Inverse Kinematics using Quaternions

Graduate project, 7.5 ECTS, Supervisor: Kenny Erleben

Henriksen, Knud; Erleben, Kenny; Engell-Nørregård, Morten

Publication date:
2008

Document version
Publisher's PDF, also known as Version of record

Inverse Kinematics The state of the art

Morten Engell-Nørregård *

December 19, 2007

Graduate project, 7.5 ECTS, Supervisor: Kenny Erleben.

Figure 1: St. Michael Vanquishing Satan. Raffaello Sanzio 1505 .Superimposed is a figure posed (approximately) like St. Michael, with the BFGS method. (*The photo used is released in the public domain at* http://en.wikipedia.org/wiki/Image:Sanmichele_satana_raffaello.jpg)

Department of Computer Science, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen East, Denmark

*mort@diku.dk

Contents

Abstract

In this project I describe the status of inverse kinematics research, with the focus firmly on the methods that solve the core problem. An overview of the different methods are presented
Three common methods used in inverse kinematics computation have been chosen as subject for closer inspection. The three methods are described in some detail. An analysis is performed where the three methods are compared, and benchmarked against each other.
I conclude which of the methods are the most promising and what their respective forces, weaknesses and prospects are. All figures shown are made using the test suite, developed in this project and in [4].
Source code developed for this project includes the CCD method , improvements on the BFGS method and Jacobian inverse originally developed in [4]. [1]

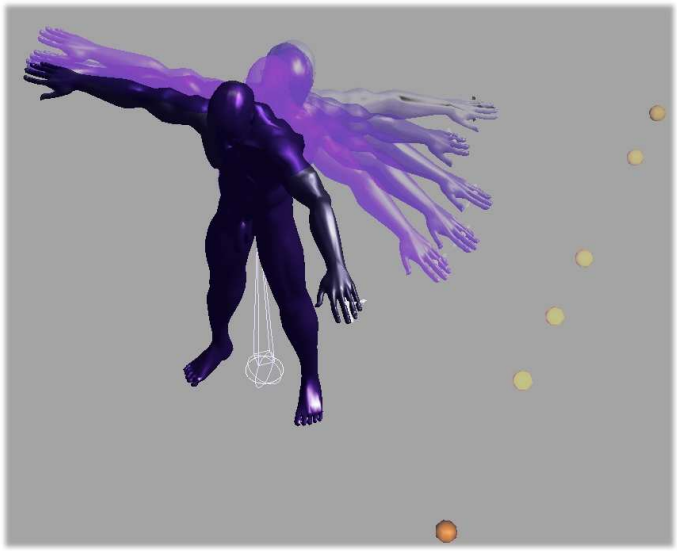

Figure 2: Figure reaching for a moving ball, posed with bfgs. The figure have been constrained to obtain a more realistic pose.

[1] The source can be found at http://www.opentissue.org/svn/OpenTissue/branches/IK/.

1 Introduction

According to [9] "*Kinematics is the study of the motion of rigid bodies without consideration of Newtonian laws*".
This is the broadest possible definition, and I will be using a much more narrow definition in the rest of this text. When i refer to kinematics in the following, i refer to algorithms for calculating the placement of skeletal joints with regard to a given base. I.e. given a skeleton structure, We want to be able to calculate the placement of the joints in the skeleton relative to the root of the skeleton. The easiest way to do this, is by forward kinematics. Forward kinematics computes the placement of an end-effector given the placement of all the joints from the root and out.
This is not a very intuitive way of posing a skeleton though, and since my object of study is interactive kinematics for animation purposes, it is quite vital that the interaction interface is intuitive. This leads to the development of inverse kinematics.
As the name implies inverse kinematics is the inverse of forward kinematics. Given the intended position and orientation of an end-effector we wish to compute the parameters of each of the joints.
This project is about inverse kinematics (IK).

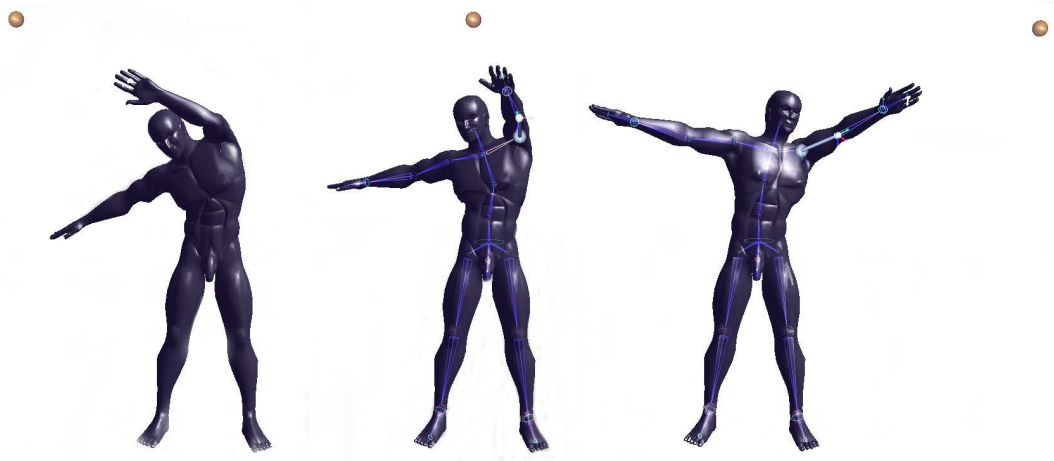

Figure 3: Figure reaching for a moving ball, posed with BFGS. The figure has been constrained preventing it from reaching the ball.

1.1 Motivation and goals

I was motivated to this project by a wish to obtain a broader knowledge and a deeper understanding of the methods used in IK. It was born in part from previous work done in the field by me, and others [4], partly with the intention of doing future work on the subject.

1.2 Overview of the project

This section describes the project, and gives an overview of the work done,as well as a walkthrough of the document.
To be able to make a comparison of the three methods, I needed an implementation of them all. This was only possible because two of the methods were made available in OpenTissue [1], by another project in which I participated simultaneously with this project [4].The Document is split in the following parts:

Section 1 Here I introduce the subject, and explain certain terms and notations special to this project. The foundation on which this work is build is described.

Section 2 explains in more details the methods that i have chosen to evaluate, to give the reader an understanding of the forces and weaknesses of the methods.

Section 3 is perhaps the most important in the project, since it is here i discuss and test the strengths and weaknesses of the methods.

Section 4 I conclude on the project and the results obtained in section 3.

Section 5 is a look into the future where I try to describe possible future work.

1.3 Notes on language and notation

Since this text deals with linked skeletons I will be using terminology from this area of research. The terminology is not coherent though and some terms are found in many variants. This section will try to clarify some of the terminology so as to prevent it from inhibiting the readers understanding.
The notation used in the formulas troughout the document is for the most part standard notation. However there are some peculiarities that can not be expected to be known, and some variables so widely used in the text, I have decided to list them here for reference.
Throughout the document I use J to mean the Jacobian matrix and J^+ to

mean its pseudo inverse. H means the Hessian B its approximation. Other notations and variables are explained in the context where necessary.

Constraint types. The position and orientation of a joint can be constrained by several factors. The physical limits of the joint,collisions with external objects, gravity or other external forces. All of these can be grouped in two groups, joint limits and global constraints, when i talk about constraints in this text it may refer to one or both.

Figure 4: E.R.I.K. The human model used throughout the document, posed with the BFGS method.

1.4 Background

IK has been the object of study for decades and a large and rich literature is available for the interested student of the problem.
Getting an overview of the problem however, is made difficult by the very fact that the subject is so well studied. Many texts that claim to deal with IK is in fact not IK but rather motion blending [3]. Motion blending is a fast and effective way of posing figures given a number of poses to blend between. It has little to do with IK though, and even though impressive results can be obtained this way [10, 3], the fact remains that a convex boundary on the possible poses are given by the example poses.
IK on the other hand computes the pose given one or more goal positions and possibly a number of constraints, which means that within the constraints, any pose can be computed. This comes at a cost in computation

though, and many solutions have been proposed, most of them based on optimization theory.
To make a shortcut I have chosen to start with someone who has done the same thing that I'm trying to, namely Martin Fedor [5]. His article is from 2003 however, and this text can be seen as an update on that work. As will become apparent later, the problem is still solved in much the same way as it was when Fedor wrote his article.
One quite recent article [8] proposes to solve the problem using linear programming. One problem pose itself when using this approach though. The simplex algorithm is used to solve the problem and even though it performs well in most cases, the worst case complexity of the algortihm is unviable in real time applications. This implies that an approach using an iterative nonlinear solver of some kind would be better. More research is necessary before this approach can be completely ignored though.
Martin Fedor [5] describes three methods for inverse kinematics, two of which are allmost kanonical in all literature concerning IK. this is the Jacobian Inverse and the CCD [2]. A third method, (the analytical approach) is also described, however it is tailored specifically to a certain skeleton type, which of course is a limit to its generality.
The CCD method is also described in [12] where a comparison with the jacobian transpose method is done.
The Jacobian inverse can be combined with constraints to make a constrained IKsolver . This is described in [13] which also give a thorough description of the CCD method, with constraints. The constrained Jacobian-inverse [3] is not implemented in [13].
In [15] a method for calculating constrained inverse kinematics are presented, which is widely cited and very interesting indeed. It uses a quasi-newton BFGS optimisation [11], combined with a constraint function that finds a Karusch-Kuhn-Tucker point [11]. The constraint method used is a gradient projection method mentioned in [4, 11].
Much of the recent research on IK has mostly concerned itself with how to augment the core solver, wtih different ways of making life easier for the artist. These subjects are beyond this project, even though they are part of the knowledge base. The interested reader is refered to e.g. [2, 6, 14].
On the basis of the presented, I have chosen to concentrate my attention on the following three methods: CCD as presented in[5], Jacobian Inverse[7] and BFGS [15, 11].

[2]Cyclic Coordinate Descent
[3]or as it is the Jacobian transpose

2 Theory

In the following I will shortly describe the methods I have chosen to concentrate on. I will briefly describe the theory and ideas behind them, and give a short desriptionof the pros and cons of each of them, based on their theoretical properties.

2.1 Chosen methods

I have chosen to focus on three methods for solving the IK problem. They are:

- Cyclic coordinate descent. A heuristic approach that takes the local greedy choice to solve the problem, one joint at a time. This method is elaborated in section 2.2
- Jacobian Inverse. A Newton based iterative optimization method that tries to solve the problem by iteratively updating the parameter vector x by the following equation

$$x_{i+1} = x_i + \Delta x_i; \tag{1}$$

  As can be seen from the above, this is a first order approximation of the updated parameter vector. This method is explained in more detail in section 2.3 where it is also explained how to find Δx.
- The BFGS method is an optimization method that tries to solve the problem as a minimization problem. the method used for solving this is similar in many ways to the above Jacobian inverse. The BFGS is a second order approximation to a scalar function. This fact makes the calculations very similar, since the gradient is a vector and the hessian is a matrix. The search direction is found using these two, in much the same way as the parameter vector and the Jacobian is used in Jacobian inverse. More details on this method is included in section 2.4

2.2 Cyclic Coordinate Descent

Cyclic coordinate descent, (CCD) is a heuristic approach that uses the greedy paradigm. For each joint we perform a transformation of that joint that brings the endeffector as close as possible to the goal. We then move to the next joint and do the same. This is done until a satisfactory solution is obtained.

2.2.1 The basic idea

in the following I consider revolute joints, but prismatic joints can be treated similarly.
Given a position of the current joint p, a position of the end effector e and a goal position g, we can construct two vectors $\overline{u} = \frac{e-p}{\|e-p\|}$ and $\overline{v} = \frac{g-p}{\|g-p\|}$. I can now compute the axis of rotation and the angle by the following

$$axis = \overline{u} \times \overline{v} \tag{2}$$

$$angle = \arcsin \|axis\| \tag{3}$$

Since arcsin isn't uniquely determined, it is possible to look at the dot product of the two vectors to obtain the cosine and thereby determine the angle uniquely. This implies that the use of arctan would be a better choice and specifically the function $arctan2$. For an explanation of $arctan2$ see [9]. Thus the angle is computed as

$$angle = \arctan 2 \left(\frac{\|axis\|}{\vec{u} \cdot \vec{v}} \right) \tag{4}$$

The next step is to perform the rotation on the joint in question. Since I am using a quaternion representation I need to construct the rotation quaternion. This is given by

$$Q = cos(angle/2), sin(angle/2) \left(\frac{axis}{\|axis\|} \right) \tag{5}$$

2.2.2 Incorporating constraints

Constraints can be incorporated in the Ccd method. Some constraints are easier than others though. Since the method is local it is difficult to implement global constraints such as equality constraints between several joints.

2.2.3 Pros and cons

The CCD method is simple and easy to implement. Furthermore the cost per iteration is quite low, which makes it an obvious first choice of method. The method is not without problems though. As will become apparent in the following the method has problems with the convergence properties.

2.3 Jacobian inverse

The Jacobian inverse method (JI)and its variant the jacobian transpose (JT), takes a global approach by trying to solve the problem for all joints simultaneously.

2.3.1 The basic idea

The basic idea is that we try to solve the equation:

$$y_1 = f(x_0 + \Delta x) \tag{6}$$

where y_1 is a desired endeffector position [4], x_0 is a current parameter vector and Δx is an unknown modifikation to x_0 which will satisfy y_1.
Knowing the desired goal and the current value of $x = x_0$, we can find the value of Δx. First we need to perform a Taylor expansion of the function $f(x_0 + \Delta x)$:

$$y_1 = f(x_0) + \frac{\partial f}{\partial x}(x_0)\Delta x + O\|\Delta x\|^2 \tag{7}$$

Since $f(x)$ is a vector function, $\frac{\partial f}{\partial x}$ is a matrix of first order partial derivatives. Such a matrix is usually denoted J so ignoring the remainder term we can shorten the expression to:

$$y_1 = f(x_0) + J(x_0)\Delta x \tag{8}$$

Since the solution to $f(x_0)$ is y_0 we can Isolate the unknown Δx:

$$\Delta x = J(x_0)^{-1}(y_1 - y_0) \tag{9}$$

The expression $(y_1 - y_0)$ is the desired solution minus the current solution which means that we now have an iterative scheme which can be used to find the correct configuration.

2.3.2 Incorporating constraints

It is not straight forward to implement constraints in the Jacobian method. [13, 4] uses a simple projection where the unconstrained solution is projected unto the feasible region, in the hope that it will still lie close to an optimal solution. In [5] a penalty based method is used with the adverse effect of damping the system, thus resulting in bad convergence.

2.3.3 Pros and cons

One great advantage of the Jacobian inverse is, that it minimizes the work done, in the sense that any given solution is a minimum change solution, compared to the previous iterate. This gives a more fluid motion without the erratic discontinuities of the CCD.
On the other hand Jacobian inverse has its own problems. The method requires the inversion of a matrix that might be singular, it might not even be square[5].

[4] or position and orientation
[5] actually it will usually be either over or underdetermined

The problem with non square Jacobians can be solved by using the pseudoinverse instead [7]. This does not take care of singularity though, so we still need to adress this.
Another problem is the fact that we dont have a general way of incorporating constraints. The problem is not bigger than it is for the CCD method, but it is still a problem.

2.4 BFGS

The problems with the ill conditioning of the jacobian matrix is avoiding in BFGS,where the solution is found as a second order approximation of a real function.

$$minG(\theta) \tag{10}$$
$$\text{subject to:} \quad a_i^T\theta = b_i, i = 1, 2, ..., l \tag{11}$$
$$a_i^T\theta <= b_i, i = l+1, l+, 2, ..., k \tag{12}$$

2.4.1 The basic idea

The method for solving this problem is a Quasi-newton method (see [11] for a more thorough explanation of this), where the search direction p_k^N is found by solving the equation:

$$p_k^N = -\nabla^2 f_k^{-1} \nabla f_k \tag{13}$$

where$\nabla^2 f_k$ is the hessian matrix and ∇f_k is the gradient of the objective function. This is given by

$$\nabla f_k = J(e - g) \tag{14}$$

where e is the end-effector position and g is the goal position.

Hessian approximation Calculating the exact Hessian is a time consuming business. Quasi Newton methods avoid this by working with an approximation of the Hessian. The main difference between Quasi Newton methods lie in the way they aproximate the Hessian. BFGS uses the following formula to obtain an estimate of the Hessian:

$$B_{i+1} = B_i - \frac{B_i s_i s_i^T B_i}{s_i^T B_i s_i} + \frac{g_i g_i^T}{g_i^T s_i} \tag{15}$$

where i denotes the present iteration, $s_i = x_{i+1} - x_i$, $g = \nabla f(x_{i+1}) - \nabla f(x)$, and B is an positive definite approximation of the Hessian
The Initial value of the Hessian is not uniquely defined. several different versions have been proposed such as $H = I$ or $H = \nabla f(x)^T \nabla f(x)$, both versions seem to work fine, but i have decided on the latter since it seemed the more precise approximation.

2.4.2 Incorporating constraints

Since the BFGS method is posed as a minimization problem it is straight forward to incorporate constraints and several possible methods present themselves. In [15] a gradient projection method is used, and it does seem the most obvious choice, since the IK problem has exactly the properties described in [11] to make it a candidate. These are that the constraints take the form:

$$l_i \leq x_i \leq u_i \tag{16}$$

where l_i and u_i are upper and lower bounds on individual DOFs.

2.4.3 Pros and cons

The BFGS method have all the advantages of the JI method and does not suffer from the singularity problems. It is also well described [11, 15] how to incorporate constraints. It is however complex and difficult to understand and implement, and has a high per iteration cost.

3 Validation

The following three methods have been chosen as the subject of my investigation and hence my validation.

1. The cyclic coordinate descent method
2. The Jacobian inverse
3. BFGS method

I am interested in the following properties of the different approaches:

1. Efficiency measured as rate of convergence and benchmarking of the time until convergence is obtained.
2. Scaling. How well do the methods scale (scale is measured in the complexity of the model i.e. number of joints.)
3. Precision measured as compliance with goal position

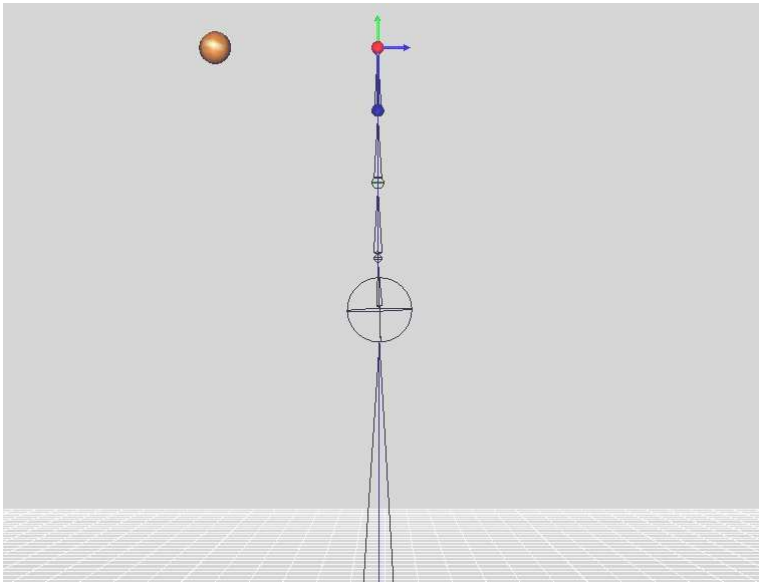

Figure 5: Test case for convergence. The chain consists of 5 joints and the goal is placed 25 units away from the goal, out of reach of the end effector.

3.1 Efficiency

There are two parts to measuring the efficiency of the different methods. First I will have to analyse the convergence properties of each of the methods.

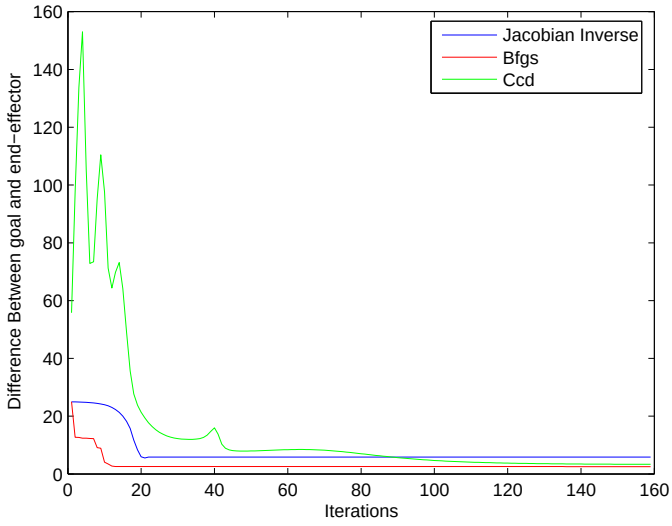

Figure 6: The convergence of the three methods measured in iterations

This is done as simple as possible by plotting the objective function value for each iteration as shown in figure 6. Second I will have to benchmark the three methods to be able to compare their actual performance.

3.2 The test suite

To test the methods I have made three different models, one very simple arm with one single chain skeleton with 5 joint (see figure 5), one slightly more complicated with 5 chains(see figure 7) and one human model with 32 joints and 78 DOFs. I have mainly used the two simple skeletons for this section, since they present the simplest results which lend itself well to this kind of analysis. Results was written to log files and plotted using simple matlab functions.

3.3 Tests

I have performed test both with regard to precision, scaling, and convergence. The results are presented in the following.

3.3.1 Convergence

In this test a 5 joint chain was used. The goal was placed 25 units from the end-effector and each of the three methods was allowed to run until they converged. The convergence criteria in this test was the relative one

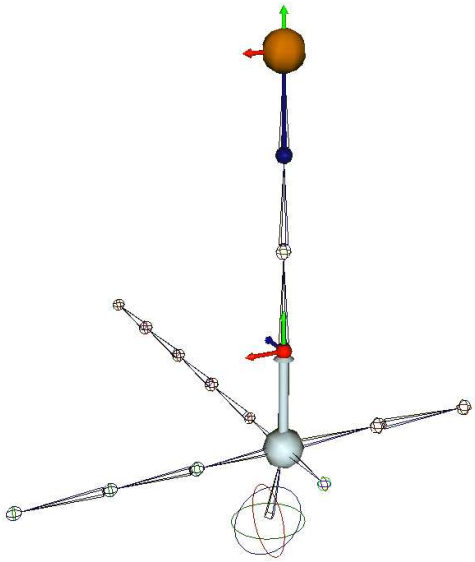

Figure 7: The model used in the benchmarking tests. It consists of 5 chains with 2,3,4,5 and 6 joints each. Notice that the first joint after the root is shared by all chains.

of testing change in the solutio compared to previous iteration. This was chosen because it gives a better picture of the real convergence properties of the methods. The reason for this is that the goal is out of reach so none of the methods could "get lucky" and find a solution that fullfilled the absolute stopping criteria. The test setup is shown in figure 5.

The results of the test is shown in figure 6. As can be seen from the figure, CCD shows erratic behaviour for the first many iterations and then it converges to a solution. The two other methods show nice monotonically decreasing behavior. The monotonically decreasing behavior is of great importance in real time applications where a fixed number of iterations is often used to ensure an upper limit on the time spent.

3.3.2 Scaling

For testing the scaling of the methods I used a special skeleton consisting of 5 chains with increasing length. The skeleton used can be seen in figure 7. Obviously I could have chosen to use longer chains but my experiments have shown that for the purposes of this discussion i would gain little from this approach.
I have performed two comparisons of the methods with regards to scaling.

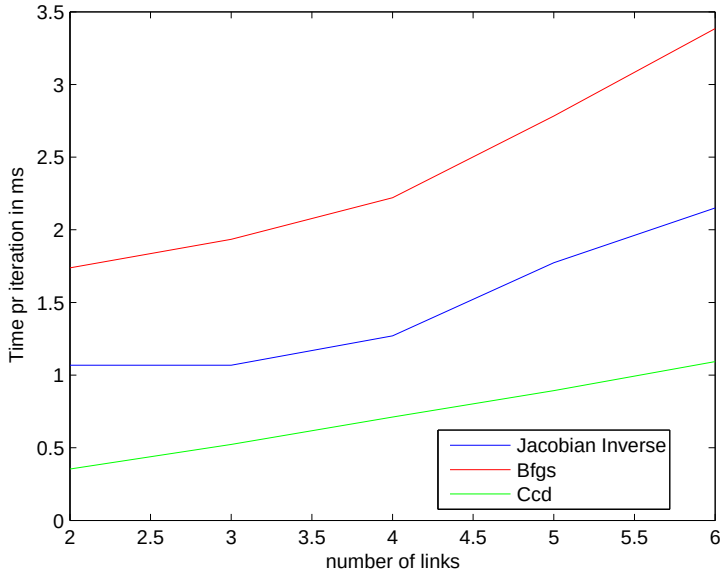

Figure 8: milliseconds pr. iteration for increasing number of joints

First I have compared the time per iteration when the number of joints grow. The results of this test can be seen in figure 8. Second I have tested the number of iterations each method has to use before they converge, as the number of chains grow, the result of this test can be seen in figure 9.

As can be seen from figure 8 both the BFGS and JI methods seem to have quadratic scaling here, whereas the CCD looks linear.
This fits well with what we know about the methods, since the JI must perform an inversion of a $n \times n$ matrix where n is the number of DOFs, and the BFGS method does something similar, with the Hessian matrix.
Properly implemented the BFGS method should avoid this behaviour though, so the results may be because of inacurate measurements or bugs in the implementation.

The time it takes to perform one iteration is not the only criterion. More important is the number of iterations, and how they scale, since this might just as easily dominate the result.
Figure 9 shows how the iteration count rises with the number of joints. As can be clearly seen here, the CCD method obtains the worst result. The JI seems to actually fall in iteration count. This is the result of the fail safes that are implemented to prevent divergence in the singular cases though, and not an inherent property of the method. Without these fail safes the

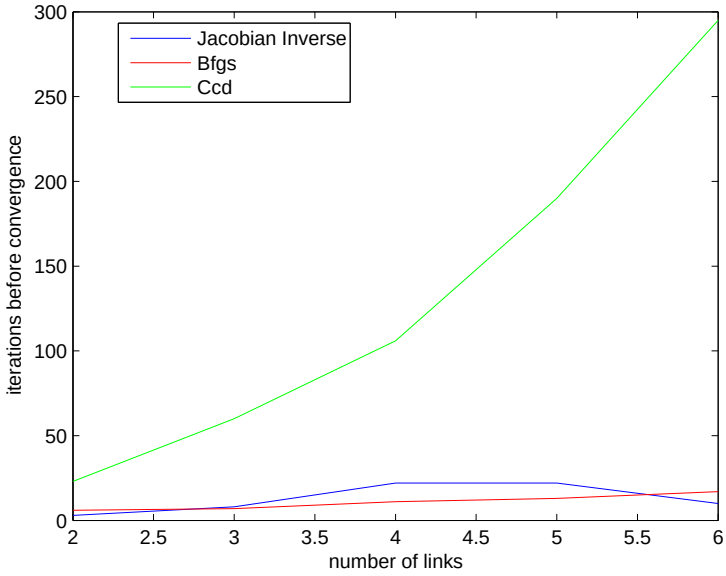

Figure 9: number of iterations as a function of chainlength

method shows properties similar to the BFGS method, with a sligthly higher iteration count. Unfortunately because of the risk of divergence it has not been possible to show the results of the unaugmented JI. Having measured both the time per iteration and the iteration count for different chains it is now easy to give an estimate on the scaling of the methods with regard to convergence time.

As it should be obvious from figure 10 it is the iteration count that dominates the convergence time. Again the JI seems to fall in the time it takes it to converge with 6 joints.

3.3.3 Precision

The precision has been tested by placing the end effector out of reach of the chain and measuring which method came closest to the optimal solution. The test case used for this test is similar to the one decribed in section 3.3.1 and an example is shown in figure 5. Several tests where run with different configurations and placement of the end effector. For simplicity I have only included one example. It is however a result which is general for the tests I have done.

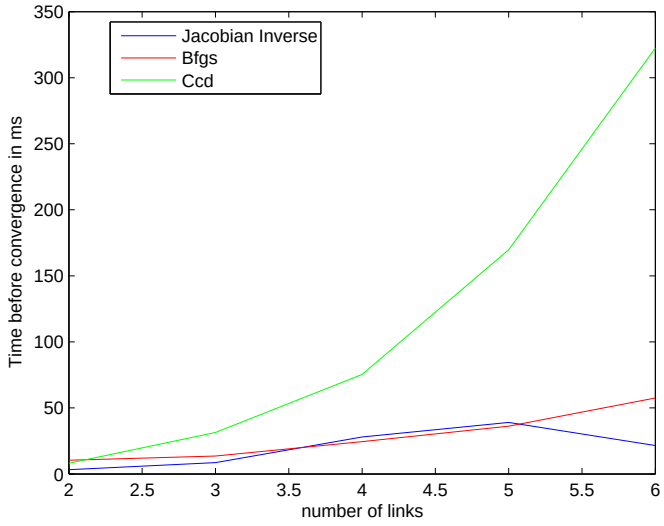

Figure 10: The convergence of the three methods measured in ms

method:	distance to goal
JI	5.853667
CCD	3.368891
BFGS	2.534875

These results are in fact the same as can be read from figure 6, but I have repeated them here for convenience since it may be difficult to read from the figure. As it can be seen, the BFGS method comes closest to reaching the goal. In fact it is very near to the optimal possible solution which lies at approximately 2. The Jacobian inverse obtains the worst result in this test, mainly because the method has a relative stopping criteria to prevent it from diverging near singular configurations. The CCD method could be expected to get closest given enough time, but tests have shown that more than 1000 iterations would be needed to obtain this solution, making it intractable in this context.

4 Conclusion

The main purpose of this project was to obtain knowledge of and compare a number of methods used in IK today. i have chosen three methods and implemented/improved them. I have compared the methods with regard to precision, convergence, and scaling, and found strength and weaknesses in them all.

CCD is simple to implement, has low per iteration cost, and is intuitive to understand. It suffers from several drawbacks though. First and foremost is the fact that it is a local method, and like most local methods, it oscillates its way to a solution. This means that the method does not have a monotously decreasing objective function, which may well disqualify it in the face of a ceiling on the iteration count. For practical purposes CCD seems to work fine and many implementations use it.
The JI method seems to have excellent convergence properties, acceptable iteration time and tolerable precision. The main drawback is the problems with singularities, which have led to me having to implement, certain failsafes to prevent divergence. Since methods exist that can cope with these problems e.g. singular value decomposition, coupled with , the method is a sound candidate for an IK solver.
BFGS seems to be the allround most effective algorithm. It is somewhat expensive per iteration, but has great convergence and no problems with divergence near singularities. It is also the method best suited for incorporating constraints, as described in [11, 15, 4].
All of the methods are at a proof of concept stage which means they have not been optimized. much performance can be gained from trimming the implementations. This has been beyond the scope of this project though.

5 Future Work

Having worked with IK intensively for 3 month now, I have found that the subject is both very interesting and very large. I therefore have numerous ideas for future projects and one of them I have already started work on.
The most obvious candidate is the incorporation of "real" constraints in the developed system. Allthough the simple constraints implemented seems to work fine, there are both performance and precision issues that would imply that a more theoretically sound system should be developed.
The interaction with the system could be improved considerably. For example by adding visualisation of the reach cones of the joints, and even more interesting, by implementing reach cones like the ones mentioned in [14].
Next is the robustness of the Jacobian inverse, which have nothing that handles the possibly singular inverted Jacobian matrix. Several solutions exist, the most obvious of which is to do a Singular Value Decomposition of the matrix to test for singularity. Other faster ways exist though, and will have to be adressed in the future.
The last possibility but by far not the least, is the comparison of performance between quaternion implementations and matrix implementation. This is a project which I not only wish to do, but in fact have already begun preliminary work on.

References

[1] , *OpenTissue*, OpenTissue Library, http://www.opentissue.org/ (2007).

[2] K. G. Der, R. W. Sumner, and J. Popovic;, Inverse kinematics for reduced deformable models, *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM Press, New York, NY, USA (2006), 1174–1179.

[3] J. Edsall, *Animation Blending: Achieving Inverse Kinematics and more*, gamasutra.com (2003).

[4] M. P. Engell-Nørregård, S. M. Niebe, and M. B. Bonding, Inverse kinematics with constraints, *Department of Computer Science, University of Copenhagen* (2007). graduate project.

[5] M. Fědor, Application of inverse kinematics for skeleton manipulation in real-time, *SCCG '03: Proceedings of the 19th spring conference on Computer graphics*, ACM Press, New York, NY, USA (2003), 203–212.

[6] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popovic;, Style-based inverse kinematics, *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM Press, New York, NY, USA (2004), 522–531.

[7] K. Henriksen and K. Erleben, Inverse kinematics using quaternions (2007). unpublished.

[8] E. S. L. Ho, T. Komura, and R. W. H. Lau, Computing inverse kinematics with linear programming, *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, New York, NY, USA (2005), 163–166.

[9] Kenny Erleben, Jon Sporring, Knud Henriksen and Henrik Dohlmann, *Physics based animation*, Charles River Media (2005).

[10] L. Kovar and M. Gleicher, Flexible automatic motion blending with registration curves, *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003), 214–224.

[11] Nocedal, J. and Wright, S.J., *Numerical Optimization*, Springer (1999).

[12] J. Vigsted and M. Parm, Invers kinematik med denavit-hartenberg notation, *Department of Computer Science, University of Copenhagen* (2007). student project.

[13] C. Welman, Inverse Kinematics and geometric constraints for articulated figure manipulation, M. Sc. Thesis, SIMON FRASER UNIVERSITY (1993).

[14] J. Wilhelms and A. V. Gelder, Fast and easy reach-cone joint limits, *J. Graph. Tools* **6**, 2 (2001), 27–41.

[15] J. Zhao and N. I. Badler, Inverse kinematics positioning using nonlinear programming for highly articulated figures, *ACM Trans. Graph.* **13**, 4 (1994), 313–336.