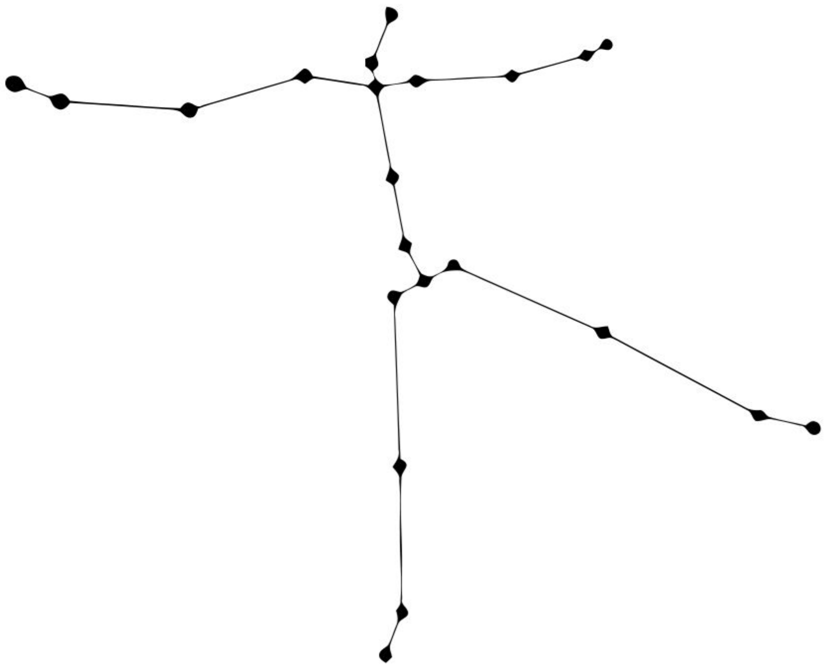


A Real-Time Adaptation of Inverse Kinematics for Motion Capture

Master of Science Thesis Computer Science Algorithms, Language and Logic

JOANNA SVANTESSON
JONAS BORNOLD

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2015

A Real-Time Adaptation of Inverse Kinematics for Motion Capture

JOANNA E. SVANTESSON,
JONAS M. BORNOLD,

Examiner: ULF ASSARSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
A skeleton constructed from motion captured data.

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Abstract

In this thesis, the process of animating characters from motion capture in real-time is presented. A model of a human skeleton is defined, and the joints of this skeleton are located using positions of markers placed on the body during motion capture. These methods for locating joints are presented and evaluated. During motion capture, markers might be occluded from cameras, resulting in joint positions that can not be determined. This problem is, in this project, solved by estimating the missing joint positions with the technique Inverse Kinematics, in a way that, to our knowledge, has not been used before. Four different inverse kinematics solvers were implemented in order to determine if one algorithm performs better than other. The implemented algorithms are the Jacobian Transpose, Damped Least Squares (DLS), Cyclic Coordinate Descent (CCD), and Forward and Backward Reaching Inverse Kinematics (FABRIK). The implemented system receives marker positions from the motion capture system in real-time and first performs joint localization followed by estimation of missing positions with IK. The result is a skeleton with a similar pose as the actor frame by frame. The application works well, but need improvements for smooth animation. The fastest and most accurate algorithm was CCD, but the other three are promising for possible future implementations.

Keywords: motion capture, inverse kinematics, mocap, ik, animation, real-time.

Acknowledgements

We want thank our supervisor Doctor Erik Sintorn for his insightful comments during the course of this work. We also want to thank Qualisys AB for the use of their motion capture system, and Doctor Vincent Fohanno for his biomechanical expertise.

Joanna Svantesson and Jonas Bornold, Gothenburg, June 2015

Contents

1

Introduction

Motion capture is the process of digitally recording motion and it has a variety of applications within areas like animation, biomechanics, sports, medicine and other motion analysis [1]. In biomechanics, motion capture can be used for analysis to, e.g., increase performance and prevent injuries during sports or improve treatment during rehabilitation [2], and it can be used in the industry for things like control systems, analysing wave motion under water, or optimising aerodynamics [3]. Motion capture is also a very popular technique for animating characters for films and video games [4]. This thesis addresses animation for humanoid characters, using optical motion capture where multiple cameras are used to capture the positions of reflective markers attached to an actor's body (as described in, e.g., the work by Guerra-Filho [5]).

Realistic motion can be hard to achieve since characters can be very complicated and can consist of many joints with different degrees of freedom (DoF). A common technique is to construct a skeleton with joints positioned corresponding to the actor's current pose, and map this to a virtual character [6, 7]. Depending on the purpose of the animation, the skeleton can be modeled in a more or less complex and anatomically correct way. Furthermore, there exist limits, or constraints, for how the joints can twist and rotate with respect to each other. Incorporating constraints in the skeleton model will lead to more realistic poses [4].

There is a number of techniques for localizing the joints of the skeleton to determine its pose [8, 9]. For medical or biomechanical analysis, it is important with as correct joint positions as possible [1], but when animating for games or virtual reality, corners can be cut and the resulting poses do not have to be correct, as long as the animation looks realistic. Joint localization is described further in Section 2.1.

Furthermore, the animation presented in this thesis shall be done in real-time. This means concretely that when an actor moves, a character should move in the same way on a screen. It should be experienced as if they move simultaneously even though a slight delay is inevitable. Because of the real-time aspect, the use of too heavy calculations, to get very precise results, will not be possible. Some techniques for joint localization also use the whole sequence of frames, and calculate skeleton poses retrospectively [10], which is not possible for a real-time implementation.

One purpose for a real-time system is to be able to analyse the motion data directly as a human actor performs the motion. It can take time to set up and calibrate the motion capture system, and therefore it is good to be able to see the animation in real-time to increase the chance of collecting the desired data during one session. Real-time evaluation of motion data also makes virtual reality and real-time game applications possible.

A problem that might occur in motion capture is marker occlusion; when a marker is not seen by enough cameras its position can not be determined, resulting in loss of joints in the skeleton [11]. There are numerous techniques for dealing with this problem and estimating joint positions that could not be located. A technique for dealing with joint estimation using inverse kinematics (IK) in a, to our knowledge, new way is presented in Section 3.6. IK is a method for determining the posture of a chain of joints given a goal [4], and is more described in Section 3.5.

1.1 Purpose

There exist many already available techniques and methods for animation from optical motion capture, inverse kinematics, and joint localization in real-time. This thesis aims to collect and increase the knowledge in the area and get a comprehensive picture of a real-time implementation. The whole process, from performing joint localization based on marker positions to animating a virtual human character, is addressed and a method for dealing with missing joint positions using IK is described. An investigation of different IK algorithms and their performance for this purpose is presented.

1.2 Problem

The process of mapping marker data to a character is not trivial, regardless of the purpose. A model of the skeleton needs to be defined, the joints need to be located from the motion capture data, and if the motion capture data is not complete, joints need to be estimated with a suitable method.

The poses of the character are defined by a model of a human skeleton, with a number of joints in specific locations. When animating for medical analysis or similar purposes, the skeleton might need to be modeled in an as physically correct way as possible. For purposes where the animation just needs to look realistic, a simpler model can suffice. The model might contain both actual joints, like the elbow or knee joints, and because of simplifications, virtual joints that are not actually a part of the human body. The model can also include rotation constraints for the skeleton joints to make sure that limbs can not bend in the wrong direction. The first step in animating a character from motion capture is to find a suitable model for the skeleton. The model we arrived at is described in Section 3.2.

The positions of the skeleton joints need to be located in 3D space to obtain the same pose of the skeleton as the actor has. Locating the joints from the positions of the markers placed on the body can be done in a variety of different ways depending on the marker set. Markers can be placed on limbs or close to the joints, and the number of markers can vary. For this thesis, one marker set with 35 markers, placed close to joints on anatomical landmarks of the body, has been used (see Appendix A). Methods to localize each joint position with the help of the available markers have been developed based on research in the field.

During motion capture, occlusion of markers (i.e., when not enough cameras are able to see the marker) is not uncommon and different ways of solving this problem exist (see Chapter 2). For this thesis, inverse kinematics is used to estimate the joint positions and orientations when the joints can not be properly located, as described further in Section 3.6. There exist several algorithms for solving the IK problem, and for this thesis four IK methods have been investigated in order to study this way of estimating joint positions, and to determine if some IK solvers are more suitable than others for this particular purpose.

Two obvious problems that arise with the task of animating from motion capture data in real-time, is accuracy and performance. The joint *localization* needs to be as accurate as possible in the sense that the joints in the skeleton model that are actual joints in the human body should not be too far from the actual location. When the joints are *estimated* using IK, they should be accurate in the sense of being in realistic positions, preferably not too far from the actual pose of the actor. Since the whole process will be done in real-time the algorithms used can not be too computationally intensive, but the accuracy and appearance must still be acceptable.

This thesis examines which methods and techniques that are best suited for animating from motion capture in real-time. The steps and adaptations necessary for motion capture data are described with the balance between correctness and performance in mind.

1.3 Limitations

Several methods for joint localization from motion capture exist, and different techniques have been studied for this thesis. However, joint localization depends on the marker set used during motion capture and due to time constraints only one method was implemented for one specific marker set. This marker set applies to human actors only, so the animation in this project can therefore only be done for humanoid characters. The marker set has few markers and, e.g., fingers are not tracked, so therefore the skeleton is modeled without such features.

Testing the accuracy of the joint localization from the marker positions would be desirable. One way would be to compare the joint positions in the skeleton model to the actor's actual joints, but there is no convenient method to measure these positions. Moreover, there are joints in the skeleton model that do not exist in a real human skeleton. One possibility would be to use a tool for biomechanical analysis, which uses the whole motion capture sequence and calculates very accurate joint positions. Those positions could then be compared against the joints in the model which has corresponding actual joints. Unfortunately, due to limited access of such tools, accuracy tests for the joint localization was omitted. Instead, the visual aspect was considered in the sense of the poses being realistic and comparable to the actor's actual poses.

The number of studied IK algorithms was limited by time and four algorithms were chosen for implementation. More advanced methods for animating from motion capture data, which might consider mass, forces, and torques exist [12, 13], but in this thesis only marker positions are used.

One common problem in animation is self-collisions, such as when an arm penetrates the body. But the target character for the skeleton used in this thesis is unknown, so no mesh or volume properties can be considered, and self-collisions are therefore not in the scope of this work.

Another animation problem is so called foot planting, which is when the soles of the feet are placed in level with the ground. This might be needed, e.g., when the character or its surroundings are different from the actor and the floor and the retargeting of the motion might result in the feet penetrating the ground or floating above it. Foot planting is not considered in this thesis. Furthermore, a character might be of different size and proportions and, during retargeting, the motion could be scaled and adapted in various ways. In this project only a simple approach of retargeting was implemented, since it alone is a big topic.

1.4 Method

To study the necessary steps for animating from motion capture data, an application was implemented as a middleware between Qualisys Track Manager (QTM) [14] and Unity [15]. QTM is an application developed by Qualisys AB used to process the data received from their motion capture system. QTM streams marker positions to the developed application, which then performs joint localization and inverse kinematics to determine the pose of a skeleton. The skeleton is then mapped to a character in Unity which is animated in real-time. The system was developed in C# using Visual Studio and is described in detail in Section 3.1.

One approach for joint localization and four inverse kinematics algorithms were implemented. Joint localization was tested and evaluated by visually inspecting and comparing the character's poses to the actor's actual poses. The IK algorithms were tested by using a skeleton built from a motion capture recording with no occluded markers and after joint localization, one or more joints were removed to simulate missing markers that resulted in unknown joint positions. The difference in positions between the original joint positions and the estimated joint positions was measured over time. Some error is expected, and the main goal is that the postures look realistic, and therefore the visual aspect needs to be considered when evaluating the IK solvers as well.

1.5 Outline of the report

In Chapter 2, papers in related areas are presented; work done in animation of characters from motion capture data, and previously used methods for joint localization and inverse kinematics algorithms are presented.

Chapter 3 explains the system implemented for the character animation. A description of the skeleton model is presented in Section 3.2, and an explanation of the constraints used for ensuring that joints are in realistic positions during the inverse kinematics phase follows in the next Section 3.3. The joint localization methods for each joint in the skeleton model are presented in Section 3.4. A more detailed explanation of the different inverse kinematics algorithms implemented and tested is presented in Section 3.5, and the chapter is concluded by Section 3.6 where a way of dealing with imperfect motion capture data using inverse kinematics is presented.

The following Chapter 4, presents the result of the tests measuring speed and accuracy of the different IK algorithms for solving joint gaps. Discussion, Conclusion and Future Work follows in Chapters 5 and 6.

2

Previous Work

In this chapter, work in related areas of motion capture animation, joint localization and inverse kinematics are presented. Joint localization is a big area and the first section presents different techniques both from animation and biomechanics. The second section deals with inverse kinematics algorithms, and the final section presents approaches using inverse kinematics when animating from motion capture data.

2.1 Joint Localization

Numerous techniques exist for localizing joint center positions with varying accuracy and computational time. Some techniques are specialized for specific joints and some are more suitable for real-time applications. In biomechanics the joint localization is usually divided into two different approaches: regression and functional methods [16].

Two common functional methods are coordinate transformation and sphere fitting. Coordinate transformation techniques (e.g., Cameron [17]), are suitable for real-time applications but need a minimum of three markers on each limb linked to the joint. Other functional methods are described by Stoddart [18], and by Ehrig et al. [16]. Both methods use a transformation between frames to locate center of rotation. Sphere fitting based techniques, were the center of the sphere is the joint, and the radii of spheres are optimised to fit the trajectories of marker positions [10, 19], usually need a complete motion capture recording when calculating joint center positions, and thus are not suitable for real-time applications.

Regression methods use anatomical landmarks to find the joint center and are therefore joint specific. For the knee joints, such methods include the ones presented by Gardner and Chief [20]; for the shoulder joints, methods include those of Campbell et al. [21], Lloyd et al. [22], Meskers et al. [23] and Sholukha et al. [24]; and methods for the hip joints are presented by Davis III et al. [25], Harrington et al. [26] and Bell et al. [27].

There are animation specific joint localization techniques that uses the whole motion capture data set and is therefore not suitable for real-time. Silaghi [8] compares global and local skeleton fitting, where global fitting considers building the whole skeleton at once, whereas the local approach fits one limb at a time, ignoring relations between remote limbs.

Real-time applicable methods, however, usually need markers to be positioned on the center of the limbs, and preferably not near the joints. Such is the method of Cameron and Lasenby [17], which takes advantage of the simplification that all markers on a body segment move roughly as a rigid body.

2.2 Inverse Kinematics

Solving an inverse kinematics problem is not limited to the algorithms used in this thesis. Several approaches and algorithms have been proposed and many have been developed for animating realistic human motion.

A commonly used group of IK algorithms is based on the inverse of the Jacobian matrix [28], as described in Section 3.5.1. This inverse, however, might not exist, and there are several methods for approximating the inverse for all possible matrices, including using the transpose [29] or the pseudoinverse [30]. Another approach is to use Damped Least Squares to calculate the Jacobian inverse [31]. Jacobian algorithms generally produce good results, but are computationally intensive and suffer from singularity problems (which occur when a matrix does not have an inverse) [7].

A very popular IK algorithm is Cyclic Coordinate Descent (CCD) [33], which is an iterative heuristic search technique, and it is significantly faster than Jacobian based methods [7]. Another heuristic and iterative method is Forward And Backward Reaching Inverse Kinematics (FABRIK) recently presented by Aristidou and Lasenby [4]. According to the authors, FABRIK is both faster and gives more natural looking solutions than CCD.

Newton methods are a family of IK solvers that uses the Hessian matrix and a first order approximation to equations that can be highly nonlinear [34]. These methods might converge slowly and are both computationally intensive and complicated [4].

Lately, several techniques using machine learning and probabilistic methods for solving the inverse kinematic problem have been presented. Such techniques include the one presented by Ong and Hilton [35] where a learning-based approach for solving the IK problem, specifically for animation, is used and a statistical model for capturing physical or kinematic constraints of articulated structures is presented. Experiments were done for bodies of both humans and dogs.

In the work of Grochow et al. [36] another approach for solving the IK problem using machine learning is presented. A scaled gaussian process latent variable model is used to produce the most probable poses given a training set with poses of a certain style. A big disadvantage of this method is that training data with realistic poses needs to be available.

Other probabilistic approaches are presented by Sapra et al. [37] and Hecker et al. [38]. In the work by Sapra et al. the IK problem is solved using a sampling importance re-sampling particle filter, which has no problems with singularities. It can find multiple solutions, but no method is proposed for choosing one of them. Hecker et al. also present a particle-based IK solver used for animating characters with unknown morphologies. The solver treats character skeletons as sets of 3 DoF particles with 1 DoF length constraints between the particles. It is specifically designed to support flexible tuning and ad hoc preconditions.

2.3 Inverse Kinematics in Motion Capture Animation

Inverse kinematics can be used in many different phases and areas when adapting motion capture data to a skeleton model. In the work of Unzueta et al. [39], IK is used in motion capture for full body reconstruction using only the pelvis and the end-effectors: head, ankles and wrists. IK algorithms are applied on different parts of the body in a specific order. Techniques for ball- and socket joint limits and self-collision avoidance are also presented respectively. This method gives visually acceptable results, but it does not necessarily achieve the same poses as the actor since only a few parts of the body are tracked.

Similarly, in the endeavor for realistic animation, Tolani et al. [34] use analytical methods and hybrid analytical and numerical methods to solve the IK problem for human-like arms and legs. The goal in this work is not to generate realistic postures but to generate all feasible solutions to choose from. Good results are presented compared to, e.g., Jacobian based methods.

A different approach in using IK in animation from motion capture data comes from Aristidou and Lasenby [40]. Markers are estimated using a variable turn model within an unscented Kalman filter and IK is used to correct the joint position estimates to keep bone length constant. Marker information from previous frames is used as well as 2d data when a marker's 3d position can not be determined, due to occlusion from too many cameras. The center of rotation of joints is calculated using both the known markers and the predicted ones. Good results are obtained even when markers are occluded for long time periods.

Skeletons can be of different proportions and different topology; Inverse kinematics can also be used for retargeting and adapting the motion from one skeleton to another, as described by, e.g., Monzani [41]. Multon et al [13] describe the use of IK in real-time for adaptation of a motion captured skeleton to its environment using spacetime constraints, mass position, and mechanical laws.

Also using IK in real-time for adapting motion-captured animation are Meredith and Maddock [42]. The focus in their work is on retargeting motion to characters of different size as well as to individualizing the motion. Characters can for example appear to move in a stiffer or bouncier way, or to be injured. A character can also maneuver uneven terrain not present during motion capture. A half-Jacobian based IK solver is used to easily embed weighting parameters to manipulate the joint angles in real-time. The results are computationally cheaper than, but not as realistic as, similar techniques using dynamics or biomechanical algorithms.

Fêdor [12] describes various constraint methods when using different inverse kinematics solvers in motion capture. Constraints in this regard can regulate many things, like rotation, angular velocity, or center of mass placement. Aristidou and Lasenby also propose a method for constraining the rotation of joints [4].The constraints used in our implementation are based on this method.

3

Analysis

THE following chapter provides an overview of the implemented system and the approaches, algorithms, and methods used when animating from motion capture data. Key concepts in the project such as the skeleton model developed, the methods used for finding joint positions, and the algorithms for missing joint-location estimation using IK are presented in detail. The theory and implementation of the IK solvers are presented in the later sections.

3.1 System Overview

The system implemented for this project uses Qualisys' motion capture system and the software Qualisys Track Manager (QTM) [14]. QTM processes the data from the cameras and performs automatic marker identification. It also has a real-time functionality which streams the marker positions together with labels frame by frame. The implemented system calculates positions of the joints in the skeleton for each frame to obtain the pose of a character.

The first task for the system when receiving a frame from QTM is to use the marker positions to calculate the position and orientation of each joint in the skeleton model. This is referred to as joint localization and is further described in Section 3.4. If not enough cameras are able to see a marker, that marker position will be missing, and joint positions may not be identifiable. Some methods solve this problem by estimating positions of the missing *markers* before the joint localization [40]. In this thesis the positions of the missing *joints* are estimated instead.

Each unknown joint position in the skeleton model is estimated using an inverse kinematics solver. Four different solvers have been implemented and tests have been performed to investigate which of them are most suitable. The joint positions from the previous frame are used when solving the gaps with IK in a way that is described in Section 3.6. To ensure that no unnatural poses occur, constraints for each bone are incorporated in the model and all bones with an estimated position are checked to be within these constraints during the IK stage.

When the skeleton is complete, it should correspond to the actor's current pose or, if markers were occluded, to a realistic pose close to the actor's. The skeleton is now ready to drive the animation and can be mapped to the character rig. This is called retargeting and might include scaling and translating the rotations of the skeleton to the character rig. Retargeting is outside the scope of this thesis, but a very basic method was implemented for characters in Unity. The rotation, and not the position, of each bone is mapped to the corresponding bone of the target character.

The following sections describe the skeleton model used for this project and present the theory and details of the methods used for joint localization and inverse kinematics.

3.2 Skeleton Model

When animating characters, the animation is often driven by a skeleton, or a character rig. Since the target character's properties is unknown in this project, a general skeleton is needed to define the poses obtained each frame from the motion capture system. This skeleton is then retargeted to the character rig. The skeleton model is, in this case, a simplified model of a human skeleton with defined limbs and joints. The developed model is designed with respect to the available marker positions (see Appendix A) and is similar to models that are commonly used within animation (e.g. the models presented by Aristidou and Lasenby [7], and Unzueta et al. [39]).

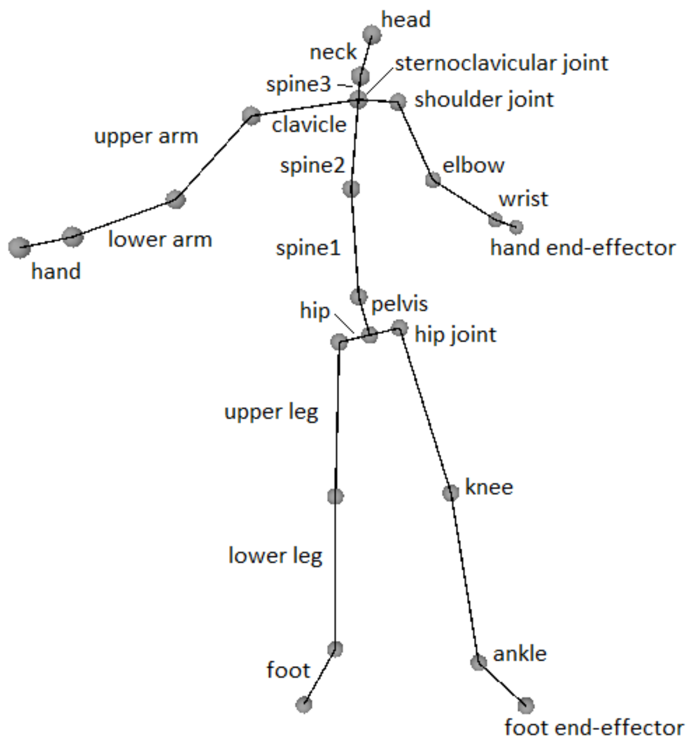


Figure 3.1: The skeleton model, with the notation for joints and bones.

The skeleton is modelled with the most obvious real human joints needed to achieve believable motion, like the elbows, wrists, hips, knees, and ankles. The shoulder joints (the joints between the clavicles and the upper arms) are also obvious, but are complicated in a real skeleton and their motion is also dependent on the corresponding shoulder blade [24], which is not modeled. In the model, the clavicles are simplified compared to a real skeleton as they have a shared starting point on the spine instead of unique starting points on the rib cage. The spine is modeled with two more joints in locations corresponding to marker placements on the back in order to use all information known about the pose of the spine.

The skeleton is represented as a tree hierarchy of bones, with the pelvis as root, and each node has a reference to the parent and can have any number of children. Each node contains a name, position, orientation, and constraints for how the bone is allowed to twist and rotate. The leaves in the hierarchy are called end-effectors, which in this case are the hands, feet, and head. Most bones have orientations that are defined as the bone being rotated towards its child. The head is a special case and its orientation is determined by the marker positions on the head. The end-effectors for the hands and feet do not have an orientation, since they just mark the end of the respective chain, and are thus actually defining the orientation of their parents. Between every bone in the tree is a joint, but joints are not modelled directly; the position of a bone is assumed to be the position of the joint linking the bone and its parent together. In other words, bones that are linked to the same parent share a joint and have the same position (e.g, the sternoclavicular joint is shared by spine3 and the clavicles).

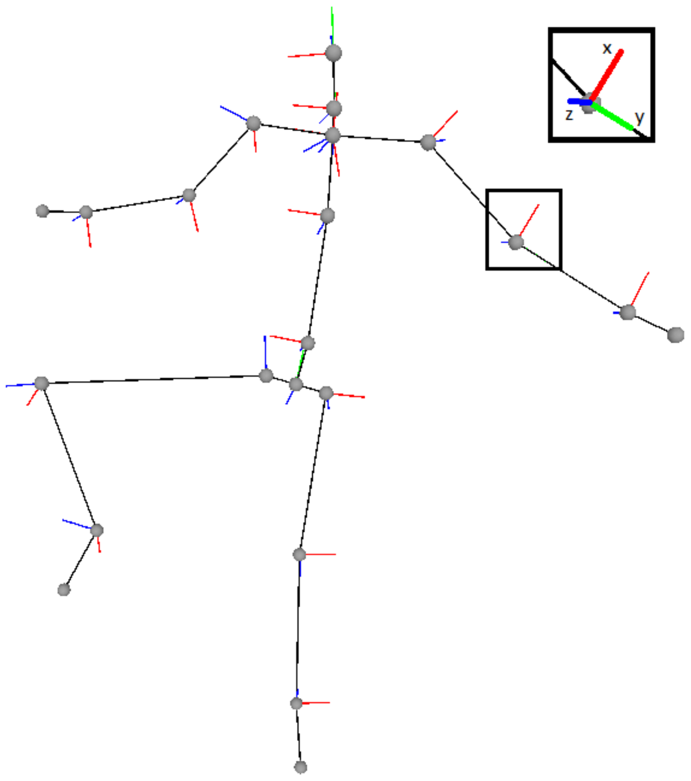


Figure 3.2: The skeleton model with rotations

In the model, world positions and world orientations are used, where the positions are represented as 3-dimensional vectors and the orientations as quaternions. Each bone has x-, y-, and z-axes, defining the orientation, as can be seen in Figure 3.2. The y-axis is along the bone, and the z-axis is towards the front of the bone. The front of the bone is defined as the front of the skeleton when standing upright, with its arms straight and parallel to the sides of the body, and the palms of the hands facing towards the thighs.

3.3 Constraints

Constraints, or joint limits, regulate the relative motion between the bones a joint connects. In a human skeleton, all joints have limits and different kinds of joints have different kinds of constraints. E.g., the elbow in a human skeleton behaves as a hinge joint and can only move with 1 DoF. In the model for this project all joints are treated as ball and socket joints with 3 DoF, even though this is not the case in a real skeleton. To treat all joints the same way is a simplification, but this is compensated by narrowing the constraints on joints with less degrees of freedom such that unrealistic motion is not allowed.

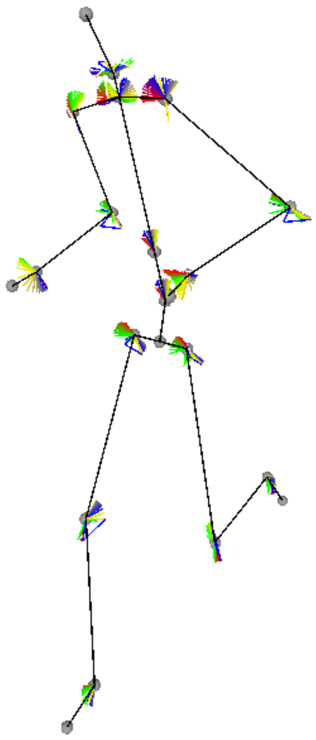


Figure 3.3: Rotational constraints on a skeleton.

Every joint, or bone in the skeleton model, has two kinds of constraints: orientational and rotational. Orientational constraints regulate how much a bone can twist around its direction axis and rotational constraints regulate how much a bone can rotate in all other directions. The rotational constraint can be visualized as an irregular cone defining the allowed volume for the bone with the constraint.

In the human body, the bones' freedom of movement can depend on the configuration of the direct child joint [39]. E.g., the upper leg is allowed more movement if the knee is bent. To simplify calculations, and since visually pleasing results are good enough, this is not taken into consideration for this thesis. Furthermore, to have visually pleasing, realistic results, the constraints might be more strictly defined than the limits that most humans actually have; even if a pose is possible, it is not necessarily probable. Also, one human might be more or less agile than another, which means that it might be desirable to let the constraints be user definable.

Constraints are checked during the IK phase to make sure that the poses obtained are legal. The IK algorithms are iterative and during each iteration, the bone under consideration is checked to be within the orientational and rotational constraints, and if it is not, it is rotated to an allowed pose. The following two sections describe in detail the orientational and rotational constraints respectively.

3.3.1 Orientational Constraints

The orientational constraint, or twist constraint, regulates how much a bone can twist around its own direction-axis (the y-axis). It is defined as an allowed range of angles a bone can be twisted with respect to its parent bone. An example is the lower arm, which can twist a certain amount with respect to the upper arm. The allowed range is between a start angle and an end angle on a circle which is looked at in the direction of the bone (see Figure 3.4). After each bone has been repositioned during the IK algorithm, a check is done to make sure that it stayed inside the constraints and if the bone is outside, it is rotated back inside.

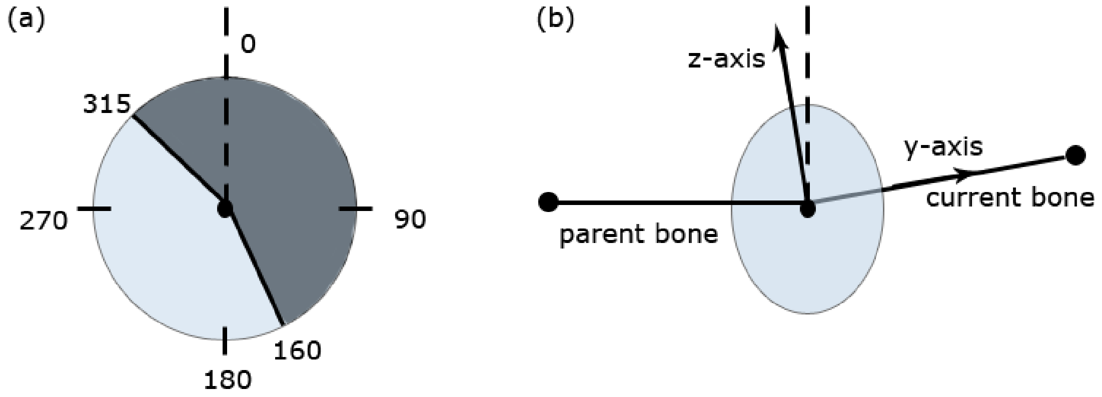


Figure 3.4: a) The twist angle in this example is allowed to be between 315° and 360° or between 0° and 160°. b) The circle is looked at in the direction of the bone.

The check consists of making sure that the bone under consideration is not twisted too much around its own direction vector, according to the bone's orientational constraint. First, the twist of the bone is calculated as the difference in rotation between the bone and its parent. Then the direction of the twist (right or left) is calculated in order to obtain the correct twist angle and decide if the twist is outside the allowed range or not.

The twist is calculated by comparing the z-axis of the bone under consideration and the z-axis of its parent, and calculate the angle between them. But, since the two bones can be rotated in any direction from each other, the two z-axes can not be compared directly; a reference vector for the parent bone needs to be found, which is the parent's z-axis rotated such that the parent bone is pointed in the same direction as the bone under consideration, but without twisting the bone around its direction-axis. The bone's z-axis and the reference vector are now located in the same plane (the plane with the current bone's y-axis as normal) and the angle between them defines the twist. The reference vector is always at 0° on the circle. The angle between the reference and the z-axis however does not tell us if the bone is twisted to the right or to the left of the parent. If it is twisted to the left the twist angle x would actually mean the angle $360 - x$ on the circle. Therefore an additional check needs to be done to determine if the twist angle denotes an angle to the left or the right and translate it accordingly to our circle (see Figure 3.6). This is easily checked by calculating the angle between the x-axis of the bone under consideration and the reference vector; if the angle is larger than 90° the bone is twisted to the left, and otherwise to the right.

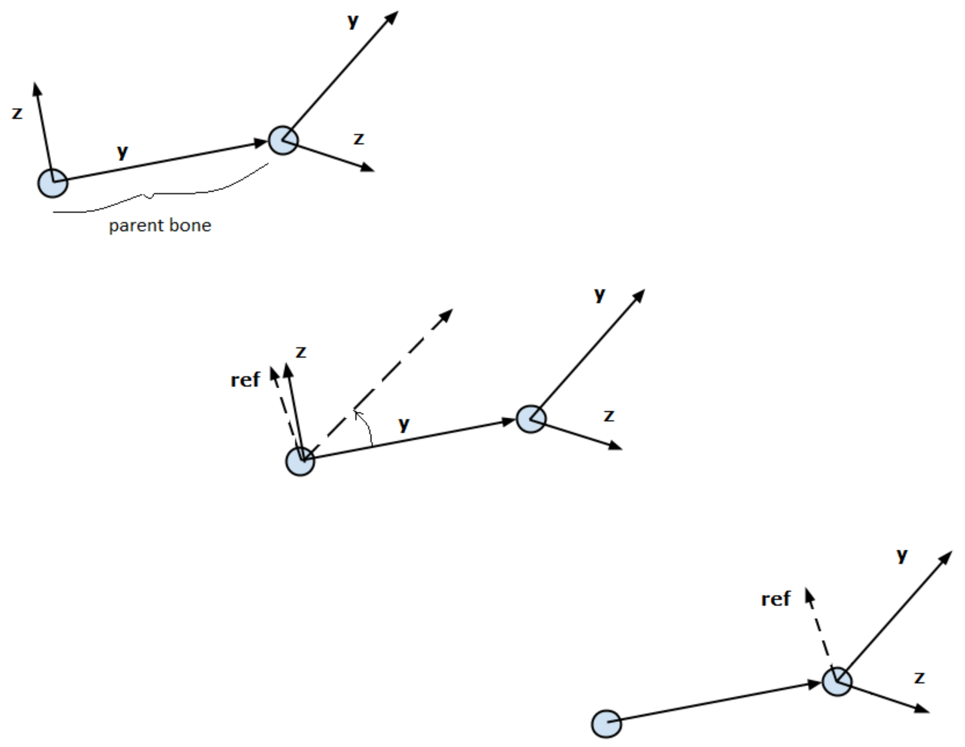


Figure 3.5: The reference vector is obtained by rotating the parent bone such that its y-axis points in the same direction as the child bone.

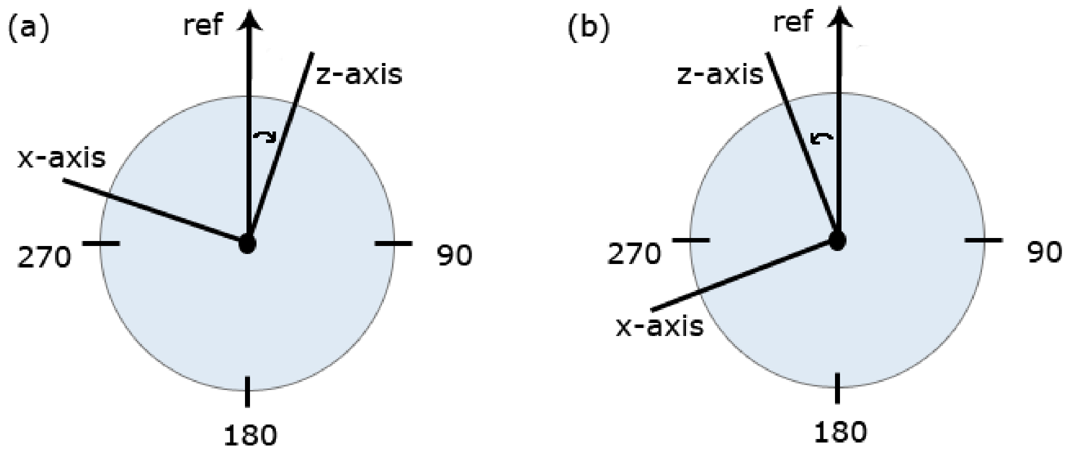


Figure 3.6: The z-axis is twisted 20° in both a) and b). If the z-axis is twisted to the right of the reference vector, the twist angle is 20°, as in a), and if the z-axis is twisted to the left of the reference vector the twist angle is 360° - 20° = 340°, as in b).

If the bone under consideration is outside the constraints it is twisted to either the start limit or the end limit, depending on whichever is closest.

3.3.2 Rotational Constraints

The rotational constraints regulate where in space a bone can be located with respect to the bone's parent, and ensure that, e.g., a knee does not bend in the wrong direction. The constraint is modeled as an irregular cone defining the bone's allowed volume of space (see Figure 3.7). If the IK-algorithm proposes a rotation outside the defined volume, it is deemed illegal and the bone needs to be rotated to the nearest legal rotation. The method described in this section is based on the work by Aristidou and Lasenby [4].

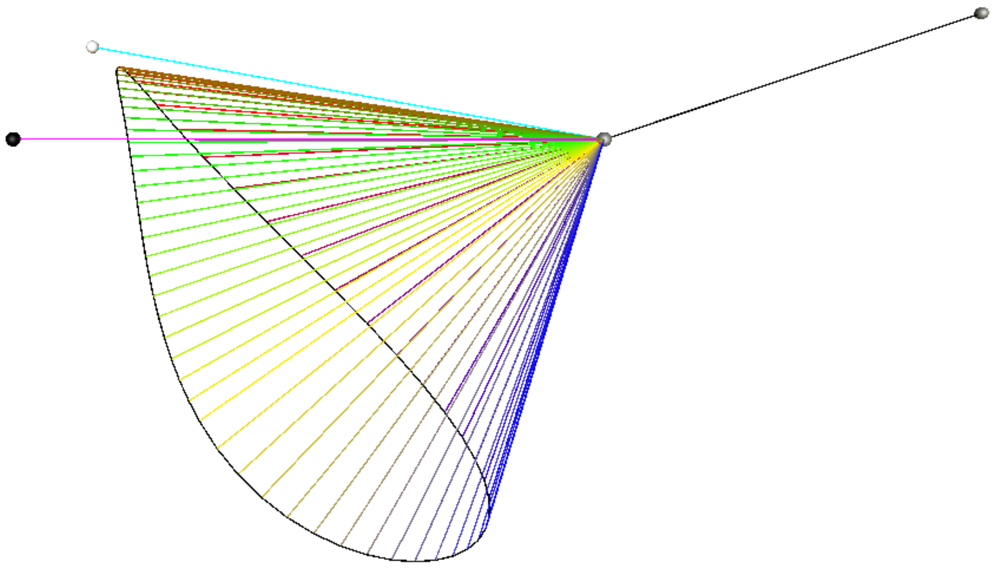


Figure 3.7: An irregular cone representing the area in which the bone can move. The cyan line represents the proposed rotation of the bone from the IK algorithm. The magenta line is the closest legal rotation.

An ordinary cone can be described by a radius of the base and the height of the cone. An irregular cone does not have a circular base and thus needs to be described by more than one radius. For an irregular cone with two radii, the base is an ellipse instead of a circle. In this implementation, four radii are used to define the irregular cone, where each quadrant of the cone's base is defined by two of the radii, creating a quarter of an ellipse (see Figure 3.8).

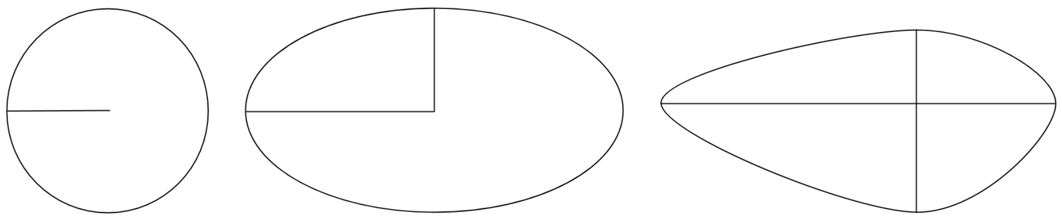


Figure 3.8: The base of a cone defined by one, two and four radii.

To decide if the bone under consideration has a legal rotation, the position of the connecting child joint is determined to be inside or outside the cone defined by the constraints. Performing this test in three dimensions is complicated, and therefore the problem is simplified into two dimensions. The two dimensional problem can be thought of as the cone being viewed from the base, seeing an irregular ellipse, and the connecting joint as a 2d-point. The bone then has a legal rotation if the 2d-point is within the irregular ellipse. See Figure 3.9.

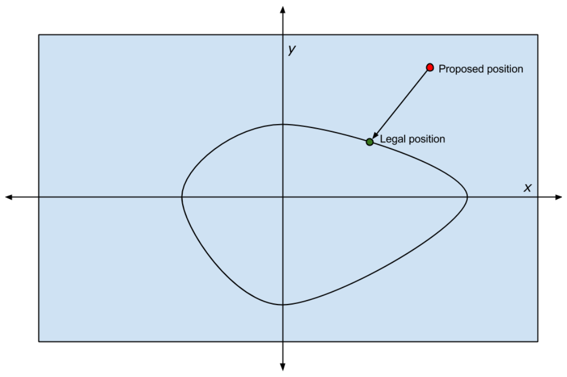


Figure 3.9: The three dimensional problem translated into two dimensions.

The cone is not defined by the radii but by four angles at the top of the cone, as can be seen in Figure 3.10. The radii defining the base of the cone can be calculated with trigonometry, using the associated angle θ and the height of the cone, as

$$radius = height \times \tan\theta.$$

To calculate the height of the cone, the vector between the current joint and the next bone's joint is projected onto the line defined by the direction of the parent bone (the *Origin Vector*). The length of the Origin Vector is the height of the cone. By translating the proposed position onto the plane with the Origin Vector as normal, the 3d-problem is now a 2d-problem. The proposed position of the next joint is a point on the plane, and the irregular cone is an irregular ellipse with its center in the origin.

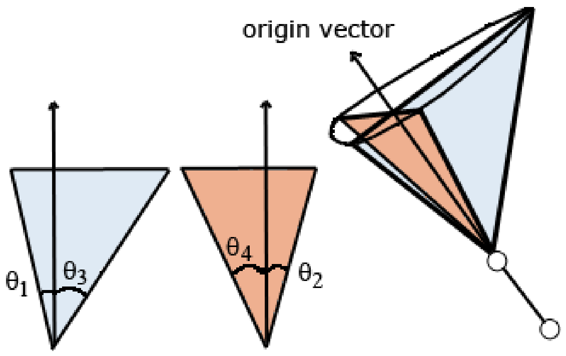


Figure 3.10: The cone is defined by four angles.

By locating which quadrant the 2d-point is in, the associated angles can form the ellipse of that quadrant. It is then checked whether the point is inside this ellipse or not. If the point is outside, the bone has an illegal orientation and the closest point to the ellipse is calculated using the method described by Eberly [43]. This new point is used to find the closest legal rotation for the bone (see Figure 3.9).

A special case is when the proposed rotation of a bone is at an angle larger than 90° from the Origin Vector. In this case the target rotation is referred to as being *behind* the parent bone. The proposed rotation is deemed illegal, and the closest point to the ellipse is located as normal, but when translating back to 3d space, the Origin Vector is negated, (see Figure 3.11).

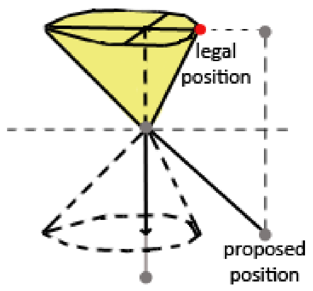


Figure 3.11: The bone is behind its parent.

The proposed method for constraints works well on cones with angles smaller than 90°. However, with one or more cone angles larger than or equal to 90° the ellipse is harder to define, but with joints being able to rotate in angles larger than 90° this is a necessity for the constraint model to handle.

The easiest case is when all four angels are over 90°. The method works in a similar fashion as before, but the rotation is illegal if the bone is *inside* the cone, as described by Figure 3.12

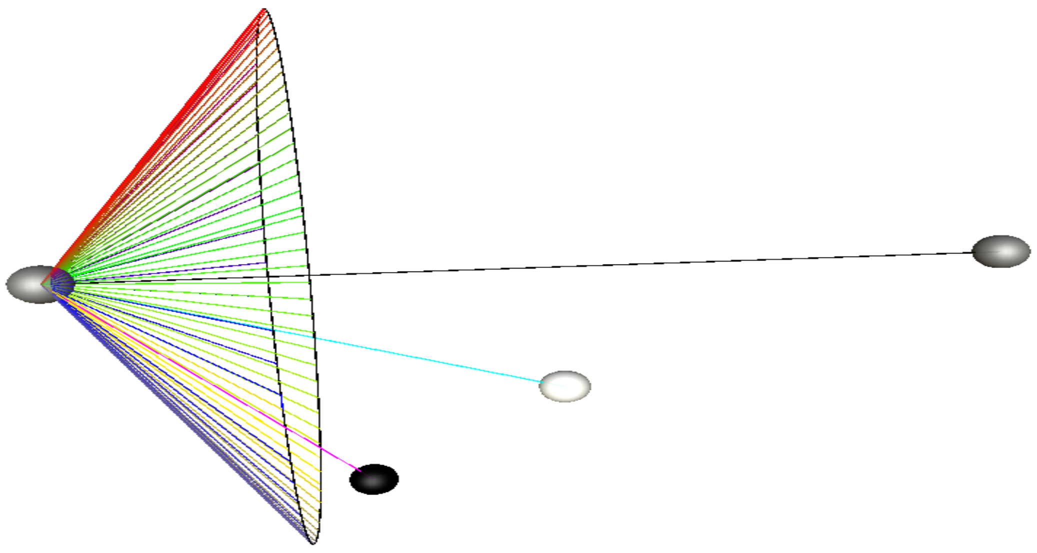


Figure 3.12: All angles of the cone are over 90°. The magenta bone is allowed everywhere but *inside* the cone.

A more complicated case is when *one* of the four angles defining the cone is over 90°. This is the case for hinge joints such as the elbow or the knee. The cone can still be constructed with the angles not associated with the quadrant where the angle over 90° defines the cone. If, however, the proposed rotation is in the other two quadrants, the cone can be constructed with the Origin Vector on a different axis than the direction axis defined by the parent bone. See Figure 3.13 for examples of how the origin vector can be located.

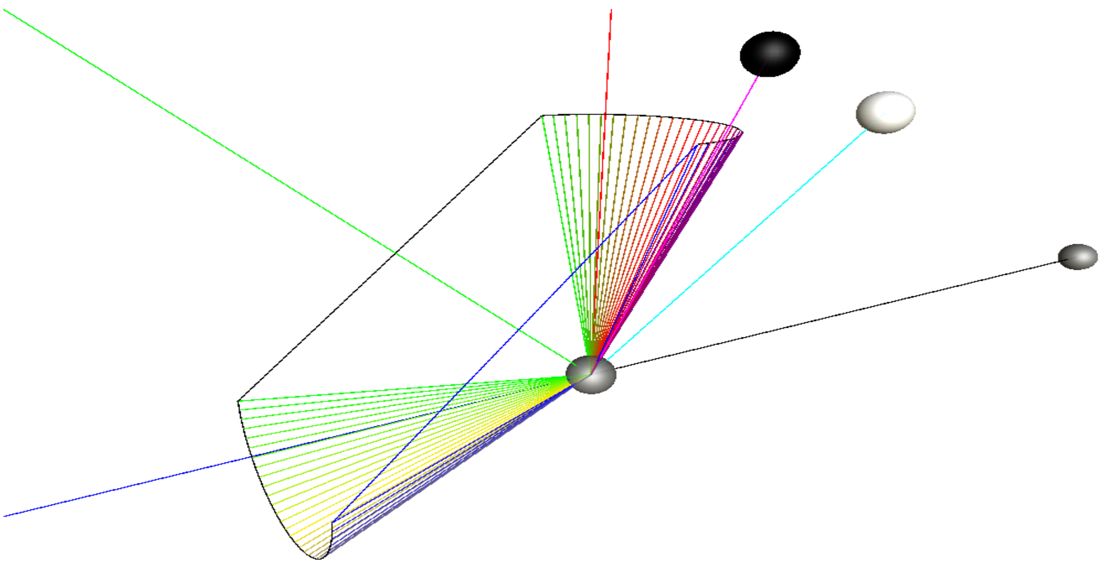


Figure 3.13: One angle is over 90°, like in a hinge joint. The blue line is the original Origin Vector, the green when rotation is in front of the parent, and the red when rotation is behind the parent, as is the case of the proposed cyan bone.

If the proposed rotation is within the quadrant associated with the angle over 90° and also *behind*, the Origin Vector is not the direction axis of the parent, but rather the axis on which the angle over 90° is associated. This way, a cone can be constructed with the angle over 90° as $\theta' = \theta - 90°$ and the 2d-point is checked to be within the ellipse on the new plane as before.

If the bone rotation is instead in front of the parent, the angle over 90° is set to a value close to 90°. The ellipse (in this case two almost parallel lines because the angle is close to 90°) is constructed on the plane with the Origin Vector as normal. The Origin Vector is in this case located between the parent directional axis and the axis associated with the angle over 90°.

3.4 Joint Localization

Joint localization is the process of finding the position of the joints defined in the skeleton model, in the three dimensional space. Motion capture marker positions are used for joint localization, and the placement of the markers is relevant to how this is done. A description of the marker set used and the anatomical placements of the markers can be viewed in Appendix A.

Localizing a joint center from motion capture data is a well studied subject, especially in biomechanics, where precise joint localization is very important (e.g., in gait analysis).

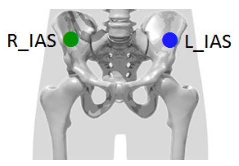
Some joint localization techniques are best suited when using a marker set where markers are placed on the middle of the limbs rather than as close to the joints as possible, such as the techniques used by Ringer and Lasenby [44], and Cameron and Lasenby [17]. When markers are placed on anatomical landmarks, as in this project, the best technique for many joints is to find the center of rotation by using empirical relations between the landmarks and the specific joint center [45].

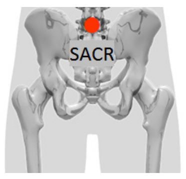
Techniques for localizing the center of rotation for joints are usually divided into functional and regression methods. Functional methods (iterative techniques usually using larger data sets) are much more precise than the heuristical regression methods on many key joints; e.g. the hip and the shoulder joints [16]. In animation, subcentimeter precision is not as necessary as in biomechanics, and regression methods are usually sufficient. Furthermore, the marker set used in this implementation is not suited for any real-time functional method.

When using regression methods, the markers are usually placed on anatomical landmarks as close to the joints as possible, such as the case when locating the hip joint in the method provided by Bell et al. [27]. The following sections describe how the joint center localization for the different joints in the skeleton model is performed in our implementation.

3.4.1 Hips and Pelvis

The centers of rotation of the hip joints are hard to locate since no markers can be placed on anatomical landmarks close to the joints [26] and thus, no straight lines can be drawn to the center of rotation.





The hip joints are a well studied area of joint localization. Some accurate and commonly used regression methods for finding the hip joints from the existing landmarks include methods by: Bell et al. [27], Davis et al. [25], and Harrington et al.[26]. Among these, the equation provided by Harrington et al. provided the most accurate and consistent results [46] with a measured maximum error of 21 mm [26], and was therefore chosen for implementation.

The Harrington method is defined from an offset of the anterior superior iliac spine with only the pelvis properties as variables where the joint position is defined as an X, Y, and

Z offset from the anterior superior iliac spine (markers L_IAS and R_IAS) where

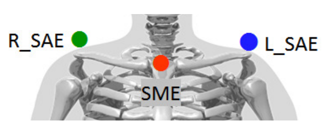
$$\begin{aligned} X &= 0.33 \times \text{pelvis width (mm)} - 7.3, \\ Y &= -0.30 \times \text{pelvis width (mm)} - 10.9, \text{ and} \\ Z &= -0.24 \times \text{pelvis depth (mm)} - 9.9 \end{aligned} \tag{3.1}$$

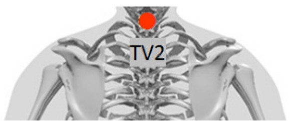
for the right hip. For the left hip, the X is negated.

The pelvis is located on the middle of the line between the hip joints.

3.4.2 Shoulders

The shoulder joint is another complicated joint center to locate. Soft tissues surrounding the shoulder joint moves in large part independently from the joint center of rotation [21].





The shoulder joint localization method used in this project is described by Campbell et al. [21]. The joint location is calculated from the acromial edge of the scapula, or the tip of the shoulder (the markers R_SAE and L_SAE), with the X, Y, and Z offsets:

$$\begin{aligned} X &= 96.2 - 0.302 \times \text{depth of chest (mm)} - 0.364 \times height(cm) + 0.385 \times mass(kg), \\ Y &= -66.32 + 0.30 \times \text{depth of chest (mm)} - 0.432 \times mass(kg), \text{ and} \\ Z &= 66.468 - 0.531 \times \text{width of shoulder (mm)} + 0.571 \times mass(kg) \end{aligned} \tag{3.2}$$

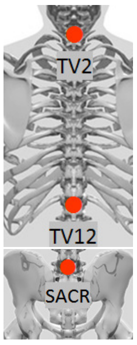
for the left shoulder. Z is negated for the right shoulder.

Campbell et al. present promising results of locating the shoulder joint without using many anatomical landmarks. The average error calculated in the study is approximately 6 mm. Such precise results are unlikely for this project because the equation uses body height and mass, which are parameters that have to be estimated. The height is estimated by calculating the distance from the marker positions at the feet to the marker positions at the head and the mass is estimated by using height and a body mass index of 24.

Another possible source of error comes from calculating depth of chest and shoulder width. The proposed method uses the seventh cervical vertebra (CV7) while the marker set used in this project uses the second thoracic vertebra (TV2), which is two vertebrae further down on the spine.

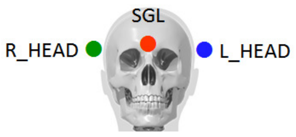
3.4.3 Spine

Three different spine joints are included in the skeleton model, corresponding to the marker set described in Appendix A: the lumbosacral joint, the second thoracic vertebra, and the twelfth thoracic vertebra. The positions of these joints are located by finding the orientation of the spine and offsetting the marker positions towards the body.



3.4.4 Head

The head joint, or the first cervical vertebrae, is located by a displacement from the middle point of the head markers, L_HEAD and R_HEAD, along the the negative z-axis of the orientation identified by the three head markers.



3.4.5 Knees and Ankles

The locations of the knee and ankle joints are based on a method described by Gardner and Chief [20] and further by Tranberg [45]. The method used creates virtual marker placements closer to the skin and finds a local coordinate system to produce more accurate localizations.

3.4.6 Elbows and Wrists

The elbow and the wrist positions are defined as the middle point between the markers located close to the respective joints.

3.4.7 Hands and Feet (End-Effectors)

An end-effector is the last joint in the hierarchy of the skeleton model, without any child joint connected to it. The end-effectors in the model are the hands, the feet and the head. The hands and feet are not joints in the same manner as the others, since they do not include any orientation, and they do not have any constraints. The positions of the markers placed on them are directly used as joint positions.

3.4.8 Hip Markers Interpolation

A special case exists for the hip markers. The pelvis is root in the skeleton model presented in Section 3.2 and as a result of how the joint estimation is implemented (see Section 3.6), the pelvis can not be missing after the joint localization. When hip markers are missing from a frame, they need to be estimated. This estimation is done by interpolating marker positions from the position of the previous frame.

If one of the three key markers for defining the hip, R_IAS, L_IAS, or SACR, is missing, the vectors from the previous frames between the other two markers are added to the existing marker positions. This ensures that the hip has the same proportions even when markers are missing. However, even if the proportion and position of the hip will be close to the real value, any displacement of the missing marker, that the other markers do not reflect, will not be estimated, resulting in incorrect hip orientation.

The same principle is used if two of the hip markers are missing, using previously known positions to estimate the other two. If all hip markers are missing, an existing non-hip marker in this frame that also existed in the previous frame is used and with the help of vectors from that marker to the hip markers, the hip is approximated. In this case, the hip rotation will deteriorate rapidly until the hip markers are again located.

Another possibility would be to use any known joint positions as root and target in a kinematic chain and estimate the position of the hip by using IK, as described in Section 3.6. This has neither been implemented nor tested.

3.5 Inverse Kinematics

Inverse Kinematics can be thought of as the reverse of forward kinematics, which is a technique for deciding the position of the last joint (the end-effector) in a kinematic chain when the positions and rotations of all other joints are known. Inverse kinematics, on the other hand, is used to determine the configuration of the joints in a kinematic chain when the end-effector position is known. The IK problem is defined as finding the posture of a kinematic chain to satisfy a given goal [4]. IK originated in robotics but is now commonly used in animation and related areas [34].

A kinematic chain consists of joints and links with a joint configuration specified by the scalars $\theta_1, \ldots, \theta_n$, where n is the number of joints [7]. The last joint in a chain is called an end-effector. In our case the joint configuration θ_i is an angle of rotation around some known axis for joint i. Each end-effector has a target position, and the joint angles, end-effector positions, and target positions are described by the vectors:

$\theta = (\theta_1, \ldots, \theta_n)^T$, column vector of joint angles,

$\mathbf{s} = (s_1, \ldots, s_k)^T$, column vector of end-effector positions, and

$\mathbf{t} = (t_1, \ldots, t_k)^T$, column vector of target positions.

The IK problem is solved when each end-effector position is located at the corresponding target position. The vector $\mathbf{e}$ is defined as

$\mathbf{e} = (e_1, \ldots, e_k)^T$

where $e_j = t_j - s_j$ is a vector of the desired change of position in end-effector j, and k is the number of end-effectors and the number of targets.

Each end-effector position can be seen as a function of the joint angles:

$$\mathbf{s} = f(\theta).$$

This means that given the angles for all joints, the positions of the end-effectors can be obtained; this is what is known as forward kinematics [7]. In inverse kinematics, joint angles need to be found such that the end-effector positions are equal to the given target positions:

$$\theta = f^{-1}(\mathbf{s}).$$

A solution to this equation might not exist, and if it does, the solution is not necessarily unique. There are several different methods for solving the IK problem approximately. It is often done by obtaining a better solution, in iterations, until a sufficiently good solution is reached. Some common methods are described in the following sections.

In this project, IK is used on simple chains with only one end-effector, and thus only one target. This simplifies calculations since the vectors $\mathbf{s}$ and $\mathbf{t}$, defined above, consist of only one element each.

When constraints are incorporated in the model, all the implemented IK algorithms suffer from deadlocks. There are cases when the target can not be reached because of constraints, but also cases when the target can be reached, but the algorithm fails to find a solution. This is solved by checking in each iteration of the IK algorithm if the end-effector is any closer to the target compared to the previous iteration. If it is not closer, the chain is likely to be stuck in a deadlock state. By rotating the chain a few degrees, it is forced to change its position, and hopefully allow for the target to be reached. This rotation is done two degrees at a time until the end-effector moves closer to the target, or until the chain has been rotated 360°. If the chain has been rotated 360° without success the solution can not be found and the algorithm is terminated. This was implemented as suggested by Aristidou et al. [47].

3.5.1 Jacobian Methods

Jacobian based IK methods use the Jacobian matrix (which is a matrix containing partial derivatives of functions) and its inverse to solve the IK problem. These methods are known to give good results, but also to converge slowly [7]. Jacobian methods can easily handle multiple end-effectors, but it can be hard to incorporate constraints efficiently and some of the methods might have problems when the target is unreachable [28].

The Jacobian used for the IK problem is a matrix of partial derivatives of the chain system relative to the end-effectors [7]. θ is a vector of scalars describing the joint configurations. In the case of rotational joints the scalar is the joint angle around some known axis [28].

The Jacobian matrix is defined by

$$J(\theta) = (\frac{\delta s_i}{\delta \theta_j})_{i,j}.$$

The matrix describes the relationship between a small change in angles and the resulting change in end-effector position. One entry is the derivative of the end-effector i with respect to the angle of joint j. The derivatives can be calculated by the equation

$$(\frac{\delta s_i}{\delta \theta_j})_{i,j} = v_j \times (s_i - p_j),$$

where v_j is the current rotation axis of joint p_j.

The problem is to find values for the angles such that each end-effector reaches its target; i.e. $\mathbf{t} = \mathbf{s}$ [28]. The solution is a linear approximation of the actual motion of the end-effector [7]. In other words, not only an end-pose for the kinematic chain is obtained; the whole trajectory for the end-effector is calculated [34].

The Jacobian methods are iterative, and a small step in the right direction is taken at each iteration. In each iteration a small change in end-effector position is obtained and it can be calculated by $\Delta\mathbf{s} = J\Delta\theta$. This means that $\Delta\theta$ should be chosen such that $\Delta\mathbf{s}$ is approximately equal to $\mathbf{e}$.

If $\Delta\mathbf{s}$ is replaced by $\mathbf{e}$ and the function $\mathbf{e} = J\Delta\theta$ is used, it is clear that $\Delta\theta = \mathbf{e}J^{-1}$. However, this function might cause problems since the Jacobian matrix in many cases is singular (i.e. not invertible) and it may not work well if J is nearly singular either [28]. There are several methods to avoid these problems. Some of the most common ones are described in the following sections.

Two Jacobian methods were implemented and tested, despite the fact that these methods often converge slowly. In this project, however, the kinematic chains to solve are small and have only one end-effector and target. Furthermore, the end-effector is never very far away from the target. The Damped Least Squares method was chosen since it has proven to be among the fastest alternatives [7], and the Jacobian Transpose method was chosen because of its simplicity and for comparison.

The Jacobian Transpose Method uses the transpose of the Jacobian instead of the inverse [7]. The transpose is not always the same thing as the inverse, but it can be shown that for a sufficiently small scalar $\alpha > 0$ the equation

$$\Delta\theta = \alpha J^T\mathbf{e}$$

will reduce the magnitude of the vector $\mathbf{e}$. I.e. the end-effector position will be moved closer to the target. α can be chosen such that the new value of $\mathbf{e}$ after the update is minimized, as described by Buss [28], and the following equation can be used:

$$\alpha = \frac{\mathbf{e} \cdot JJ^T\mathbf{e}}{JJ^T\mathbf{e} \cdot JJ^T\mathbf{e}.}$$

The Jacobian Transpose method has shown poor quality, but fast results compared to other Jacobian methods when the target is reachable and multiple end-effectors have been used. It has, however, been seen to work well with a single end-effector [28], which is one of the reasons why it was chosen for implementation for this project. The Jacobian Transpose was in our implementation sufficiently fast and provided acceptable results, even without constraints.

The Pseudoinverse Method uses the pseudoinverse of J instead of the regular inverse. A pseudoinverse, or general inverse, is a kind of partial inverse of a matrix which is singular or non-square [48]. It has some properties of a regular inverse and reduces to the usual inverse when the matrix is non-singular, and it is defined for all matrices. In this case, it gives the best possible solution to the equation $J\Delta\theta = \mathbf{e}$ using a least squares method [7]. The equation to be solved is

$$\Delta\theta = J^{\dagger}\mathbf{e}$$

where $J^{\dagger}$ denotes the pseudoinverse of J.

This method has shown poor results because of instability near singularities. When the Jacobian is exactly at singularity the pseudoinverse is well behaved, but when it is close to singular the changes in joint angles will be very large even for small changes in target positions, which is problematic [28]. Because of these problems this method was not chosen for implementation.

The Damped Least Squares (DLS) method finds the changes in joint angles, $\Delta\theta$, that minimizes the quantity

$$\|J\Delta\theta - \mathbf{e}\|^2 + \lambda^2\|\Delta\theta\|^2,$$

where λ is a damping constant and $\lambda > 0$.

This is equivalent to minimizing the quantity

$$\left\| \begin{pmatrix} J \\ \lambda I \end{pmatrix} \Delta\theta - \begin{pmatrix} \mathbf{e} \\ 0 \end{pmatrix} \right\|$$

where I is the identity matrix. This has the corresponding normal equation

$$\begin{pmatrix} J \\ \lambda I \end{pmatrix}^T \begin{pmatrix} J \\ \lambda I \end{pmatrix} \Delta\theta = \begin{pmatrix} J \\ \lambda I \end{pmatrix}^T \begin{pmatrix} \mathbf{e} \\ 0 \end{pmatrix}$$

which can be rewritten as

$$(J^T J + \lambda^2 I)\Delta\theta = J^T \mathbf{e}.$$

Then, the equation

$$\Delta\theta = (J^T J + \lambda^2 I)^{-1} J^T \mathbf{e} \tag{3.3}$$

can be used, and since it can be shown that $(J^T J + \lambda^2 I)$ has an inverse, this equation will not have any singularity issues [28]. The matrix to be inverted is $n \times n$ in size (where n is the number of joints in the chain), but this equation can be further rewritten to obtain

$$\Delta\theta = J^T (J J^T + \lambda^2 I)^{-1} \mathbf{e} \tag{3.4}$$

where the matrix to be inverted has the dimensions $m \times m$. $m = 3k$ and is usually much smaller than n, but since no multiple end-effectors are handled in this project, $k = 1$, and the matrix has the dimensions 3×3. The chains are also very short, usually containing two or three joints, which makes a matrix with the dimensions $n \times n$ smaller or equal to the matrix with dimensions $m \times m$. Therefore, the difference in computation time using Equation 3.3 or Equation 3.4 is insignificant.

The DLS method avoids problems with singularities and is numerically stable with a properly chosen damping constant [7]. The damping constant depends on the kinematic chain and the target positions. It should be large enough to avoid singularity problems, but small enough for the algorithm to not converge too slowly. The damping constant can be set dynamically depending on the configuration of the kinematic chain [28], but in our implementation it is set to a static value of $\lambda = 1.1$. This value was proposed by Buss and Kim [49].

The DLS method was implemented and tested since it has converged relatively fast in previous studies [7, 28]. It did provide sufficiently fast results, and the damping constant did not cause too much oscillation. The tests are further described in Section 4.

3.5.2 Cyclic Coordinate Descent

Cyclic Coordinate Descent (CCD) is an iterative heuristic inverse kinematics algorithm, often used in computer games [7]. It was first introduced by Wang and Chen, in 1991 [33], for robotic manipulators. It rotates and translates one joint at a time in a simple way, making it very fast [4].

For every joint in the chain, starting with the one closest to the end-effector, the angle between the vectors from the current joint to the end-effector, and the current joint to the target is found. The end-effector together with the affected joints are then rotated by this angle towards the target (see Figure 3.14). This is repeated until the end-effector reaches the target or can be considered close enough [7].

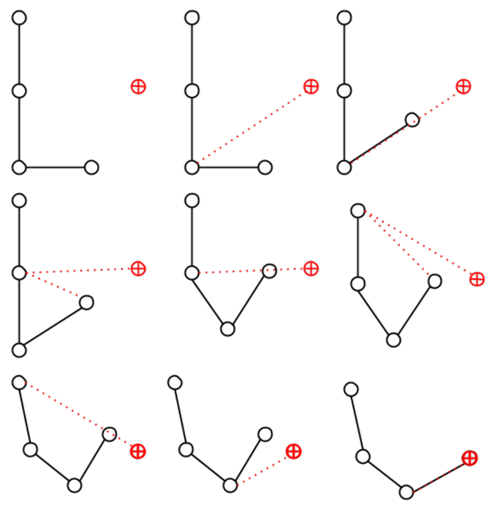


Figure 3.14: In this example CCD reaches the target in two iterations.

CCD is fast, but it suffers from unrealistic motion since it often overemphasizes the movement close to the end-effector [7], and therefore, incorporating constraints is extra important to obtain realistic poses. It also has no easy way of handling multiple end-effectors, but since the implementation for this project only handles single end-effectors this is not a problem. CCD was chosen for implementation because of its popularity and its speed.

No real-time problems did occur during testing and CCD provided poses with small error. CCD does, however, run into problems when the chain is straight, and the target is located on the chain, or on the chain's reflection in the root. The target, in that case, cannot be reached, and the chain needs to be rotated a few degrees to allow for the algorithm to find a solution.

3.5.3 Forward And Backward Reaching Inverse Kinematics (FABRIK)

Forward And Backward Reaching Inverse Kinematics (FABRIK) is an IK algorithm introduced by Aristidou and Lasenby [4] that uses an iterative approach, finding joint positions by locating a point on a line. FABRIK first checks if the target is reachable, and if it is, the algorithm iterates until the end-effector reaches the target or a maximum number of iterations is reached. The first step in one iteration is done forwards; it adjusts each joint angle one at a time, starting from the last joint in the chain and working its way to the root. The second step is done backwards in the same way. It then iterates until the end-effector has reached the target or is close enough.

The joint-transforms in the forward step are computed by moving the end-effector to the target and the new positions of the following joints are found on the lines between the next joint position and the current position, as seen in Figure 3.15. This is likely to make the root move away from its original position, and that is why the backward step is done the same way, but the root's original position is treated as target and the root is moved to its original position.

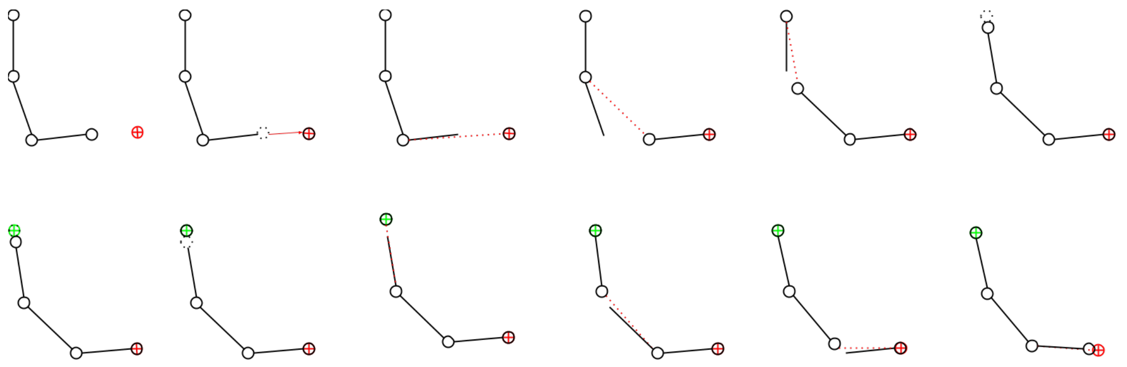


Figure 3.15: A forward (top row) and backward (bottom row) reach in FABRIK.

The algorithm is fast and it takes few iterations to converge. There are ways to incorporate constraints and handle multiple end-effectors [4].

Because FABRIK is an algorithm with promising results in previous studies [7, 40], it was chosen for implementation in this project. There are, however, many special cases where the algorithm does not work well. It has, like CCD, problems when the target is located on the chain or its reflection (when the chain is straight). In the tests, FABRIK was found to be much slower than expected and it suffers greatly from deadlocks. The tests and results are described in Section 4.

3.6 Estimation of Joint Positions

This section describes the way IK is used, in this project, for finding missing joints. The joint localization might not be able to determine the position of every joint in the skeleton model, due to marker occlusion, and it is the task of the IK solvers to estimate these joints. Because IK solvers need a chain and a target, an initial position of the chain needs to be estimated.

If a joint somewhere in the middle of a joint chain is missing, the initial chain is constructed by taking the closest existing parent to the missing joint as root of the chain. The rest of the chain is constructed from the joints of the previous frame, but offsetted by the directional vector from the missing joints parent in the last frame, as seen in

Figure 3.16. This chain is then passed to the chosen IK solver together with the next known joint as target.

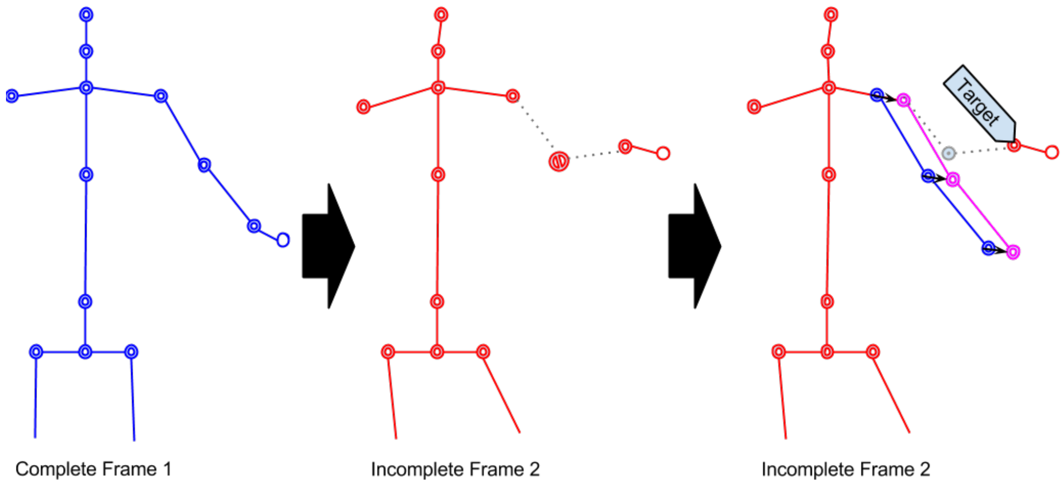


Figure 3.16: In the second frame the elbow position is missing. This is estimated with IK by constructing a chain from the shoulder to the wrist from the previous frame. This chain is offsetted such that the root is located at the shoulder joint position of the current frame. The resulting chain is shown in magenta. The wrist position of the current frame is set as target.

In the case of a missing end-effector there is no possible target to pass to a solver. Therefore the end-effector from the previous frame is used such that the joint closest to the end effector has the same rotation as in the last frame. A full description can be seen in Algorithm 1.

```
input  : A skeleton skeleton of size $l$
input  : A complete skeleton lastskeleton of size $l$
for $i \leftarrow 1$ to $l$ do
    if skeleton[$i$] has no position then
        offset $\leftarrow$ skeleton[$i-1$] $-$ lastskeleton[$i-1$];
        Add lastskeleton[$i-1$] to chain;
        for $j \leftarrow i$ to $l$ do
            Add (lastskeleton[$i$] $-$ offset) to chain;
            if skeleton[$j$] is not missing then
                target $\leftarrow$ skeleton[$j$];
                chain $\leftarrow$ IKSolve(chain, target);
                break;
            end
            else if skeleton[$j$] is end effector then
                break;
            end
        end
        Add chain to skeleton;
        $i \leftarrow j$;
    end
end
lastskeleton $\leftarrow$ skeleton
```

Algorithm 1: Algorithm for joint positions estimation

4

Results

The first section of this chapter describes how well the skeleton model and the joint localization worked for this project. The model and the joint localization algorithms are related since the result is dependent on both the joints to be located and the methods for locating them. The following sections address the inverse kinematics algorithms and present results for both accuracy and performance.

The tests were performed on motion capture recordings of a person running, a person biking, and a person dancing. The chosen sets of motion capture data were free from gaps in marker positions, and tests were created where strategically selected joints were removed to simulate marker occlusion gaps. These removed joint positions were then estimated with IK solvers and compared against the positions obtained by joint localization, in order to test accuracy of the estimated joint positions. All the different IK algorithms were tested and compared against each other, both with and without constraints. Tests to measure time were also performed for the different IK algorithms.

The different parts of the implemented system works well; the joint localization methods calculate good joint positions when none of the markers needed are occluded, and the estimated poses are realistic as long as constraints are incorporated. The full results are presented in the following sections.

One problem with the system as a whole is that even though the poses frame by frame are realistic, the animation might not always be smooth. The reason for this is that as soon as an occluded marker is found again, the true position of a chain can be obtained. The estimated joint position might have been far from the true position, resulting in twitching motions.

As mentioned in Section 3.6, the skeleton can not be solved with IK if an end-effector is missing. The rotation of the last bone in the chain is instead set to the rotation it had in the last frame. This still passes as acceptable motion, but when many markers on the bones connected to the end-effector are occluded, that part of the body will be stiff, since the joint positions can not be estimated in a good way.

4.1 The Skeleton Model and Joint Localization

The skeleton model has a total of 22 joints, twelve of which correspond to existing joints in the human body. The end-effectors hands and feet are not really joints, but model the end of the chain. They are located by directly using the positions of the markers, which are located on the hands right before the fingers start, and on the feet before the toes. As a result of this, the orientation of the hands and feet differs from the real orientation; this is most noticeable on the hands, which are rotated too much towards the upper side of the arm. Since no fingers are modeled, movement of these is not possible.

The spine is modeled with three joints from the pelvis to the head. The pelvis is a special case as well, and acts as a root in the skeleton hierarchy. The head joint is an end-effector, but it acts more like a regular joint, defining the position and orientation of the head. But since it is an end-effector, the orientation of the head is not defined by rotating the joint towards the next joint in the chain, as it is for all other joints; it is calculated from the three head markers. The full definition of the skeleton model can be viewed in Section 3.2.

The constraints incorporated in the model allow for exact definitions of the joint limits. Each bone in the model can have its specifically defined limits, but since all joints are modeled in the same way, as ball- and socket joints, the same types of constraints applies for all bones. The constraints are checked for each bone in an iterative manner during the execution of the IK algorithms, which makes sure that the bones never twist and rotate in an unrealistic manner. The constraints are, however, increasing the execution time of the IK algorithms, as can be seen in Section 4.3. There are also problems with deadlock when using the constraints, i.e., a solution is never found because of the constraints limiting the chain's rotation, even though a solution exists.

A description of the joint localization can be seen in detail in Section 3.4. There are no problems locating the joints as long as no marker positions are missing and the markers are located at the exact positions defined by the marker set. Every joint in the skeleton model is located quickly, and as can be seen in Figure 4.1, the joint positions are reasonable compared to the human body.

One, sometimes problematic, method is the one for locating the knee and ankle. It uses the same markers for both joint positions which results in the loss of both the knee and the ankle position if one or more of those markers are missing. The biggest problem

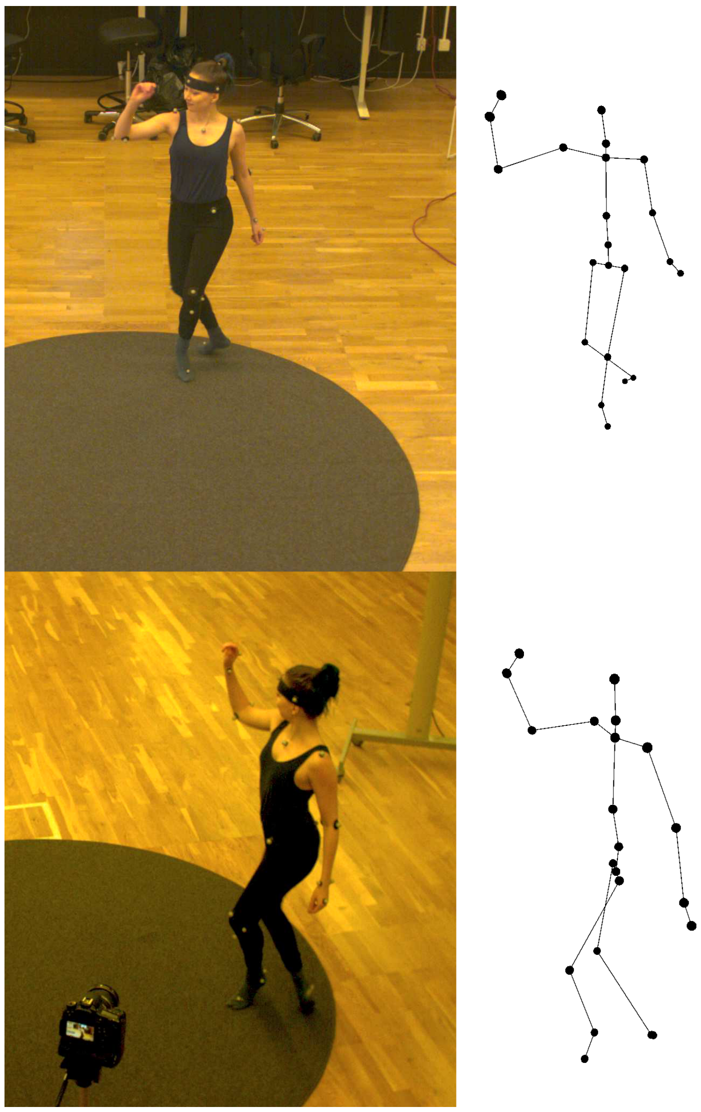


Figure 4.1: A pose of the skeleton obtained only by joint localization compared to the actor's actual pose.

when this happens is if the end-effector on the foot is occluded as well. This results in a chain of three joints that can not be solved with IK, and the orientation of the leg is set to the orientation it had in the last frame, which leaves the leg stiff for as long as the markers are missing. The pose of the skeleton is always hard to estimate with many markers missing for both limbs and end-effectors, but the described situation can occur when only two markers are missing.

4.2 Inverse Kinematics Accuracy Comparison

The tests presented in this section were performed on motion capture sequences with generated gaps in joint positions, and these missing joint positions are estimated with the different inverse kinematics algorithms. Joints in the arms or legs were removed, since those are the most interesting joints for comparison. The spine does not move much and the end-effectors are not estimated with IK if they are missing, since no target exists in that case. One or more joints in sequence were removed, and the error in joint position from one of these joints compared to the skeleton obtained from joint localization was measured over time. An as small error as possible in joint position is a good result, since the goal for the joint estimation is to get a realistic pose close to the actor's actual pose.

The tests were done both with and without constraints for the joints. In general, all algorithms performed better with constraints. FABRIK sometimes got a very large error with constraints, because it suffers from problems with deadlocks in that case, and never reaches the target. In some tests, FABRIK actually obtained better poses without constraints. Constraints are, however, necessary to obtain realistic poses. CCD performs the best, with smooth motions, but the two Jacobian solvers are not far behind.

In conclusion, after analysing all tests, FABRIK has poor performance with constraints and constrained CCD produces, overall, the best results.

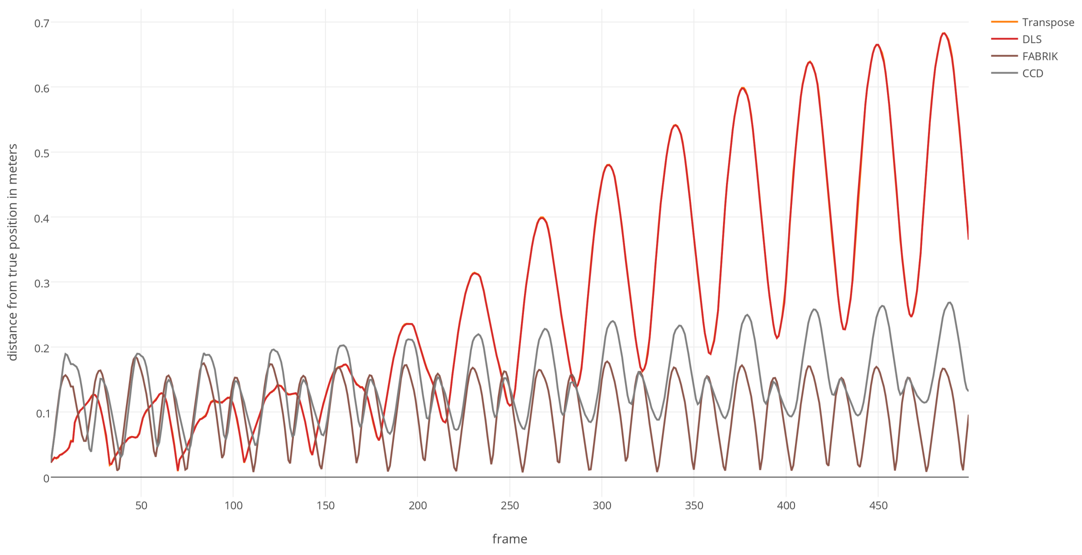


Figure 4.2: Distance to the true position of the knee when no constraints were incorporated. Tested on a biking sequence with the positions of the knee joint and the ankle joint removed. The Jacobian Transpose and DLS follows each other exactly.

Figure 4.2 shows the error of the knee joint during a motion capture sequence of a person riding a bicycle on a treadmill, where the positions of both the knee joint and the ankle joint are missing. The knee and ankle are localized with the help from the same markers, which means that both joints are missing when one of those markers are occluded. No constraints were applied on the skeleton. FABRIK has an accuracy always below 20 cm and does not increase over time, whilst CCD exceeds that error early and is approaching an error of 30 cm after 450 frames. The Jacobian Transpose and DLS find the same resulting position, as seen in Figure 4.8, and therefore the errors follow each other exactly. The smallest error is in the beginning, but it increases quickly over time. This is the result of the knee rotating in the wrong direction.

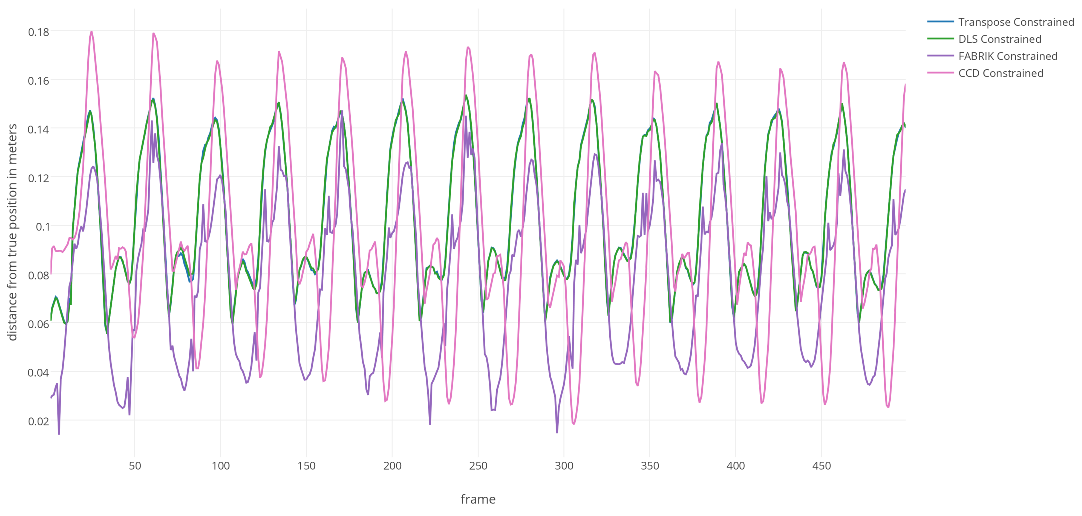


Figure 4.3: Distance to the true position of the left knee when constraints were incorporated. Tested on a biking sequence with the positions of the knee joint and the ankle joint removed.

When constraints are applied for the same motion capture sequence, none of the algorithms differ more than 18 cm from the true position, as can be seen in Figure 4.3. CCD gets the biggest error and also the biggest amplitude. The Jacobian Transpose and DLS follows each other in this case as well. The errors are much smaller when using constraints and are not increasing over time, since the constraints ensure that the rotation of the knee is kept within natural limits. The error oscillates for all algorithms, since biking is a cyclic motion, and Figure 4.4 shows the poses where the error is large and where it is small. The poses obtained are still realistic and the biking motion is acceptable.

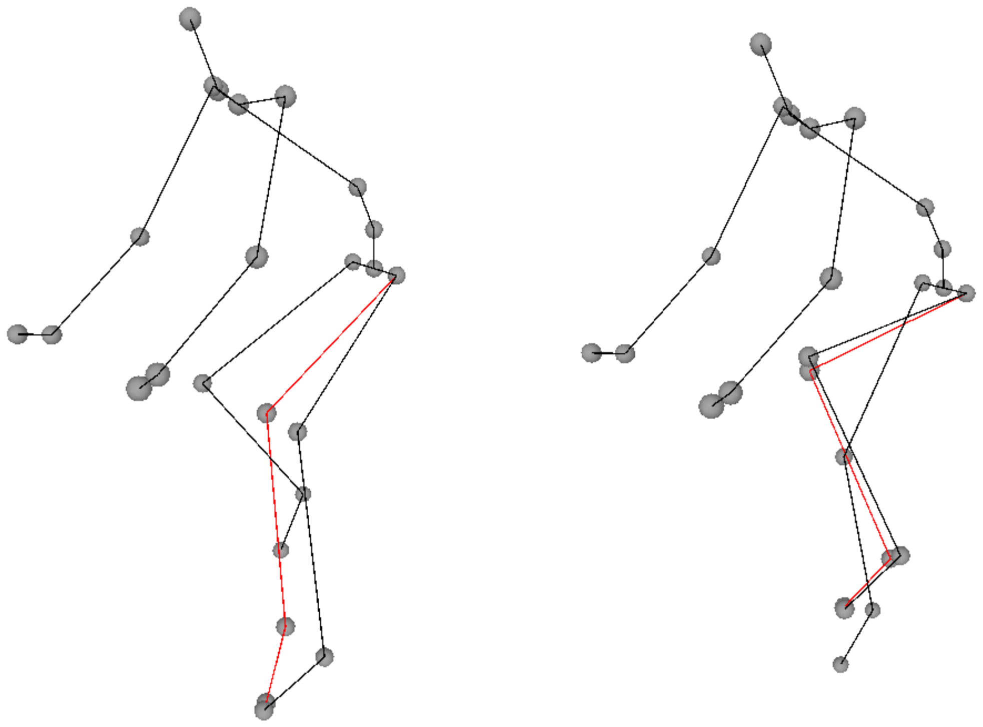


Figure 4.4: The located positions of the bones, in black, and the estimated position of the leg in red. The knee and ankle positions are estimated using CCD.

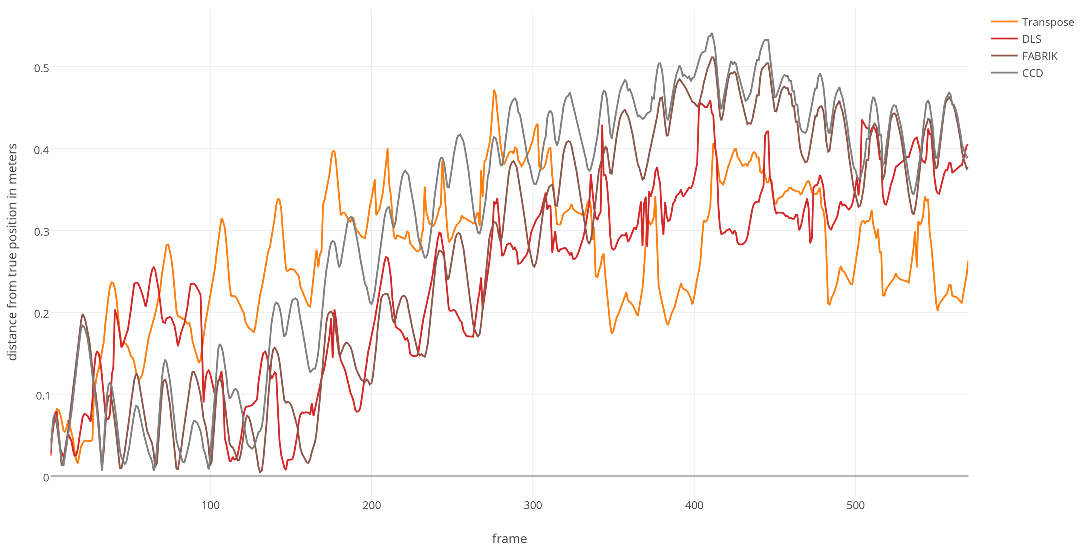


Figure 4.5: Distance to the true position of the elbow when no constraints were incorporated. Tested on a running sequence with the position of the elbow joint removed.

Figure 4.5 shows the error in position for the elbow joint during a running trial. All algorithms are unconstrained, and the error is increasing much over time. The Jacobian Transpose is more even during the whole interval, compared to the other algorithms.

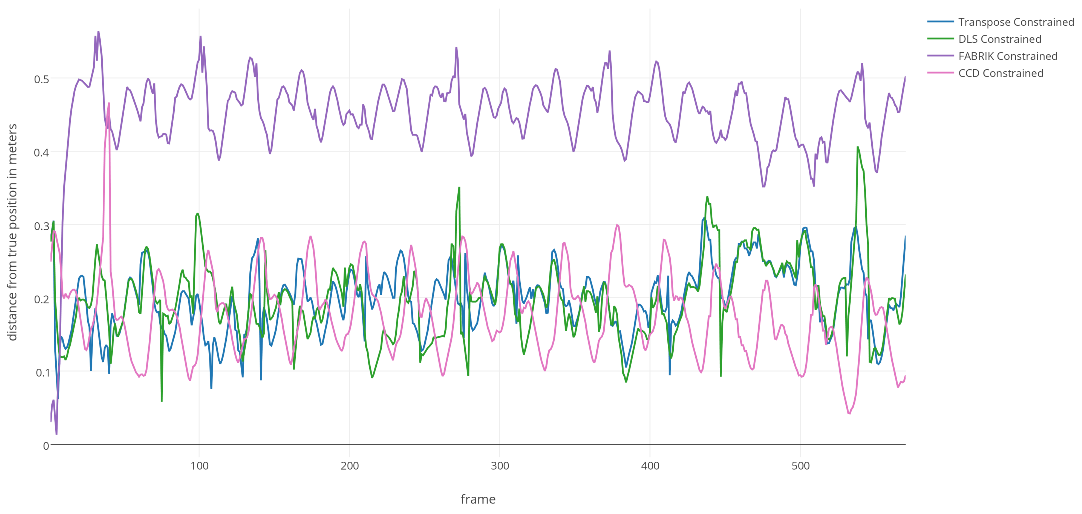


Figure 4.6: Distance to the true position of the elbow when constraints were incorporated. Tested on a running sequence with the position of the elbow joint removed..

Figure 4.6 shows the same trial with the algorithms constrained. The Jacobian Transpose is the only one to stay below an error of 30 cm, while DLS and CCD sometimes peak, and FABRIK has a consistently high error with a mean of 45 cm.

		Min	Max	Mean	Median	Std Dev	Variance
Transpose	Unconstrained	0.02	0.47	0.27	0.28	0.09	0.01
	Constrained	0.06	0.31	0.2	0.2	0.04	0
DLS	Unconstrained	0.01	0.46	0.24	0.27	0.11	0.01
	Constrained	0.06	0.41	0.2	0.2	0.05	0
FABRIK	Unconstrained	0	0.51	0.27	0.3	0.16	0.03
	Constrained	0.01	0.56	0.45	0.46	0.06	0
CCD	Unconstrained	0.01	0.54	0.32	0.38	0.16	0.03
	Constrained	0.04	0.47	0.18	0.18	0.06	0

Table 4.1: Statistics over the distance in meter to the true position of the elbow in a motion capture sequence of a person running. The best cases are written in green and the worst in red.

Table 4.1 shows the same data as Figure 4.6 and Figure 4.5. FABRIK is performing badly in general and CCD gets much better when constraints are applied. Both the Jacobian methods have comparable results, and CCD gets slightly better results than both of them. This is consistent with other tests, where FABRIK gets worse with constraints and constrained CCD gets the best results.

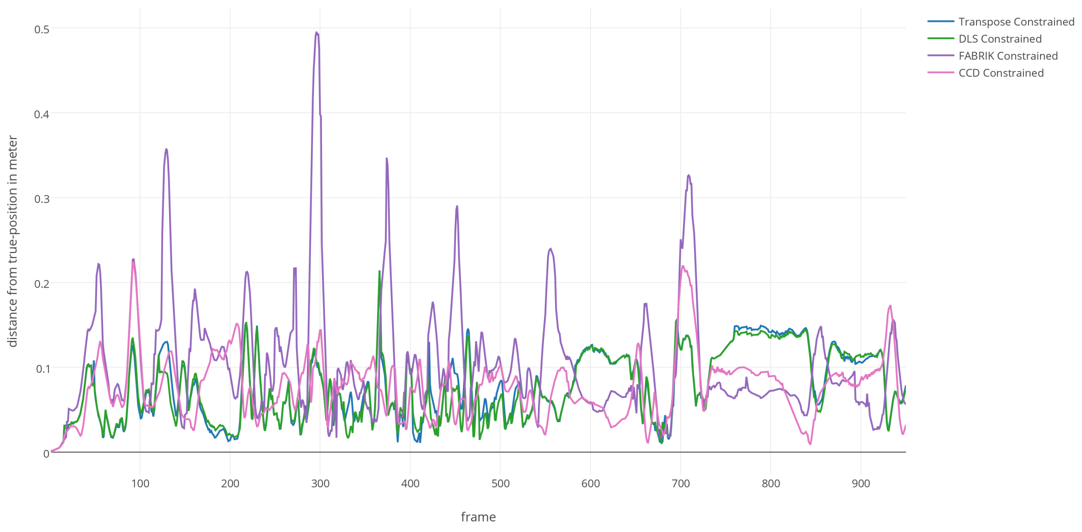


Figure 4.7: Distance to the true position of the knee when constraints were incorporated. Tested on a dancing sequence with the position of the knee joint removed.

Tests were also done on more non-cyclic motions. Figure 4.7 shows how the algorithms behave, with constraints, for a short dance sequence with the knee removed. The average error is below 20 cm for most algorithms, but FABRIK has peaks going almost as high as 50 cm. FABRIK is behaving this way because of its problems with deadlocks. If deadlock occurs, the algorithm tries to rotate the chain to allow for more rotation in other joints than before, but if the target still can not be reached, the chain might be stuck in a position far from the target, resulting in a large error.

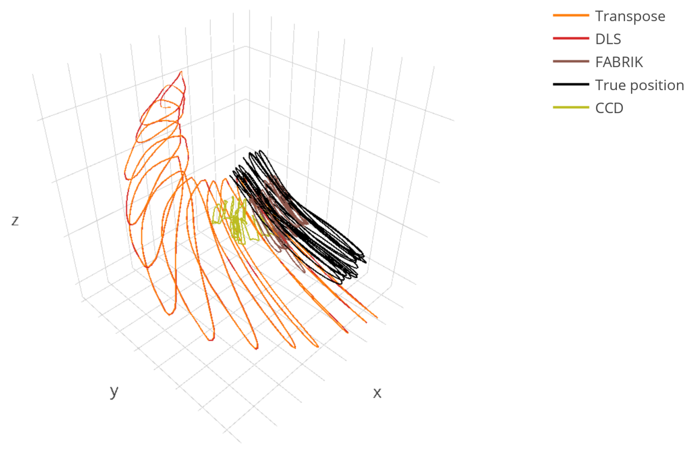


Figure 4.8: The position of the knee joint in 3D space when no constraints were incorporated, measured over time. Tested on a biking sequence with the positions of the knee joint and the ankle joint removed. The Jacobian Transpose and DLS follows each other exactly.

Figure 4.8 is a 3D plot of the same data as shown in Figure 4.2. It shows how the Jacobian algorithms slowly move away from the true position. CCD, and in a smaller degree FABRIK, stays closer to the true position but moves in a much smaller amplitude. This is the result of the joints closer to the end-effector moving more than the joints further away.

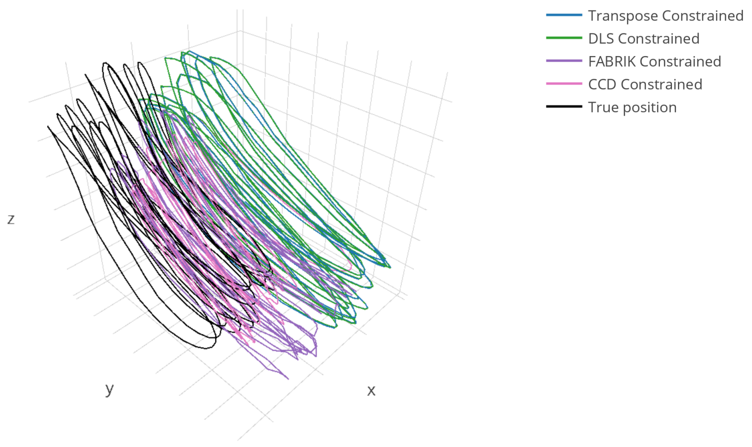


Figure 4.9: The position of the knee joint in 3D space when constraints were incorporated, measured over time. Tested on a biking sequence with the positions of the knee joint and the ankle joint removed.

The 3D plot in Figure 4.9 shows the same trial as the previous plot, but with constraints applied. It can be seen that even though the estimated joint positions are a small distance away from the correct ones, the cyclic motion is preserved. The results are much better with constraints.

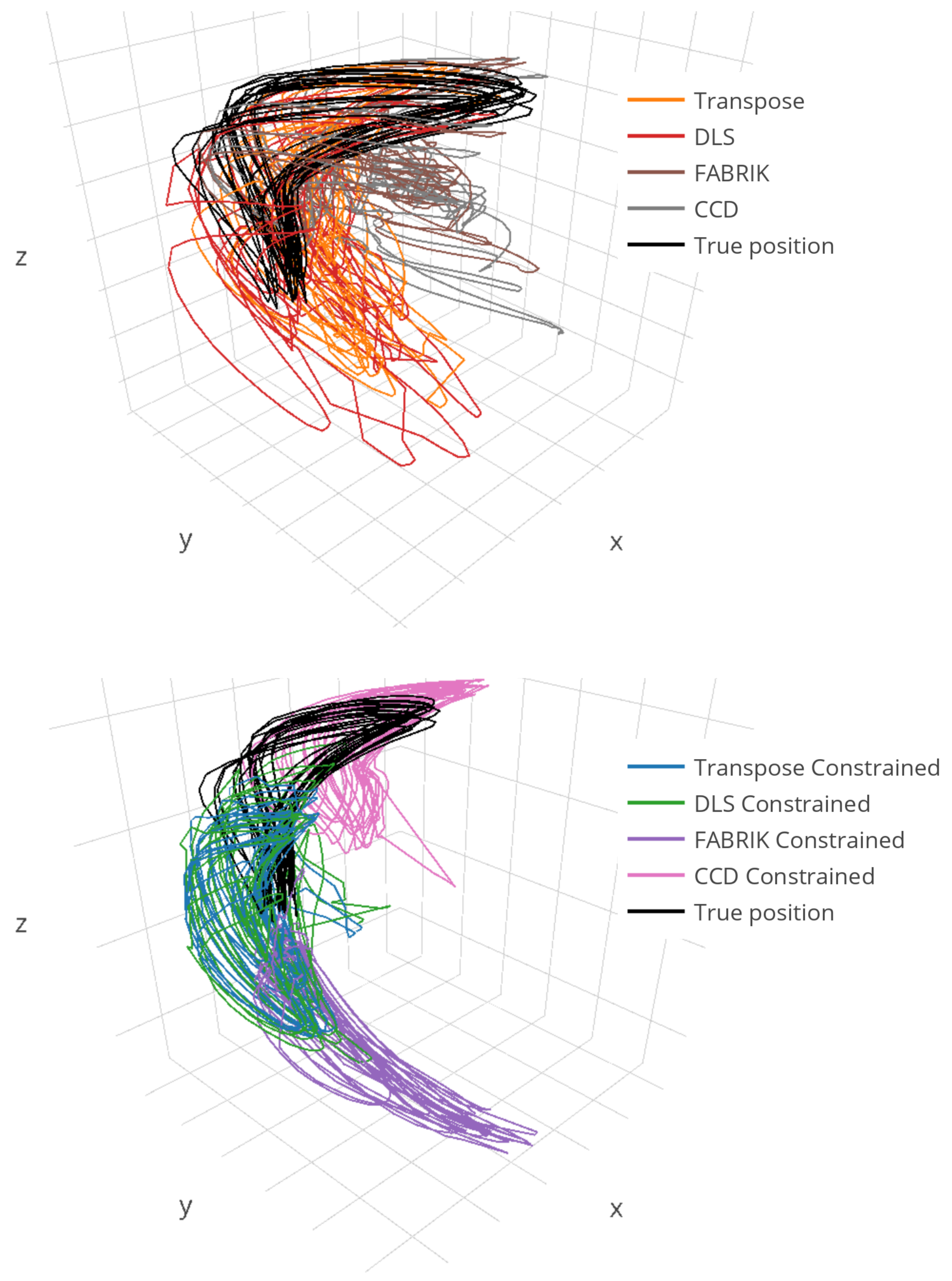


Figure 4.10: The position of the elbow joint in 3D space, measured over time. Tested on a running sequence with the position of the elbow joint removed.

In Figure 4.10 the advantage of constraints in IK is clear, where the motion of the elbow is smoother, and the position is closer to the true elbow position. The error of FABRIK is also evident, where the elbow is locked in a position further away from the true position when constraints are applied than when not. The trial is the same as in Figure 4.5 and Figure 4.6 showing the distance to the true position.

4.3 Inverse Kinematics Execution Time

To measure the execution time of the different IK algorithms, all joints except the end-effectors and the pelvis were removed from the skeleton after the joint localization of a person running, to test a worst-case scenario. All tests were run on a Windows 7 Professional 64-bit PC with an Intel(R) Core(TM) i3 CPU M 370 @ 2.40GHz (4 CPUs) processor and 3072 MB of RAM.

The IK algorithms were tested both with and without using constraints. Without constraints all algorithms were very fast. With constraints, the execution time increased much but it was still acceptable for real-time for all algorithms except FABRIK.

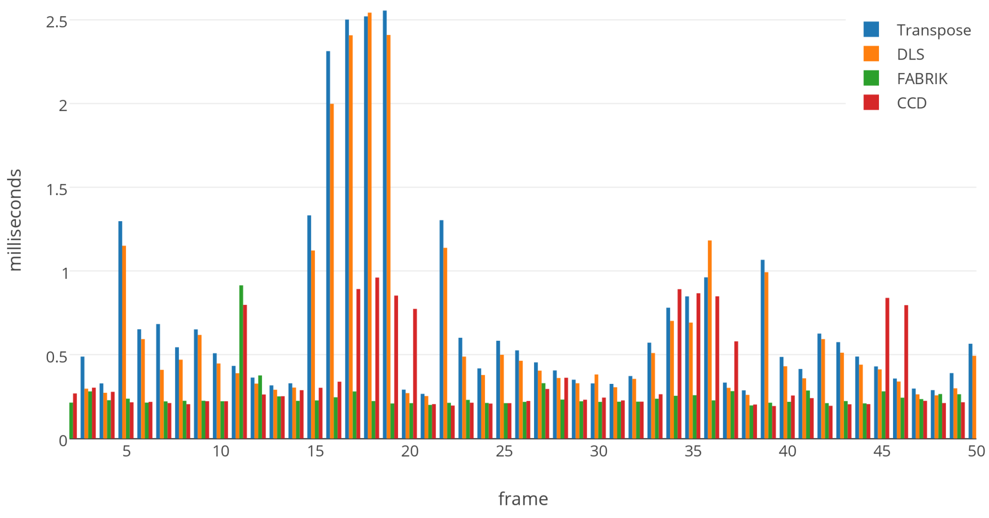


Figure 4.11: The execution time of the fifty first frames when solving the full body without using constraints.

		Min	Max	Mean	Median	Std Dev	Variance
Transpose	Unconstrained	0.17	2.59	0.6	0.48	0.39	0.15
DLS	Unconstrained	0.17	4.16	0.54	0.43	0.37	0.14
FABRIK	Unconstrained	0.17	3.24	0.24	0.23	0.13	0.02
CCD	Unconstrained	0.16	1.07	0.41	0.24	0.28	0.08

Table 4.2: Statistics of the execution time, for all 569 frames, in milliseconds when solving the full body without using constraints. The best cases are written in green and the worst in red.

As seen in Figure 4.11, unconstrained IK algorithms have no problem with real-time performance. Except for a few frames where the Jacobian algorithms, DLS and Jacobian Transpose, peaked to around 2.5 ms, the algorithms were completed in under 1 ms. FABRIK had the fastest average computational time, with a mean of 0.24 ms, and the slowest was the Jacobian Transpose with a mean of 0.6 ms.

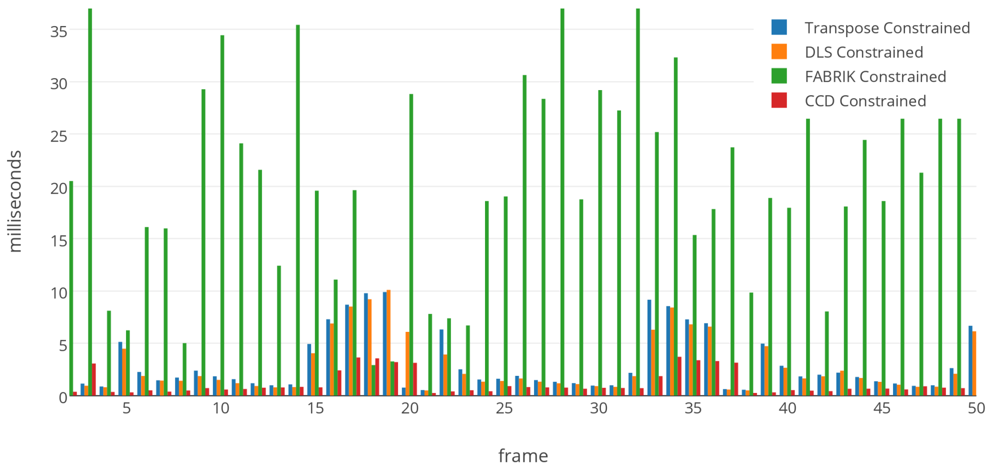


Figure 4.12: The execution time of the fifty first frames when solving the full body when using constraints.

		Min	Max	Mean	Median	Std Dev	Variance
Transpose	Constrained	0.17	23.21	3.69	1.86	3.59	12.87
DLS	Constrained	0.17	24.69	3.48	1.6	3.65	13.29
FABRIK	Constrained	2.42	60.41	22.03	19.71	12.23	149.52
CCD	Constrained	0.16	5.08	1.4	0.72	1.27	1.61

Table 4.3: The execution time in milliseconds, for all 569 frames, when solving the full body when using constraints. The best cases are written in green and the worst in red.

When applying constraints to the skeleton model, the execution times get severely degraded. As seen in Figure 4.12, FABRIK is slow in finding the solutions. From having the best execution time unconstrained, FABRIK has the worst when constrained, with a mean of 22.03 ms and peaks up to 60 ms, which is not suited for real-time performance. The reason for FABRIK's slow results is most likely that it gets stuck in deadlocks. The IK algorithm with the best execution time, with constraints, was CCD, with a mean of 1.4 ms and a maximum execution time of 5.08 ms.

5

Discussion

AMONG the biggest problems with the system is the non-smooth animation that occurs between frames when a joint position jumps from an estimated position to a true position. This occurs when a marker position that has been missing is found again, and causes a twitching motion. It is most noticeable if more than one joint position are missing, and especially if one of them is an end-effector. If the animation should be used for analysis, the true joint positions might be more important than smooth animation, but for this application it would be desirable to solve the problem with sudden position jumps. One solution would be to interpolate towards the true joint position when it is again located, in order for the chain to move towards its position in an appropriate number of frames.

The pelvis position is never estimated with IK; if hip markers are missing the position is estimated during the joint localization phase. This could, however, also cause twitching motions when markers are found again by the cameras. This is most evident when two markers are missing for a long time, which might cause the pelvis to be skewed in unnatural positions. As described above the motion would be smoother by interpolating between the estimated and the true position . The unnatural position could possibly be improved by estimating the pelvis position with IK in a similar manner as the rest of the joints, by constructing a chain between one of the legs and the spine. This was not implemented due to the way the skeleton hierarchy is defined, which complicates implementation.

If estimating the hip positions with IK, it would probably be preferable to perform the estimation of the pelvis by solving the IK problem with multiple end-effectors. Multiple end-effectors is something that would be beneficial for the estimation of the sternoclavicular joint and its children, as well. There are different methods for handling multiple end-effectors, depending on the IK algorithm, presented in the literature. CCD has no easy way of doing this, while it is straight forward to implement for the Jacobian methods.

Many similar implementations do not differentiate between the joint localization and the joint estimation phase, but estimate positions during the localization phase and might correct these estimations afterwards. One advantage of doing this in separate phases, like in our implementation, is that the methods and algorithms easily can be replaced without changing the whole system. For example, the joint localization methods can be replaced by other methods for all joints, or just one or two joints. The joint estimation can be replaced by other IK algorithms, or by a totally different approach.

One interesting change would be to implement joint localization for other kinds of marker sets. The system could be extended both for other marker sets for humans, but also for other animals, like horses or dogs. A marker set with more markers placed on the limbs, instead of close to the joints, could avoid the necessity for a predefined marker set with exact positions, and even the necessity for a predefined skeleton hierarchy. This would allow for animation of both humans and other animals, without having to change technique.

A good feature with the current implementation is that it does not require any calibration or startup phase. This allows for a user to quickly get started with the session, since it already can be time consuming to set up and calibrate the motion capture system.

5.1 Improving the Skeleton Model

Acceptable results were achieved with the skeleton model used in this project, but some design choices might not have been optimal. The use of global bone positions and rotations was one choice that in retrospect could have been different. If local positions would have been used instead, each position would have been defined from the position of its parent, and it would have simplified some calculations. The algorithm seen in Section 3.6 would not be as complex, only copying the missing bones from the last skeleton to the current without the need for the offsets.

The skeleton model could also have been made a lot more complex than it is in this project, modeling more joints. E.g., with more joints along the spine the poses could look more realistic. Fingers are also something that would be desirable for most applications. Another marker set would, however, have been needed to locate more joints in a good way.

The constraint model explained in Section 3.3 worked well. They are intuitive and allow for very exact definitions. However, finding good values for the constraints on the different joints was done by trial-and-error, and the constraints as they were run in the tests could be far from optimal. Fine tuning the constraints could lead to better solutions. It would also be preferable if the constraints could be user defined since humans can be more or less agile, and if specific motions are captured, specific limits might be known. E.g., when biking it is probable that the feet have approximately the same rotation the entire time, and then it could be desirable to ensure this with constraints.

5.2 Joint Localization Improvements

As mentioned, the marker set used is small and does not allow for much change in the skeleton model or the joint localization. The disadvantage of this marker set is that when mistakes occur in identifying the markers, or when the markers are placed far from its defined landmark, localization of joints might suffer greatly. Also, when markers are put on clothes they might be displaced when the fabric is moving in relation to the skin. Even when placed on the skin, the markers might move in relation to the bones. This might result in the joints being slightly displaced from one frame to another, but in our implementation it was not noticeable. This is, however, a problem that previously have been solved using IK to keep the bone lengths constant.

If a marker set with more markers were to be used, the joint localization might be more accurate and robust. But the advantage of using this small marker set is, naturally, that less markers need to be put on the body, and more markers might lead to less freedom of movement for the actor. On, e.g., the leg the markers are placed for ease of movement, with no markers placed on the inside of the knees, since it is easy to drop those markers when moving.

As previously mentioned, the method for localizing the knee and ankle uses the same markers for both joints, causing a loss of two joints every time one of those markers are occluded. To use different methods to localize the knee and the ankle would make the skeleton more stable and less error-prone. But on the other hand, the method used now is locating the positions well, and is well researched in biomechanics.

Another flaw in the joint localization is that the marker positions on the hands and feet are used directly as end-effector positions. This means that the end-effector will not be located inside the hand or foot, but slightly above it. A solution would be to offset the positions a small distance to be located more inside the body. It would be necessary to create a local coordinate system from the markers placed on the wrist, making the joint position of the end-effector dependent on three markers instead of just one. If losing one of the three markers, the hand positions would also be lost, and would thus be more prone to disappear.

5.3 Joint Estimation and IK Improvements

Some of the results in the tests were really surprising, such as the poor results for FABRIK when using constraints. The constraints are based on the same paper as the introduction of the FABRIK algorithm [4], and the expectation was that FABRIK would be a frontrunner in performance and accuracy. The reason for this is probably, as mentioned, that the algorithm gets stuck in deadlock where it can not find the solution for reaching the target. This problem needs to be solved in order to use FABRIK, and would probably require some remodelling of the constraints, or the way the constraints are checked during the IK phase. FABRIK has potential, since it is so fast and finds good poses even without constraints, and it could be worth the work. However, the constraints work very well together with CCD, and deadlocks does not occur as often.

Other ways to improve the estimation of joint positions would be to use previous frames in order to take advantage of the velocity of joints. Both speed and direction of the joint could result in more accurate joint estimations. Often when positions can not be located due to marker occlusion, not all markers associated with that joint is missing. If two markers were to be occluded on the elbow, for example, there would still be one position still known. This would be a good hint of the approximate position of the joint, and to take advantage of that would greatly improve the joint estimations.

6

Conclusion and Future Work

This report have presented a study on animation in real-time from motion capture. Methods for localizing the joints of a defined skeleton model has been presented. The techniques for joint localization are based in biomechanics and computationally intensive algorithms have been avoided.

A, to our knowledge, new way of dealing with imperfect motion capture data using inverse kinematics with incorporated constraints have been presented and tested. IK is used to estimate joint positions that could not be located during the joint localization phase due to marker occlusion. Tests were performed on four different IK algorithms (FABRIK, CCD, Jacobian Transpose, and Damped Least Squares), in order to conclude which, if any, algorithm that performs the best in terms of accuracy and execution time.

The quickest and most accurate of the IK algorithms was CCD. The way CCD works is well suited for the constraints model used and the poses resulting from CCD was acceptable to really good when constraints were applied. The estimation of joint positions could improve if chains with multiple end-effectors are used as input to the IK solvers. Estimations can also be more accurate if information from previous frames are taken into account, or positions of not occluded markers close to the joint are used.

FABRIK performed the worst of the IK algorithms, when using constraints. Since constraints are necessary to obtain realistic poses, we do not recommend using FABRIK for estimating joint positions as described in this thesis. FABRIK does, however, show potential since it is very fast when not using constraints. Improving the constraint model, and the algorithms for ensuring that the constraints are kept, would be a possible future task to improve the overall performance of the system.

Also, to be acceptable, the system needs to provide smooth animation. An important fix would therefore be to implement the previously mentioned interpolation between frames in order to avoid uneven motion.

In conclusion, the different parts of the system work well, but smoother motion needs to be obtained in order to use it for animation. If multiple end-effectors are to be implemented, Jacobian methods have easier ways, than CCD, of handling this. However, it need to be further studied if these methods achieve acceptable execution times with multiple end-effectors. An alternative would be to improve the implementation of FABRIK, to better avoid deadlocks, since FABRIK in this case should be fast and has described methods for handling multiple end-effectors. The best IK algorithm to use for the proposed method in its current state is CCD.

Bibliography

[1] Qualisys, Capture the motion of anything (March 2015).
URL `http://www.qualisys.com/applications/`

[2] Qualisys, Biomechanics - motion capture for improved performance and research (March 2015).
URL `http://www.qualisys.com/applications/biomechanics/`

[3] Qualisys, Industrial - fast and precise tracking (March 2015).
URL `http://www.qualisys.com/applications/industrial/`

[4] A. Aristidou, J. Lasenby, Fabrik: a fast, iterative solver for the inverse kinematics problem, Graphical Models 73 (5) (2011) 243–260.

[5] G. Guerra-Filho, Optical motion capture: Theory and implementation., RITA 12 (2) (2005) 61–90.

[6] A. G. Kirk, J. F. O'Brien, D. A. Forsyth, Skeletal parameter estimation from optical motion capture data, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 2, IEEE, 2005, pp. 782–788.

[7] A. Aristidou, J. Lasenby, Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver, Tech. rep., Cambridge University Engineering Department (2009).

[8] M.-C. Silaghi, R. Plänkers, R. Boulic, P. Fua, D. Thalmann, Local and global skeleton fitting techniques for optical motion capture, in: Modelling and Motion Capture Techniques for Virtual Environments, Springer, 1998, pp. 26–40.

[9] V. B. Zordan, N. 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, Eurographics Association, 2003, pp. 245–250.

[10] V. Pratt, Direct least-squares fitting of algebraic surfaces, ACM SIGGRAPH computer graphics 21 (4) (1987) 145–152.

[11] A. Aristidou, J. Cameron, J. Lasenby, Real-time estimation of missing markers in human motion capture, in: Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on, IEEE, 2008, pp. 1343–1346.

[12] M. Fêdor, Application of inverse kinematics for skeleton manipulation in real-time, in: Proceedings of the 19th spring conference on Computer graphics, ACM, 2003, pp. 203–212.

[13] F. Multon, R. Kulpa, L. Hoyet, T. Komura, From motion capture to real-time character animation, in: Motion in Games, Springer, 2008, pp. 72–81.

[14] Qualisys, Qualisys motion capture software (April 2015).
URL `http://www.qualisys.com/products/software/qtm/`

[15] U. Technologies, Unity cross-platform game engine (2015).
URL `http://unity3d.com/`

[16] R. M. Ehrig, W. R. Taylor, G. N. Duda, M. O. Heller, A survey of formal methods for determining the centre of rotation of ball joints, Journal of biomechanics 39 (15) (2006) 2798–2809.

[17] J. Cameron, J. Lasenby, A real-time sequential algorithm for human joint localization, in: ACM SIGGRAPH 2005 Posters, ACM, 2005, p. 107.

[18] A. J. Stoddart, D. Ewins, D. Hynd, A computational method for hip joint centre location from optical markers (1999).

[19] S. S. H. U. Gamage, J. Lasenby, New least squares solutions for estimating the average centre of rotation and the axis of rotation, Journal of biomechanics 35 (1) (2002) 87–93.

[20] H. F. Gardner, O. S. Chief, A method for location of prosthetic and orthotic knee joints, Artificial limbs 13 (1969) 31–35.

[21] A. Campbell, D. G. Lloyd, J. Alderson, B. Elliott, Mri development and validation of two new predictive methods of glenohumeral joint centre location identification and comparison with established techniques, Journal of biomechanics 42 (10) (2009) 1527–1532.

[22] D. Lloyd, J. Alderson, B. Elliott, An upper limb kinematic model for the examination of cricket bowling: A case study of mutiah muralitharan, Journal of sports sciences 18 (12) (2000) 975–982.

[23] C. Meskers, F. Van der Helm, L. Rozendaal, P. Rozing, In vivo estimation of the glenohumeral joint rotation center from scapular bony landmarks by linear regression, Journal of biomechanics 31 (1) (1997) 93–96.

[24] V. Sholukha, S. V. S. Jan, O. Snoeck, P. Salvia, F. Moiseev, M. Rooze, Prediction of joint center location by customizable multiple regressions: application to clavicle, scapula and humerus, Journal of biomechanics 42 (3) (2009) 319–324.

[25] R. B. Davis, S. Ounpuu, D. Tyburski, J. R. Gage, A gait analysis data collection and reduction technique, Human Movement Science 10 (5) (1991) 575–587.

[26] M. Harrington, A. Zavatsky, S. Lawson, Z. Yuan, T. Theologis, Prediction of the hip joint centre in adults, children, and patients with cerebral palsy based on magnetic resonance imaging, Journal of biomechanics 40 (3) (2007) 595–602.

[27] A. L. Bell, D. R. Pedersen, R. A. Brand, A comparison of the accuracy of several hip center location prediction methods, Journal of biomechanics 23 (6) (1990) 617–621.

[28] S. R. Buss, Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods, IEEE Journal of Robotics and Automation 17 (2004) 1–19.

[29] A. Balestrino, G. De Maria, L. Sciavicco, Robust control of robotic manipulators, in: Proceedings of the 9th IFAC World Congress, Vol. 5, 1984, pp. 2435–2440.

[30] D. E. Whitney, Resolved motion rate control of manipulators and human prostheses., IEEE Transactions on man-machine systems.

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

[32] A. N. Pechev, Inverse kinematics without matrix inversion, in: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, IEEE, 2008, pp. 2005–2012.

[33] L.-C. Wang, C.-C. Chen, A combined optimization method for solving the inverse kinematics problems of mechanical manipulators, Robotics and Automation, IEEE Transactions on 7 (4) (1991) 489–499.

[34] D. Tolani, A. Goswami, N. I. Badler, Real-time inverse kinematics techniques for anthropomorphic limbs, Graphical models 62 (5) (2000) 353–388.

[35] E.-J. Ong, A. Hilton, Learnt inverse kinematics for animation synthesis, Graphical Models 68 (5) (2006) 472–483.

[36] K. Grochow, S. L. Martin, A. Hertzmann, Z. Popović, Style-based inverse kinematics, in: ACM Transactions on Graphics (TOG), Vol. 23, ACM, 2004, pp. 522–531.

[37] R. Sapra, M. Mathew, S. Majumder, A solution to inverse kinematics problem using the concept of sampling importance resampling, in: Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on, IEEE, 2014, pp. 471–477.

[38] C. Hecker, B. Raabe, R. W. Enslow, J. DeWeese, J. Maynard, K. van Prooijen, Real-time motion retargeting to highly varied user-created morphologies, in: ACM Transactions on Graphics (TOG), Vol. 27, ACM, 2008, p. 27.

[39] L. Unzueta, M. Peinado, R. Boulic, Á. Suescun, Full-body performance animation with sequential inverse kinematics, Graphical models 70 (5) (2008) 87–104.

[40] A. Aristidou, J. Lasenby, Real-time marker prediction and cor estimation in optical motion capture, The Visual Computer 29 (1) (2013) 7–26.

[41] J.-S. Monzani, P. Baerlocher, R. Boulic, D. Thalmann, Using an intermediate skeleton and inverse kinematics for motion retargeting, in: Computer Graphics Forum, Vol. 19, Wiley Online Library, 2000, pp. 11–19.

[42] M. Meredith, S. Maddock, Adapting motion capture data using weighted real-time inverse kinematics, Computers in Entertainment (CIE) 3 (1) (2005) 5–5.

[43] D. Eberly, Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid, Geometric Tools, LLC.

[44] M. Ringer, J. Lasenby, A procedure for automatically estimating model parameters in optical motion capture., in: BMVC, 2002, pp. 1–10.

[45] R. Tranberg, Analysis of body motions based on optical markers. accuracy, error analysis and clinical applications., Ph.D. thesis, University of Gothenburg. Sahlgrenska Academy (2010).

[46] M. S. Andersen, S. Mellon, G. Grammatopoulos, H. S. Gill, Evaluation of the accuracy of three popular regression equations for hip joint centre estimation using computerised tomography measurements for metal-on-metal hip resurfacing arthroplasty patients, Gait & posture 38 (4) (2013) 1044–1047.

[47] A. Aristidou, Y. Chrysanthou, J. Lasenby, Extending fabrik with model constraints, Computer Animation and Virtual Worlds.

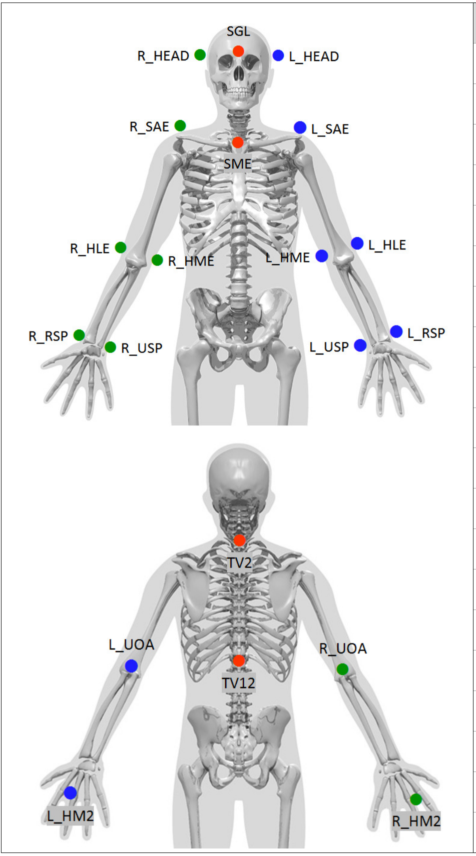
[48] A. Ben-Israel, T. N. Greville, Generalized inverses: theory and applications, Vol. 15, Springer Science & Business Media, 2003.

[49] S. R. Buss, J.-S. Kim, Selectively damped least squares for inverse kinematics, journal of graphics, gpu, and game tools 10 (3) (2005) 37–49.

A

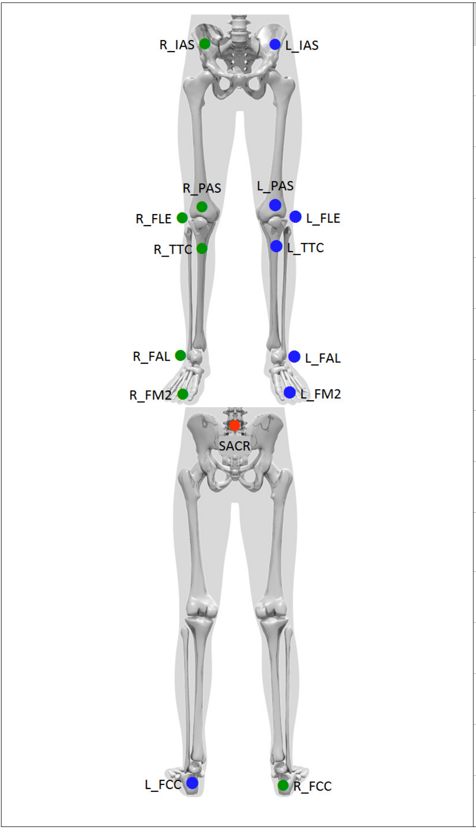
Appendix

Qualisys PAF Running Package Marker Set



Name	Ref. [1]	Location	Static (35)	Dyn. (35)
L_HEAD		On headband, just above ear	X	X
R_HEAD		On headband, just above ear	X	X
SGL	SGL	Skull - Glabella (Forehead)	X	X
SME	SME	Sternum – Manubriosternal Edge	X	X
TV2	TV2	Spinous Process of the 2nd Thoracic Vertebra	X	X
TV12	TV12	Spinous Process of the 12th Thoracic Vertebra	X	X
L_SAE	SAE	Scapula – Acromial Edge	X	X
L_HLE	HLE	Humerus – Lateral Epicondyle	X	X
L_UOA	UOA	Ulna – Olecranon (Apex)	X	X
L_HME	HME	Humerus – Medial Epicondyle	X	X
L_RSP	RSP	Radius – Styloid Process	X	X
L_USP	USP	Ulna – Styloid Process	X	X
L_HM2	HM2	Hand/Metacarpus – Head, Medial Aspect of the forefinger	X	X
R_SAE	SAE	Scapula – Acromial Edge	X	X
R_HLE	HLE	Humerus – Lateral Epicondyle	X	X
R_UOA	UOA	Ulna – Olecranon (Apex)	X	X
R_HME	HME	Humerus – Medial Epicondyle	X	X
R_RSP	RSP	Radius – Styloid Process	X	X
R_USP	USP	Ulna – Styloid Process	X	X
R_HM2	HM2	Hand/Metacarpus – Head, Medial aspect of the forefinger	X	X

Qualisys PAF Running Package Marker Set



Name	Ref. [1]	Location	Static (35)	Dyn. (35)
L_IAS	IAS	Ilium – Anterior Superior Iliac Spine	X	X
SACR	(IPS)	Sacrum – Midpoint between left and right Posterior Superior Iliac Spine	X	X
R_IAS	IAS	Ilium – Anterior Superior Iliac Spine	X	X
L_PAS		Along the central line of the patella at the rectus femoris tendon, 1 cm proximally of the superior border of the patella when the knee is extended	X	X
L_FLE	FLE	Femur - Lateral Epicondyle	X	X
L_TTC	TTC	Tibia – Tibial Tuberosity	X	X
L_FAL	FAL	Fibula – Apex of the Lateral Malleolus	X	X
L_FCC	FCC	Foot/Calcaneus – Aspect of the Achilles Tendon insertion	X	X
L_FM2	FM2	Foot/Metatarsus – 2nd head	X	X
R_PAS		Along the central line of the patella at the rectus femoris tendon, 1 cm proximally of the superior border of the patella when the knee is extended	X	X
R_FLE	FLE	Left Lateral Epicondyle	X	X
R_TTC	TTC	Tibia – Tibial Tuberosity	X	X
R_FAL	FAL	Fibula – Apex of the Lateral Malleolus	X	X
R_FCC	FCC	Foot/Calcaneus – Aspect of the Achilles Tendon insertion	X	X
R_FM2	FM2	Foot/Metatarsus – 2nd head	X	X

[1] Sint Jan, S. Van (2007). Color Atlas of Skeletal Landmark Definitions. Guidelines for Reproducible Manual and Virtual Palpations. Edinburgh : Churchill Livingstone.