-visible marker. The observation vector describing the occluded marker m_1 in frame k, $\hat{\mathbf{x}}_1^k$, can be assigned as the shortest distance between the intersection of the two spheres with centre the positions of the markers m_2 and m_3, radii the distances D_{12} and D_{13}, and the point in space, $\tilde{\mathbf{x}}_1^k$. The solution presented here is solved using Conformal Geometric Algebra (CGA, see [LLW04]) as shown in the next paragraphs. The intersection of 2 spheres is discussed in section 2.2.6. The distance from the point $\tilde{\mathbf{x}}_1^k$ (which can be considered as any point in space) to $\hat{\mathbf{x}}_1^k$ can be solved as the problem of finding the nearest point on a circle to a point in space.

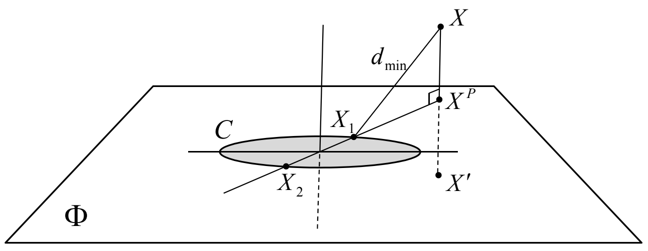


Figure A.1.: *The nearest point on circle to point in space. The point X is projected on the circle's plane Φ. A line is then formed through the midpoint of X and its projected counterpart and the centre of the circle. The intersection between the line and the circle returns two possible solutions, the one that is shorter to the point X is chosen.*

The minimum distance on a circle from a point in space is related to the projection of that point onto the plane Φ of the circle. This can be achieved by reflecting the point in the plane and finding the mid-point of the reflected and the original point. Hence, let the circle $C = H(b) \wedge H(c) \wedge H(d)$, where b, c and d are points that lie on the circle and the operation of $H(x)$ is the Hestenes' mapping in 2.2. The centre c of the circle C can be calculated as,

$$c = CnC \tag{A.1}$$

and the plane Φ of the circle can be formulated as:

$$\Phi = C \wedge n = \langle Cn \rangle_4 \tag{A.2}$$

Having the plane Φ and the point $X = H(x)$ in space, the nearest point on circle can be found by reflecting that point in the plane Φ.

$$X' = \Phi X \Phi \tag{A.3}$$

The mid-point X^P is then calculated as:

$$X'^P = X^P + \alpha n = H\left(\frac{1}{2}\left(H^{-1}\left(X'\right) + x\right)\right) \tag{A.4}$$

Then, a line, L, is formed through this midpoint and the centre of the circle,

$$L = X^P \wedge c \wedge n \tag{A.5}$$

The intersection between line L and circle C will return a bivector, $A \wedge B$, which represents the shortest and longest distances on the circle from the point in space. The vectors X_1 and X_2 can be extracted from $X_1 \wedge X_2$ using projectors (see equation 2.65). The nearest point is then selected using a simple distance comparison method. This method is also illustrated in figure A.1.

$$\begin{aligned}(X_1, X_2) &= L \vee C \\ X &= \arg\left(\max\left(X_1 \cdot X, X_2 \cdot X\right)\right)\end{aligned} \tag{A.6}$$

A.2. Nearest Point on a Line from a Circle

Section A.2 considers the problem of finding the nearest point on a line from a circle in order to obtain a better estimate of the missing marker position. In section 4.6.2 we presented an efficient model for estimating the missing marker on a limb segment, in the case where 2 markers are visible on the same limb segment. Also, in section 4.6.5 we talked about instances where missing markers are visible in a single camera. This information identifies a line, L_1, starting from the camera and passing through the position of the missing marker. Here we present an alternative way of estimating the missing marker which is visible to one camera, by finding the nearest point on line L_1 from circle C_1. Circle C_1 is the circle created after the intersection of the spheres Σ_1 and Σ_2, as described in section 4.6.2.

Consider a line L_1 passing through the points a and b, which are represented in CGA as:

$$A = \frac{1}{2}\left(a^2 n + 2a - \bar{n}\right) \tag{A.7}$$

$$B = \frac{1}{2}\left(b^2 n + 2b - \bar{n}\right) \tag{A.8}$$

Line L_1 will then be equal to:

$$L_1 = A \wedge B \wedge n = \langle\langle AB\rangle_2 \, n\rangle_3 \tag{A.9}$$

Assume that circle C_1 lies on the plane Φ. Line L_1 intersects with the plane Φ, otherwise L_1 and Φ are parallel. Thus, the bivector B is given by

$$B = \Phi \vee L_1 = \left[\langle\Phi L_1\rangle_3\right]^* = \langle\Phi L_1\rangle_3 \, I \tag{A.10}$$

If $B^2 > 0$, then L_1 intersects with plane Φ, while if $B^2 = 0$, then L_1 and plane Φ are parallel. Firstly, consider the case where line L_1 and plane Φ intersect. In such a case the bivector B will be equal to $B = P \wedge n$ or $B = n \wedge P$. The vector P can be extracted from bivector B as discussed in section 2.2.5. Thereafter, we need to find the projection of the centre of the circle, $C = H\left(c\right)$, on line L_1. This is achieved by reflecting the centre of the circle in the line and then finding the mid-point, E, of point C and its reflection C'.

$$C' = L_1 C L_1 \tag{A.11}$$

$$E = H\left(\frac{1}{2}\left(H^{-1}\left(C'\right) + H^{-1}\left(C\right)\right)\right) \tag{A.12}$$

We then need to find the point Y which is the projection of point E onto the plane Φ. A

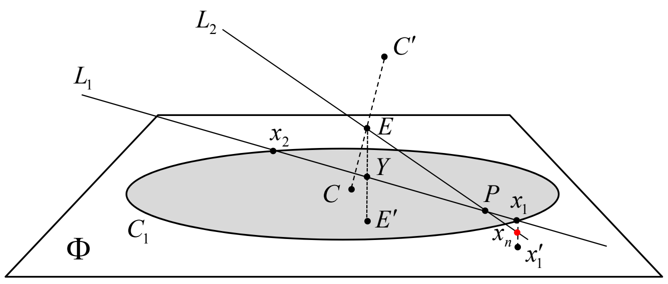


Figure A.2.: *Finding the nearest point on a line from a circle using CGA. Case where the distance from the centre of the circle C_1 to the point Y is smaller than the radius of the circle and point P is inside the circle area.*

similar procedure as above is used;

$$E' = \Phi E \Phi \tag{A.13}$$

$$Y = H\left(\frac{1}{2}\left(H^{-1}\left(E'\right) + H^{-1}\left(E\right)\right)\right) \tag{A.14}$$

After finding the point Y, the algorithm can be separated into three possible subcases:

- If the distance from the centre of the circle, C, to the point Y, is smaller than the radius of the circle, then a line $L_2 = P \wedge Y \wedge n$ will be established. The intersection of line L_2 with the circle C_1 will return 2 points, x_1 and x_2, i.e. those points on the circle with the shortest and longest distances from point P. The selection of the appropriate point can be achieved using a simple distance comparison.

$$(x_1, x_2) = L_2 \vee C = \left[\langle L_2 C\rangle_4\right]^* \tag{A.15}$$

$$X = \arg\left(\max\left(X_1 \cdot P, X_2 \cdot P\right)\right) \tag{A.16}$$

where X_1 and X_2 are the CGA representations of x_1 and x_2 respectively.

Assume that the point with the shortest distance from P is point x_1. Obviously, the point on L_1 which is closest to x_1 is the projection of that point onto the line. Hence, the point we are looking for, x_n, is equal to:

$$X_1' = L_1 X_1 L_1 \tag{A.17}$$

$$x_n = H\left(\frac{1}{2}\left(H^{-1}\left(X_1'\right) + H^{-1}\left(X_1\right)\right)\right) \tag{A.18}$$

Figures A.2 and A.3 illustrate the above. Point x_n is given in red.

However, there are some extreme instances to be considered:

- In the case where point P and Y are identical (Line L_1 is perpendicular to plane Φ), then $L_2 = C \wedge Y \wedge n$. The rest of the algorithm remains the same.
- In the case where line L_1 passes through the centre of the circle, line L_2 cannot be established using the above procedure. Hence, we consider line L_2 as the projection

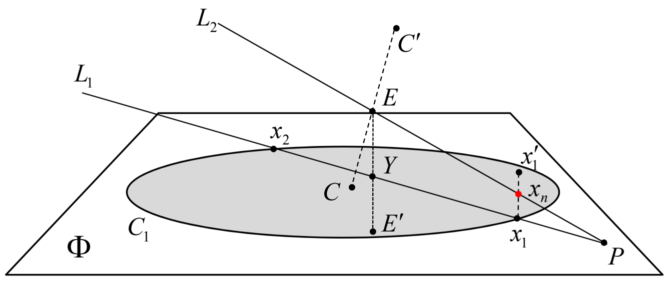


Figure A.3.: *Finding the nearest point on a line from a circle using CGA. Case where the distance from the centre of the circle C_1 to the point Y is smaller than the radius of the circle and point P is outside the circle area.*

of line L_1 onto the plane Φ. The rest of the algorithm remains the same.

  – In the case where line L_1 is perpendicular to plane Φ and also passes through the centre of the circle, then any point on circle C_1 has the same distance from line L_1.

- If the distance from the centre of the circle, C, to the point Y, is equal to the radius of the circle, then $x_1 = Y$. In such a case, the point on line we are looking for, x_n can be achieved by projecting the point x_1 onto the line L_1 as above.

- If the distance from the centre of the circle, C, to the point Y is greater than the radius of the circle, then a line L_2 will be introduced which is equal to $L_2 = C \wedge Y \wedge n$. Hence, we have a similar situation as above but line L_2 is different. Again, the intersection of line L_2 with the circle C_1 will return 2 points, x_1 and x_2, and the appropriate point can be selected as the point with the shortest distance from point P. Thus, point x_n is related to the projection of point x_1 onto line L_1. Figure A.4 illustrates the above.

In the second case, where $B^2 = 0$, plane Φ and line L_1 are parallel. We can check if the circle and the line intersect by calculating the *meet* between them.

$$G = C \vee L_1 = [\langle CL_1\rangle_4]^* = \langle CL_1\rangle_4 I \tag{A.19}$$

There are three further possible subcases:

- If $G = 0$ then line L_1 belongs to plane Φ and intersects with the circle at two points. In such a case we can extract the two points as in section 2.2.5. Both points are on line L_1 and on circle C.

- If $G = X$ and $X^2 = 0$ then line L_1 lies in the plane Φ and intersects with the circle at one point. That is the point we are looking for.

- If $G = X$ and $X^2 \neq 0$ then line L_1 and circle do not intersect at any point. This case can be further separated in two subcases; the instance where line L_1 lies in the plane Φ and the instance where it does not.

  – In the case where line L_1 lies in the plane Φ, then the nearest point on the line from the circle can be achieved by finding the projection of the centre of the circle onto the line.

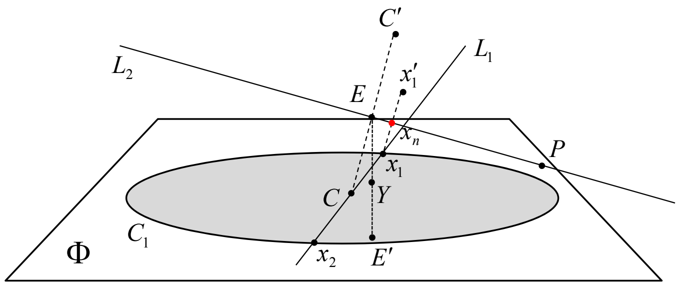


Figure A.4.: *Finding the nearest point on a line from a circle using CGA. Case where the distance from the centre of the circle C_1 to the point Y is greater than the radius of the circle and point P.*

– In the case where line L_1 does not lie in the plane Φ, then the problem can be solved by finding the projection of line L_1 onto the plane Φ. Hence, we can continue as above, having a line on plane Φ.

A.3. Nearest Point on a Line from a Sphere

This section focuses on the problem of finding the nearest point on a line from a sphere. In section 4.6.3, an efficient model for estimating missing markers has been proposed for the case where 2 markers on a limb segment are occluded. However, there are instances where the missing marker is visible in a single camera, giving us more information about the position of this missing marker. This information identifies a line, L_1, starting from the camera and passing through the position of the missing marker. Section 4.6.5 presents a reliable approach of estimating the missing markers for such instances. Here we discus an alternative way of estimating the occluded marker, which is visible to one camera, by finding the nearest point on line L_1 from sphere Σ_1. Σ_1 has as its centre the position of the visible marker m_2 and radius the distance $|\hat{\mathbf{D}}^k_{j,2}|$, as calculated in section 4.6.3.

Consider a line L_1 passing through the points a and b, which are represented in CGA as:

$$A = \frac{1}{2}\left(a^2 n + 2a - \bar{n}\right) \tag{A.20}$$

$$B = \frac{1}{2}\left(b^2 n + 2b - \bar{n}\right) \tag{A.21}$$

Line L_1 is then:

$$L_1 = A \wedge B \wedge n = \langle \langle AB \rangle_2 \, n \rangle_3 \tag{A.22}$$

We now need to find the projection of the centre of the sphere, C, onto line L_1. This can be achieved by reflecting the centre of the circle in the line and then finding the mid-point, Y, between the original, C, and the reflected point C'.

$$C' = L_1 C L_1 \tag{A.23}$$

$$Y = H\left(\frac{1}{2}\left(H^{-1}\left(C'\right) + H^{-1}\left(C\right)\right)\right) \tag{A.24}$$

The problem can now be separated into three possible subcases; the case where line L_1 intersects the sphere at two points, the case where line L_1 touches the sphere and the case where the line L_1 and the sphere have no intersection points. The solutions for all cases is given below:

- If the distance between the centre of the sphere and the point Y is greater than the radius of the sphere, then line L_1 does not intersect with sphere Σ_1. Hence, Y represents the point on line we are looking for.
- If the distance between the centre of the sphere and the point Y is equal to the radius of the sphere, then line L_1 intersect with sphere Σ_1 at exactly one point, the point Y. Obviously, Y represents the point on the line with minimum distance from the sphere.
- If the distance between the centre of the sphere and the point Y is smaller than the radius of the sphere, then line L_1 intersect with sphere Σ_1 at two points. Both points could represent the point we are looking for. However, we select as the position of the missing marker the point which returns minimum distance from the position in the previous frame.

A.4. Nearest Point on a Sphere from a Point in Space

This section presents how to calculate the nearest point on a sphere from a point in space using CGA. Assume that a sphere has centre $\underline{c}$ and radius ρ. The sphere Σ_1 can be expressed as blades in CGA as follows:

$$\Sigma_1 = \left(c - \frac{1}{2}\rho^2 n\right) I$$

where

$$c = \frac{1}{2}\underline{c}^2 n + \underline{c} - \frac{\bar{n}}{2}$$

Assume a point in space q. In order to find the nearest point on the sphere from that point, we need to find the intersection of the line L_1, that passes through the point q and the sphere centre $\underline{c}$. Thus,

$$L_1 = Q \wedge c \wedge n = \langle\langle Qc\rangle_2 n\rangle_3 \tag{A.25}$$

where $Q = H(q)$ is the Hestens mapping of q. The intersection between the line L_1 and the sphere Σ_1 always returns two possible solutions, which are given by the bivector, $A \wedge B$.

$$A \wedge B = L_1 \vee \Sigma_1 \tag{A.26}$$

Finally, the vectors A and B can be extracted from $A \wedge B$ using equations 2.65. Then, the nearest point on the sphere is assigned as the point that returns the minimum distance from the point in space.

A.5. IK Solutions using Conformal Geometric Algebra

In general, CGA gives us the ability to describe algorithms in a geometrically intuitive and compact manner; it simplifies the mathematical model of the inverse kinematics solver, since basic entities, such as spheres, lines, planes and circles, are simply represented by algebraic objects. It is worth noting that FABRIK performs three times faster when it is implemented by simply taking distances along lines rather than intersecting with spheres, as described in chapter 3.3. However, when we wish to incorporate constraints, we often need the sphere-line information, so in this appendix we present the entire work in this unified framework. In addition, several optimisations can be applied using CGA; for instance, a direct estimation of a missing joint, when it is between 2 true positions, can be achieved by intersecting the 2 spheres rather than finding an approximation using the normal iterative algorithm. Another simple optimisation is the direct construction of a line pointing towards the target, when the latter is unreachable. The FABRIK algorithm is still real-time implementable and is described in pseudocode in Alg. 5. The FABRIK solution in CGA was also presented in [AL10a].

The implementation of FABRIK in CGA is similar to the normal algorithm. Thus, assume that $\mathbf{p}_1, ..., \mathbf{p}_n$ are the joint positions of a manipulator. For the simple case where only a single end effector exists, take $\mathbf{p}_1$ as the root joint and $\mathbf{p}_n$ as the end effector. The target is symbolised as $\mathbf{t}$ and the initial base position by $\mathbf{b}$. First calculate the distances between each joint $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$, for $i = 1, ..., n-1$. Then, to 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, $\mathbf{p}'_n = \mathbf{t}$. The new position of the $(n-1)^{th}$ joint, $\mathbf{p}'_{n-1}$, is assigned as the nearest point on the sphere Σ_{n-1}, with centre the joint position $\mathbf{p}'_n$ and radii the distance d_{n-1}, from the joint position $\mathbf{p}_{n-1}$. Similarly, the new position of the $(n-2)^{th}$ joint, $\mathbf{p}'_{n-2}$, is selected as the nearest point on sphere Σ_{n-2}, with centre the joint position $\mathbf{p}'_{n-1}$ and radii the distance d_{n-2}, from the joint $\mathbf{p}_{n-2}$. The algorithm continues until all new joint positions are calculated, including the root, $\mathbf{p}'_1$. The trigonometric solution of the nearest point on a sphere from a point in space in given in section A.4.

A full iteration is completed when the same procedure is repeated but this time starting

Algorithm 5: A full iteration of the FABRIK algorithm using CGA.

```
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 $i = 1, ..., n$.
5.1  % The distance between root and target
5.2  $dist = |\mathbf{p}_1 - \mathbf{t}|$
5.3  % Check whether the target is within reach
5.4  if $dist > d_1 + d_2 + ... + d_{n-1}$ then
5.5      % The target is unreachable
5.6      for $i = 1, ..., n-1$ do
5.7          % Find the nearest point on sphere, with centre the joint position $\mathbf{p}_i$ and radius the distance
             $d_i$, from a point is space, $\mathbf{t}$
5.8          $\mathbf{p}_{i+1} = \mathbf{NearestPointSphere}(\mathbf{p}_i, d_i, \mathbf{t})$;
5.9      end
5.10 else
5.11     % The target is reachable; thus, set as $\mathbf{b}$ the initial position of the joint $\mathbf{p}_1$
5.12     $\mathbf{b} = \mathbf{p}_1$
5.13     % Check whether the distance between the end effector $\mathbf{p}_n$ and the target $\mathbf{t}$ is greater than a
         tolerance.
5.14     $dif_A = |\mathbf{p}_n - \mathbf{t}|$
5.15     while $dif_A > tol$ do
5.16         % STAGE 1: FORWARD REACHING
5.17         % Set the end effector $\mathbf{p}_n$ as target $\mathbf{t}$
5.18         $\mathbf{p}_n = \mathbf{t}$
5.19         for $i = n-1, ..., 1$ do
5.20             % Find the nearest point on sphere, with centre the joint position $\mathbf{p}_{i+1}$ and radius the
                 distance $d_i$, from a point is space, $\mathbf{p}_i$
5.21             $\mathbf{p}_i = \mathbf{NearestPointSphere}(\mathbf{p}_{i+1}, d_i, \mathbf{p}_i)$;
5.22         end
5.23         % STAGE 2: BACKWARD REACHING
5.24         % Set the root $\mathbf{p}_1$ its initial position.
5.25         $\mathbf{p}_1 = \mathbf{b}$
5.26         for $i = 1, ..., n-1$ do
5.27             % Find the nearest point on sphere, with centre the joint position $\mathbf{p}_i$ and radius the
                 distance $d_i$, from a point is space, $\mathbf{p}_{i+1}$
5.28             $\mathbf{p}_i = \mathbf{NearestPointSphere}(\mathbf{p}_i, d_i, \mathbf{p}_{i+1})$;
5.29         end
5.30         $dif_A = |\mathbf{p}_n - \mathbf{t}|$
5.31     end
5.32 end
5.33
5.34 % Where the function NearestPointSphere($X$,$Y$,$Z$), finds the nearest point on a sphere from a
     point in space. $X$ is the sphere's centre, $Y$ is the sphere's radii and $Z$ is the point in space.
```

from the root joint and moving outwards to the end effector. Thus, let the new position for the 1st joint, $\mathbf{p}_1''$, be its initial position $\mathbf{b}$. Then, the new joint position $\mathbf{p}_2''$ is assigned as the nearest point on the sphere Σ_1, with centre the $\mathbf{p}_1''$ and radii the distance d_1, from the joint $\mathbf{p}_2'$. This procedure is repeated for all the remaining joints, including the end effector. FABRIK is illustrated in pseudo-code in Algorithm 5 and a graphical representation of a full iteration of the algorithm is demonstrated in figure A.5.

The forward and backward 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. Finally, the CGA framework offers several algorithm optimisations such as for cases where the ‘end effectors’ are not positioned at the end of the chains (i.e. it is a leaf). For instance, assume

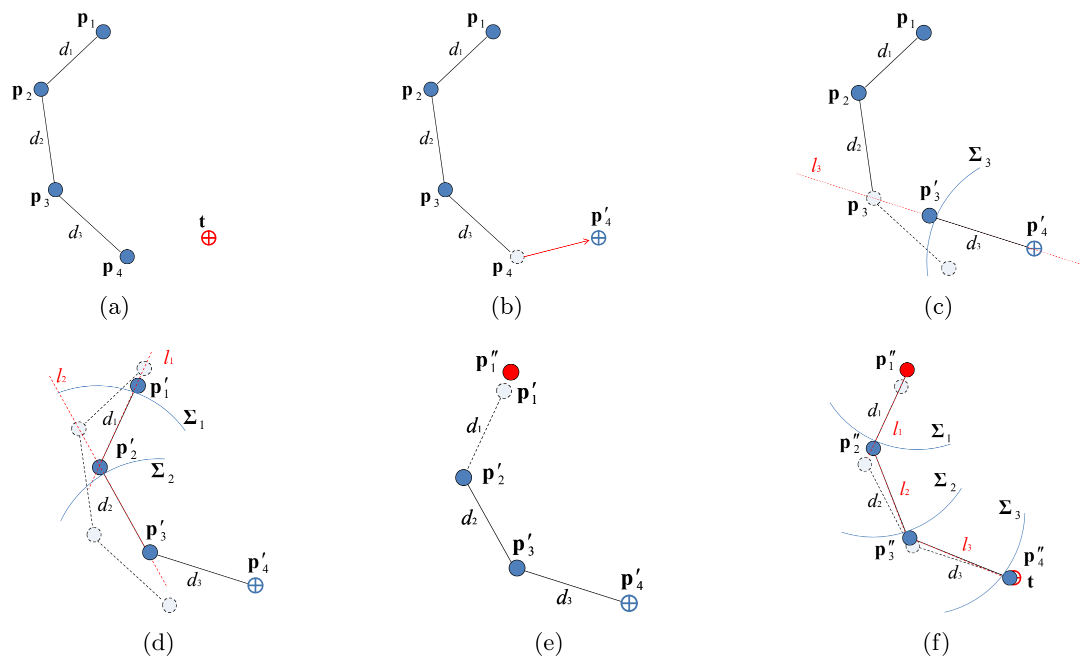


Figure A.5.: *A full iteration of FABRIK for the case of a single target and 4 joints using CGA. (a) The initial position of the manipulator and the target, (b) move the end effector $\mathbf{p}_4$ to the target, (c) find the joint $\mathbf{p}'_3$ which is the intersection of the sphere Σ_3 and the line l_3 which passes through the points $\mathbf{p}'_4$ and $\mathbf{p}_3$, (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.*

that the joint positions $\mathbf{p}_i$ and $\mathbf{p}_{i-2}$ are known, and that we want to estimate the joint position $\mathbf{p}_{i-1}$. This can be done by finding the intersection of the spheres Σ_1 and Σ_2 (as is described in section 2.2.6) with centres the known joint positions $\mathbf{p}_i$ and $\mathbf{p}_{i-2}$ and radii the distances $d_i = |\mathbf{p}_i - \mathbf{p}_{i-1}|$ and $d_{i-2} = |\mathbf{p}_{i-2} - \mathbf{p}_{i-1}|$, respectively. If the intersection is a circle, then the estimated joint position can be assigned as the nearest point on that circle from its previous position (as described in section A.1). If the intersection is a single point, the estimated joint position is assumed to be that point, otherwise, if the two spheres do not intersect, the estimated joint position is equal to $\mathbf{p}_{i-1} = \frac{\mathbf{p}_i + \mathbf{p}_{i-2}}{2}$. Another simple optimisation is the direct construction of a line pointing towards the target, when the latter is unreachable. In that case, each joint $\mathbf{p}_i$ is assigned to be the nearest point on the sphere, with centre the previous joint $\mathbf{p}_{i-1}$ and radii the distance d_{i-1}, from the target.

Glossary

BFGS	-	Broyden, Fletcher, Goldfarb, Shanno method
CCD	-	Cyclic Coordinate Descent
CDF	-	Cumulative Distribution Function
CGA	-	Conformal Geometric Algebra
CoR	-	Centre of Rotation
CRBM	-	Conditional Restricted Boltzmann Machine
CV	-	Constant Velocity
DLS	-	Damped Least Squares
DoF	-	Degrees of Freedom
EKF	-	Extended Kalman Filter
FABRIK	-	Forward And Backward Reaching Inverse Kinematics
FIK	-	Feedback Inverse Kinematics
FK	-	Forward Kinematics
FTL	-	Follow The Leader
GA	-	Geometric Algebra
GPDM	-	Gaussian process dynamical model
GPLVM	-	Gaussian Process Latent Variable Model
HCI	-	Human Computer Interaction
HMM	-	Hidden Markov Model
IIK	-	Inductive Inverse Kinematics
IK	-	Inverse Kinematics
KF	-	Kalman Filter
LDS	-	Linear Dynamical System
MMSE	-	Minimum Mean Square Error
Mocap	-	Motion Capture
NBP	-	Nonparametric Belief Propagation
NCT	-	Nearly Constant Turn
PCA	-	Principal Component Analysis
SDLS	-	Selectively Damped Least Squares

SIK	-	Sequential Inverse Kinematics
SMCM	-	Sequential Monte Carlo Methods
SVD	-	Singular Value Decomposition
VTM	-	Variable Turn Model
UKF	-	Unscented Kalman Filter
UPM	-	Uniform Posture Map

Supplementary Materials

A DVD containing supplementary materials accompanies this thesis. The DVD includes videos demonstrating our algorithms and comparing their performance against other state of the art methods. In addition, the Kine application (see section 3.6.1) is included, offering the opportunity to interact and evaluate the results obtained in this work.

The supplementary materials included in the DVD are:

1. **FABRIK.avi** A video that demonstrates FABRIK and compares its performance against other methods and under different scenarios.
2. **MarkerPrediction.avi** A video presenting experimental examples of our marker prediction and CoR methodology, with and without incorporating FABRIK for length constraints. It also compares its performance against other methods, such as a simple EKF model, interpolation and extrapolation.
3. **HandPoseTracker.avi** A video showing examples of our hand pose tracker under different frame rates.
4. **The Kine Application** Kine is a 2D real-time gaming application which offers to the user the opportunity to interact and evaluate the performance of the FABRIK algorithm against CCD. There is also an option where the user clicks and drags on the screen, and the snake (the kinematic chain is drawn as a snake) will track the mouse.

Bibliography

[AA82] S. J. Asseo and R. J. Ardila. Sensor independent target state estimator design and evaluation. In *Proceedings of the National Aerospace and Electronics Conference (NAECON)*, pages 916–924, 1982. 91

[ACL08] 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*, volume 5098, pages 238–247, Mallorca, Spain, 2008. LNCS. 7, 84, 96, 106

[AL09] Andreas Aristidou and Joan Lasenby. Inverse kinematics: A review of existing techniques and introduction of a new fast iterative solver. Technical Report F-INFENG/TR. 632, CUED, 2009. 104

[AL10a] Andreas Aristidou and Joan Lasenby. Inverse kinematics solutions using conformal geometric algebra. In *Proceedings of the 4th conference on Applied Geometric Algebras in Computer Science and Engineering (AGACSE'10)*, Amsterdam, The Netherlands, June 14-16 2010. 55, 57, 146

[AL10b] Andreas Aristidou and Joan Lasenby. Motion capture with constrained inverse kinematics for real-time hand tracking. In *Proceedings of the 4th International Symposium on Communications, Control and Signal Processing (ISCCSP 2010)*, pages 1–5, Limassol, Cyprus, March 3-5 2010. IEEE. 57, 75, 115, 121

[AL10c] Andreas Aristidou and Joan Lasenby. Real-time marker prediction and CoR estimation in optical motion capture. *Submitted to Image and Vision Computing*, 2010. 57, 75, 125

[AL10d] Andreas Aristidou and Joan Lasenby. FABRIK: a fast, iterative solver for the inverse kinematics problem. *Submitted to Graphical Models*, 2010. 53, 101, 104, 123

[ALC09] Andreas Aristidou, Joan Lasenby, and Jonathan Cameron. Methods for real-time restoration and estimation in optical motion capture. Technical Report F-INFENG/TR. 619, CUED, 2009. 96, 97, 99

[ARW01] 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

[AW00] Golam Ashraf and Kok Cheong Wong. Dynamic time warp based framespace interpolation for motion editing. In Sidney Fels and Pierre Poulin, editors, *Graphics Interface*, pages 45–52. Canadian Human-Computer Communications Society, 2000. 84

[Bac09] Adam S. Backer. Estimating Missing Motion Capture Data with Accelerometers. *Mphil thesis, Cambridge University Engineering Department.* Cambridge, UK, August 2009. 95

[Bai85] 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. 5, 45

[BB01] 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. 58

[BDMS84] 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. 5, 46

[BK05] Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, 2005. 5, 48, 65, 71

[Ble] Blender foundation: 3d content creation suite program. http://www.blender.org. 39, 65, 126

[BLM04] Joel Brown, Jean-Claude Latombe, and Kevin Montgomery. Real-time knot-tying simulation. *The Visual Computer*, 20(2):165–179, 2004. 6

[Blo02] Jonathan Blow. Inverse kinematics with quaternion joint limits. *Game Developer*, April 2002. 58, 62

[BP07] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transaction on Graphics (TOG)*, 26(3):72, 2007. 126

[BPW93] N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, New York, Oxford, 1993. 40, 41

[BSR07] Jurgen Broeren, Katharina S. Sunnerhagen, and Martin Rydmark. A kinematic analysis of a haptic handheld stylus in a virtual environment: a study in healthy subjects. *Journal of NeuroEngineering and Rehabilitation*, 4:13, May 2007. 2, 80

[BVU$^+$06] 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. 5, 53

[CA08] Nicolas Courty and Elise Arnaud. Inverse kinematics using sequential monte carlo methods. In *Proceedings of the V Conference on Articulated Motion and Deformable Objects*, volume 5098, pages 1–10, Mallorca, Spain, 2008. LNCS. 5, 50

[Cam07] Jonathan Cameron. *Aspects of Geometric Algebra with applications in motion capture.* PhD thesis, Cambridge University Engineering Department, Cambridge, UK, 2007. 26

[CD03] 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. 3, 5, 38, 51

[CDML$^+$07] P. Cerveri, E. De Momi, N. Lopomo, G. Baud-Bovy, R. M. L. Barros, and G. Ferrigno. Finger kinematic modeling and real-time hand motion estimation. *Annals of Biomedical Engineering*, 35(11):1989–2002, November 2007. 115

[CFAT07] Weiying Chen, Ryuji Fujiki, Daisaku Arita, and Rin-ichiro Taniguchi. Real-time 3d hand shape estimation based on image feature analysis and inverse kinematics. In *Proceedings of the 14th International Conference on Image Analysis and Processing (ICIAP '07)*, pages 247–252, Washington, DC, USA, 2007. IEEE Computer Society. 115

[CH05] Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics (TOG)*, 24(3):686–696, 2005. 8, 85

[CKM97] Kwan W. Chin, B. R. von Konsky, and A. Marriott. Closed-form and generalized inverse kinematics solutions for the analysis of human motion. In *Proceedings of the 19th IEEE International Conference of Engineering in Medicine and Biology society*, volume 5, pages 1911–1914, 1997. 49

[CL05] 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. xii, 6, 84, 88, 89, 90

[Cod] CodaMotion: Optical motion capture systems. `http://www.codamotion.com`. 80

[CP07] Lillian Y. Chang and Nancy Pollard. Constrained least-squares optimization for robust estimation of center of rotation. *Journal of Biomechanics*, 40(1):1392 – 1400, 2007. 86

[Cra89] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., 1989. 40, 41

[DDFG01] Arnaud Doucet, Nando De Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. Springer, New York, 2001. 96

[DDHF06] Guillaume Dewaele, Frédéric Devernay, Radu P. Horaud, and Florence Forbes. The alignment between 3-d data and articulated shapes with bending surfaces. In *Proceedings of the 9th European Conference on Computer Vision, Graz, Austria*, volume III of *LNCS*, pages 578–591. Springer, May 2006. 115

[DH93] Andreas W. M. Dress and Timothy F. Havel. Distance geometry and geometric algebra. *Foundations of Physics*, 23(10):1357 – 1374, October 1993. 21

[DL03] Chris Doran and Anthony Lasenby. *Geometric Algebra for Physicists*. Cambridge University Press, Cambridge UK, 2003. 11, 55

[DMR06] Pushkar Dhawale, Masood Masoodian, and Bill Rogers. Bare-hand 3d gesture input to interactive systems. In *Proceedings of the 7th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction (CHINZ '06)*, pages 25–32, New York, NY, USA, 2006. ACM. 115

[Dor05] Leo Dorst. First order error propagation of the Procrustes method for 3d attitude estimation. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 27(2):221–229, February 2005. 83

[DSP06] Kevin G. Der, Robert W. Sumner, and Jovan Popović. Inverse kinematics for reduced deformable models. In *ACM SIGGRAPH Papers*, pages 1174–1179, New York, NY, USA, 2006. ACM. 6, 50

[DU03] Klaus Dorfmüller-Ulhaas. Robust optical user motion tracking using a Kalman filter. Technical Report TR-2003-6, Institut fuer Informatik, Universitatsstr. 2, 86159 Augsburg, May 2003. 7, 84

[Enc] Microsoft Encarta Online Encyclopedia. Types of joints in the human body. ©1997-2008 Microsoft Corporation, `http://encarta.msn.com`. 41

[ETDH06] Rainald M. Ehrig, William R. Taylor, Georg N. Duda, and Markus O. Heller. A survey of formal methods for determining the centre of rotation of ball joints. *Journal of Biomechanics*, 39(15):2798–2809, 2006. 6, 84

[Fê03] 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. 49

[FAT05] Ryuji Fujiki, Daisaku Arita, and Rin-ichiro Taniguchi. Real-time 3d hand shape estimation based on inverse kinematics and physical constraints. In *Proceedings of the 13th International Conference on Image Analysis and Processing (ICIAP '05)*, volume 3617 of *Lecture Notes in Computer Science*, pages 850–858. Springer, 2005. 115

[Fle87] R. Fletcher. *Practical methods of optimization; (2nd Ed.)*. Wiley-Interscience, New York, NY, USA, 1987. 5, 49

[FRF08] Jonas Fredriksson, Sven Berg Ryen, and Morten Fjeld. Real-time 3d hand-computer interaction: optimization and complexity reduction. In *Proceedings of the 5th Nordic conference on Human-computer interaction*, pages 133–141, New York, NY, USA, 2008. ACM. 8, 114

[GL02] Sahan S. Hiniduma Udugama Gamage and Joan Lasenby. New least squares solutions for estimating the average centre of rotation and the axis of rotation. *Journal of Biomechanics*, 35(1):87–93, January 2002. 6, 83, 84

[GMHP04] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *ACM Transactions on Graphics (TOG)*, pages 522–531, New York, NY, USA, August 2004. ACM. 5, 7, 50, 85

[Gol] Motion reality golf systems. `http://www.motionrealitygolf.com`. 2, 80

[GPF08] Martin de La Gorce, Nikos Paragios, and David J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008. 8, 115

[Gra62] Hermann Grassmann. *Ausdehnungslehre*. Enslin, 1862. 12

[Hal03] K. Halvorsen. Bias compensated least squares estimate of the center of rotation. *Journal of Biomechanics*, 36:999–1008(10), July 2003. 6, 83

[HDSB05] Maureen K. Holden, Thomas A. Dyar, Lee Schwamm, and Emilio Bizzi. Virtual-environment-based telerehabilitation in patients with stroke. *Presence: Teleoper. Virtual Environ.*, 14(2):214–233, 2005. 2, 80

[Hes01] D. Hestenes. Old wine in new bottles: a new algebraic framework for computational geometry. In *Geometric Algebra with Applications in Science and Engineering*, pages 1–16. Birkhauser, 2001. 21

[HFP+00] Lorna Herda, Pascal Fua, Ralf Plänkers, Ronan Boulic, and Daniel Thalmann. Skeleton-based motion capture for robust reconstruction of human motion. In *Proceedings of the IEEE Computer Animation (CA'00)*, pages 77–86, Pennsylvania, USA, May 3-5 2000. 7, 85

[HFP+01] Lorna Herda, Pascal Fua, Ralf Plänkers, Ronan Boulic, and Daniel Thalmann. Using skeleton-based tracking to increase the reliability of optical motion capture. *Human Movement Science Journal*, 20(3):313–341, 2001. 7, 85

[HGP04] Eugene Hsu, Sommer Gentry, and Jovan Popović. Example-based control of human motion. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '04)*, pages 69–77, Grenoble, France, 2004. Eurographics Association. 86

[HLL99] K. Halvorsen, M. Lesser, and A. Lundberg. A new method for estimating the axis of rotation and the center of rotation. *Journal of Biomechanics*, 32:1221–1227, 1999. 6, 83

[HNT$^+$06] Junichi Hashiguchi, Hiroki Nivomiya, Hiroshi Tanaka, Mari Nakamura, and Katsuya Nobuhara. Biomechanical analysis of a golf swing using motion capture system. In *Proceedings of Annual Meeting of Japanese Society for Orthopaedic Biomechanics*, volume 27, pages 325–330, 2006. 2, 80

[Hol91] S. Holzreiter. Calculation of the instantaneous centre of rotation for a rigid body. *Journal of Biomechanics*, 24(7):643–647, 1991. 6, 83

[Hor87] Berthold Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4:629–642, April 1987. 86, 120

[HRE$^+$08] 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. 5, 50

[HS84] D. Hestenes and G. Sobczyk. *Clifford Algebra to Geometric Calculus: A unified language for mathematics and physics.* D. Reidel, 1984. 19, 20, 21, 31, 55

[HSD05] Alexander Hornung and Sandip Sar-Dessai. Self-calibrating optical motion tracking for articulated bodies. In *Proceedings of the IEEE Conference on Virtual Reality, VR '05*, pages 75–82, Washington, DC, USA, 2005. IEEE Computer Society. 7, 85

[HUHF03] 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. 41

[IP90] Augustus A. White III and Manohar M. Panjabi. *Clinical biomechanics of the spine.* J.B. Lippincott Company, Second Edition, 1990. 41

[IWZL09] Satoru Ishigaki, Timothy White, Victor B. Zordan, and C. Karen Liu. Performance-based control interface for character animation. *ACM Transaction on Graphics (TOG)*, 28(3):1–8, 2009. 100

[Jaz70] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, April 1970. 93

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999. 81

[JU97] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Proceedings of the International Symposium on Aerospace/Defense Sensing, Simululation and Controls*, volume Acquisition, Tracking and Pointing XI, pages 182–193, Orlando, Florida, USA, 1997. 94

[Kal60] Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, pages 35–45, 1960. 93

[KL07] 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. 9, 41, 71, 115

[KL09] Paris Kaimakis and Joan Lasenby. Physiological modelling for improved reliability in silhouette-driven gradient-based hand tracking. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19–26, Miami, FL, USA, June 25 2009. 9, 115, 121, 123, 124, 125, 130

[KLC$^+$03] 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, pages 305–313, St. Petersburg, Russia, June 2-4 2003. 52

[KM05] Richard Kulpa and Franck Multon. Fast inverse kinematics and kinetics solver for human-like figures. In *International Conference on Humanoid Robots, IEEE-RAS*, pages 38–43, December 2005. 5, 52, 57, 66

[KOF05] Adam G. Kirk, James F. O'Brien, and David A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 782–788, June 2005. 84

[Kor85] James Urey Korein. *A geometric investigation of reach.* MIT Press, Cambridge, MA, USA, 1985. 41, 58

[KTL07] 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. 41

[Lan98] Jeff Lander. Making kine more flexible. *Game Developer*, 5(3):15–22, 1998. 5, 51, 69

[LFLD98] Joan Lasenby, William J. Fitzgerald, Anthony N. Lasenby, and Chris J. L. Doran. New geometric methods for computer vision: An application to structure and motion estimation. *International Journal of Computer Vision*, 26(3):191–213, 1998. 120

[LG98] Ming C. Lin and Stefan Gottschalk. Collision detection between geometric models: A survey. In *Proceedings of IMA Conference on Mathematics of Surfaces*, pages 37–56, 1998. 63, 105, 130

[LGPW98] ChanSu Lee, SangWon Ghyme, ChanJong Park, and KwangYun Wohn. The control of avatar motion using hand gesture. In *Proceedings of the ACM symposium on Virtual reality software and technology (VRST '98)*, pages 59–65, Taipei, Taiwan, 1998. ACM. 115

[LH98] Cheng-Chang Lien and Chung-Lin Huang. Model-based articulated hand motion tracking for gesture recognition. *Image and Vision Computing*, 16(2):121–134, 1998. 8, 114

[Lie77] A. Liegeois. Automatic supervisory control of the configuration and behavior of multi-body mechanisms. *IEEE Transactions on Systems, Man and Cybernetics*, 7(12):868–871, December 1977. 45

[LJ03] X. Rong Li and Vesselin P. Jilkov. Survey of maneuvering target tracking. Part I: Dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1333–1364, October 2003. 90, 91

[LLW04] Anthony N. Lasenby, Joan Lasenby, and Richard Wareham. A covariant approach to geometry using geometric algebra. Technical Report F-INFENG/TR-483, Cambridge University Engineering Department, 2004. 26, 28, 30, 31, 34, 139

[LM06] Guodong Liu and Leonard McMillan. Estimation of missing markers in human motion capture. *The Visual Computer*, 22(9-11):721–728, September 2006. 8, 86

[LMPF09] Lei Li, James McCann, Nancy S. Pollard, and Christos Faloutsos. Dynammo: mining and summarization of coevolving sequences with missing values. In *Proceedings of the 15th International Conference on Knowledge Discovery and Data mining*, pages 507–516, Paris, France, 2009. ACM. 85

[LMPF10] Lei Li, James McCann, Nancy S. Pollard, and Christos Faloutsos. Bolero: A principled technique for including bone length constraints in motion capture occlusion filling. In *Proceedings of the ACM symposium on Computer animation*, Madrid, Spain, 2010. 85

[LZWM06] Guodong Liu, Jingdan Zhang, Wei Wang, and Leonard McMillan. Human motion estimation from a reduced marker set. In *I3D '06: Proceedings of the Symposium on Interactive 3D graphics and games*, pages 35–42, New York, NY, USA, 2006. ACM. 8, 86

[MAT] The Mathworks - MATLAB and Simulink for technical computing. `http://www.mathworks.com`. 65, 126

[MB91] 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. 41

[MCM07] 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. 52

[MD04] Damian Merrick and Tim Dwyer. Skeletal animation for the exploration of graphs. In *Australasian Symposium on Information Visualisation, (invis.au'04)*, volume 35, pages 61–70, Christchurch, New Zealand, 2004. ACS. 52

[MDFW00] Rudolph V. D. Merwe, Arnaud Doucet, Nando D. Freitas, and Eric Wan. The unscented particle filter. Technical Report F-INFENG/TR. 380, CUED, 2000. 94

[Men99] Alberto Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. 2, 80

[MK85] 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. 45

[MM05] 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

[MSZ94] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1994. 41

[MT00] W. Maurel and D. Thalmann. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics*, 24(2):203–18, 2000. 41

[MTHC03] Ivana Mikić, Mohan Trivedi, Edward Hunter, and Pamela Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–223, 2003. 115

[Muk09] R. Mukundan. A robust inverse kinematics algorithm for animating a joint chain. *International Journal of Computer Applications in Technology*, 34(4):303–308, 2009. 53

[Neb99a] Jean-Christophe Nebel. Keyframe animation of articulated figures using autocollision-free interpolation. In *Proceedings of the 17th Eurographics UK Conference*, 13-15 April 1999. 84

[Neb99b] Jean-Christophe Nebel. Keyframe interpolation with self-collision avoidance. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation*, pages 77–86. Springer, September 1999. 84

[NH86] 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. 5, 47

[OBBH00] James F. O'Brien, Robert E. Bodenheimer, Gabriel J. Brostow, and Jessica K. Hodgins. Automatic joint parameter estimation from magnetic motion capture data. In *Proceedings of Graphic Interface*, pages 53–60, 2000. 6, 83

[OS84] 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. 44

[Pec08] Alexandre N. Pechev. Inverse kinematics without matrix invertion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '08)*, pages 2005–2012, Pasadena, CA, USA, May 19-23 2008. 5, 49

[PH06] Sang Il Park and Jessica K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Transaction on Graphics (TOG)*, 25(3):881–889, 2006. 86

[Pha] PhaseSpace inc: Optical motion capture systems. http://www.phasespace.com. 4, 39, 80, 81, 86, 105, 114, 116, 126, 134

[PLH$^+$09] Tommaso Piazza, Johan Lundström, Alexander Hugestrand, Andreas Kunz, and Morten Fjeld. Towards solving the missing marker problem in realtime motion capture. In *Proceedings of the International Design Engineering Technical Conference*, 2009. 7, 84, 106

[PTP$^+$05] Lamberto Piron, Paolo Tonin, Francesco Piccione, Vincenzo Iaia, Elena Trivello, and Mauro Dam. Virtual environment training therapy for arm motor rehabilitation. *Teleoperators and Virtual Environments*, 14(6):732–740, 2005. 2, 80

[PY06] Jun Park and Yeo-Lip Yoon. LED-glove based interactions in multi-modal displays for teleconferencing. In *Proceedings of the 16th International Conference on Artificial Reality and Telexistence (ICAT '06)*, pages 395–399, Washington, DC, USA, 2006. 115

[RCB98] Charles Rose, Michael Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998. 7, 84

[RG91] Hans Rijpkema and Michael Girard. Computer animation of knowledge-based human grasping. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 339–348, New York, NY, USA, 1991. ACM. 41

[RK94] James Rehg and Takeo Kanade. Digiteyes: Vision-based hand tracking for human-computer interaction. In *In Proceedings of the workshop on Motion of Non-Rigid and Articulated Bodies*, pages 16–22. IEEE Computer Society Press, 1994. 115

[RL02] Maurice Ringer and Joan Lasenby. A procedure for automatically estimating model parameters in optical motion capture. In *Proceedings of the British Machine Vision Conference*, pages 747–756, 2002. 7, 85

[RM06] Arjen Van Rhijn and Jurriaan D. Mulder. Optical tracking and automatic model estimation of composite interaction devices. *IEEE Virtual Reality Conference (VR'06)*, 00:135–142, 2006. 7, 84

[SB71] Robert A. Singer and Kenneth W. Behnke. Real-time tracking filter evaluation and selection for tactical applications. *IEEE Transactions on Aerospace and Electronic Systems*, AES-7(1):100–110, January 1971. 90

[Sin70] Robert A. Singer. Estimating optimal tracking filter performance for manned maneuvering targets. *IEEE Transactions on Aerospace and Electronic Systems*, AES-6(4):473–483, July 1970. 90

[SK07] Markus Schlattman and Reinhard Klein. Simultaneous 4 gestures 6 dof real-time two-hand tracking without any markers. In *Proceedings of the ACM symposium on Virtual reality software and technology (VRST '07)*, pages 39–42, California, USA, 2007. 115

[SMC01] Björn Stenger, Paulo R. S. Mendonça, and Roberto Cipolla. Model-based 3d tracking of an articulated hand. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 310–315, Los Alamitos, CA, USA, 2001. IEEE Computer Society. 8, 115

[SMFW04] Erik B. Sudderth, Michael I. M, William T. Freeman, and Alan S. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In *In Neural Information Processing Systems (NIPS)*, pages 1369–1376. MIT Press, 2004. 115

[SPB$^+$98] Marius-Calin Silaghi, Ralf Plänkers, Ronan Boulic, Pascal Fua, and Daniel Thalmann. Local and global skeleton fitting techniques for optical motion capture. In *Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, pages 26–40, London, UK, 1998. 6, 83, 84

[SPCM97] Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. Anatomy-based modeling of the human musculature. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 163–172, 1997. 41

[Sto91] Jorge Stolfi. *Oriented projective geometry: a framework for geometric computations*. Academic Press, Boston, 1991. 30

[STTC06] Björn Stenger, Arasanathan Thayananthan, Philip H.S. Torr, and Roberto Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384, Sept. 2006. 8, 115

[STW07] Caifeng Shan, Tieniu Tan, and Yucheng Wei. Real-time hand tracking using a mean shift embedded particle filter. *Pattern Recognition*, 40(7):1958–1970, 2007. 115

[SZGP05] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *ACM Transactions on Graphics (TOG)*, 24(3):488–495, 2005. 6, 50

[THR07] Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems*, pages 1345–1352, Cambridge, MA, 2007. MIT Press. 86

[TK05] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Transactions on Graphics (TOG)*, 24(1):98–117, 2005. 7, 84

[UPBS08] Luis Unzueta, Manuel Peinado, Ronan Boulic, and Ángel Suescun. Full-body performance animation with sequential inverse kinematics. *Graphical Models*, 70(5):87–104, 2008. 5, 53, 57

[Vic] Vicon motion system and peak performance. `http://www.vicon.com`. 80

[Wam86] 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. 5, 47

[WBV+99] Greg Welch, Gary Bishop, Leandra Vicci, Stephen Brumback, Kurtis Keller, and D'nardo Colucci. The HiBall tracker: High-performance wide-area tracking for virtual and augmented environments. In *Virtual Reality Software and Technology, VRST, ACM*, pages 1–10, December 20-22 1999. 7, 84

[WC91] 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. 5, 50

[WE84] 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, 46

[Wel93] Chris Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. *Master Dissertation, Simon Fraser University*, 1993. 5, 49, 50

[WFH08] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008. 85

[WG01] Jane Wilhelms and Allen Van Gelder. Fast and easy reach-cone joint limits. *Journal of Graphic Tools*, 6(2):27–41, 2001. 58, 62

[WH97] Douglas J. Wiley and James K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45, 1997. 7, 84

[WL08] Rich Wareham and Joan Lasenby. Bone glow: An improved method for the assignment of weights for mesh deformation. In *Proceedings of the V Conference on Articulated Motion and Deformable Objects*, volume 5098, pages 63–71, Mallorca, Spain, 2008. LNCS. 126

[WP09] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. In *ACM SIGGRAPH Papers*, pages 1–8, New York, NY, USA, 2009. ACM. 8, 114

[WV98] 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. 41

[WW91] Alan Watt and Mark Watt. *Advanced animation and rendering techniques.* Addison - Wisley, ACM Press, New York, NY, USA, 1991. 41

[YLD07] Qian Yu, Qing Li, , and Zhigang Deng. Online motion capture marker labeling for multiple interacting articulated targets. *Computer Graphics Forum (Proceedings of Eurographics 2007)*, 27(7):477–483, 2007. 8, 85

[ZB94] 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. 4, 49

[ZVDH03] Victor Brian Zordan and Nicholas C. Van Der Horst. Mapping optical motion capture data to skeletal motion using a physical model. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03)*, pages 245–250, San Diego, California, 2003. Eurographics Association. 85

Index