FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

PhD thesis

Morten Pol Engell-Nørregård

Interactive Modelling and Simulation of Human Motion

Academic advisors:

Kenny Erleben

Kim Steenstrup Pedersen

Submitted: june/01/2012

I’ve got it!
But it only works for spherical chickens in a vacuum.

Prologue

Why would I start something as serious as my PhD thesis with the punchline from a joke. Well first because it succinctly describes one of the major difficulties of physical modelling, the need for simplification. Secondly but equally important because, as the danish poet Piet Hein put it:

> Den, som kun tar spøg for spøg
> og alvor kun alvorligt,
> han og hun har faktisk fattet
> begge dele dårligt.

Which can be translated as

> Taking fun as simply fun
> and earnestness in earnest
> shows how thoroughly thou none
> of the two discernest.

For me, what this means is that as important as it may be to acquire new knowledge, the motivation and drive will always be the enjoyment of the process. I am glad to say, that even though it may have been hard at times, I have thoroughly enjoyed myself in the 3 years I have spent as a PhD student at DIKU. The opportunity to pursue the goals that I set up myself is something that makes me feel very privileged.
Returning to the Punchline, I managed to find the time to see several episodes of the series Big bang theory which tells the story of a bunch of friends who also happens to be scientists. One episode contains a joke, told by one of the characters, about a farmer who has a problem with sick chickens.
The Farmer asks a friend who is a physicist to help him and the friend starts scribbling maniacally in his notebook until finally he exclaims "I've got it! But it only works for spherical chickens in a vacuum."
This exemplifies the other part of the verse above. Even though something is Funny it can contain valuable information. In this case, the people who usually find this joke hilarious are people working in science, since they get the more subtle implication that, when modelling real world phenomena we are always forced to simplify, and the choice of simplification is a very important one indeed. My work has been full of situations where, I have had to choose which part of the real human motion I wanted to include in my models, and which to exclude. In this process, the applicability of the

end result has always been important to me. Whether I have had Graphics applications or bio-mechanical models in mind.
Therefore I hope, when you have read this text, you will agree with me that this work is not only suitable for Spherical chickens in a vacuum.

Acknowledgement

When I started at the university some years ago I came from a background as an artisan, a goldsmith to be concise. My highschool math was all but forgotten and my programming experience amounted to my experience as a kid, copying basic programs from books unto my commodore 64. I owe great thanks to the many people who, have helped me attain the knowledge and skills, which made it possible for me to get this far. Also all the people who have supported me on a more personal level.
I want to start by thanking my many mentors. Kenny Erleben, my primary supervisor for always sharing the enthusiasm for my weird ideas and always beating me to it, with even wilder ideas. Always helping me when the math made my head hurt, and making my head hurt with math when it didn't. Being not only a good supervisor but a good friend.
Kim Steenstrup Pedersen my co supervisor for teaching me that there are other research fields and other ways of solving the problem, than the physically based simulations. Knud Henriksen for initially sparking my interest for graphics, and Mathias Teschner for taking good care of me during my stay in Freiburg.

I would also like to take the opportunity to thank all my colleagues at the e-science institute. They are legion and I could not possibly mention them all but I must mention at least Søren Hauberg and Sarah Niebe with whom I have wasted more time than I care to think about, exchanging irrelevant information or plain out gibberish. All my office mates have partaken in this (whether they liked it or not) but none more than these two.

During my studies I have used several open-source tools to help me visualize my work. I would like to acknowledge the great work of the teams behind: Gimp, Inkscape and most of all the Blender team. You are doing a great job out there.

Even Though it may not always seem like it, there is a life outside the PhD project. Without the love and support of my family and friends I would have never had the energy to get through this process. In particular I want to thank my wife and kids who bore the brunt of my absence and absent mindedness, when the going got tough.

Finally I want to thank the committee for taking time from their busy schedules to read and give feedback to my work

Thank you all!

Notation

Here follows a short, incomplete list of the notation used in this Thesis. Most of the notation is explained as it is introduced but I felt it was justified to mention some of the most common notations which may be used differently elsewhere.

$\mathbf{x}$ is the vector x
$\mathbf{M}$ is the matrix M
$|| * ||$ is the euclidean norm
∇ is used to denote the gradient usually in the form ∇f
$\mathcal{L}$ is the Lagrangian function
$\mathbf{J}$ denotes the Jacobian matrix
$\mathbf{H}$ denotes the Hessian matrix
$\dot{\mathbf{x}}$ In relation to governing equations of motion, we sometimes use the physicist way of writing the time derivative.

A note on visual content

With a few exceptions all figures and 3d models are made by me in the course of my studies. I have used Blender [1] for most of the 3d work and inkscape and gimp for 2D images. This was done mainly to have access to content without having to worry about licensing. I have however used a few things that are not my own work. Of note, is the lizard used in the spline activation (see Figure 27) which I found on the web. It is made by Kevin Hayes and kindly released under a creative commons license . Also the female in Figure 1 is a mesh from the Ogre framework [2] In relation to my publications I have made media content in the form of videoclips. Since My topic is Motion I think some of the arguments are best described using these videos. They are available at http://iphys.wordpress.com/.

Summary

This PhD thesis concerns itself with modelling and simulation of human motion. The research subjects in this thesis has at least two things in common.

First they are concerned with Human Motion. Even though the models may be used for other things as well, the main focus is on modelling the human body.

Second they are all concerned with *simulation* as a tool to synthesize motion and thus, get *animations*. This is an important point since it means we are not only creating tools for animators to make fun and interesting animations, but also models for simulation of realistic motion. As the project progressed the focus has shifted from purely graphics oriented, to something which may be at least as interesting for the biomechanics community.

Scientific contributions

The main scientific contributions of this work are:

- An efficient method for solving interactive constrained inverse kinematics problems , using an optimization based approach. The method is usable for motion planning of complex articulated mechanisms, with a large degree of interdependency between different parts, such as a human body.

- A general and fast joint constraint model. The joint constraint model is well suited for modelling joints with highly non-convex limits and multiple degrees of freedom. Even though this constraint model may have many other uses we believe it is very well suited for the modelling of human joints which exhibit both non-convexity and multiple degrees of freedom

- A general and versatile model for activating soft bodies. The model may be used as an animation tool but would be equally well suited for simulation of human muscles since it adheres to basic physical principles. Further, it can be used with any softbody simulation method such as finite elements or mass spring systems.

- A control method for deformable bodies based on the space time optimization. the method may be used to control the contraction of muscles in a muscle simulation.

Dansk resumé

Denne ph.d.-afhandling beskæftiger sig med modellering og simulation af menneskelig bevægelse. Emnerne i denne afhandling har mindst to ting til fælles.

For det første beskæftiger de sig med menneskelig bevægelse. Selv om de udviklede modeller også kan benyttes til andre ting,er det primære fokus på at modellere den menneskelige krop.

For det andet, beskæftiger de sig alle med *simulering* som et redskab til at syntetisere bevægelse og dermed skabe *animationer*. Dette er en vigtigt pointe, da det betyder, at vi ikke kun skaber værktøjer til animatorer, som de kan bruge til at lave sjove og spændende animationer, men også modeller til simulering af realistiske bevægelser. I løbet af projektet er fokus flyttet fra at være rent grafik orienteret , til noget som er mindst lige så interessant for biomekanikere.

Videnskabelige bidrag

De vigtigste videnskabelige bidrag af dette arbejde er:

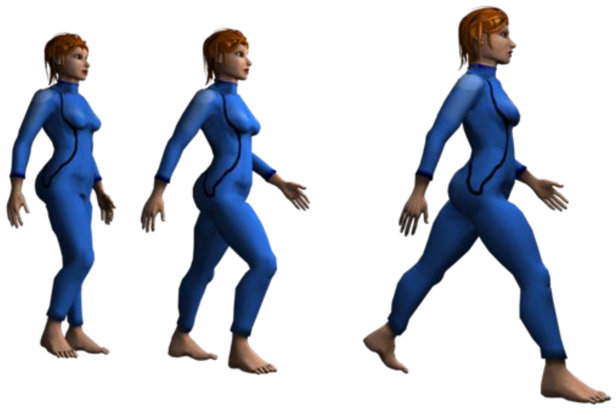
- En effektiv metode til at løse interaktive, inverse kinematik problemer med ledbegrænsninger, ved hjælp af en optimerings tilgang. Metoden er anvendelig til bevægelses planlægning for komplekse ledmekanismer, med en høj grad af indbyrdes afhængighed mellem de forskellige dele, såsom et menneskes krop.

- En generel og hurtig led-begrænsning model. Denne led-begrænsning model er velegnet til modellering af led med særdeles ikke-konvekse grænser og flere frihedsgrader. Selv om denne led-begrænsnings model kan have mange andre anvendelser,mener vi, den er meget velegnet til modellering af menneskers led, der udviser både ikke-konveksitet og flere frihedsgrader

- En generel og alsidig model for aktivering af bløde legemer. Modellen kan anvendes som et animations værktøj, men er lige så velegnet til simulering af menneskelige muskler, da den opfylder de grundlæggende fysiske principper. Endvidere kan den anvendes med enhver softbody simuleringsmodel som finite elements eller mass spring systemer.

- En kontrol metode til deformerbare legemer baseret på rum tids optimering. fremgangsmåden kan anvendes til at styre sammentrækning af muskler i en muskel simulering.

Contents



Introduction

This Thesis describes the work I have done over the past 3 years on Modelling and simulating human motion. With such a broad area it is necessary to impose some restrictions on the subject. I have chosen to describe human motion as two steps, the planning phase and the execution phase.

In the planning phase, the wanted motion is chosen from the space of possible motions. Essentially this means that a number of goals are converted into a pose. A simple example is taking a step. If a person wants to take a step, we can pose the problem as moving one foot from position a to position b while clearing the ground

This way of deciding a motion based on position and/or orientation of a subset of the joints is an intuitive and easy way of specifying motion which appeals to the way humans think about motion. This can be seen by the fact that this is the way animators and robot controllers work.

In the execution phase the actual forward dynamics simulation of the planned motion should be performed. here I have chosen to focus on the motor of human motion namely the Muscles. I have done this through the development of a deformable spline activation model suitable for simulation of simple muscles and with the added advantage of the passive deformable mesh which is of interest both from a visualization point of view and for collision detection the strength of the method is its ability to be combined with the deformable mesh of choice for a given simulation .

Finally I have recently started exploring the possibilities for using the motion planning to control the simulation that is use the poses calculated using the inverse kinematics and joint constraints to drive the motion of the dynamically simulated mesh. This is still work in progress but I have chosen to include it since it represents the connection between the two other parts of the project and since it is also the most recent research activity I have undertaken.

The Research presented in this thesis is to a large extend identical to the results presented in my published papers, but I have taken the opportunity to elaborate on some of the subjects which had to be left out or shortened in the publications due to page restrictions. I have rewritten and reformatted the papers to fit into the context of this thesis.That is, I present the research as one text combined from the papers, but as much as possible, I have kept the original thoughts and arguments unchanged.

The Papers, on which this thesis is based, all have co-authors, and I felt that it would be most honest to keep the plural form in this text.

Part I

Inverse Kinematics

In the first part of my PhD project I worked with optimization methods in inverse kinematics. This resulted in the following publications:

- 'A Projected Back-tracking Line-search for Constrained Interactive Inverse Kinematics' [3] .
  Published in Computers and Graphics. Together with Kenny Erleben.

- 'Interactive inverse kinematics for human motion estimation' published at Vriphys 09: 6th Workshop on Virtual reality Interactions and Physical Simulations [4] .
  This work was done together with Søren Hauberg ,Jerome Lapuyade , Kenny Erleben and Kim Steenstrup Pedersen.

- Three Dimensional Monocular Human Motion Analysis in End-Effector Space.
  Published in Lecture Notes in Computer Science.
  This work was done together with Søren Hauberg ,Jerome Lapuyade , Kenny Erleben and Kim Steenstrup Pedersen.

The following is a description of this work, based on [3]. with a description of the results of the application described in [4].

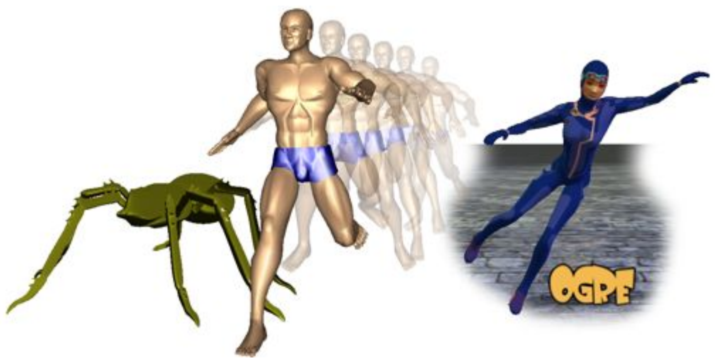


Figure 1: Inverse kinematics solved figures, the method is general and poses any conceivable figure. The developed method was integrated in an Ogre demo.

1 Constrained Inverse Kinematics

Inverse kinematics is the problem of manipulating the pose of an articulated figure in order to achieve a desired goal disregarding inertia and forces. One can approach the problem as a non-linear optimization problem or as non-linear equation solving. The former approach is superior in its generality and ability to generate realistic poses, whereas the latter approach is recognized for its low iteration cost. Therefore, many prefer equation solving over optimization for interactive applications. In the following, we present a projected gradient method for solving the inverse kinematics problem interactively, which exhibit good performance and precision. The method is compared to existing work in terms of visual quality and accuracy. Our method shows good convergence properties and deals with joint constraints in a simple and elegant manner. Our main contribution lies in the explicit incorporation of joint limits in an interactive solver. This makes it possible to compute the pose in each frame without the discontinuities exhibited by existing key frame animation techniques. In this section we have limited ourselves to using a box-limit joint constraint model. This is done because the focus in this section is the Inverse kinematics solver and not the joint limits. The method works with any projection based joint limit method as we will demonstrate later.

Inverse kinematics is used for a wide range of applications such as robot simulation or motion planning, creation of digital content for movies or commercials, or for synthesizing motion in interactive applications such as computer games and other types of virtual worlds. In Figure 1 I have illustrated animations created with our own interactive inverse kinematics method.

Inverse kinematics is a standard tool in many applications like Maya, 3D Studio Max [5] or Blender [1]. Recently inverse kinematics have been employed as a dimensionality reduction tool in a tracker of human motion [6]. In short, it is a well-known and wide-spread technique, and better numerical methods will therefore be valuable to a large community. A generally applicable method should be easy to use and thus minimize the number and complexity of user-defined parameters, while giving as realistic a pose as possible. Furthermore speed is essential, whether the method is used interactively to pose a figure, or to simulate movement in a virtual world. Most applications have chosen one of two avenues. Either, they are specialized closed form solutions for specific low-dimensional manipulators, the approach often taken in Robotics, or they are general type methods. Even though inverse kinematics has been around for quite some time, there seems to be very little work done in exploring methods which can bridge the gap between the two extremes, perhaps because the animation industry has not felt the need for further improving their methods, and the design of robots have followed the same line of thought. However, with the development of more and more humanoid robots and the move towards more physics based animations in media, the need for interactive general purpose inverse kinematics methods is again topical. Our focus is on the underlying method of solving the inverse kinematics problem. This can be extended to handle more user control, but that is not our interest.

We focus on a general, interactive method which includes joint limits. To state the problem more formally: Given a serial mechanism, we can set up a coordinate transformation from one joint frame to the next. Thus, we can find one transformation that takes a point specified in the frame of the end-effector into the root frame of the mechanism. We write it in general as

$$\mathbf{y} = \mathbf{F}(\theta). \tag{1}$$

We can change the values of the joint parameter θ and gain explicit control over the position and orientation of the end-effector, $\mathbf{y}$. This is commonly known as forward kinematics. Given a desired goal position, g, one seeks the value of θ such that

$$\theta = \mathbf{F}^{-1}(\mathbf{g}). \tag{2}$$

This is known as inverse kinematics and it is the problem we address in this part of the thesis.

1.1 Related Work

Inverse kinematics was introduced in robotics [7] and to the graphics community early on [8]. In robotics, the problem is usually phrased as a dynamic system which may lead to different schemes [9]. An overview of numerical methods used in computer graphics can be found in [10].

Inverse Kinematics methods In [11] a new mathematical formalism is presented for solving inverse kinematics as a constrained non-linear optimization problem. This work allows for general types of constraints. Inverse kinematics is known to suffer from problems with redundancies and singularities [12]. In robotics, redundancy problems have been addressed by adding more constraints [13]. In animation the redundancy is most often handled by using the spatial temporal coherency of consecutive solutions. Thus, the correct solution closest to the previous solution is chosen. This is also true in our system (see e.g. Figure 2).

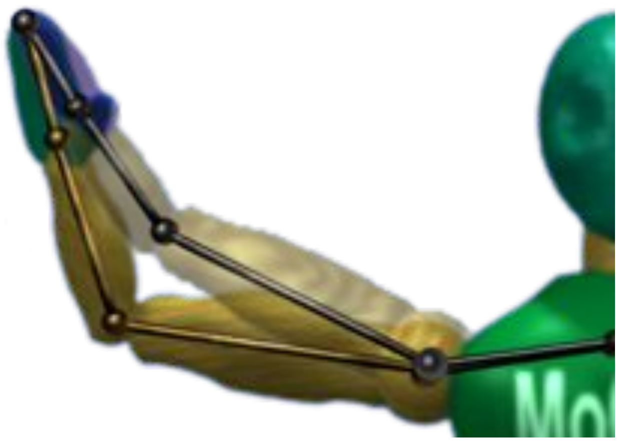


Figure 2: An example of a mathematically correct solution which deviates from the reference due to redundancy. The Ghosted overlay with the grey skeleton is the projected line-search inverse kinematics solution. Notice the large difference on the wrist and elbow and the small but noticeable difference on the shoulder joint. A positional goal was used in this example.

In [14], three methods are revisited and evaluated for computer game usage: an algebraic method, Cyclic-Coordinate-Descent [15], and a Newton-Raphson method inspired by [11]. The Newton-based method is used for complex manipulation and claimed to give the most realistic looking poses, but it is the slowest. However, joint limits were not dealt with in this work. Other formulations have been investigated for instance in [16] the problem is solved using linear programming. In [17] a mesh based inverse kinematics method was presented. This relies on example poses to manipulate the mesh directly and does not handle joint limits. Furthermore the mesh based approach runs with only 1 frame per second for even moderately sized meshes.

In robotics closed-form methods are popular [18]. Closed-form methods often result in algebraic systems that can be solved very fast and reliably, but these methods are highly specialized for a specific low-dimensional manipulator (up to 7 degrees of freedom).

In Computer Graphics which is our main focus in this work, even a single arm will usually have more than 7 degrees of freedom. In the simplified male human skeleton from Figure 1 and 3 this is 11. The shoulder has 5 degrees of freedom (2 in the collar bone 3 in the shoulder joint) 1 in the elbow , 3 in the wrist and 2 in the hand. Thus, general purpose methods capable of dealing with many degrees of freedom are desired.

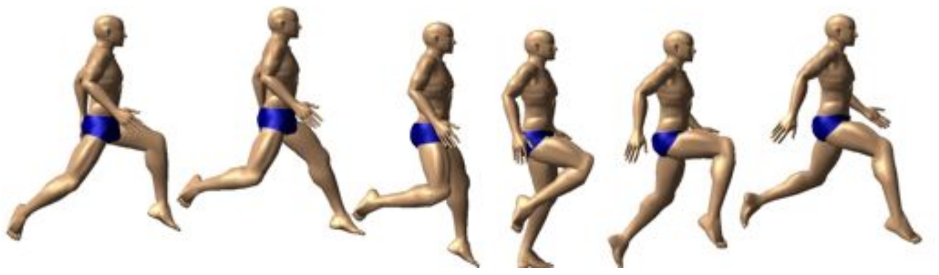


Figure 3: A running animation made using the projected line-search optimization approach. The presented method supports interactive editing of animated characters.

In [19], inverse kinematics is combined with other techniques and a sequential quadratic programming problem is solved. The running times are in minutes which prohibits interactive usage. Motion synthesis using space-time optimization and machine learning has also been tried [20, 21] although running times are not yet within the grasp of the real-time domain. An example of a widely used general method is the Blender Software [1] which uses a Jacobian Inverse scheme with the pseudo inverse being computed using SVD. To sum up, there is still a need for fast general purpose methods for posing characters with direct manipulation.

Joint Limits Often constraints such as joint limits are omitted [14] or dealt with as a post-processing step [15]. The added value of handling joint limits are shown clearly in Figure 5. Several approaches for handling joint limits has been proposed. Joint sinus curves was used in [22] to describe the boundaries of the feasible motion space of the joints. If a joint exceeds its boundary then it is projected back to the boundary and kept fixed at the boundary until the goal position is changed. In [23], quaternion boundary fields were created, and a bisection algorithm was used to back-project infeasible joint positions onto the closest-point on the quaternion boundary field. In [24] a back-projection is used after the joint-parameters have been updated. In [25], joint reach cones are introduced and later refined in [26] to handle moving center of rotations. Here back-projection of infeasible joints is also used. In [11] they keep track of currently active joint limits and modify their scheme in such way that joint limits will not be violated. An example of a method currently used in Commodity software, is the Jacobian Inverse method used in the Blender software, which uses a projection to move the solution unto a feasible region. This projection is performed as a separate step after an unconstrained solution has been found. It has not been possible to get information regarding the methods used in other major 3D systems, but their performance and quality are comparable. Thus, Blender has been chosen for the explicit comparisons in this work.

1.2 Contribution

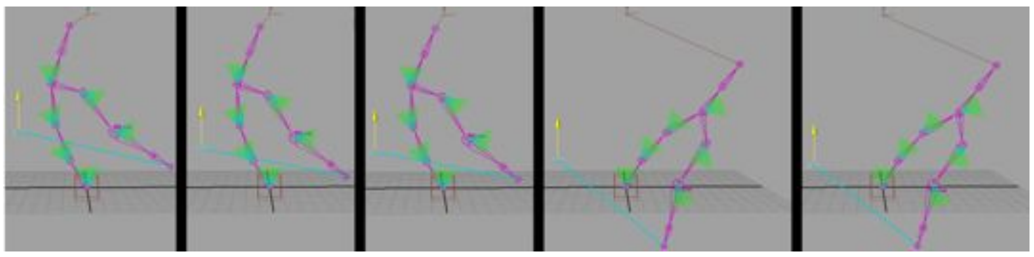


Figure 4: The Inverse kinematics system in Maya exhibiting the typical flipping behavior of systems which project the joint limits after solving the system. The skeleton has 16 degrees of freedom in total and 2 positional end effector goals. Joint limits are shown as green cones. One goal (shown by yellow arrow) was moved vertically to force the system to move along the joint limits. The flip is clearly seen from the third to the fourth picture.

In previous work, limits are dealt with as a post-processing step that simply back-projects infeasible iterates to a feasible iterate or an active set approach is used. These approaches either disregard a sufficient improvement in the solution or result in computationally expensive book-keeping. Our major contribution is a line-search method that guarantees an improvement of the solution, and robust handling of joint limits.

Our experience and the literature seem to indicate that constrained non-linear optimization is not the favourite choice for interactive applications. This is a shame since such a formalism offers more realistic motion and general constraints. This has motivated our work. We believe the task lies in creating a simple and elegant numerical method well suited for the purpose of inverse kinematics and which is easily implemented by programmers in industry.

In this work, we will present and evaluate a numerical approach for solving the interactive inverse kinematics problem as a constrained non-linear optimization problem. We will demonstrate how our numerical approach can be used interactively and can deal with joint limits. Our method does not exhibit the flipping behavior of methods which solve the unconstrained inverse kinematics problem before projecting the solution unto the feasible set. It avoids this without having to resort to such pre or post processing as e.g. the preferred pose of Maya. in fig 4 an example of the unwanted flipping behavior exhibited in Maya for a constrained solution with two positional goals and joint limits are shown.
Our focus has not been on developing a finished interactive animation suite but rather to present a method which can be used by others in their systems. We have done this by making all code available in the OpenTissue library [27]. From this the implementation details can be seen and the method may be included in any project.

Explicit comparison is performed with the inverse kinematics solver found in the Blender software package. Blender has been chosen because it is fully comparable in functionality with the major 3D animation softwares Maya and 3D Max. Furthermore, Blender is open–source. Thus, we could compare the timings of the IK-solvers directly without e.g. render-time disturbing the measurements.

2 The Traditional Approach

In the following we will describe the traditional approach for solving the inverse kinematics problem.

In the context of human modelling, a skeleton is often modelled as a collection of rigid bodies connected by rotational joints of 1–3 degrees of freedom. An example is shown in Figure 6. All joints are constrained in

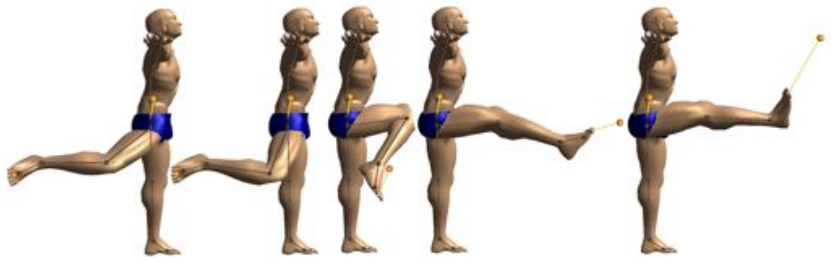


Figure 5: Joint limits shown by posing a leg in its extreme positions using positional goals. The goal position of the end-effector is shown as a small orange ball. Notice in pose 3 that the goal would be reachable if no joint limits were present, and that both pose 4 and 5 would be different without joint limits.

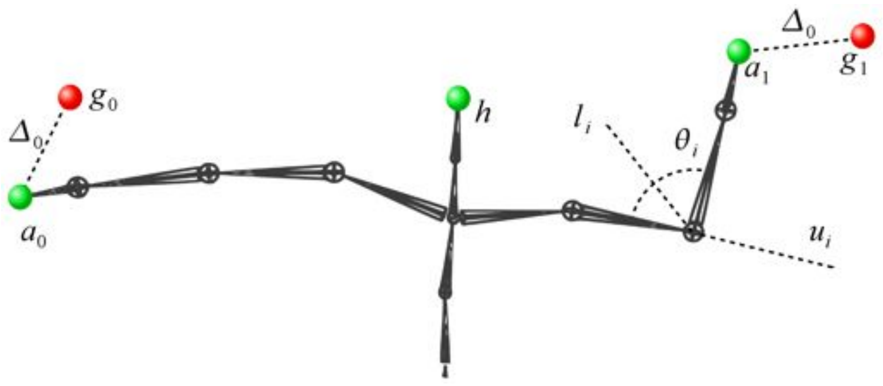


Figure 6: An illustration of the kinematic model. End–effector positions are shown as green dots, while the desired positions (goals) are shown as red dots.

their rotation, as exemplified by joint i in Figure 6 with l_i and u_i showing the limits of the angle θ_i.

To compute the position and orientation of a joint in space, we perform a transformation of the bone relative to its parent joint. The transformation consists of a rotation and a translation corresponding to the shape and orientation of the joint, relative to its parent. These transformations are then nested to create chains of joints. Each chain ends in an end–effector, which can be regarded as the handle for controlling the chain. Thus, the full transformation of a joint from local space to global space can be performed.

Initially, we know the value of the joint parameters, θ^k, and a desired goal state for the end-effector, g. The corresponding initial state of the end-effector is given by

$$\mathbf{y}^k = \mathbf{F}(\theta^k). \tag{3}$$

Writing

$$\theta = \theta^k + \Delta\theta^k, \tag{4}$$

where $\Delta\theta^k$ is the change in joint parameter values. Our task is now to compute $\Delta\theta^k$ such that

$$\mathbf{g} = \mathbf{F}(\theta^k + \Delta\theta^k). \tag{5}$$

We perform a Taylor series expansion of the right-hand side

$$\mathbf{g} = \mathbf{F}(\theta^k) + \frac{\partial \mathbf{F}(\theta^k)}{\partial\theta}\Delta\theta^k + \mathcal{O}(\| \Delta\theta^k \|^2). \tag{6}$$

We introduce the notation

$$\mathbf{J}(\theta^k) = \frac{\partial \mathbf{F}(\theta^k)}{\partial\theta}. \tag{7}$$

and call the matrix $\mathbf{J}$ the Jacobian. Next, we ignore the remainder term of the Taylor series expansion, to obtain the approximation

$$\mathbf{g} \approx \mathbf{F}(\theta^k) + \mathbf{J}(\theta^k)\Delta\theta^k. \tag{8}$$

Recall that $\mathbf{y} = \mathbf{F}(\theta^k)$ and for the moment assume that $\mathbf{J}(\theta^k)$ is invertible. Then we can isolate the unknowns of our problem

$$\Delta\theta^k = \mathbf{J}(\theta^k)^{-1}(\mathbf{g} - \mathbf{y}^k). \tag{9}$$

This is a linear model for taking us as close to g as possible with a linear step. Thus, we may not get to g in one step. To solve this we will keep on taking more steps until we get sufficiently close. That is, we compute $\theta^{k+1} = \theta^k + \Delta\theta^k$ and repeat the above steps with k replaced by $k+1$. This results in a non-linear Newton method. The important thing to notice is that the method only needs to know how to evaluate the function-value, $\mathbf{F}(\theta)$, and the Jacobian $\mathbf{J}(\theta)$.

From [28] we know that if $\mathbf{F}$ is continuously differentiable and the Newton sub-system is solved with sufficient accuracy then the non-linear Newton method will have quadratic convergence. We also know, that we are guaranteed to find a solution to the vector equation $\mathbf{g} = \mathbf{F}(\theta)$ given the initial iterate θ^1 is sufficiently close to the solution. If the initial iterate is not sufficiently close then we may only get linear convergence.

In practice, J is rarely invertible. To overcome these problems one may use the Moore–Penrose pseudo inverse in which case the method is known as the Jacobian Inverse method [8]. The pseudo inverse update is the solution of the least squares problem of minimizing $\frac{1}{2} \parallel \mathbf{J}\Delta\theta^k - (\mathbf{g} - \mathbf{F}(\theta^k)) \parallel^2$. The residual function, $\mathbf{r}$, can be written as

$$\mathbf{r}(\Delta\theta) = \mathbf{y}^{k+1} - \mathbf{g} = \mathbf{F}(\theta^k + \Delta\theta) - \mathbf{g}. \tag{10}$$

Taking a first-order approximation yields

$$\mathbf{r}(\Delta\theta) \approx \mathbf{J}\Delta\theta - (\mathbf{g} - \mathbf{F}(\theta^k)). \tag{11}$$

Using this linear residual model we wish to minimize $\parallel \mathbf{J}\Delta\theta - (\mathbf{g} - \mathbf{F}(\theta^k)) \parallel$ or equivalently

$$f(\Delta\theta) = \frac{1}{2} \parallel \mathbf{J}\Delta\theta - (\mathbf{g} - \mathbf{F}(\theta^k)) \parallel^2 . \tag{12}$$

From the first-order optimality conditions we have the minimizer

$$\nabla f(\Delta\theta^*) = \mathbf{J}^T\mathbf{J}\Delta\theta^* - \mathbf{J}^T(\mathbf{g} - \mathbf{F}(\theta^k)) = 0. \tag{13}$$

Setting $\Delta\theta^k$ to be the minimizer and re-arranging terms while assuming full column-rank of $\mathbf{J}$, we have

$$\Delta\theta^k = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T(\mathbf{g} - \mathbf{y}^k). \tag{14}$$

Thus, the pseudo-inverse is a Gauss-Newton type of method that yield the solution of a least square problem.

A major draw-back of the pseudo-inverse method is the discontinuity of the pseudo-inverse near a singularity [12]. A damped least square (Levenberg-Marquardt) type method can be used to overcome this problem. That is, one seek to minimize $\frac{1}{2} \parallel \mathbf{J}\Delta\theta - (\mathbf{g} - \mathbf{F}(\theta^k)) \parallel^2 + \lambda^2 \parallel \Delta\theta \parallel^2$, where $\lambda > 0$ is a regularization/damping parameter. Performing a similar derivation as above results in the update formula,

$$\Delta\theta^k = (\mathbf{J}^T\mathbf{J} + \lambda^2\mathbf{I})^{-1}\mathbf{J}^T(\mathbf{g} - \mathbf{y}^k). \tag{15}$$

However, one must deal with the problem of selecting a regularization value. Actually, Levenberg-Marquardt is using $\mathbf{J}^T\mathbf{J}$ as the Hessian approximation and can be understood as a modified Newton method combined with a trust region [28]. Notice that in most work on inverse kinematics only a single Gauss-Newton or Levenberg-Marquardt iteration is taken to solve the Newton sub-system.

In some cases, the inverse Jacobian can be approximated by the transpose, $\mathbf{J}^{-1} \approx \mathbf{J}^T$, this variant of the method is known as the Jacobian Transpose [15]. The Jacobian Transpose method has linear convergence to the unconstrained minimizer.

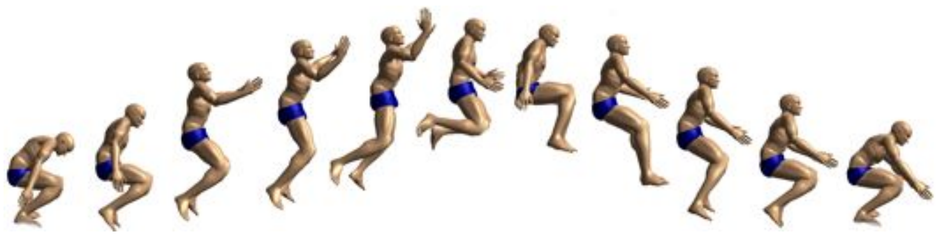


Figure 7: Example of the visual quality showing 11 key-frames in an animation of a jump. The animation was edited manually using a simple editing suite made in connection with this work.

One may also use singular value decomposition to obtain a minimum norm solution. Singular value decomposition has the benefit that one can deal with the singularities and ill-conditioning arising from the loss of freedom [12], while it retains the ability to handle secondary goals.

The open source software package Blender [1] uses a singular value decomposition based Jacobian Inverse method, which deals with joint limits by projection of the final solution unto the feasible region.

If one uses a matrix splitting method [29] for solving the Newton equation then one would obtain the equivalent of the Cyclic-Coordinate-Descent method [15]. The iteration cost of this method is very low. However, it has poor convergence rate.

All of the above variants suffer from the following two drawbacks. First their extension to deal with joint limits is not an explicit part of the mathematical model and can be described as applying a back-projection of non-feasible iterates disregarding the optimality of the projected iterate. Second, the Newton sub-system is not well-posed and approximate solutions are needed in one way or the other. This often results in poor convergence rates and maybe even divergence.

Taking a non-linear optimization approach to the inverse kinematics problems allows one to model joint limits in the underlying mathematical model of the problem, and the numerical problem of singularities of the Jacobian is avoided.

3 A Non-Linear Optimization Approach

We use a squared weighted norm formulation measuring the distance between the goal positions and the end-effector positions. This formulation is similar to [11], and like them we can support numerous goal types including both position and orientation goals. The main difference being that we have agglomerated all goals and introduced a general weighting matrix instead of dealing with K square weighted summation terms. This formulation is well suited for calculation of the solution to a global kinematics problem with multiple end-effectors, because it takes into account the interdependency between various branches. The Jumping motion in Figure 7 is an example of such an animation with multiple dependencies.

Given a branched mechanism containing K kinematic chains, where each chain has exactly one end-effector. We agglomerate the K end-effector functions into one function

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_j \\ \vdots \\ \mathbf{y}_K \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1(\theta) \\ \vdots \\ \mathbf{F}_j(\theta) \\ \vdots \\ \mathbf{F}_K(\theta) \end{bmatrix} = \mathbf{F}(\theta), \tag{16}$$

where $\mathbf{y}_j$ is the world coordinate position of the j^{th} end-effector, and $\mathbf{F}_j(\theta)$ is the end-effector function corresponding to the j^{th} kinematic chain. Using the agglomerated end-effector function, we create the objective function

$$f(\theta) = (\mathbf{g} - \mathbf{F}(\theta))^T \mathbf{W} (\mathbf{g} - \mathbf{F}(\theta)), \tag{17}$$

where $\mathbf{W}$ is a symmetric positive definite and possible diagonal matrix and $\mathbf{g} = \begin{bmatrix} \mathbf{g}_1^T & \cdots & \mathbf{g}_K^T \end{bmatrix}^T$ is the agglomerated vector of end-effector goals. The optimization problem is

$$\theta^* = \min_{\theta} f(\theta) \tag{18}$$

subject to the linear box-constraints

$$\theta \geq \mathbf{l} \tag{19a}$$

$$\theta \leq \mathbf{u}, \tag{19b}$$

which models the minimum and maximum joint parameter values. Here $\mathbf{l}$ is a vector containing the minimum joint limits and $\mathbf{u}$ is a vector of the maximum joints limits. This implies $\mathbf{l} \leq \mathbf{u}$ at all times.

If $\mathbf{F}$ is sufficiently smooth and $\theta^k \to \theta^*$ as $k \to \infty$ then $\mathbf{F}$ behaves almost as a quadratic function at θ^*. This intuition suggest that when we get close to a solution the formulation behaves as a convex quadratic minimization problem. Further, by design all constraints are linear functions defining a convex feasible region. Thus, a simple constraint qualification for the first-order necessary Karush-Kuhn-Tucker optimality conditions is always fulfilled [28]. This would imply that a Newton method would be the method best suited for solving this problem.

Unfortunately this approach is infeasible, since the interactivity requirements of the system prohibits the costly computation of a Hessian, and even Quasi-Newton methods such as the Broyden - Fletcher - Goldfarb - Shanno (BFGS) method [28] would still be costly compared to methods relying solely on the first order information. Originally, [11] combined a Quasi-Newton method with the active set idea used in the projected gradient method of [30, 31]. The idea can be stated as modifying the Newton direction by applying a modification to the current Hessian approximation such that the resulting Newton search direction obtained from the Newton sub-system is kept inside the feasible region. Of course then one must search for blocking constraints when performing a line-search, and furthermore, one must update the set of active constraints and conduct the corresponding projections on the Hessian approximation.

The approach of Zhao et al. does not exploit the fact that the above formulation has a convex feasible region. In fact the feasible region is a boxed feasible region. This suggest that all the book-keeping and modifications of the Hessian matrix can be omitted if the active set idea is replaced by a projected line-search. Unfortunately preliminary testing showed us that projection of a Quasi-Newton direction was not a feasible approach, since the projected direction can no longer be guaranteed to give a reduction in the objective function. To be able to guarantee this reduction a method which is more perpendicular to the iso-contour of the objective function must be chosen.

4 Projected Gradient

The simplest idea for a projected line-search method is to use the gradient descent as a basis. This method is called Gradient projection or the projected gradient method. It is very well suited for nonlinear optimization with box constraints and is robust. In fact, it is not even necessary to assume feasibility of the previous iterate to ensure feasibility of the current iterate, since the method relies on projection. The unprojected search direction of the gradient descent is orthogonal to the iso-contour of the objective function. Thus, we can guarantee that the projected search direction will always be a descent direction for a sufficiently small step-length.

4.1 Computing the Gradient

From (17) we have

$$f(\theta) = (\mathbf{g} - \mathbf{F}(\theta))^T \mathbf{W} (\mathbf{g} - \mathbf{F}(\theta)). \tag{20}$$

The differential can be computed as follows

$$df = d(\mathbf{g} - \mathbf{F}(\theta))^T \mathbf{W} (\mathbf{g} - \mathbf{F}(\theta)) + (\mathbf{g} - \mathbf{F}(\theta))^T \mathbf{W} d(\mathbf{g} - \mathbf{F}(\theta)), \tag{21}$$

which reduces to

$$df = 2(\mathbf{g} - \mathbf{F}(\theta))^T \mathbf{W} d(\mathbf{g} - \mathbf{F}(\theta)), \tag{22}$$

where

$$d(\mathbf{g} - \mathbf{F}(\theta)) = -\frac{\partial \mathbf{F}(\theta)}{\partial \theta} d\theta, \tag{23a}$$

$$= -\mathbf{J} d\theta. \tag{23b}$$

Which means

$$df = -2(\mathbf{g} - \mathbf{F}(\theta))^T \mathbf{W} \mathbf{J} d\theta. \tag{24}$$

From this we have

$$\frac{df}{d\theta} = -2(\mathbf{g} - \mathbf{F}(\theta))^T \mathbf{W} \mathbf{J}, \tag{25}$$

and the gradient can now be written

$$\nabla f = \frac{df}{d\theta}^T = -2\mathbf{J}^T \mathbf{W} (\mathbf{g} - \mathbf{F}(\theta)). \tag{26}$$

How to compute the Jacobian is treated in 5.

4.2 Finding a Step-Length

Given this search direction we can update our parameter vector θ by

$$\theta^{k+1} = \theta^k - \tau \nabla f, \tag{27}$$

where τ is some scalar. If τ is a constant or given by a formula this is equivalent to solving the Jacobian Transpose method as can be seen from (26).

Several values of τ have been tried for the Jacobian Transpose method. In [10], they compute the step-length according to,

$$\tau = \frac{\mathbf{e}^T \mathbf{z}}{\mathbf{z}^T \mathbf{z}} \quad \text{where} \quad \mathbf{e} = \left(\mathbf{g} - \mathbf{y}^k\right) \quad \text{and} \quad \mathbf{z} = \mathbf{J}\mathbf{J}^T \mathbf{e}. \tag{28}$$

Whereas in [32] they compute the step-length such that

$$\| \left(\mathbf{I} - \mathbf{J}\mathbf{J}^{+}\tau\mathbf{e}\right) \| \leq \varepsilon \tag{29}$$

where $\varepsilon > 0$ is a user-specified constant. The rationale behind both approaches is to measure the deviation from the linear approximation. If the deviation is too big then the step-length is reduced. In Robotics, rate control is used [33, 9] in which $\| \Delta\theta \|$ is clamped to a specified maximum, and only a single iteration is used. Others [34] clamp the goal displacement if it exceeds a threshold. Due to linearity down-scaling, the goal-displacement is equivalent to using a smaller step-length.

A non linear optimization method uses some dynamic scheme to find a suitable step-length in each iteration. Usually, a simple inexact line-search is used, such as the Armijo back-tracking. As our approach needs to satisfy the constraints we need a modification of this approach as we will describe next.

4.3 Projected Armijo Back-Tracking Line-Search

An important part of a projected method is the projected line-search since it is here that the actual projection and thus the constraining of the solution is done. Numerous ways of performing inexact line-searches exist and most of them could be combined with projection. We have chosen the Armijo back-tracking approach because of its beneficial properties with regard to speed and robustness, and because it guarantees good improvements in the objective function when such is possible.

We can think of $f(\theta)$ as being a function of the step-length parameter, τ, thus we may write

$$f(\tau) = f(\theta + \tau\Delta\theta) \tag{30}$$

A first order Taylor approximation around $\tau = 0$ yields

$$f(\tau) \approx f(0) + f'(0)\tau \tag{31}$$

The sufficient decrease condition, the Armijo condition [28], is

$$f(\tau) < f(0) + \alpha f'(0)\tau \tag{32}$$

for some $\alpha \in [0..1]$. Observe that

$$f' = \frac{d}{d\tau} f(\theta + \tau\Delta\theta) = \nabla f(\theta)^T \Delta\theta \tag{33}$$

This is nothing more than the directional derivative of f taken at θ and in the direction of $\Delta\theta$. The idea is now to perform an iterative step reduction by setting $\tau^1 = 1$ and verify the above test. If the test fails one updates the step-length as

$$\tau^{k+1} = \beta\tau^k \tag{34}$$

where $\alpha \leq \beta < 1$ is the step-reduction parameter. Performing back-tracking ensures the longest possible step is taken. Therefore there is no need for a curvature condition to avoid unnecessarily small steps. We can now rephrase the test as follows

$$f(\theta + \tau^k\Delta\theta) < f(\theta) + \left(\alpha\nabla f(\theta)^T\Delta\theta\right)\tau^k \tag{35}$$

This is the Armijo test used in an un-projected line-search. If a projected line-search is done, then we can think of θ as a function of τ^k so we write

$$\hat{\theta}(\tau^k) = \theta + \Delta\theta\tau^k, \tag{36}$$

moving some terms around results in

$$\Delta\theta\tau^k = \hat{\theta}(\tau^k) - \theta. \tag{37}$$

Using this in the original Armijo condition we have

$$f(\hat{\theta}(\tau^k)) < f(\theta) + \alpha\nabla f(\theta)^T(\hat{\theta}(\tau^k) - \theta) \tag{38}$$

Next we will keep $\hat{\theta}(\tau^k)$ feasible by doing a projection onto the feasible region

$$f(P(\hat{\theta}(\tau^k))) < f(\theta) + \alpha\nabla f(\theta)^T(P(\hat{\theta}(\tau^k)) - \theta) \tag{39}$$

where P is a projection operator, for instance it could be

$$P(\hat{\theta}(\tau^k)) = \max(\min(\hat{\theta}(\tau^k), \mathbf{u}), \mathbf{l}) \tag{40}$$

where the comparison are element–wise. The vectors $\mathbf{l}$ and $\mathbf{u}$ would be constant lower and upper bounds for θ. This ensures that even if a previous iterate was infeasible then the current iterate will be feasible. Given this projected line-search it is possible to perform a fast and robust computation of the pose.

5 A Geometric Approach to the Differential

To give the reader a more full picture of the method we have included details of the Jacobian computation. The derivations in this section are based on previous work such as [11], but describes this in more detail for the multiple end-effector case.

Without loss of generality, we will choose homogeneous coordinates to develop the theory in the following. Given a chain with n links, we have ${}^{0}\mathbf{T}_1, \ldots, {}^{n-1}\mathbf{T}_n$ homogeneous coordinate matrices. We will assume that the i^{th} joint depends on the parameters θ_i. That is ${}^{i-1}\mathbf{T}_i$ can be thought of as a function of θ_i. We will specify the tool held by the end-effector and the goal placement of the tool by the agglomerated vectors

$$[\mathbf{y}]_n = \begin{bmatrix} \mathbf{p} \\ \hat{\imath} \\ \hat{\jmath} \end{bmatrix}_n, [\mathbf{g}]_0 = \begin{bmatrix} \mathbf{p}_{\text{goal}} \\ \hat{\imath}_{\text{goal}} \\ \hat{\jmath}_{\text{goal}} \end{bmatrix}_0 \in \mathbb{R}^3 \times S^2 \times S^2 \tag{41}$$

where $\mathbf{p}$ is the position while $\hat{\imath}$ and $\hat{\jmath}$ are unit vectors specifying the orientation. The bracket notation $[\cdot]_i$ is used to make it explicit that vectors are expressed in the coordinates of the i^{th} joint frame. Using homogeneous coordinates we can write the instantaneous position of the tool as

$$y = \begin{bmatrix} \mathbf{p} \\ \hat{\imath} \\ \hat{\jmath} \end{bmatrix}_0 = \begin{bmatrix} {}^{0}\mathbf{T}_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & {}^{0}\mathbf{T}_n & \mathbf{0} \\ \mathbf{0} & 0 & {}^{0}\mathbf{T}_n \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \hat{\imath} \\ \hat{\jmath} \end{bmatrix}_n = \mathbf{F}(\theta), \tag{42}$$

where ${}^{0}\mathbf{T}_n = {}^{0}\mathbf{T}_1 \cdots {}^{n-1}\mathbf{T}_n$. Often one would use the practical choices

$$[\mathbf{p}]_n = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad [\hat{\imath}]_n = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \text{and} \quad [\hat{\jmath}]_n = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{43}$$

which could greatly simplify all expressions. However, in the following we will keep things general. Let us investigate the differential of the end-effector function,

$$d\mathbf{F} = \underbrace{\frac{\partial \mathbf{F}(\theta)}{\partial \theta}}_{\mathbf{J}} d\theta = \begin{bmatrix} \mathbf{J}_1 & \cdots & \mathbf{J}_i & \cdots \mathbf{J}_n \end{bmatrix} \begin{bmatrix} d\theta_1 \\ \vdots \\ d\theta_i \\ \vdots \\ d\theta_n \end{bmatrix} \tag{44}$$

where $\mathbf{J}$ is the Jacobian. The above equation tell us what the differential change of the end-effector would be if we induced some differential change in the joint parameters. This opens up for an intuitive way of computing $\mathbf{J}$.

We observe that $\mathbf{J}_i d\theta_i$ describes how the end-effector position is influenced by a change in θ_i. Without loss of generality, we will only focus on the position term. The remaining terms follows in a similar fashion. We have

$$\frac{\partial}{\partial\theta_i}\begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_0 d\theta_i = \underbrace{{}^0\mathbf{T}_1 \cdots {}^{i-2}\mathbf{T}_{i-1}}_{{}^0\mathbf{T}_{i-1}} \frac{\partial({}^{i-1}\mathbf{T}_i)}{\partial\theta_i} \underbrace{{}^i\mathbf{T}_{i+1} \cdots {}^{n-1}\mathbf{T}_n}_{{}^i\mathbf{T}_n} \begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_n d\theta_i \tag{45a}$$

$$= {}^0\mathbf{T}_{i-1} \frac{\partial({}^{i-1}\mathbf{T}_i)}{\partial\theta_i} \begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_i d\theta_i, \tag{45b}$$

Assume that the i^{th} joint is a revolute joint with the unit joint axis $[\mathbf{u}_i]_{i-1} = \begin{bmatrix}x_i & y_i & z_i\end{bmatrix}^T$ specified as a constant vector in the $i-1^{\text{th}}$ frame. One can show

$$\frac{\partial({}^{i-1}\mathbf{T}_i)}{\partial\theta_i} = \begin{bmatrix}\mathbf{U}_i^\times\mathbf{R}_i & \mathbf{0}\\ \mathbf{0}^T & 0\end{bmatrix} \tag{46}$$

where $\mathbf{R}_i$ is the 3-by-3 upper part of ${}^{i-1}\mathbf{T}_i$

$${}^{i-1}\mathbf{T}_i = \begin{bmatrix}\mathbf{R}_i & \mathbf{t}_i\\ \mathbf{0}^T & 1\end{bmatrix} \tag{47}$$

where $\mathbf{t}_i$ is the translational part and

$$\mathbf{U}_i^\times = \begin{bmatrix}0 & y_i & -z_i\\ -y_i & 0 & x_i\\ z_i & -x_i & 0\end{bmatrix} \tag{48}$$

is the skew-symmetric cross-product matrix. That is $[\mathbf{u}_i]_{i-1} \times \mathbf{p} = \mathbf{U}_i^\times \mathbf{p}$ for some $\mathbf{p}$-vector. This means we have

$$\frac{\partial}{\partial\theta_i}\begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_0 d\theta_i = {}^0\mathbf{T}_{i-1}\begin{bmatrix}\mathbf{U}_i^\times\mathbf{R}_i & \mathbf{0}\\ \mathbf{0}^T & 0\end{bmatrix}\begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_i d\theta_i, \tag{49a}$$

$$= {}^0\mathbf{T}_{i-1}\begin{bmatrix}\mathbf{U}_i^\times\mathbf{R}_i[\mathbf{p}]_i\\ 0\end{bmatrix} d\theta_i, \tag{49b}$$

Notice that $\mathbf{R}_i[\mathbf{p}]_i = [\mathbf{p}]_{i-1} - [\mathbf{t}_i]_{i-1}$, so

$$\frac{\partial}{\partial\theta_i}\begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_0 d\theta_i = {}^0\mathbf{T}_{i-1}\begin{bmatrix}[\mathbf{u}_i]_{i-1} \times ([\mathbf{p}]_{i-1} - [\mathbf{t}_i]_{i-1})\\ 0\end{bmatrix} d\theta_i, \tag{50a}$$

$$= \begin{bmatrix}[\mathbf{u}_i]_0 \times ([\mathbf{p}]_0 - [\mathbf{t}_i]_0)\\ 0\end{bmatrix} d\theta_i, \tag{50b}$$

Using similar derivations for i and j terms allow us to conclude that we can obtain the i^{th} column of the Jacobian corresponding to a revolute joint by

$$J_i = \begin{bmatrix} [\mathbf{u}_i]_0 \times ([\mathbf{p}]_0 - [\mathbf{t}_i]_0) \\ [\mathbf{u}_i]_0 \times [\hat{i}]_0 \\ [\mathbf{u}_i]_0 \times [\hat{j}]_0 \end{bmatrix} \tag{51}$$

If the i^{th} joint was a prismatic joint with sliding along the joint axis given by the unit vector $[\mathbf{u}_i]_{i-1}$ then one would have

$$\frac{\partial({}^{i-1}\mathbf{T}_i)}{\partial\theta_i} = \begin{bmatrix} \mathbf{0} & [\mathbf{u}_i]_{i-1} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{52}$$

from this we have

$$\frac{\partial}{\partial\theta_i}\begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_0 d\theta_i = {}^0\mathbf{T}_{i-1}\begin{bmatrix} \mathbf{0} & [\mathbf{u}_i]_{i-1} \\ \mathbf{0}^T & 0 \end{bmatrix} {}^i\mathbf{T}_n \begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_n d\theta_i, \tag{53a}$$

$$= {}^0\mathbf{T}_{i-1}\begin{bmatrix}\mathbf{u}_i\\0\end{bmatrix}_{i-1} d\theta_i, \tag{53b}$$

$$= \begin{bmatrix}\mathbf{u}_i\\0\end{bmatrix}_0 d\theta_i, \tag{53c}$$

Note that the sub-parts corresponding to orientation, $\hat{i}$ and $\hat{j}$, are zero.

$$\frac{\partial}{\partial\theta_i}\begin{bmatrix}\hat{i}\\0\end{bmatrix}_0 = {}^0\mathbf{T}_{i-1}\begin{bmatrix} \mathbf{0} & [\mathbf{u}_i]_{i-1} \\ \mathbf{0}^T & 0 \end{bmatrix} {}^i\mathbf{T}_n \begin{bmatrix}\hat{i}\\0\end{bmatrix}_n d\theta_i, \tag{54a}$$

$$= {}^0\mathbf{T}_{i-1}\begin{bmatrix} \mathbf{0} & [\mathbf{u}_i]_{i-1} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix}\hat{i}\\0\end{bmatrix}_i d\theta_i \tag{54b}$$

$$= \mathbf{0} \tag{54c}$$

similar for the $\hat{j}$-term. Thus for the case of the prismatic joint we have

$$\mathbf{J}_i = \begin{bmatrix} [\mathbf{u}_i]_0 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{55}$$

The extension to more than one translation axis is trivial.

As seen from all the above derivations the "effect" of the tool is computed from the geometry expressed in the world coordinate system, and we can conclude that the computation of the Jacobian is totally independent of what type of coordinate representation one uses. We exploit this to use a quaternion representation rather than a homogeneous coordinate representation. Since spherical linear interpolation quaternions are better suited for interpolation of the in-between key-frames and superior skinning techniques exist based on quaternions as well [35]. Thus, one can avoid conversion between quaternions and matrices completely. Further the computational effort in computing the absolute transformations of each joint is less expensive using a quaternion representation than using a matrix representation.

5.1 A Ball and Socket Joint

In the above, we only dealt with having one rotation axis and multiple translation axes. Thus, a natural question is what to do with multiple rotation axes? Traditionally, this has been handled by creating imaginary multiple joints. Thus a joint with rotation around three axes can be modeled as three revolute joints placed on top of each other. In the following, we will derive equations for dealing with such a joint in a canonical way.

Euler angles are a popular choice as parameterization in motion capture and animation formats, therefore we will use an Euler angle parameterization of a ball and socket joint. Inspired by the robotics community [7, 36], we will adopt the ZYZ Euler angle convention. This means that if the i^{th} joint is a ball-and-socket joint then it will be parametrized by the angles θ_i, ϕ_i, and ψ_i. Further, the relative transformation is given as

$$ {}^{i-1}\mathbf{T}_i = \begin{bmatrix} \mathbf{R}_Z(\phi_i)\mathbf{R}_Y(\psi_i)\mathbf{R}_Z(\theta_i) & \mathbf{t}_i \\ \mathbf{0}^T & 0 \end{bmatrix}, \tag{56} $$

where $\mathbf{R}_Z$ and $\mathbf{R}_Y$ are rotations around the z and y axes of the $(i-1)^{th}$ joint frame. Trivially we have

$$ \frac{\partial({}^{i-1}\mathbf{T}_i)}{\partial\phi_i} = \begin{bmatrix} \left(\mathbf{Z}^\times\mathbf{R}_Z(\phi_i)\right)\mathbf{R}_Y(\psi_i)\mathbf{R}_Z(\theta_i) & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{57a} $$

$$ \frac{\partial({}^{i-1}\mathbf{T}_i)}{\partial\psi_i} = \begin{bmatrix} \mathbf{R}_Z(\phi_i)\left(\mathbf{Y}^\times\mathbf{R}_Y(\psi_i)\right)\mathbf{R}_Z(\theta_i) & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{57b} $$

$$ \frac{\partial({}^{i-1}\mathbf{T}_i)}{\partial\theta_i} = \begin{bmatrix} \mathbf{R}_Z(\phi_i)\mathbf{R}_Y(\psi_i)\left(\mathbf{Z}^\times\mathbf{R}_Z(\theta_i)\right) & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{57c} $$

where

$$ \mathbf{Y}^\times\mathbf{p} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \mathbf{p} \quad \text{and} \quad \mathbf{Z}^\times\mathbf{p} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \mathbf{p} \tag{58} $$

for some vector $\mathbf{p}$. Next we define the three vectors

$$[\mathbf{u}_i]_{i-1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{i-1} \tag{59a}$$

$$[\mathbf{v}_i]_{i-1} = \mathbf{R}_Z(\phi_i) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_{i-1} \tag{59b}$$

$$[\mathbf{w}_i]_{i-1} = \mathbf{R}_Z(\phi_i)\mathbf{R}_Y(\psi_i) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{i-1} \tag{59c}$$

and by straightforward computation we have,

$$\frac{\partial [\mathbf{p}]_0}{\partial \phi_i} d\phi_i = \left({}^0\mathbf{T}_{i-1}[\mathbf{u}_i]_{i-1}\right) \times \left([\mathbf{p}]_0 - [\mathbf{t}_i]_0\right) d\phi_i, \tag{60a}$$

$$\frac{\partial [\mathbf{p}]_0}{\partial \psi_i} d\psi_i = \left({}^0\mathbf{T}_{i-1}[\mathbf{v}_i]_{i-1}\right) \times \left([\mathbf{p}]_0 - [\mathbf{t}_i]_0\right) d\psi_i, \tag{60b}$$

$$\frac{\partial [\mathbf{p}]_0}{\partial \theta_i} d\theta_i = \left({}^0\mathbf{T}_{i-1}[\mathbf{w}_i]_{i-1}\right) \times \left([\mathbf{p}]_0 - [\mathbf{t}_i]_0\right) d\theta_i, \tag{60c}$$

now renaming the vector $([\mathbf{p}]_0 - [\mathbf{t}_i]_0)$

$$r_0 = ([\mathbf{p}]_0 - [\mathbf{t}_i]_0) \tag{61}$$

we get the Jacobian entry

$$\mathbf{J}_i = \begin{bmatrix} [\mathbf{u}_i]_0 \times r_0 & [\mathbf{v}_i]_0 \times r_0 & [\mathbf{w}_i]_0 \times r_0 \\ [\mathbf{u}_i]_0 \times [\hat{\imath}]_0 & [\mathbf{v}_i]_0 \times [\hat{\imath}]_0 & [\mathbf{w}_i]_0 \times [\hat{\imath}]_0 \\ [\mathbf{u}_i]_0 \times [\hat{\jmath}]_0 & [\mathbf{v}_i]_0 \times [\hat{\jmath}]_0 & [\mathbf{w}_i]_0 \times [\hat{\jmath}]_0 \end{bmatrix} \tag{62}$$

6 Performance of Projected Back-Tracking Line-Search

The projected line-search developed in this work has been tested with two different methods: the Jacobian Transpose method, which is the same as a Steepest Descent method using a fixed step-length, and a projected line-search method with the projected Armijo back-tracking line-search we described in Section 4.3. We have compared our results with the results from the SVD based Jacobian Inverse method used in the Blender Software package.

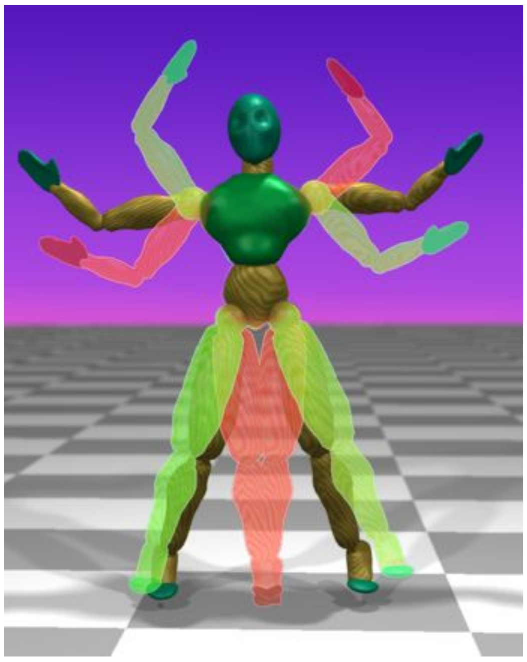


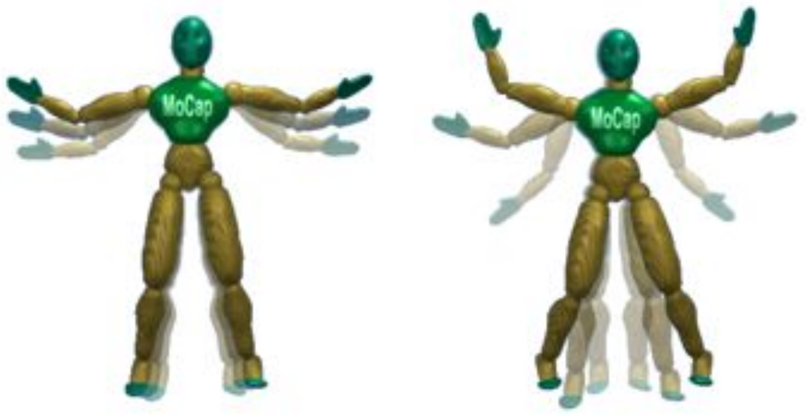
Figure 8: An example of what may happen if the step-length of the Jacobian Transpose method is chosen too high. The motion flips between the green and the red pose, superimposed on the desired result. Step-length in this example was set at 0.05. This example used 5 positional goals and joint limits.

6.1 Comparisons

Blender was chosen as an example of a widely used software package which lives up to current performance and quality demands. The reason for choosing Blender instead of e.g. Maya or 3D Max was, that it is possible to perform actual measurements on Blender due to it being open source. Since the functionality, quality and performance of Blenders inverse kinematics system is very similar to the other solutions (see e.g. Figure 4 or the so called cg survey by [37]), we chose only this one.

Our reference implementation of the Jacobian Transpose method corresponds to the one used by [15] where a fixed sufficiently small step-length is used. Experiments were done with the Jacobian Transpose method varying the fixed step-length. These experiments showed that a step-length of more than 0.005 would make the Jacobian Transpose method diverge even if the method were given an initial iterate comparatively close to a solution. Step-length below this limit slowed down the method without discernible improvement to the quality. Therefore, in all our results we used Jacobian

Transpose method with a fixed step-length of 0.005. An example of the divergence exhibited with too large step-length is shown in Figure 8.


(a) Animation using 28 frames per cycle (b) Animation using 7 frames per cycle

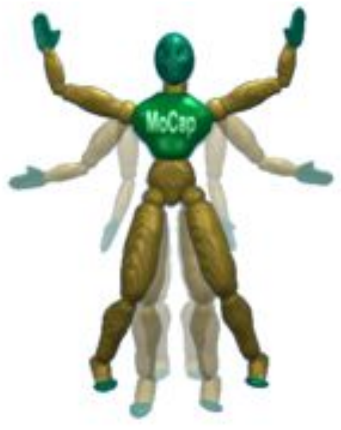

(c) Animation using 5 frames per cycle

Figure 9: The difference between the three sampling settings of motion capture data. Notice that spatial-temporal coherence is decreased from a to c.

6.2 Test Framework

The methods were tested by running a number of repeated tests under different conditions. The test scenario comprised a fixed setup using a motion captured animation of a person doing a gymnastics exercise. A time lapse of the animation showing the exercise can be seen in Figure 13. The positions of the hands, feet, pelvis and head of the motion capture animation were used as goals for the inverse kinematics skeleton. The orientation of the goals was not used in these tests. The goals were updated each frame. The skeleton consists of 71 degrees of freedom, and joint limit constraints on all joints, giving a total of 142 constraints (71 upper and 71 lower).

Since interactive performance was a major factor in the evaluation we chose to compare the quality of the method with a fixed time slot at their disposal giving each a maximum number of iterations which would make it converge in approximately 0.015 seconds or less, corresponding to approximately 66 frames / seconds.

We varied how far the goal-positions were placed from the end-effector positions. This was done by sampling the motion captured data with varying intervals. The settings used were 0.05, 0.7, and 1.0 seconds. Figure 9 shows three consecutive frames of the reference animation, using the three settings.

The test cases were run on an Intel®dual core 1.66 GHz architecture with 1 GB memory utilizing only one core.

An absolute tolerance of 0.05 units was set. The tolerance was chosen to be small enough not to interfere with the visual quality of the animation. 0.05 units is approximately 1 centimeter if the skeleton corresponds to a person that is approximately 1.80 meters high. The function value in Figure 10, 11, and 12 are the sums of all end-effector squared errors.

6.3 Visual Quality

The visual quality is graded by the closeness to the reference animation as well as the smoothness of the animation. In both cases the gradient projection gives superior results to what is being computed by the SVD based Jacobian Inverse method. This can be seen from the plots in Figure 10, 11 and 12.The figures show the Median value as a red line. The edges of the blue boxes denote the 25th and 75th percentiles, the whiskers denote the boundary of the inliers of the data set. Outliers are shown individually as red crosses. The smoothness of the animation and the handling of joint limits and orientation/position of intermediate joints can be seen from the frames in Figure 13 notice frames 7 and 9 where the animation made by the reference method clearly jitters.

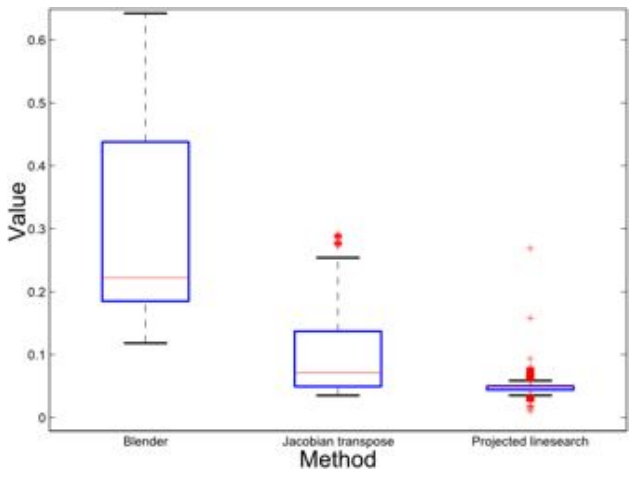


Figure 10: Plot of the distribution corresponding to the animation shown in Figure 9(a). Red crosses denote outliers in the data set. Notice that the Projected line-search has much lower Median as well as a more compact distribution.

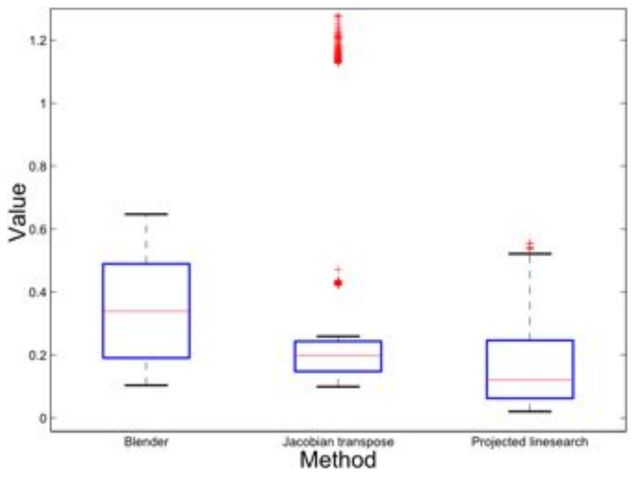


Figure 11: Plot of the distribution corresponding to the animation shown in Figure 9(b). Red crosses denote outliers in the data set.

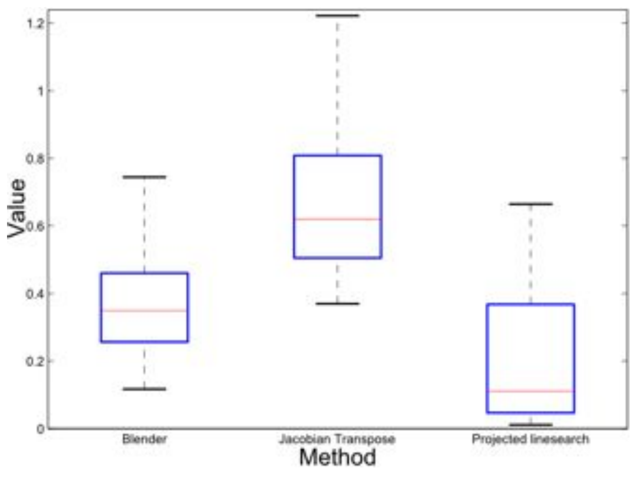


Figure 12: Plot of the distribution corresponding to the animation shown in Figure 9(c). Notice that the Jacobian Transpose method is beginning to have problems in this case due to the lack of coherence. Some significant outliers with values of around 80 has been omitted from the Jacobian Transpose method plot to make comparison possible.

6.4 Robustness

Since the method guarantees a reduction in the objective function, the method is very robust. No restrictions on previous poses are necessary. The robustness can be seen from the tight distributions of the values in Figure 10, 11 and 12.

6.5 Generality

The method can be used with any conceivable skeleton. Examples of skeletons we have tested are shown in Figure 1. The method is easily integrable in other systems. The method is implemented in the free open-source meta-library OpenTissue [27], and can be included in any source code under the Zlib license [38]. As shown in Figure 1 the method has been included in an ogre demo by a third party.

7 Alternatives to Gradient descent

Gradient descent is not the best performer in the family of optimization methods. For general constrained problems a quasi-newton approach is usually a much better choice. Even with the higher iteration cost and the

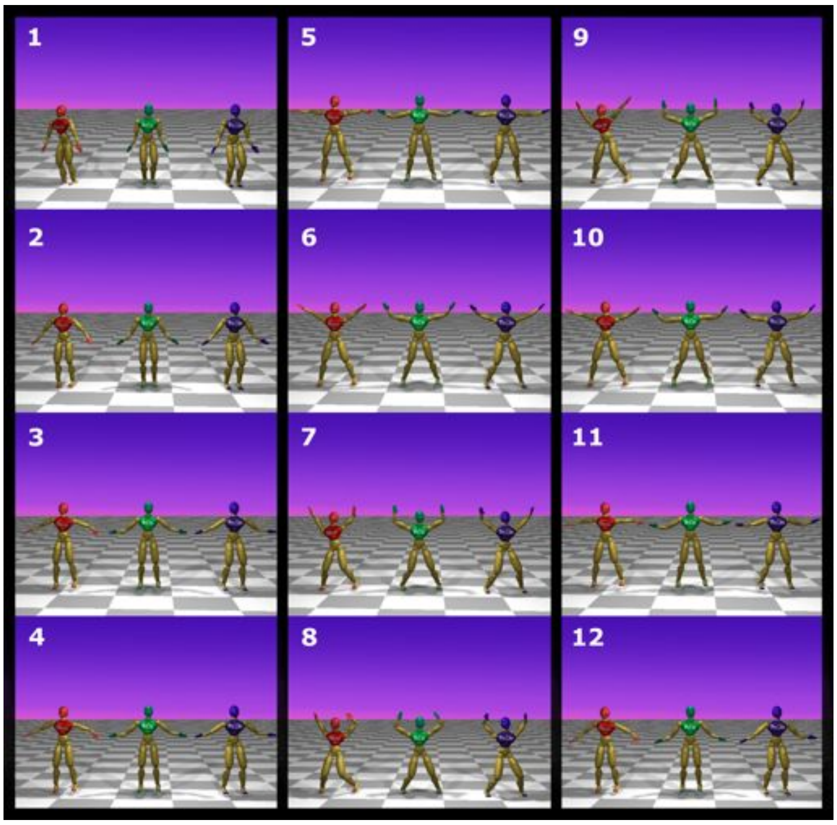


Figure 13: Selected frames from the animation used in testing the methods, each frame shows from left to right: the Blender version, the Motion Capture reference, the Projected line-search method. Notice that the Blender method jitters from frame to frame and that it differs significantly from the reference (notice especially frame 7 and 9).

necessity to keep track of the active set of constraints, better performance can usually be achieved. For the Inverse kinematics problem in an interactive setting it is often not as important to get an exact solution and when we add to this that the projection operation cannot be combined with the quasi newton update step, it becomes infeasible to use the active set method. In this PhD project we experimented with several method and as it can be seen from our earlier work [39] the projection operator can be combined with a conjugate gradient method, with good results. The projection operator was also combined with a dogleg method in [40] The performance of these alternatives are not handled in this work. The interested reader is refered to these texts.

8 An Application in motion tracking

In the following I will describe an application of the Inverse kinematics system. The system was used as a dimensionality reduction in a particle filter tracker developed by Søren Hauberg in conjunction with his Phd work [41]. The description here is an extract of the one presented in [4] here I have concentrated on presenting the results.

8.1 Motion Estimation

as a dimensionality reduction for monocular human motion estimation. The inverse kinematics solver deals efficiently and robustly with box constraints and does not suffer from shaking artifacts. The presented motion estimation system uses a single camera to estimate the motion of a human. The results show that inverse kinematics can significantly speed up the estimation process, while retaining a quality comparable to a full pose motion estimation system. Our novelty lies primarily in use of inverse kinematics to significantly speed up the particle filtering. It should be stressed that the observation part of the system has not been our focus, and as such is described only from a sense of completeness.

With our approach it is possible to construct a robust and computationally efficient system for human motion estimation.

Inverse kinematics has found widespread use as an intuitive posing system for articulated figures in Computer Graphics [14] and for motion planning in Robotics [42]. We propose a novel use of inverse kinematics as a means to reduce the dimensionality of a particle filter based tracking algorithm. Preliminary results show that this approach significantly reduces the time demands compared to existing approaches with comparable results. The method may make it possible to perform visual tracking of general human motion in an interactive way.

Three dimensional human motion analysis is the process of estimating the configuration of body parts over time from sensor input [43]. Traditionally motion capture equipment has been used to track this motion. In motion capture, markers are attached to the body and then tracked in 3 dimensions. Usually this requires multiple tracking devices so motion capture is most often performed in pre–calibrated laboratory settings.

Our long term goal is to use human motion analysis as part of a physiotherapeutic rehabilitation system. In the system, the motion of a patient is tracked and analyzed during exercise sessions performed both at the clinic and at the patient's home. This application rules out the traditional motion capture approach, and results in the need for a simpler system with fewer cameras. These limitations make it necessary to formulate a better model to compensate for the lack of information from the image data. Our approach is to use inverse kinematics as a way to impose this information.

Our approach utilizes the fact that the pose of a skeleton can be deduced from the end–effector positions. While a full human skeleton may have more than 100 degrees of freedom, the end–effectors space of the same articulated figure can have e.g. only 15 degrees of freedom (positional parameters of head, hands and feet). This dimensionality reduction accounts for a significant speedup in the computational demands of the system, compared to analyzing the motion in the full pose configuration space of the skeleton.

Estimation of human motion is an inherently high dimensional problem, since human motion is both diverse and has many degrees of freedom. The traditional approach to reducing the dimensionality has been to utilize manifold learning, i.e. to try and restrain the motions in a subspace of the full space. This approach is used in [44, 45, 46], but seems to be most suited for a constrained set of motions like walking or golf–swinging.

We want to be able to process a larger set of motions, and thus need some other means of reducing the dimensionality. Inverse kinematics has been used in motion estimation before [47], but not as a dimensionality reduction tool. Our work differs in that it utilizes the posing abilities of the inverse kinematics system to infer the pose of the remaining joints. Thus, the estimation can be performed on the end–effector joints only.

Our inverse kinematics system is based on the robust system described in previous sections.

The motion estimation system used is the one described in [6]. While [6] concentrated on the motion estimation, this work focuses on the interactive inverse kinematics solver.

9 Visual Motion Estimation

Visual motion estimation is the process of inferring the motion of a moving object from a sequence of images. In this work we wish to infer the 3 dimensional pose of a human moving in front of a camera.

In this Thesis the Visual motion estimation will not be described in detail since this part of the work must be attributed mainly to Søren Hauberg. The interested reader is referred to the papers [6] and [4] where a detailed description of the motion tracking can be found. In this workk we will

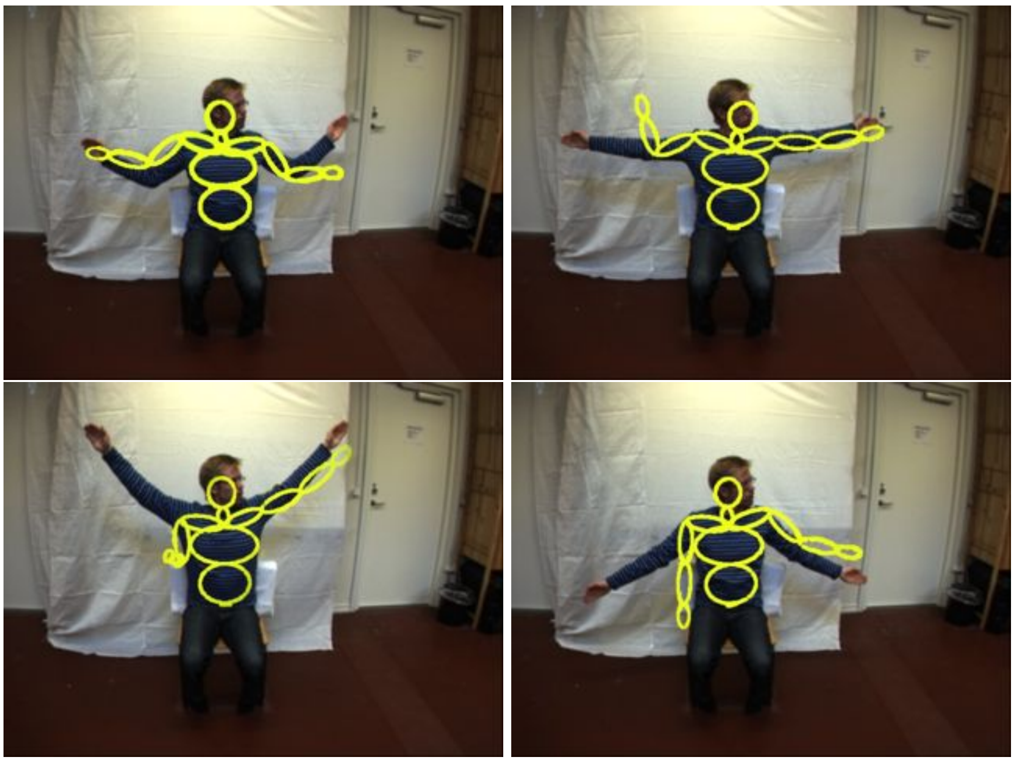


Figure 14: Traditional motion estimation in full pose space using 100 particles. Notice that the superimposed figure looses track of the person in the image.

concentrate on the effects of introducing the inverse kinematics solver as a dimensionality reduction tool. and thus we present the results of our tests.

9.1 Results

To verify the quality and to measure the time improvement we performed some simple tests. The tests consisted of estimating the motion of a person sitting on a chair moving his arms about. The resulting skeleton had 3 end–effectors , the hands and the head. The skeleton was fixed at the hip and the legs were not modelled. The method can handle a full skeleton but for these tests we chose a simple skeleton. The original video clip used was app. 45 seconds long, at 15 fps.

The purpose of the tests was to compare the time expenditure and quality of our method to a traditional method.

We performed tests and timing of the system, using a Lenovo T400 Thinkpad® laptop with an Intel ®core™2 duo 2.40 Ghz.

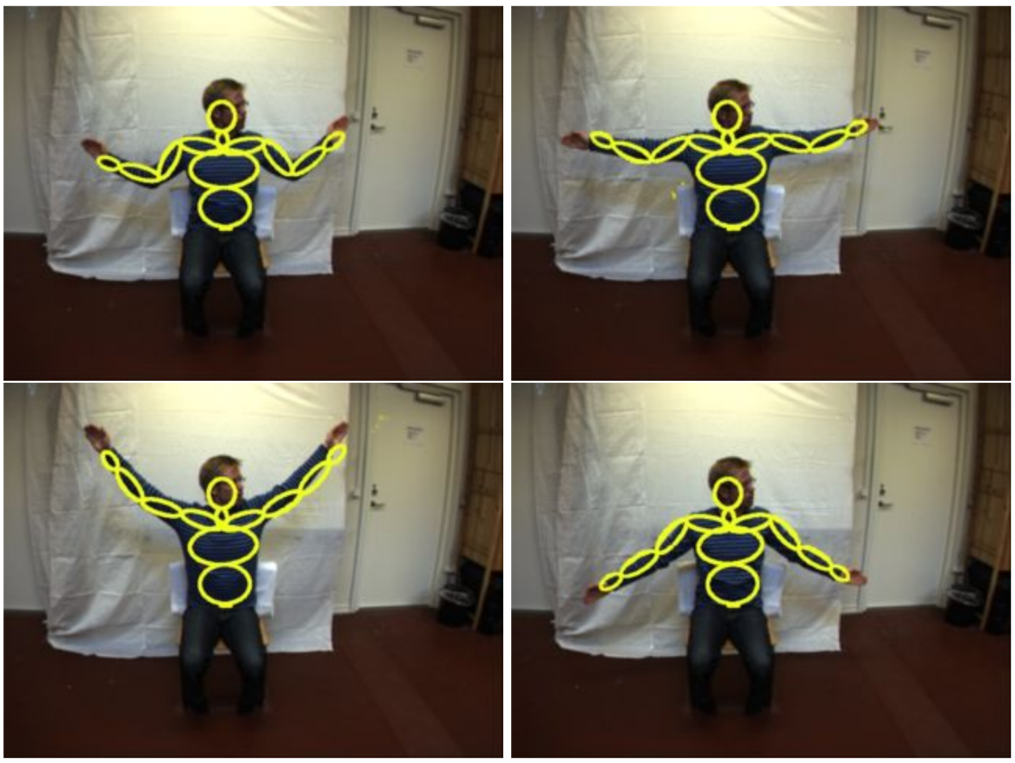


Figure 15: Traditional motion estimation in full pose space using 5000 particles. The results of this motion analysis are satisfactory but the computation took more than 10 hours.

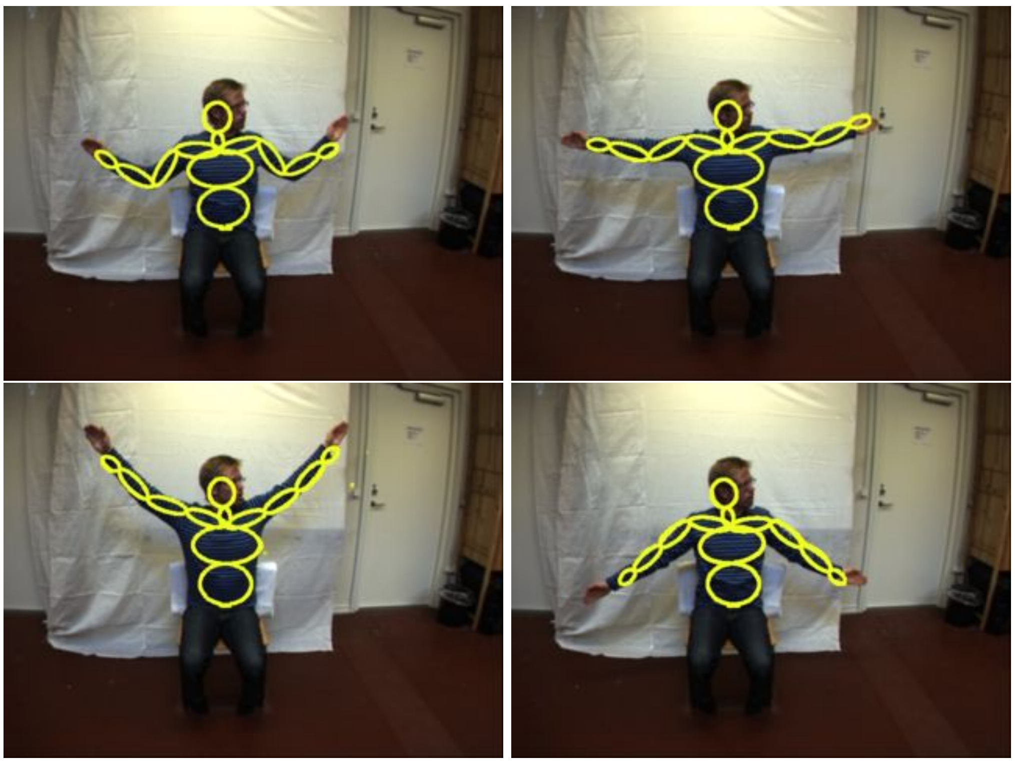


Figure 16: Motion estimation using inverse kinematics with 25 particles. As it can be seen the results are very similar to Figure 15. However the time spent was only 5 minutes.

Figure 14, 15 and 16 show selected frames from an image sequence with the pose estimation results superimposed. The images in each of the figures correspond to the 32^{nd}, 94^{th}, 126^{th} and the 196^{th} frame of the sequence. A video version of the test results are available at:

`http://humim.org/vriphys2009/`

The first test was a traditional full pose motion estimation without inverse kinematics using 100 particles. Figure 14 shows the result. Here the system quickly looses track of one arm and produces a large amount of shaking in the motion estimation. The computations took approximately five minutes on the test hardware. We then increased the number of particles to 5000, which resulted in a successful motion estimation with only little shaking. Unfortunately, this required more than 10 hours of computation time. The results of this test can be seen in Figure 15.

The final experiment was run using the inverse kinematics system for pose calculation, making it possible to track in only 9 dimensions. Only 25 particles were used and the running time was approximately 5 minutes. This is a speedup factor of approximately 120 compared to the 5000 particle run and a comparable quality, while the 100 particle run is comparable in time spent, but in this case, the quality of the inverse kinematics tracker is much better.

The results show that motion estimation in end–effector space is possible and that large speedups can be achieved using this approach. Our long term goal is to create a motion estimation system for use in a physiotherapeutic rehabilitation program and here it is essential to have real–time performance, in order to provide feedback to both patients and therapists. Estimation in end–effector space makes this requirement more feasible compared to estimation in full–pose space.

Using only a single camera gives no depth information which means that the system has difficulty in placing the goals correctly in this direction. This problem might be solved by using a camera type which can infer some depth information such as a stereo camera or a time of flight camera. Weighting the depth parameter of the goals with a small weight might also help since this would give the inverse kinematics solver more freedom in the placement with regards to the depth. It could however, also result in poses which would fit badly with the visual data so it could not stand alone.

10 Discussion and Conclusion

The non-linear optimization approach has shown beneficial behavior as a strong mathematical formulation of the inverse kinematics problem. We have presented an efficient and robust numerical approach for solving the problem. We believe our approach is simple and elegant and easy to implement, even for non-specialists in numerical optimization. Further, we have demonstrated that our approach is sufficiently exact and responsive for interactive manipulation of multiple end-effectors while handling joint-limits.

We have shown the connection between using a Projected gradient method for the non-linear optimization and the traditional Jacobian Transpose method. Further, we showed how the Jacobian Transpose method could be improved by using our projected line-search method.

The shaky and jittery motion often reported near singular configurations can be avoided completely when using an optimization approach, since the variable step-length cancels out the adverse effect of the high angular velocity. It is thus only a matter of a sufficiently exact line-search and a sufficiently low minimum step-length.

We have demonstrated how the inverse kinematics solver can be used in an articulated tracker application with a significant speedup as the result.

In our opinion, we have only touched upon the subject of seeking the best suited mathematical formulation for interactive inverse kinematics and matching numerical methods. There are many possible avenues for further work. It could be interesting to reformulate the first order necessary optimality conditions into a complementarity problem and further reformulate this as a possible smooth or non-smooth non-linear equation solving problem. This would in a sense take us back to the equation solving approaches but with the difference that constraints are dealt with implicitly without the need for projections at all. Quasi-Newton and Steepest Descent methods are but a few out of many methods for solving the constrained minimization problem we have stated. Trust region methods or hybrids may be other interesting methods to adapt to the formulation given.

Part II

Joint Constraint Models

Box constrained joint limits are excellent for modelling the joint constraints of e.g. robots, but it is not a very exact reconstruction of actual joint limits in the human body. Therefore, my research began to concentrate on finding a good alternative, that would still be compatible with the Inverse Kinematics method described in the previous part, as well as being usable for other methods. This led to an investigation into whether such a model existed. As it turned out there was no method for describing joint limits which could supply a constant time projection operator, thus, we decided to invent our own. This research resulted in the following publications

- ’ A Joint-Constraint Model for Human Joints using Signed Distance-Fields.
  Special issue of Multibody System Dynamics [48].
- Local Joint-Limits using Distance Field Cones in Euler Angle Space.
  Computer graphics international CGI 2010 [49].
- Distance-Field Based Joint-Limits for Biomechanic Joint Models.
  Euromech Colloquium 511 - Biomechanics of Human Motion.

All made in collaboration with Sarah Niebe and Kenny Erleben.
The topics of these 3 publications are the development and presentation of the signed distance field joint constraint model. The application moved towards biomechanics and clinical motion analysis. This Is a very nice example of the way a method developed initially for a purely graphics oriented application can be used in e.g. biomechanical work.

11 Joint constraints using signed distance fields

We present a local joint-constraint model for a single joint which is based on distance fields. Our model is fast, general and well suited for modelling human joints. In this work, we take a geometric approach and model the geometry of the boundary of the feasible region, i.e. the boundary of all allowed poses. A region of feasible posed can be build by embedding motion captured data points in a signed distance field. The only assumption is that the feasible poses form a single connected set of angular values. We show how signed distance fields can be used to generate fast and general joint-constraint models for kinematic figures. Our model is compared to existing joint-constraint models, both in terms of generality and computational cost.
The presented method supports joint-constraints of up to three degrees of freedom and works well with sampled motion data. Our model can be extended to handle inter-joint dependencies, or joints with more than three degrees of freedom. The resolution of the joint-constraints can be tweaked individually for each degree of freedom, which can be used to optimize memory usage. We perform a comparative study of the key-properties of various joint-constraint models, as well as a performance study of our model compared to the fastest alternative, the box limit model. The study is performed on the shoulder joint, using a motion captured jumping motion as reference.

12 Introduction

When simulating articulated figures, one needs a model that describes the range of motion of the individual joints. An example of such a model is the inverse kinematics (IK) skeleton which is a hierarchy of bones where each bone is connected to a parent bone by a joint [11]. The relative coordinate transforms between connected bones are given by a set of joint parameters. The focus of this work is the presentation of a novel accurate model for representing legal values of these joint parameters based on experimental kinematic data.

From a mathematical viewpoint, an IK skeleton is a hierarchy of homogeneous coordinate transformations, where each bone corresponds to a homogeneous transformation. The different joints of the skeleton require varying numbers of parameters for representing a given pose. For instance, the elbow joint only needs a single parameter describing the angle between the upper and lower arm. Joints such as the shoulder or hip joints, have a higher degree of freedom (DOF) and thus require more angle parameters. One could even use a translational joint parameter to model the sliding of the scapula on the thorax in the shoulder complex.

These joint parameters are not unbounded, they are constrained by a highly non-convex, continuous connected and closed subset of the parameter space. We use the term *feasible pose* when all the joint parameters of a given pose are in the legal parameter space. Joint-constraints describe the boundaries of the feasible region of poses. Figure 12 illustrates the geometric complexity of the feasible regions we face within our application perspective. It shows joint-constraints sampled from several motions, found in the Carnegie Mellon university database of motion capture recordings [50].

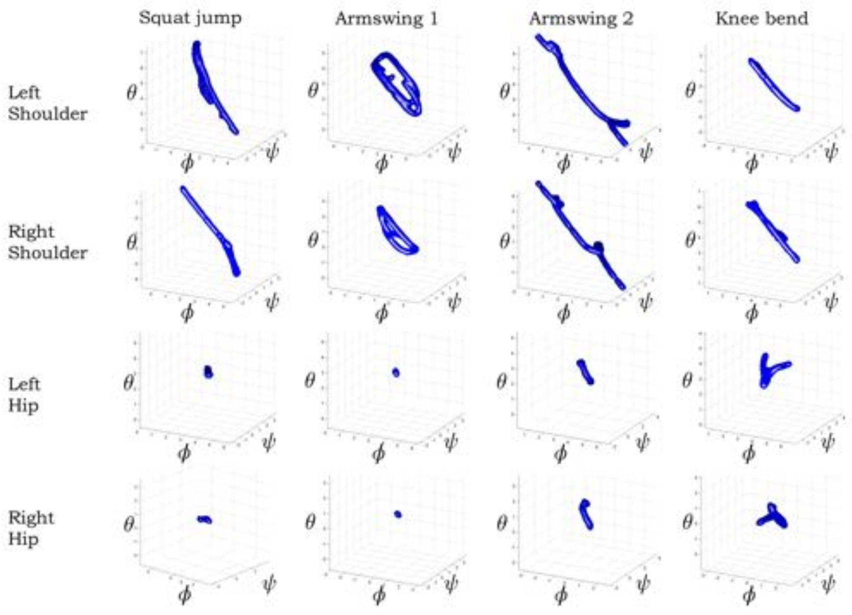


Figure 17: Examples of joint-constraints for different joint types and different motions. Each row corresponds to one joint type and each column to one motion. Observe the complex geometry of the feasible region.

The method we present is comparable to the method presented in [23]. However, the use of Euler angles and distance-fields results in a constant time projection operator, making our method a faster, more accurate and attractive approach. Our model, the signed distance-field joint-constraint model, is generalized, supports highly nonconvex joint-constraints, has a simple geometric interpretation and shows no performance problems that would prohibit use in interactive applications.

The work presented here is highly motivated by tracking human poses using computer vision and machine learning based methods [6]. Within this application perspective, there is a need for a joint-constraint model that is computationally fast and gives an accurate description of the feasible region of a single specific human motion. An early version of this research, with an animation focus, was presented in [51].

13 Related Work

Several researchers have investigated the range of motion of joints [52, 53], and numerous models exist e.g [23, 22, 25]. The box-limit model is the prevalent model, used in motion file formats such as Acclaim's asf/amc [54]. Animation tools such as Maya [5] and Blender [1] also use the box-limit model. With box-limits, the individual parameters are bounded within a minimum and maximum of allowed values, thus the feasible region forms a product of independent intervals. As this is a linear model with a convex feasible region, the box-limit model is easy to combine with optimization methods []. However, there are two major drawbacks of box-limits: They tend to result in a loose fitting boundary and they fail to include interdependencies between joint parameters. In practice, these short-comings are handled by tweaking the box-limits for specific motions, thus for a running motion one set of values would be used whereas, for a jumping motion, another set of joint parameters would be used. Despite these shortcomings, the box-limit model is widely used. As shown in previous work, box-limits can be determined automatically from motion capture data [6, 55]. The alternatives to the box-limit model offer more descriptive models, at the cost of increased complexity. The signed distance-field model, presented in this work, takes a geometric approach. For each joint, we model the geometry of the boundary of the feasible region of the joint parameters. Our model is local, in the sense that we only model joint parameter dependencies for each individual joint. The model requires a two-manifold feasible region, so it can be embedded as the zero level-set of a signed distance-field, allowing us to handle any nonconvex joint-constraint regions. In this work, we have used the shoulder joint as the primary test case because it exhibits some of the more complex behaviour of joint-constraints and thereby stresses the joint-constraint model used. The legal parameter space of the human shoulder joint is bounded by a nonconvex nonlinear joint-constraint [23, 22].

The authors of [22] use a sinus-cone model from [56], a human shoulder is modelled by a hierarchical IK skeleton. The scapula-thoracic joint is modelled by breaking the closed chain and using the scapula as an end-effector constrained to the surface of an ellipsoidal thorax. The sinus-cone model is more general than the box-limit model. A reach-cone model based on an idea from Korein [57] is presented by Wilhelms and Gelder [25]. This is extended in [26] where a general joint component framework is described. In [58], a spline based implicit joint model is suggested for multibody dynamics. Due to the implicit nature of the model, the geometry of the boundary of the feasible region can be modelled as box-limits in the spline parameter space, interdependency of parameters is omitted. Shoulder joint-constraints are modelled in [23] using quaternion field boundaries. The orientation of the shoulder joint is sampled from motion capture data in quaternion space. Radial basis-functions are used to reconstruct an implicit surface representation of the boundary of the feasible quaternions. Our approach has similarities with this method, as it uses an implicit surface and supports general nonconvex joint-constraints. However, our projection operator is superior as it is a constant time operation, i.e. it is not an iterative procedure.

The approach of [23] is further developed in [59] where a hierarchical model of joints is presented. It seems that this approach can only handle a single parent hierarchy as there is no information about deeper hierarchies. In both papers, the main focus is on the machine learning part of building the joint-constraints, while our focus is on the joint-constraint model itself.

Table 1 is a summary of a comparison study between key-properties of the above mentioned methods and our signed distance-field (SDF) joint-constraint model. All the compared models can essentially be seen as geometric models of the boundary of the feasible region. Their differences lie mostly in their choice of geometric model for representing the boundary and in the actual computational representation. Finally, there are some differences in the technicalities of how the back-projection operator and feasibility tests are supported.

As Table 1 shows, the SDF model offers more generality than its alternatives, while supporting constant time operations.

Projection operators fall in two categories: constant time operation or iterative search schemes. One major feature of our SDF model is that the projection is a constant time operation. The only other model that can offer this, is the box-limit model. On the other hand, the memory usage can be high. In our work, each joint needs $I \times J \times K$ cells of a map, storing a regular sampling of the distance-field, where I, J, K denote the resolution of the map along the three axes. Adaptive distance-maps could be used in place of a regular sampling. However, our results show that in most practical cases, coarse maps can be used and so, the memory usage can be

Model	Model Representation	Data Representation	Projection Operator
Box Limits	Explicit	Vector of intervals	Constant time projection
Sinus Cones	Implicit	Implicit function	Root-search problem, Iterative Newton method
Reach Cones	Explicit	Set of connected tetrahedra	Linear search for closest tetrahedron
Spline Joints	Para-metric	B-splines	Not available
Quaternion Boundary Fields	Implicit	Radial basis functions (RBF)	Root-search problem, Iterative Newton method
Signed Distance-Field	Implicit	Signed distance-field	Constant gradient projection

Model	Feasibility Test Operator	Model Capabilities
Box Limits	Constant time verification of enclosing interval	Convex/ Boxed
Sinus Cones	Evaluation of closed form solution	Convex/ Ellipsoid
Reach Cones	Linear search for enclosing tetrahedron	Nonconvex, but no holes
Spline Joints	Infeasible poses are not allowed	General nonconvex
Quaternion Boundary Fields	Global support of RBF convolution of all samples	General nonconvex, difficulties with holes
Signed Distance-Field	Constant time lookup of distance value	General nonconvex

Table 1: A comparison study of key-properties of various joint-constraint models.

kept at an acceptable level.

The predominant trend in previous work in this research area is to consider only local joint-constraints models, the exception being [26] where the global dependency of joint parameters is considered in the case of forward kinematics. In this respect, our work is no different, our model is also a local model. However, we do consider the full dependency between joint parameters within a single joint. Neither box-limits, sinus-cones nor reach-cones offer this.

One final aspect of joint-constraints, on which we will elaborate below, is the ease with which one can set up the model. In machine learning, this is termed model selection and could be approached as a nonlinear regression problem [28]. Some models can employ sampled feasible poses for setting up the joint-constraints, the SDF model is one such model. Quaternion boundary fields share similarities with our approach in this regard, although their coordinate basis (imaginary part of a quaternion) is non-trivial to work with. To our knowledge, no prior work addresses model selection of sinus-cones or reach-cones.

The presented SDF model offers full modelling generality of local joint-constraints with constant time operations and easy model selection. This makes the presented model a novel method for obtaining more accurate joint-constraints, highly suitable for motion simulation of articulated figures.

14 The Signed Distance-Field Model

Even though the model presented here is a local model, it could be extended to cover inter-joint dependencies. Our base assumption is that the boundary of the feasible parameter space of a joint forms a single connected component. This implies that feasible joint motion is a connected subset. Thus, the test for feasibility is reduced to a simple inside/outside test. Thus, in case of an infeasible configuration, the point is projected unto the closest feasible point.

We use Z-Y-Z Euler angles as basis, where the orientation is specified by the angles $\mathbf{p} = \begin{bmatrix}\phi & \psi & \theta\end{bmatrix}^T$. This allows us to work in a 3D space rather than 4D or 9D as would be the case for quaternions and homogeneous coordinates. By modelling the motion range geometrically, we have a broader basis of well-known geometrical representations to choose from, e.g. polygonal meshes, tetrahedral meshes etc.

As performance is of great importance, the geometric representation must support two constant time operations: Verification of a feasible joint pose and projection of infeasible joint poses back onto the boundary of the feasible region. Distance-fields are known to offer both these qualities, but at the cost of extra memory usage. A distance-field is an implicit representation of the geometry, defined by the function $\Phi : R^3 \mapsto R$ where $\mid \nabla\Phi(\mathbf{p}) \mid = 1$ everywhere and $\Phi(\mathbf{p}) = 0$ for all $\mathbf{p}$ corresponding to the geometry.

14.1 Building the Signed Distance-Map

In the following we describe how we compute the discrete signed distance-map from the continuous signed distance-field. In principle, any signed distance-field algorithm could be used [60]. For this work, we used a simple brute-force approach. Because the signed distance-field rigging is done as a preprocessing step, the cost of generating the signed distance-map is of minor importance. In fact, we did not even bother storing the preprocessing for our tests. Our experience shows that a fairly coarse resolution is sufficient, as the set of motions we study only requires a low number of temporal samples. Running this process off-line means that generating the distance-map is not a bottleneck, as might have been suspected. Table 2 in section 15.1 lists runtime statistics for generating the signed distance-maps of a jumping motion.

Although we use a simple brute-force method in constructing the distance-map, we will – for completeness of presentation – give a full detail description. Initially, we create a regular grid of a user specified resolution. The grid is located in space such that the minimum and maximum corner points of the grid is within the angle interval bounds of the Euler angles:

$$\begin{bmatrix}\phi\\ \psi\\ \theta\end{bmatrix} \in \begin{bmatrix}0, 2\pi\\ 0, 2\pi\\ 0, 2\pi\end{bmatrix}. \tag{63}$$

Once the grid has been created all distance values stored at the grid nodes are initially set to ∞.

$$\Phi(\mathbf{p}) = \infty \quad \forall \mathbf{p} \in \mathcal{N}, \tag{64}$$

where $\mathcal{N}$ is the set of all grid nodal positions in the map.

1. Next we sample the joint motion from some exemplar based motion,$\{q^i\}_{i=1}^{i=N}$, for instance some motion capture data.

2. For each time sample q^i we extract the Euler angles of the present joint

$$\mathbf{p}^i = \begin{bmatrix}\phi^i\\ \psi^i\\ \theta^i\end{bmatrix} \leftarrow q^i. \tag{65}$$

The Euler angles are collected in a chronologically ordered list $\{\mathbf{p}^i\}_{i=1}^{i=N}$.

3. For each node of the grid, we compute the distance to the closest sample point. Running through the entire list of samples we check the distance to the current sample. If the newly computed distance is less than the distance value currently stored in the corner node then we replace the stored value with the new value. This gives the following selection operation:

$$\Phi(\mathbf{p}) \leftarrow \min_{i=1..N} \left\{ \Phi(\mathbf{p}), \| \mathbf{p}^i - \mathbf{p} \| \right\} \; \forall \mathbf{p} \in \mathcal{N} \tag{66}$$

The temporal sampling of the motion might be too coarse, thereby creating multiple disjoint regions in the final signed distance-map. There are at least two ways of dealing with the sampling problem. One may adopt a naive approach: Detect if the problem occurs and then redo the entire motion sampling at a finer resolution. This could be done by detecting how many connected components we have in the final map. However, a more intelligent approach is to use an adaptive sampling: Given a grid resolution we can adaptively modify the motion capture sampling rate to ensure that the distance between any two consecutive motion samples is never larger than half the maximum grid spacing. We use the adaptive motion sampling strategy in our implementation.

14.2 Adaptive Sampling of Motion

We use a chronologically ordered list of our samples to interpolate between neighbouring samples. Coarsely sampled regions are subsampled using spherical linear interpolation(slerp [61]) in angle space. We can subsample the region between two neighbouring samples as densely as necessary, to ensure that the distance between the new samples are never greater than half the grid resolution. The resulting quaternions are then converted into Euler angles and added to the list between the existing samples.

Figure 18 summarizes the adaptive sampling algorithm. If disjoint motion is encountered, some interpolation scheme must be established to connect the manifolds of the unconnected motions. For testing purposes, we simply interpolate between the last frame of one animation with the first frame of the second animation. For the test sets used, this simple approach works well. It should be noted that this would not generally be sufficient. It should be ensured that the motion samples are not disjoint, meaning the motions should share at least one common point.

```
$n = 0$
$t = 0$
$p^n \leftarrow q^n \leftarrow$ **current sample at time** $t$
**for** $n = 1$ **to end**
    $q^{n+1} \leftarrow$ **next sample at time** $t + \Delta t$
    **while** $distance(q^n, q^n + 1) >$ **grid spacing**$/2$
        $\Delta t \leftarrow \Delta t/2$
        $q^{n+1} \leftarrow$ **next sample at time** $t + \Delta t$
    **end**
    **if** $($ $distance\ to\ small)$ **increase** $\Delta t$
    $n \leftarrow n + 1$
**end**
```

Figure 18: Adaptive motion sample algorithm. Observe that he Δt variable can be seen as a kind of trust region radius.

14.3 The Signed Distance Property

We know that all motion samples are feasible poses and we can assume with some certainty that not all feasible poses are represented. Further, we know that some poses are in the interior of the region and some are on the boundary. In some cases we want to treat all samples as if they were lying on the boundary. This would be the case when we have a very restricted motion from which we have built our signed distance-field, and we want to use these to make it possible to mimic the specific motion from which the samples have been taken. Thus, one may wish to add an additional pass to the distance-map generation algorithm. We want to implicitly add the interior void to our representation. The above part of the algorithm only represents the samples as the feasible position and has no notion of what the interior is. To add such a notion one may convert the distance-map into a signed distance-map. The idea is to place a positive sign on some outside border cell of the grid. Then one simply performs a region filling operation where all neighbouring cells not crossing the zero level-set is given a plus sign. In the end, all unassigned cells must be interior and are given a minus sign. One flaw in this approach is that the region must be a closed manifold, otherwise it will be hollowed out and there will be no interior region assigned. This can be helped by either ensuring that the motion is sufficiently dense sampled or by using an alternative method for determining boundary cells. Ensuring the density of the samples has shown to be difficult [23], and for this application the extra work is not needed. The single motion sampling represents a very limited area. In this setting the above mentioned subsampling is sufficient to ensure the signed property of the map. For sculpted joint-constraints on the other hand, the signed distance-map property must be handled as we will show in the next section.

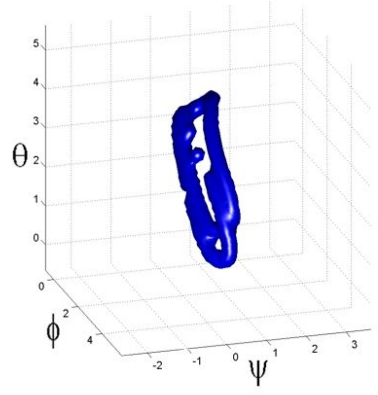
14.4 Sculpted Signed Distance-Field Joint-Constraints

It may be difficult to obtain enough motion samples to automatically generate signed distance-fields. Also, one may wish for the option to have specific control over the motion and directly model the behaviour. Using a geometric representation, such as distance-fields, allows for an easy and direct approach for modelling signed distance-fields, for instance by using sculpting tools [62]. As a proof-of-concept to support our claim we used a simple setup exporting the surface of a signed distance-field as a polygonal mesh and then used the open source 3D modelling program Blender [1] to tightly fit a two-manifold to the data, using an enclosing sphere and the shrink-wrap modifier. An example of a feasible region obtained in this way is shown in Figure 19(b). An advantage of this approach is that a non-convex manifold can be easily obtained and is easily modifiable, either by directly manipulating the mesh or by manipulation through some intuitive interface such as modifying the feasible region as a polygon on a sphere. Figure 20 shows the result of applying the sculpted signed distance-fields.

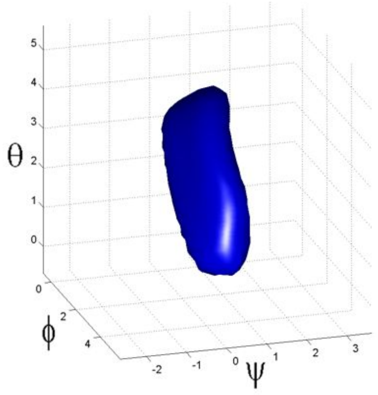
The shrink wrap method used in this example is a way of filling out the empty parts of the feasible region. For completeness, we will give a brief description of the way this modifier works. For a full description we refer to Blender documentation [1].

1. Choose an enclosing two-manifold object e.g. a sphere.
2. Project all points of the enclosing object inward onto the surface of the target object.
3. Generate a new signed distance-field from this object.

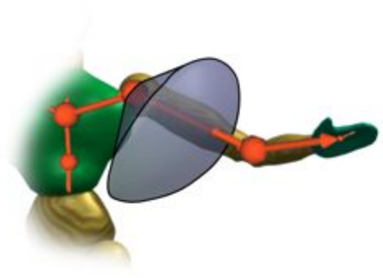
Note that the original shape of the two-manifold object has a significant impact on the quality of the resulting joint-constraints. We have chosen this approach as an easy-to-use and intuitive approach, to show the possibilities of signed distance-field joint-constraints. Other more elaborate techniques for constructing a surface from a point cloud, such as the one described



(a) Sampled joint-constraints.



(b) Sculpted joint-constraints using shrink-wrap.



(c) Sculpted joint-constraints shown in Cartesian space.

Figure 19: An example of refining the sampled joint-constraints of the left shoulder using an enclosing sphere and a simple shrink-wrap modifier from Blender [1].

in [23], may give more correct results. Either technique could be used together with our method.

14.5 Applying Signed Distance-Field Joint-Constraints for Inverse Kinematics

We use an inverse kinematic (IK) modelling approach similar to [11], where the IK problem is formulated as an optimization problem which is solved using an iterative line-search method. The feasibility of a given IK iterate, $\mathbf{p}$, is determined by testing the corresponding Euler angle distance value in the signed distance-fields:

$$\Phi(\mathbf{p}) \leq \varepsilon \Rightarrow \mathbf{p} \text{ is feasible}. \quad (67)$$

The parameter ε is a threshold value to counter round-off errors. The impact of this parameter is analysed in our results section. The signed distance-field lookup is performed in constant time by indexing the surrounding grid points and performing a tri-linear interpolation at the lookup position. Infeasible poses are projected back onto the feasible region, by moving in the opposite direction of the distance-map gradient. The gradient is computed as a central finite difference approximation:

$$\nabla\Phi_{i,j,k} = \begin{bmatrix} \frac{\Phi_{i+1,j,k}-\Phi_{i-1,j,k}}{2\Delta i} \\ \frac{\Phi_{i,j+1,k}-\Phi_{i,j-1,k}}{2\Delta j} \\ \frac{\Phi_{i,j,k+1}-\Phi_{i,j,k-1}}{2\Delta k} \end{bmatrix}, \tag{68}$$

where Δi,Δj,Δk denote the grid spacing along each coordinate axis. The gradients at the grid nodes surrounding $\mathbf{p}$ are interpolated using a tri-linear interpolation on each component of the gradient. The projection of the infeasible $\mathbf{p}$ is then:

$$\mathbf{p} \leftarrow \mathbf{p} - \Phi(\mathbf{p})\nabla\Phi(\mathbf{p}). \tag{69}$$

The central difference approximation of the gradient may cause numerical dissipation in the computation of $\Phi(\mathbf{p})$ and $\nabla\Phi(\mathbf{p})$. To alleviate this, the projection (69) can be applied twice, this does not make the procedure iterative. It is merely an implementation safeguard against numerical dissipation. Due to the distance-map properties, no more than two iterations is needed, this gives a constant time operation.

15 Results and Discussion

We have chosen to verify and validate the signed distance-fields in the context of IK. We use the box-limit model as a base of reference, mostly due to its widespread use in interactive application and because it is the model currently used in human motion tracking. We have excluded the alternative methods from the tests, since none of them live up to both the time and modelling demands of our application. We focus on performance and accuracy.

15.1 Building the signed distance-maps

In our tests, a brute-force approach was used for building the distance-maps from motion samples. Table 2 shows the timings for generating the signed distance-maps for the joint-constraints of a single jumping motion.

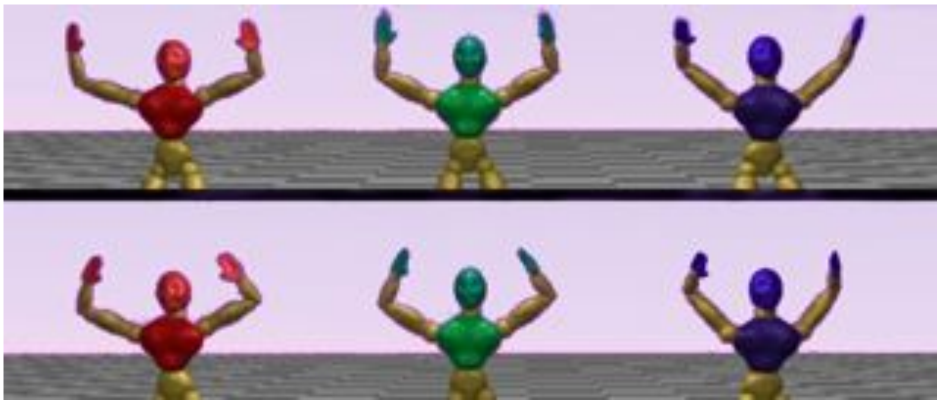


Figure 20: Comparison study of animation quality when using box-limits versus signed distance-fields. The figure on the left shows signed distance-fields, the figure on the right shows box-limits. The figure in the center is the motion capture reference.

Grid Resolution	# Motion Samples	Storage Requirement	Computing Time (secs.)
$8 \times 8 \times 8$	200	4 kB	0.21
$16 \times 16 \times 16$	200	32,7 kB	1.34
$32 \times 32 \times 32$	200	262 kB	8.30

Table 2: Timings of brute-force approach for distance-map generation for 30 joints in a jumping motion sample. Observe that even for detailed maps, the processing time is acceptable as this is a preprocessing step.

15.2 Parametrizing the Distance-Field Joint-Constraints

The signed distance-field model is dependent on the user specified grid resolution $I \times J \times K$ and the ε parameter. The parameters are orthogonal in the sense that grid resolution mostly influences the signed distance-field generation, while the feasibility threshold parameter is a run-time only parameter.

We have tested different grid resolutions and it turned out that for our single motion sampled signed distance-fields there seemed to be a threshold around a resolution of $16 \times 16 \times 16$. For finer resolutions, the animations ran smoothly and with acceptable quality. For coarser resolutions, the generated motions were jagged and tended to get stuck.
For sculpted signed distance-fields, which tend to be more connected (see Figure 19(b)), the resolution could be set much lower. Resolutions of $8 \times 8 \times 8$ or lower were acceptable in this case.
For the single motion case, the ε parameter needs to be large enough to ensure the existence of a solution for all poses, yet small enough to counter

loose constraints. We observed that for values above 0.2 radians, the constraints are too loosely fitted, and for values below 0.2 radians the motion is jittery and the IK solver tends to stall.

15.3 Constant Time Performance

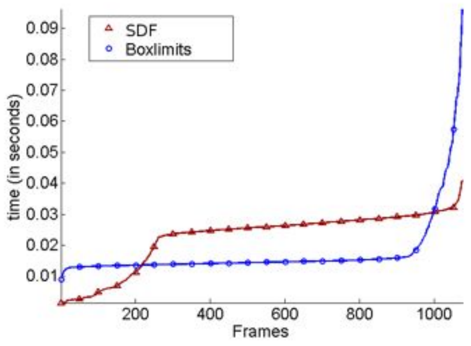


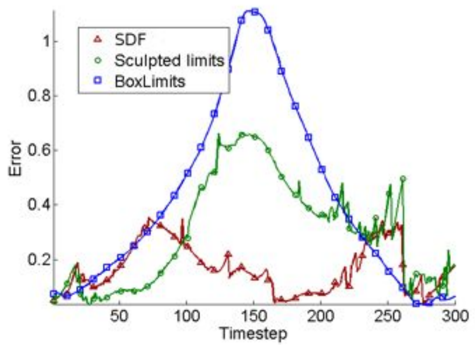
Figure 21: Computation times of box-limits versus signed distance-fields when used in inverse kinematics. The measurements are ordered by value to make comparison easier. Notice that the box-limits are generally faster but cannot guarantee the same upper bound on computation time as the signed distance-fields.

We have measured the computational time in an application, where the end-effectors of an IK solver are driven by the corresponding end-effector positions in a motion capture example. The measurements are performed on the IK solution. Figure 21 shows a plot of our measurements.

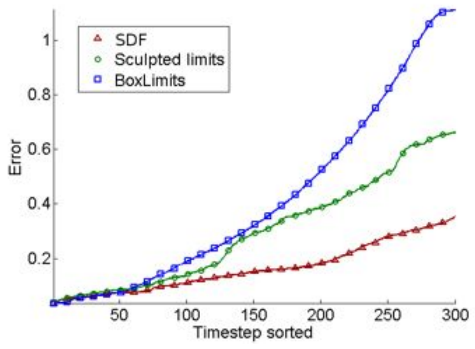
Our experiments show that the signed distance-fields are slower than the box-limits. This was to be expected. We can also see that the worst case performance of the signed distance-fields is much better than worst case for the box-limits. In the worst case the signed distance-fields still achieves approximately 20 fps, while the box-limits only reaches approximately 10 fps.

15.4 Increased Modelling Accuracy

Using the more constrained signed distance-field model, we expect a gain in accuracy as this should reduce the redundancy of the simulation. The test uses an elbow joint, this joint is a child joint of the shoulder which has a wide rang of motion. We measured the deviation of the joint, using both



(a) Plot of error compared to motion capture.



(b) Accumulated error over time.

Figure 22: Plots showing the difference between the right elbow 's global position using the different joint-constraints. The more restrictive joint-constraints cancel out the effect of the redundancy of the joint, thus getting a solution closer to the motion capture reference. 1 unit on the axis corresponds roughly to 20 cm in real world measurements.

sampled, sculpted, and box-limits, shown in Figure 22(a). As expected, the results show that the box-limits does not constrain the position very much. The box-limits result in an error which, transformed into real world measures, corresponds to an error of 20 cm. The simulated motion is shown in Figure 20.

16 Conclusion

We have presented a novel joint-constraint model using signed distance-fields.

The fitting of our model, although loose on account of the threshold, is still the tightest fitting model, compared with the other presented models as seen in Figure 1.

The model supports general non-convex joint-constraints of 3 degrees of freedom and works well with sampled motion data. The memory usage is cubic in the resolution of the mesh. However, in most cases it is possible to get by with a comparably low resolution, in which case the memory usage is acceptable. The assumption of locality has shown to be insufficient, e.g. the orientation of the hip joint indeed has an effect on the joint-constraints of the knee. Therefore, an interesting venue for further research would be to extend our model to handle more than 3 degrees of freedom.

Part III

Activation splines

The Joint Limits conclude the work I have done with motion planning. Now we move on to something even more interesting. How to actuate a body in a dynamics simulation, to actually do the motions, we have planned in the previous chapters.

The first thing you need, when getting something to move, is a motor. The human body use muscles for actuation, so the development of a muscle model is the logical next step.

To develop a muscle model, it is necessary to decide on the application of the model. Software packages, such as Maya, has inbuilt methods for animating muscles, but what I wanted was something more similar to the models used in bio-mechanics something that would have the possibility to replace the existing methods in software such as anybody but would still retain the interactive nature of simple linear actuators. The Idea was to let the actuator remain a one dimensional spline but embed it into a deformable body of arbitrary complexity. This way we get something that is potentially as complex or simple as is needed.

The Work presented in this part is done in collaboration with Kenny Erleben. It is not yet published but a paper presenting this method has been submitted to the SCA 2012 Symposium on computer animation. The following is an extended and modified version of that work.

17 Activation splines

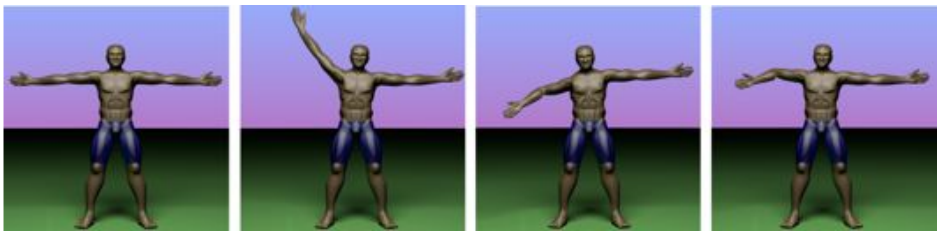


Figure 23: A man flexing his arm. The simulation was made using 5 spline muscles in the shoulder and arm.

We present a method for simulating the active contraction of deformable models, usable for animation of characters and soft deformable objects. We propose to embed splines into a deformable model to control deformation patterns. The splines model a type of contraction force that is transferred to the ambient space of the deformable model. We present a novel physical principle as the governing equation for the coupling between the low dimensional 1D activation force model and the higher dimensional 2D/3D deformable model. The coupling model is robust towards artistic force splines that might overlap and share their region of influence in the deformable model. Our activation force method works well with both mass-spring systems, finite volume, and finite element simulations. The contraction method we propose for our splines is shown through experiments to be robust and allows for rapid and responsive control of the deformations. Our activation splines are easy to set up and can be used for physics based animation of deformable models such as snake motion and locomotion of characters. Our approach generalises easily to both 2D and 3D simulations and is applicable in physics based games or animations due to its simplicity and low computational cost.

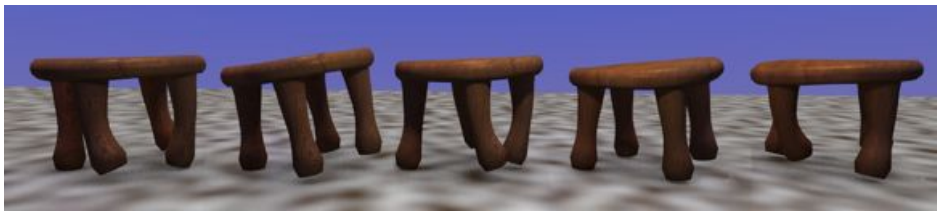


Figure 24: A cartoon stool rigged with an activation spline in each leg. A simple impulse activation signal controlling the contraction of the leg activation splines produces a gait for the locomotion of the stool.

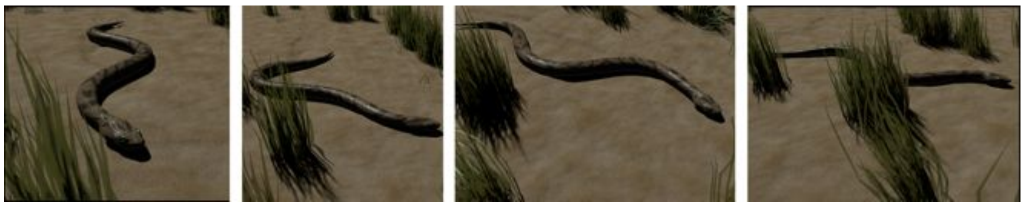


Figure 25: Our system is used to make a snake crawl on the ground. The snake contains a left and right activation spline running from head to tail. Two cosine functions with a phase shift control the contraction that activates the spline. A simple anisotropic Coulomb friction model is used to model friction between the snake's skin and the ground.

18 Introduction

Making deformable models move by themselves in computer animation, as though they have internal muscles, is a time consuming process both in terms of rigging and computational cost. Existing approaches work through external dynamic constraints acting between bodies like creating point-to-nail constraints on the fly. Although an animator may control this, the approach uses an external fictitious view to create the actuation rather than an internal more physically correct view. Other approaches use accurate muscle models with a micro scale view of muscle fibres. These models are hard to come by as only specialists know how to create the correct fibre data and the models require a substantial computational effort. Thus, we believe such models are not easily adaptable by animators or feasible in an interactive application context.

We present a method which allows animators to model activation forces with splines embedded in the passive mesh. Each spline affects a region of the mesh. The splines are intuitive, as they can be understood as tendency lines of the true continuum field of activation forces. Thus, our spline model reduces time consumption in rigging compared to a muscle based model. Existing physically realistic methods are computationally heavy, taking hours of wall clock time to simulate seconds. Our method is conceptually closer to the traditional line-based models such as Hill [63] and Zajac [64] . However, our approach naturally overcomes the difficulties in having to model via points to ensure proper physical correct bending moments. Our work allows for a natural coupling between the activation forces and the resulting deformations. This is ignored in most line-based modelling. Our spline model bridges the gap between the physically accurate, but computationally heavy, methods, and the classical biomechanics models which are computationally efficient but typically lack deformation and collision support. Our novel contributions are,

- A novel force equivalence principle for transferring coarse low 1 dimensional representation of activation forces to a fine high 2/3 dimensional representation of the activation force field.
- A generalization of line-based force models to arbitrary smooth curved lines.
- An orthogonal approach to extend line based force models to include a physical realistic passive deformation of the deformable models.
- A unique robust and responsive method for contraction, which works by altering the rest shape of a spline, rather than tweaking physical parameters such as stiffness.

We present parameter studies, demonstrating that our method is not sensitive to specific modelling of the splines and that complicated deformations are possible with the novel contraction parameter. Thus, our approach is suitable for artistic created models as it allows for overlapping splines and can transfer activation forces to all regions of the deformable model . Several animation examples show that the method produces robust animations, resulting in interesting motion with little rigging or animation time.

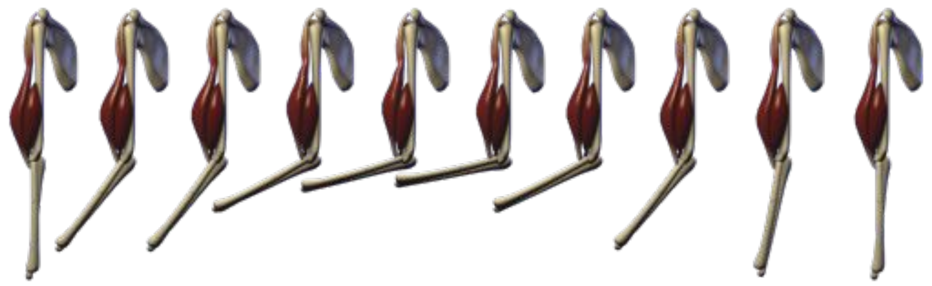


Figure 26: An animation of a biceps muscle bending an arm. The simulation does simple collision handling and uses gravity to pull the arm down after flexing. Our method was not developed for biomechanics. However, as the example shows it can capture some of the real world muscle effects.

19 Related Work

Early work in computer graphics focused on simple kinematic deformations [65] or cage-based deformations such as free form deformation (FFD) [66]. Later, energy constraints were introduced for adding constraints between parametrized models [67] and generalized [68]. A more physical approach used the finite element method (FEM) for simulation of deformable models and added constraints using Lagrange multiplier method [69]. Some of these early ideas have been optimized by precomputing of principal components [70] to build responsive dynamic textures [71]. There have been kinematic approaches to make a connection between deformation and muscle motion. An example of this is the approach by Aubel and Thalmann [72] in which a visualization mesh is wrapped around a spline and volume preservation is approximated, or Lee and Ashraf [73] which makes passive deformation of simple fusiform muscles. None of these kinematic or animation approaches handle the activation of the deformable models which is our main focus in this work.

Activation of mass spring systems have been investigated previously in computer graphics where a mass-spring lattice/cage mesh is used for both activation forces, the deformable model, and handling of collisions [74, 75]. The cage ideas have been extended to use hierarchical FFD as the lattice [76] . These are all best classified as Eulerian approaches to the activation forces whereas ours is a Lagrangian approach. We embed lower dimensional 1D spline polylines into the 2D/3D deformable model rather than caging the deformable model. For the coupling between the two different physical systems we apply a physical principle to tie together the deformable model and the actuation model. No such physical principle is used in previous work[74, 77, 75, 76] as their cage holds both the deformable model and collision/contact response.

Our contribution is based on a physical principle for coupling our activation model with the deformable model – no previous work is as far as we know based on a physical principle for this. Our approach allows the environment to interact directly with the deformable model rather than with a cage. Our approach also allow for any deformable model being mass-spring systems, finite element/volume methods or to be used together with our activation force model.

Tu, Grzeszczuk and Terzopoulos [78, 75] use mass spring systems in the same manner as [74]. The focus in this work is on learning motion patterns for these types of mass spring models the focus on our work is on the actual activation force model. These authors vary their activation force springs using a linear interpolation between a prescribed fully contracted rest length

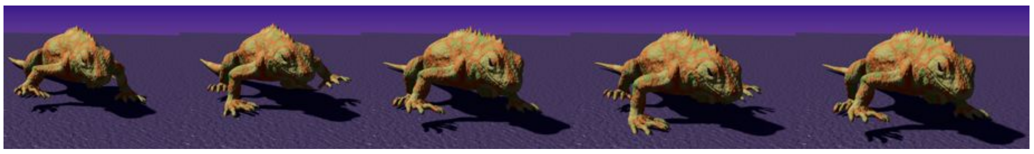


Figure 27: A children toy rubber lizard walking using 6 activation splines. Two are used to flex the spine and one is placed in each leg. The activation is controlled by phase shifted cosine functions. (3D model by Kevin Hays.)

and the initial relaxed rest length in contrast we use a simpler linear scaling of the initial rest length. This allow our activation forces to contract to zero length if needed or even expand far beyond the initial rest length. Our choice seems more reasonable for purpose of general animation.

Space-time optimization problems use a force field applied on the whole 2D/3D deformable model to actuate the deformable model and make it move in some desired manner [79, 80]. This means that any external force can be applied to a model to make it move into some desired shape. This is an unnatural activation force. Our approach models activation forces much more similar to the approaches taken for articulated figures where activation forces live in the joint space of the model [81, 19]. Hence for deformable models the control parameter vector scales linear with the number of vertices in the deformable model mesh. In our case we have as little as a single parameter for a whole embedded spline. Hence our control parameter vector scales linear in the number of splines we apply. Space-time optimization problems are not the focus of this work but we speculate that these could benefit from a low dimensional activation force model such as ours.

Simulation of muscles has been researched in computer graphics, computational physics and engineering, and biomechanics communities for years. Viceconti et al [82] published a review of the current state of the art of modelling the musculoskeletal apparatus. They state that most full body simulators use linear muscle-actuators, ideal joints and infinitely rigid bones, and stress that this is not an optimal modelling, since it neglect the factors of muscle interaction and non-linearity of the force splines. In the review by Pandy and Andriacchi [83], the function of human muscles is extensively reviewed and several ways of measuring muscle activation are compared to current models. Computational modelling is described as a valuable tool in the analysis and explanation of human joint and muscle function.

Most approaches use a FEM method [84, 85, 86] or a finite volume method (FVM) [87, 88] to model the muscle and use a three dimensional vector field to represent the fiber direction. The fiber direction may be modeled from real life observations as a constant vector field. All of these approaches are off-line methods. In comparison, our approach simplifies the modelling of the direction of the activation force field in exchange for less computational complexity. Our method can handle branches of the deformable models and multiple force directions.

More accurate muscle models in computer graphics includes Teran et al [87] which use B-spline solids to represent the fibre direction. Their approach is further developed and applied to a 3D muscle test case of the arm and shoulder complex in [88]. Their fibre representation is mixed with the constitutive equations for the muscle tissue. In contrast our work completely separates the activation force model from the constitutive equation of the deformable material. Hence, we do not need to worry about non-linear elasticity, plasticity or volume preservation as these properties are determined from the deformable model simulator. Lastly, our work use 1D spline curves rather than spline solids that must cover the whole deformable model. This makes for a much more intuitive interface. In principle our model can be perceived as a graceful degradation of the spline solid. In the limit of adding infinite many 1D splines our model would become a spline solid. As our 1D embedded splines do not need to cover the whole region of the deformable model they are much more easy to rig. One major benefit of the 1D splines over the spline solids is the ability to handle multiple directions within the same tissue. The spline solid approach lacks this ability completely.

Porcher-Nedel and Thalmann[89] used a polyline representing the action line of a muscle. The muscle surface is modelled using a mass spring system. The idea is refined by Aubel & Thalmann[72]. The polyline is moved using either pre defined behavior or a 1D damped mass spring system. The mass spring system is not unlike our activation spline. However, our splines are not restricted to follow the line of action but can be embedded anywhere in the deformable model, Thus providing much more generality and flexibility for an animator. Aubel and Thalmann use attractive and repulsive force fields to constrain the polyline and pre-defined behaviors – pinned guide points – for dealing with attachments and proper wrap around of the line of action around joints. In contrast our approach does not rely on force fields to deal with constraints or guide points instead we use the physical simulation of the deformable model to deal implicitly with all these issues. Thus, our approach requires much less modelling input and simpler rigging.

During simulation Aubel and Thalmann updates local coordinate frames at points on their polyline then the muscle surface vertices are kinematically coupled to the motion of the polyline. In our work the coupling is fully dynamic and based on a physical equivalence principle. Further, in our work we address how to choose the mass spring parameters to yield rapid response and we introduce a control parameter for controling the dynamic behavior of our activation spline.

Sueda et al. [90] describe a framework for modelling muscle and specifically tendon movement in hand animation, originally presented in [91]. It has some similarity to our approach.The fact that it uses a spline along which it activates and transfers force to the bones is similar. However, it does not have the two way coupling between the active spline and the deformable mesh.

In teh work by Pandy and Garner [92], the focus is on modelling the muscle path, that is the path spanned by the centroid line of the muscle. In this way, it is similar to the piecewise linear Hill type muscles, but it extends the linear segments with curved segments. Their approach cannot handle muscles with fibres, not following the centroid line. The work by Pandy and Garner did not consider deformation – their work allowed classical line models to wrap around bones using via points – their focus is the obstacle problem for classical usage of 1D muscle models . Our work makes via points unnecessary as the wrap around effects would be handled implicitly by the deformable model simulation (contact forces between deformable model and rigid body model).

We extend these existing approaches by presenting interactive behaviour, general non-muscle shapes, such as a stool, and intuitive rigging of the muscle fibre direction allowing animators to e.g. paint activation splines into a deformable model.

20 The Activation Spline Method

We present a system consisting of a passive deformable model represented by a volume mesh and one or more splines, used to represent the activation forces in the deformable model. On a coarse level, our approach can be described as follows

- As a pre-processing step, activation splines are discretized and embedded into the deformable model such that the splines will follow the motion of the deformable model once simulation starts.

- During run time, our simulation loop's first step is to compute activation forces on the activation splines. This can be viewed as a coarse low dimensional representation of the true activation force field in the deformable model. A novel approach is taken to control the activation forces, where we change the rest shape of the activation splines rather than their material parameters such as elastic stiffness.

- We formulate a novel force equivalence principle as the governing equation for the force transfer between spline and volume mesh. This allow us to transfer the coarse activation forces to the entire influence region in the deformable model.
- We feed the whole volume mesh activation force field to a deformable model simulator as an external force field. This allows us to use any deformable model simulator as a black box simulator. Note that it is only an "implementation" trick to apply the activation forces as external forces. In reality the activation forces are internal forces in the whole model – Hence, our activation force model is a Lagrangian model. In principle this can be viewed as an interleaved simulation approach [93, 77].
- Having obtained a new state for the deformable model we update the spline position before we initiate the next step of our simulation loop.

The simulation loop is sketched out in Figure 28.

To ease notation, and without loss of generality, in the following, we will describe our method using a single spline. We assume that an animator or modeller have created a spline with K control points and any point on the spline can be found using

$$\mathbf{p}(s) = \sum_{k=1}^{K} N_k(s)\mathbf{g}_k \tag{70}$$

where $\mathbf{p} \in \mathbb{R}^D$ with $D = 2, 3$ and $N_k : \mathbb{R} \mapsto \mathbb{R}_+$ is the k^{th} global basis function of the spline. The vector $\mathbf{g}_k \in \mathbb{R}^D$ is the corresponding control point and $s \in [0..L]$ is the spline parameter. The spline is assumed to be inside the volume mesh having V vertices, where $\mathbf{x}_j \in \mathbb{R}^D$ is the coordinates of the j^{th} vertex.

The governing equation of motion for any deformable model can be written abstractly as

$$\mathbf{M}\frac{d^2\mathbf{x}}{dt^2} + \mathbf{C}\frac{d\mathbf{x}}{dt} + \mathbf{k}(\mathbf{x} - \mathbf{x}_0) = \mathbf{F}_{\text{ext}} \tag{71}$$

where $\mathbf{M}$ is a mass matrix, $\mathbf{C}$ is a damping matrix and $\mathbf{k}$ is the elastic forces that depends on the current displacement field $\mathbf{x} - \mathbf{x}_0$. For linear elastic

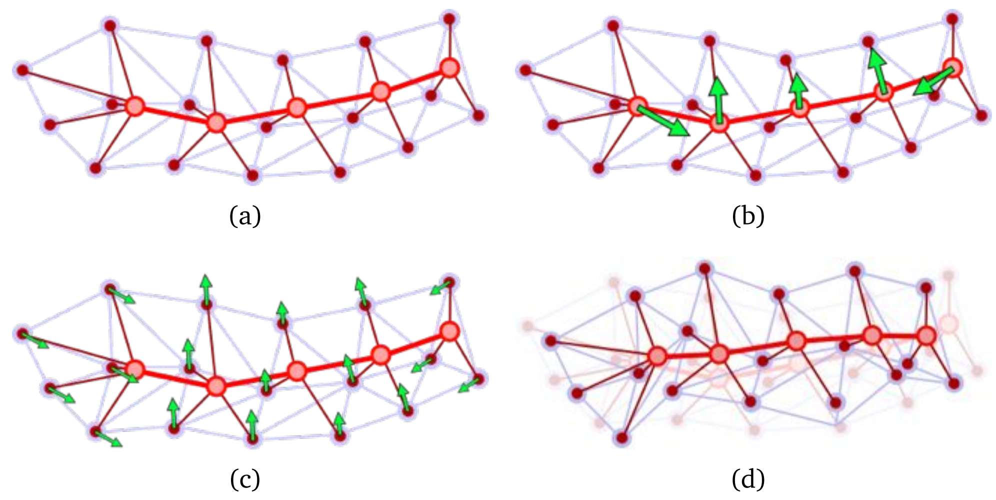


Figure 28: Illustration of our pre-processing phase and simulation loop. As a pre-processing step (a) we bind a spline to a volume mesh. During the simulation loop we first compute spline forces (b), afterwards we transfer the forces to the volume mesh using neighbourhoods (c). Finally we update the deformable model and move the embedded spline (d).

materials there is $\mathbf{k} = \mathbf{K}(\mathbf{x} - \mathbf{x}_0)$ where $\mathbf{K}$ is the stiffness matrix. The vectors $\mathbf{x}$ and $\mathbf{x}_0$ are the concatenation of the current and initial (rest) mesh vertex positions respectively. Thus, $\mathbf{x}, \mathbf{x}_0, \mathbf{k} \in \mathbb{R}^{DV}$ and $\mathbf{M}, \mathbf{C}, \mathbf{K} \in \mathbb{R}^{DV \times DV}$. $\mathbf{F}_{\text{ext}} \in \mathbb{R}^{DV}$ is the concatenation of external forces. In our simulations, we mostly use gravity which in 3D means

$$\mathbf{F}_{\text{ext},j} = (0, 0, -g)^T \,,\ \forall j = 1..V \tag{72}$$

where g is the gravitational acceleration. We have simple ground reaction forces incorporated which work by inserting a spring force, but without loss of generality we do not describe these here. During a simulation, $\mathbf{k}$ is the elastic passive forces coming from the deformable model itself. The activation spline forces will be coupled to the model by adding a right hand side force term, $\mathbf{F}_{\text{act}} \in \mathbb{R}^{DV}$. We will later add a contact force term to the right hand side, $\mathbf{F}_{\text{con}} \in \mathbb{R}^{DV}$. The resulting equations of motion to be time integrated is written as

$$\mathbf{M}\frac{d^2\mathbf{x}}{dt^2} + \mathbf{C}\frac{d\mathbf{x}}{dt} + \mathbf{k}(\mathbf{x} - \mathbf{x}_0) = \mathbf{F}_{\text{ext}} + \mathbf{F}_{\text{act}} + \mathbf{F}_{\text{con}} \tag{73}$$

In the following, we will detail the steps of the simulation loop and how the forces in the governing equation can be computed.

20.1 Discretization of the Activation Spline

For simplicity we choose to discretize a spline into a sequence of N discrete spline points $\mathbf{p}_i$ and corresponding line segments l_i between two consecutive discrete spline points $i+1$ and i. We label the discrete spline points by $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$,..., $\mathbf{p}_N$ where $\mathbf{p}_i \in \mathbb{R}^D$ for all i. The spline has $N-1$ line segments and the length of the i^{th} segment is given by

$$l_i = \parallel \mathbf{p}_{i+1} - \mathbf{p}_i \parallel \tag{74}$$

For simplicity, we discretize the spline equidistant in parameter space. Thus if $s = [0..L]$ then

$$\mathbf{p}_i = \sum_k N_k(s_i)\mathbf{g}_k \quad \text{for} \quad i = 1..N \tag{75}$$

where $N_k : \mathbb{R} \mapsto \mathbb{R}_+$ is the k^{th} global basis function of the spline and $\mathbf{g}_k \in \mathbb{R}^D$ is the corresponding control point and $s_i = \frac{(i-1)L}{N-1}$. For our test examples this has been sufficient. One could have taken curvature of the spline into account to create a more adaptive discrete version of the spline.

20.2 Embedding the Activation Spline

We embed the discrete spline into a volume mesh by binding the spline points to a volume mesh using mesh coupling. We use barycentric coordinates for an embedding tetrahedron or bi-/tri-linear interpolation for a square or cube mesh [94, 95]. Given a volume mesh with vertex positions $\mathbf{x}_j \in \mathbb{R}^D$ for $j = 1$ to V , we write

$$\mathbf{p}_i = \sum_{j \in \mathcal{C}(\mathbf{p}_i)} w_j \, \mathbf{x}_j \tag{76}$$

where $\mathcal{C}(\mathbf{p}_i)$ is the vertex index set of the mesh cell (triangle, tetrahedron, square, or cube) that contains the point $\mathbf{p}_i$. The values $w_j \in \mathbb{R}$ are the interpolation weights. At initialization, after an animator has defined the spline and its influence region, the spline is discretized. Then, all w_j's for each spline point can be precomputed and stored as attributes for each spline point. During run time the above equation can be used to re-compute and update the current position of the discrete spline point.

When $\mathbf{p}_i$ lies on the border between two elements, simply pick the element at random. If $\mathbf{p}_i$ is outside the mesh then pick the "closest" element. Barycentric coordinates may be used even when $\mathbf{p}_i$ is outside the element. For squares and cubes similar ideas can be used with little modification.

20.3 Computation of Activation Spline Forces

We compute the spline forces at the discrete spline points. This is done by creating a linear damped spring for each line segment of a spline. Let $\mathbf{e}_i \in \mathbb{R}^D$ be the unit direction vector from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$ then the spring force on $\mathbf{x}_{i+1}$ is

$$\mathbf{a}_{i+1}^{i} = -k_i \left(l_i - \alpha_i l_{0i}\right) \mathbf{e}_i - c_i \mathbf{e}_i \mathbf{e}_i^T \left(\mathbf{v}_{i+1} - \mathbf{v}_i\right) \tag{77}$$

and on $\mathbf{x}_i$ we have from Newton's third law of motion

$$\mathbf{a}_i^i = -\mathbf{a}_{i+1}^i \tag{78}$$

where the spline point velocity is given by $\mathbf{v}_i = \frac{d}{dt}\mathbf{p}_i$ for all i. Further, $k_i, c_i \in \mathbb{R}_+$ are the stiffness and damping coefficients respectively. Here l_i is the current length and l_{0i} is the initial rest length pre-computed during the binding process both given by (74). Note that $\mathbf{e}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{l_i}$.

We have extended the standard spring damper law with the parameter $\alpha_i \in \mathbb{R}_+$. This is our contraction parameter which we use to control contraction and extension of the spline. When $0 < \alpha_i < 1$ then the i^{th} segment of our spline contracts. If $\alpha > 1$, we have an extension and for $\alpha = 1$ the activation spline is passive. We can now compute the spline force at the i^{th} spline point as

$$\mathbf{f}_i = \mathbf{a}_i^{i-1} + \mathbf{a}_i^i \tag{79}$$

If a spline force is wanted for an arbitrary s-value the spline basis functions are used to interpolate the wanted value.

In our examples, we control the α-values. In some of our test cases, we use a single α-value for an entire activation spline as in the example of the stool legs shown in Figure 24. In other cases, we use a functional expression to control the contraction of each segment as in the case of the snake in Figure 25. Our initial experiments indicated that we obtained much better controllability of the activation splines by manipulating the α-parameter rather than trying to change k_i and c_i on the fly. This makes good sense as varying k_i and c_i on the fly would change the overall material. This would subsequently affect the time integration method, possibly resulting in a stiff system, which is unsuited for interactive simulations.

We choose the spring and damping coefficients to create a critically damped spring. This is motivated by behaviour of real human tissue. We follow the approach by Barzel and Barr [68] and use

$$k_i = \frac{1}{\tau^2} \quad \text{and} \quad c_i = \frac{2}{\tau} \tag{80}$$

where τ is the characteristic time. In Erleben et. al[96] it was shown that if the deviation of the spring length from rest length should be within a fraction $0 < \varepsilon \ll 1$ after the frame time, characteristic time intervals must be used. Thus, τ should be given by $\tau = \frac{\Delta t}{n}$. We apply this formula to control how fast the spline forces should be able to contract to a given deviation within a frame time-step. Our experiments showed that a value of $\epsilon < 0.01$ gave visually satisfying results.

From our 2D experiments using mass-spring systems for our deformable models, we observed that a spline should be at least twice as stiff as the embedding deformable model to be able to provide a rapid response in a controllable manner.

20.4 The Force Equivalence Principle

We will introduce our novel force equivalence principle between a force on a spline point $\mathbf{p}(s)$ and the volume integral of the activation force $\mathbf{F}(\mathbf{x})$ in a neighbourhood $\mathcal{N} \subset \mathbb{R}^D$ around the spline point,

$$\mathbf{f}(s) = \frac{1}{V_{\mathcal{N}(\mathbf{p}(s))}} \int_{\mathbf{x} \in \mathcal{N}(\mathbf{p}(s))} \mathbf{F}(\mathbf{x}) \, dV. \tag{81}$$

where $V_{\mathcal{N}(\mathbf{p}(s))}$ is the total volume of the neighbourhood $\mathcal{N}(\mathbf{p}(s))$. Thus, $\mathbf{f}(s)$can be interpreted as the average force over the entire neighbourhood.

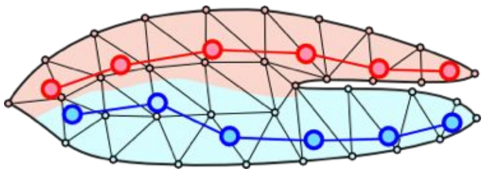
We will use this single principle to create a discrete mapping between activation forces at discrete spline points and discrete vertices in the ambient mesh where the spline is embedded. This single equation forms the main contribution of our model.

We discretize the spline parameter using our discrete spline points

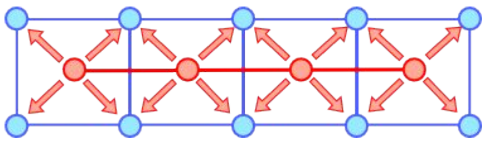
$$\mathbf{f}_i = \frac{1}{V_{\mathcal{N}(\mathbf{p}_i)}} \int_{\mathbf{x} \in \mathcal{N}(\mathbf{p}_i)} \mathbf{F}(\mathbf{x}) \, dV. \quad \text{for} \quad i = 1..N \tag{82}$$

We then define the discrete counterpart $\mathcal{N}_i$ of $\mathcal{N}(\mathbf{p}_i)$.

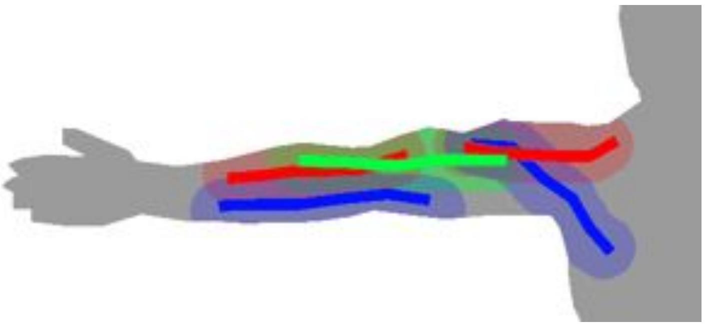
For our FEM simulations, like the muscle animation in Figure 26, we have added all vertex indices in the mesh to $\mathcal{N}_i$, where the corresponding vertex is closer to $\mathbf{p}_i$ than any other $\mathbf{p}_m$ for $m \neq i$. When computing the distance between a spline point $\mathbf{p}_i$ and a mesh vertex $\mathbf{x}_j$ care must be taken to respect the boundaries of the deformable model. The shortest path inside the volume of the deformable model should be chosen. This is important in order to avoid an activation spline in one tentacle of an object affecting a different tentacle of the object. Our method is robust towards the neighbourhood definition and allows for overlapping neighbourhoods. Neighbourhoods could even be defined by animators using a painting tool such as the ones used for skinning. For the 2D simulations using a square,



(a)



(b)



(c)

Figure 29: Illustration of different ways to define the neighbourhood set for different types of meshes.(a) A closest point set, based on geodesic distances inside the volume mesh. (b) A simpler grid based definition. (c) A region defined by a manifold mesh, a distance measure or a function.

mesh we define neighbourhoods to be the four enclosing mesh nodes. For the 3D simulations using cube meshes, we use the eight enclosing nodes. The different neighbourhood definitions are illustrated in Figure 29.

Using the discrete vertex index sets $\mathcal{N}_i$ and the mid-point rule approximation, we can rewrite the integral of (82) as a summation over the discrete volume elements

$$\mathbf{f}_i = \frac{1}{V_{\mathcal{N}_i}} \sum_{j \in \mathcal{N}_i} \mathbf{F}_j \Delta V_j, \quad \text{for} \quad i = 1..N \tag{83}$$

ΔV_j is the volume associated with the j^{th} vertex of the volume mesh. Given a lumped mass matrix $\mathbf{M}$, we get $\mathbf{M}_{jj} = m_j \mathbf{I}_{D\times D}$, where $\mathbf{I}_{D\times D}$ is the D-by-D identity matrix and m_j is the mass of the j^{th} vertex, $\mathbf{M}_{jk} = \mathbf{0}$ for $j \neq k$. For a material with a homogeneous constant mass density, ρ, then $\Delta V_j = \frac{m_j}{\rho}$ and $V_{\mathcal{N}_i} = \frac{1}{\rho} \sum_{j \in \mathcal{N}_i} m_j$. For lumped mass matrices, the following is equivalent to (83)

$$\mathbf{f}_i = \frac{\sum_{j \in \mathcal{N}_i} m_j \mathbf{F}_j}{\sum_{j \in \mathcal{N}_i} m_j}, \tag{84}$$

In all our examples we used (84). We can state the problem as a system of equations

$$\underbrace{\begin{bmatrix}\mathbf{f}_1\\ \vdots \\ \mathbf{f}_N\end{bmatrix}}_{\mathbf{b}} = \mathbf{A}\underbrace{\begin{bmatrix}\mathbf{F}_1\\ \vdots \\ \mathbf{F}_V\end{bmatrix}}_{\mathbf{y}}. \tag{85}$$

Each block of $\mathbf{A}$ contains the mass ratio coefficients defined by (84). A sub block of $\mathbf{A}$ is given by

$$\mathbf{A}_{ij} = \begin{cases}\frac{m_j}{\sum_{j\in\mathcal{N}_i} m_j}\mathbf{I}_{D\times D} & \text{If } j \in \mathcal{N}_i,\\ \mathbf{0} & \text{Otherwise}\end{cases} \tag{86}$$

Once we have solved for the unknown $\mathbf{y}$, we have $\mathbf{F}_{\text{act}} = \mathbf{y}$. In general, $\mathbf{A}$ will not be a square matrix. This depends on the resolution of the discrete spline and the resolution of the volume mesh. In most cases, $\mathbf{A}$ will have more columns than rows. We have $\mathcal{N}_i \neq \mathcal{N}_j$ when $i \neq j$, this means $\mathbf{A}$ can be assumed to have full row rank. If all index sets $\mathcal{N}_i$ are disjoint, meaning $\mathcal{N}_i \cap \mathcal{N}_j = \emptyset$ for all $i \neq j$, the matrix system $\mathbf{Ay} = \mathbf{b}$ can be broken down into N smaller independent systems. These can be solved in a naive parallel manner. This is the case for the walking stool example.

In general, we have an under-constrained system and we solve this by adding the constraint that we want to minimize the norm of $\mathbf{F}_{\text{act}}$

$$\mathbf{F}^*_{\text{act}} = \arg\min_{\mathbf{F}} \frac{1}{2}\mathbf{F}^T\mathbf{F} \quad \text{s.t.} \quad \mathbf{AF} - \mathbf{b} = 0 \tag{87}$$

The first order optimality (KKT) conditions result in the saddle point problem

$$\underbrace{\begin{bmatrix}\mathbf{I}_{VD\times VD} & \mathbf{A}^T\\ \mathbf{A} & \mathbf{0}\end{bmatrix}}_{KKT}\begin{bmatrix}\mathbf{F}^*_{\text{act}}\\ -\lambda^*\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ \mathbf{b}\end{bmatrix}, \tag{88}$$

where λ^* is the Lagrange multiplier for the solution and $\mathbf{I}_{VD\times VD}$ is a VD dimensional identity matrix. The KKT-matrix is a square symmetric indefinite matrix of full rank. Using a Schur complement method, the solution is found to be

$$\mathbf{F}^*_{\text{act}} = \underbrace{\mathbf{A}^T\left(\mathbf{AA}^T\right)^{-1}}_{\mathbf{A}^\dagger}\mathbf{b}. \tag{89}$$

The pseudo-inverse matrix, $\mathbf{A}^\dagger$, can be pre-computed during initialization and allows for a fast run-time solution.

However, the pre-computation can be even more effective, since (88) may be solved using an iterative solver such as preconditioned conjugate gradient method (PCG). This is very fast as $\mathbf{A}$ is usually very sparse. This method scales linearly in the number of mesh vertices, both in memory usage and in computational cost. The drawback is that the approximate solution will not solve (85) exactly. This can cause visible ghost forces if the solution is not sufficiently accurate.

Rather than using a direct method for minimizing (87), a gradient descent method can be used. We note that the gradient of the Lagrangian function $\mathcal{L}$ is

$$\nabla_{\mathbf{F}}\mathcal{L} = \mathbf{F} - \mathbf{A}^T\lambda \tag{90a}$$

$$\nabla_{\lambda}\mathcal{L} = \mathbf{b} - \mathbf{A}\mathbf{F} \tag{90b}$$

We need to maximize wrt. λ and minimize wrt. $\mathbf{F}$. This leads to the iterative updates

$$\lambda^{k+1} = \lambda^k + \nabla_{\lambda}\mathcal{L}^k = \lambda^k + \mathbf{b} - \mathbf{A}\mathbf{F}^k \tag{91a}$$

$$\mathbf{F}^{k+1} = \mathbf{F}^k - \nabla_{\mathbf{F}}\mathcal{L}^{k+1} = \mathbf{A}^T\lambda^{k+1} \tag{91b}$$

Letting $\lambda^0 = \mathbf{0}$ and $\mathbf{F}^0 = \mathbf{0}$, the first iterate yields

$$\lambda^1 = \mathbf{b} \tag{92a}$$

$$\mathbf{F}^1 = \mathbf{A}^T\mathbf{b} \tag{92b}$$

If $\mathbf{A}\mathbf{A}^T \approx \mathbf{I}$ little improvement can be gained from more iterations. Thus, we have the approximate solution

$$\mathbf{F}^*_{\text{act}} = \mathbf{A}^T\mathbf{b}. \tag{93}$$

It is worthwhile noting that using this approach, even if we do not get an accurate solution during the first integration step, the method will converge to the optimal solution over time. For our 2D and 3D simulations, using square and cube meshes, we have applied this gradient solver technique with great success.

Note that the condition $\mathbf{A}\mathbf{A}^T \approx \mathbf{I}$ bares similarity with the Galerkin condition for algebraic multigrid methods [97]. The matrices $\mathbf{A}$ and $\mathbf{A}^T$ can be viewed as restriction and prolongation operators in a multigrid method for computing activation forces.

If more than one activation spline is used, we obtain a linear system for each spline. We have to solve

$$\underbrace{\begin{bmatrix} \mathbf{b}^1 \\ \vdots \\ \mathbf{b}^S \end{bmatrix}}_{\mathbf{b}'} = \underbrace{\begin{bmatrix} \mathbf{A}^1 \\ \vdots \\ \mathbf{A}^S \end{bmatrix}}_{\mathbf{A}'} \underbrace{\begin{bmatrix} \mathbf{F}_1 \\ \vdots \\ \mathbf{F}_V \end{bmatrix}}_{\mathbf{y}} \tag{94}$$

where we have S splines and each $\mathbf{b}^i$ and $\mathbf{A}^i$ are defined as in (85). Observe that we still have the same number of unknowns, but our system now has more rows than previously. In principle, if we add enough splines, the linear system will become over-constrained. However, for all the examples using multiple splines, like our muscle simulation in Figure 26, the system is under-constrained and we apply all of the above solution techniques.

It is known, from real muscles, that their activation cannot move the center of mass of an isolated system, the total activation force of a muscle must sum to zero. This is obviously true for a single deformable object floating in space. This is also true for more complex systems like an astronaut in outer space. It is only through interaction with the environment that objects move. In our model this means $\sum_i \mathbf{f}_i = \mathbf{0}$ and $\sum_{j=1}^{V} \mathbf{F}_j = \mathbf{0}$. Furthermore, one should guarantee that the total torque of the splines equals the total torque of the mesh activation force field. That is, $\sum_i \mathbf{r}_i \times \mathbf{f}_i = \sum_{j=1}^{V} \mathbf{r}_j \times \mathbf{F}_j$ where $\mathbf{r}_j$ is vector from some origin to $\mathbf{p}_i$. Similarly, $\mathbf{r}_j$ is a vector from that same origin to $\mathbf{x}_j$. By construction, our spline model fulfils these criteria. If some other activation force model is used, it should be ensured that these criteria are fulfilled. All the criteria can be expressed as linear constraints for (87). Thus, our method can easily be extended to other activation force models.

20.5 Contact Forces

Our method generalizes to any collision handling strategy. For our proof-of-concept simulations however, we have chosen a simple penalty force simulator. We have a single deformable model that can collide with a ground plane. The contact force for any mesh vertex above the plane is zero. If a vertex is below the plane, we apply a penalty force in the normal direction of the plane, proportional to the penetration distance. Let $\mathbf{n} \in \mathbb{R}^D$ be the unit outer normal for the plane and let $\mathbf{q} \in \mathbb{R}^D$ be an arbitrary point on the plane. The velocity of the j^{th} vertex is given by $\mathbf{u}_j = \frac{d}{dt}\mathbf{x}_j$.The penetration depth is then computed as

$$d_j = (\mathbf{x}_j - \mathbf{q}) \cdot \mathbf{n} \tag{95}$$

The normal force is

$$\mathbf{n}_j = \begin{cases} \mathbf{0} & \text{if } d_j \geq 0 \\ -kd_j - c\mathbf{n}\mathbf{n}^T\mathbf{u}_j & \text{if } d_j < 0 \end{cases} \tag{96}$$

where k and c are the stiffness and damping coefficients, which we set using the same principles as in Section 20.3. If there is a non-zero tangential velocity

$$\mathbf{u}_{T,j} = \mathbf{u}_j - \mathbf{n}\mathbf{n}^T\mathbf{u}_j \tag{97}$$

we add a friction force according to Coulombs law of friction, opposing the sliding direction

$$\mathbf{t}_j = -\mu \parallel \mathbf{n}_j \parallel \frac{\mathbf{u}_{T,j}}{\parallel \mathbf{u}_{T,j} \parallel} \tag{98}$$

where $\mu > 0$ is the scalar coefficient of friction. In some cases (like in our snake simulation) we change the value of μ depending on the direction of $\mathbf{u}_{T,j}$. The total contact force on the j^{th} vertex is

$$\mathbf{F}_{\text{con},j} = \mathbf{n}_j + \mathbf{t}_j \tag{99}$$

21 Validation

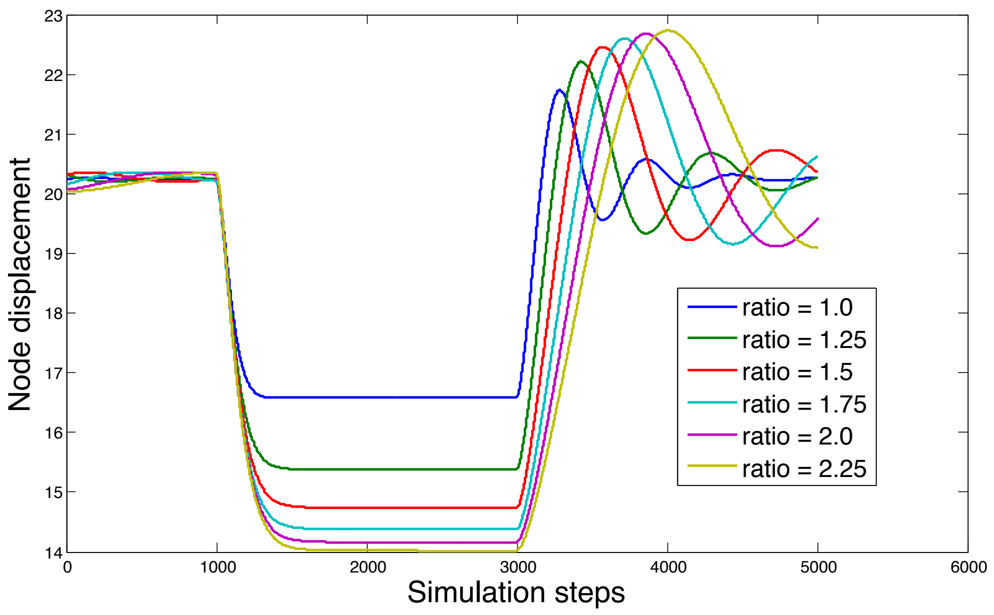


Figure 30: An example of the effect of changing the ratio between active and passive stiffness constants. The passive stiffness is kept constant and the active stiffness is changed.

To validate the method, we have implemented a proof-of-concept simulator in Matlab for both 2D and 3D. This system uses simple mass spring systems for the passive mesh, where spline points are assigned to cells in the mesh as shown in Figure 29(b). This makes it easy to assign splines and setup simulations. We also implemented a C++ version of a co-rotational linear FEM based on tetrahedral meshes. This version supports both the assignment strategies shown in Figure 29(a) and 29(c).

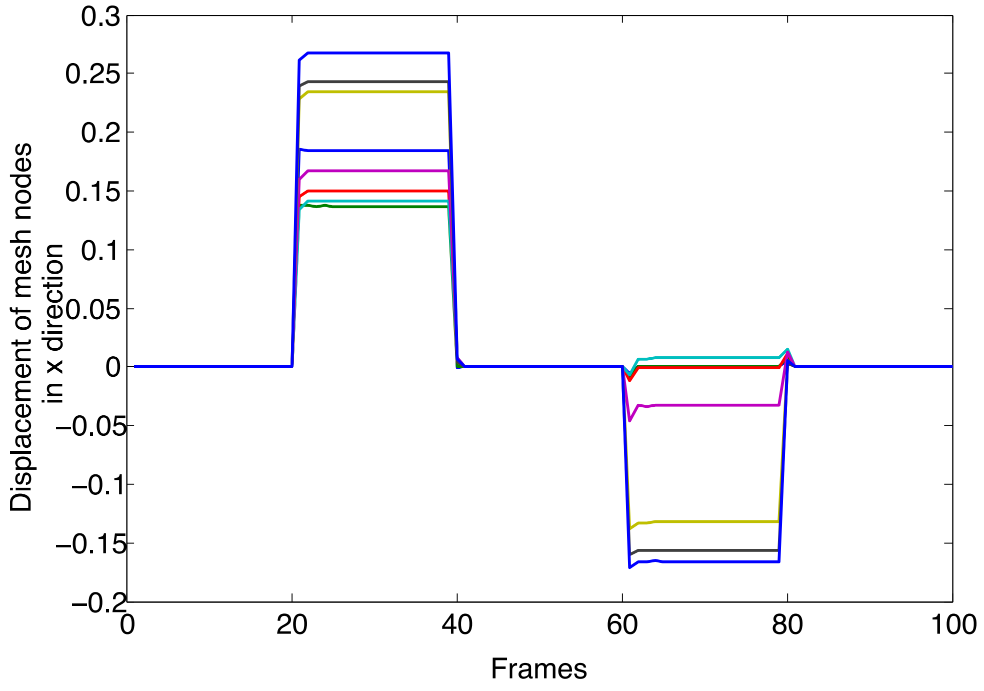


Figure 31: A plot showing the near critically damped response, given contraction of two overlapping splines at different times during a simulation. The plot shows the displacement of nodes in a mesh over time, during a contraction sequence similar to the one shown on the left in Figure 33.

We performed extensive parameter studies to investigate the effect of changing the damping and spring stiffness ratios between activation force and passive response. We have included a few of the results. Figure 30 shows a study of the effect of changing the ratio between passive and active stiffness. The passive stiffness is kept constant, while the active stiffness is changed. The simulation mesh is first contracted actively at simulation step 1000 and then released at step 3000.

It can be seen that it is important to find a ratio which gives a rapid and effective contraction but does not cause too long oscillations after release. We have used the ratio 2:1 for all our tests. Note that the oscillations are not present in the final simulation as explained in Section 20.3, and shown in Figure 31.

A number of simple tests have been performed to validate the correctness of the activation spline model as well as the generality of the activation

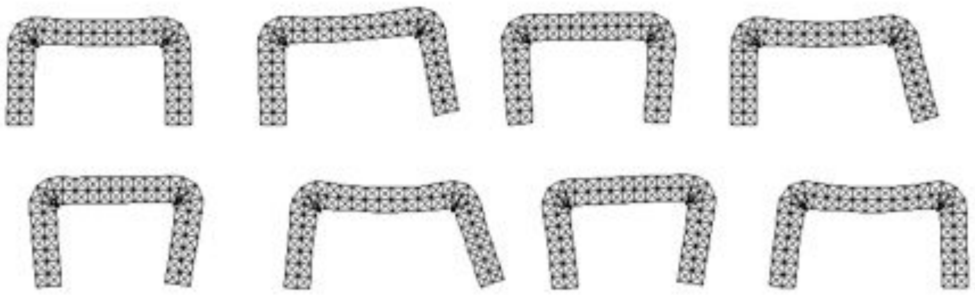


Figure 32: A time lapse of a 2D table walking, by pushing of from the ground and elongating or contracting its middle part.

force spline model. These have been designed to address both varying and multiple activation force directions, in both 2D and 3D.

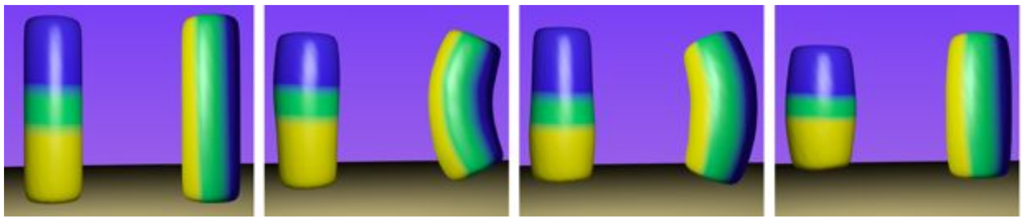


Figure 33: Test case for overlapping splines. The region of the first spline is shown in Yellow, The region of the second in blue, while the overlap is colored green.Notice that even though both bodies undergo rapid deformation no linear or angular momentum is induced.

Figure 31 shows a plot of the displacement of the mesh nodes in the x direction frame by frame. Even though the mass spring system imposes secondary oscillations, the response when the contractions are initiated in frames 20 and 60 is close to instantaneous and oscillations are minimal. Likewise, when the spline is relaxed in frames 40 and 80.

Figure 33 shows one such test case, where two cylinders is fitted with two overlapping splines and activating one after the other. One is overlapping in the direction of contraction the other orthogonal to this. The colors show the region of influence for each spline. yellow for the first spline blue for the other. The green region is the overlapping region. First the blue region is contracted, then the yellow and finally both are contracted. the leftmost image show the relaxed state.

Figure 34 show the effect of larger influence regions. In this test the the three cylinders where fitted with identical splines but each was given a different size of influence region. the red one was given an influence radius of 0.2 the green 0.4 and the blue 0.6. This meant that for the green and blue cylinders there was an overlap of the influence of the points in the spline. this turns out not to be a problem since it smooths out the force influence of individual spline points. The fundamental constraints are still upheld and notice in particular that the overall contraction is the same for all three cylinders. This makes sense since the splines are identical and thus, induce the same force on the mesh.

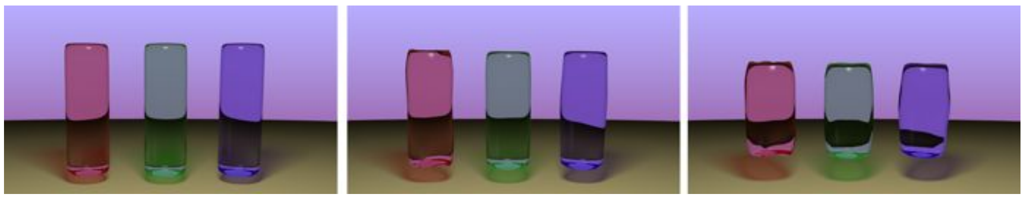


Figure 34: Three different settings for the size of influence region the red cylinder has the smallest influence region while the blue has the largest.Notice how the larger influence region smooths out the contraction as spline points share mesh nodes.

Furthermore, we have designed a number of simulations to test the capabilities of the system. In 2D, we have constructed a walking table (Figure 32). In this case, the deformable model was propelled forward using the activation splines. The table comprised two splines for pushing the legs of from the ground and one spline to elongate and contract the "back". As it turned out the resulting locomotion was surprisingly effective,

In 3D, we designed a number of simulations. To show the ability to make simple animations, a cartoon stool was rigged with activation splines in the legs. By simple impulse contractions, the stool was made to walk along the ground as shown in Figure 24. This locomotion was very similar to the one in the 2D case but since we now had four legs we could use these instead of having to contract the seat of the stool. This was animated using our matlab version of the system.

Figure 26 shows a simple muscle animation made with the C++ framework. The simulation consists of simple collision handling and two way constraints between deformable bodies and rigid bodies. It is possible to flex the arm and let it fall back to a stretched position under gravity.

A snake crawling along the ground was made using one spline in the left side from head to tail and one in the right side. The splines were activated using a phase shifted cosine function. This resulted in a surprisingly realistic behaviour for such a simple setup. Figure 25 shows a frame from the animation. Anisotropic friction was used to simulate the properties of snake skin. This animation was further developed to show the control it is possible to gain over the meshes. We let three snakes crawl and deform to spell three letters, Figure 35 shows the result of this animation.

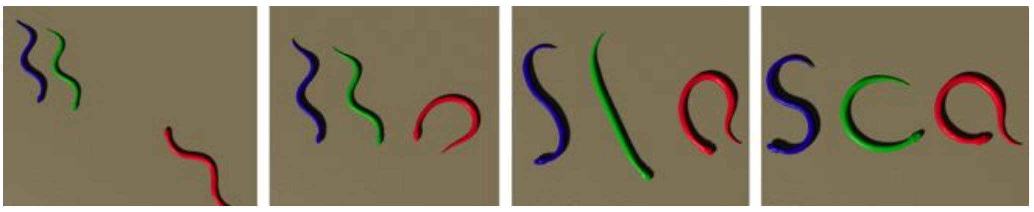
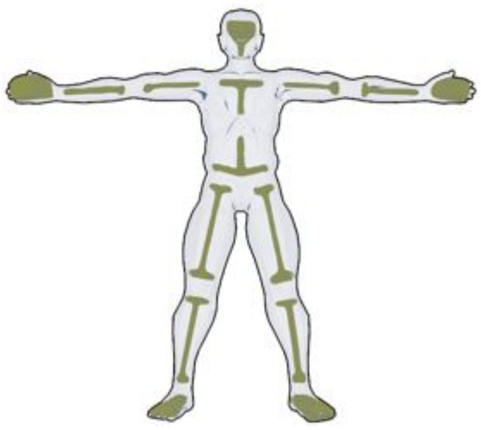
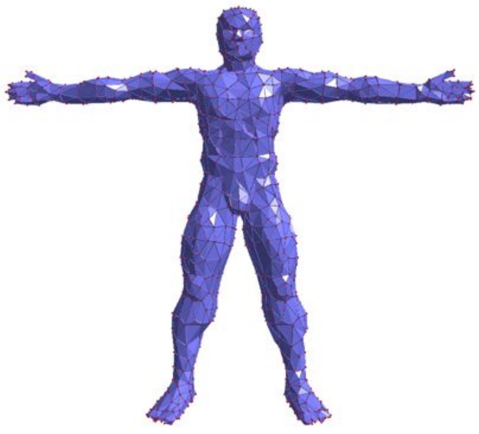


Figure 35: A time lapse of snakes spelling “SCA”. Even though the animation is based on a physical simulation, explicit control of the poses of the bodies can be obtained.

The visualization of the 3D simulation meshes has been done using embedded surface meshes which are then exported and rendered separately Using the Blender rendering engine. The 2D versions show the simulation mesh directly. Figure 24 and Figure 27 shows examples of animations where unlikely objects are articulated using the spline activation. Figure 27 shows a toy rubber lizard actuated using 6 splines. the actuation makes the toy come to life and waggle its way forward. The rubbery secondary effects of the passive mesh are a biproduct of the fact that this is a deformable model simulation and not a scripted animation. Similarly the walking stool in Figure 24 shows how an inanimate object can be brought to life. even though a very simple impulse activation is used the stool actually moves along the ground in a controllable way.

Our final test case is a human bending his arm. For this test we needed several new features. In contrast to rubber toys, humans have a skeleton which constrain the deformation of the soft parts. This was included by manipulating the stiffness of individual tetrahedrons in the simulation mesh. A skeleton was modelled and embedded into the mesh, where the skeleton mesh and the simulation mesh intersected the stiffness of the affected tetrahedra was increased 250 times. Figure 36(a) show the skeleton used. We used a comparatively fine grained mesh for this simulation, comprising 6902 tetrahedral elements. The figure was fixed at the pelvis so as not to fall, since we have no balance control in this simulation. Three muscles, the biceps the deltoid and the trapezius was activated to bent, raise and lower the arm. It proved slightly more difficult to control this particular simulation due to the rather thin limbs of humans. However even without a control parameter the figure flexes and releases his arm. Figure 23 shows


(a)


(b)

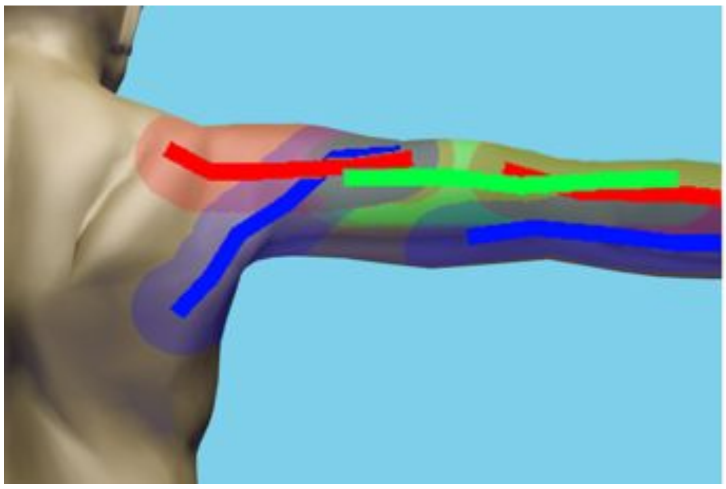

(c)

Figure 36: (a) shows the skeleton used for the human simulation. The stiffness of the tetrahedra which comprised the skeleton was set 250 times higher than the surrounding mesh. (b) shows the tetrahedral mesh used in the simulation. (c) shows the influence region of the five muscles overlaid on the visualization mesh.

four frames from this simulation. In Figure 37 the noticeable bulging of the biceps muscle under activation is clear.

Although no user interface was developed for the system, all simulations were set up in a matter of a few hours, at worst. This included creating and rigging the simulation mesh.

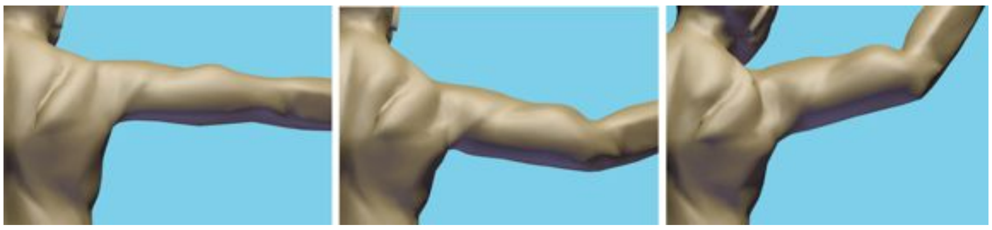


Figure 37: The bulging of the biceps muscle under activation is clearly seen in this animation

21.1 Performance measurements

To gauge the speed of the system we performed several tests with different mesh resolutions, using the c++ framework. We used a tetrahedral mesh cýlinder with mesh resolutions varying from 128 to 4011 elements. The cylinder was contracted for 1000 frames and the median values as well as the 1st and 3rd quartiles was plotted for each resolution. The results can be seen in Figure 38

As can be seen the time spent was very stable and didnt exceed $60\ ms/fr.$ for any of the test. This timing does not tell so much about our method as it tells about the FEM solver. The timings shown are for a full simulation pass including visualisation. The actual spline activation is app. $0.5 - 1.0$ percent of the full simulation time and ranges from 0.02 to 0.3 ms for the shown mesh resolutions.

Figure 39 show the time spent on the activation, which is much less and never exceeds $0.4 ms/fr$ for the tested meshes.

Further we measured the performance when changing the number of splines in the mesh. we made measurements for 1-40 splines with an interval of 5. Figure 40 show the results of these measurements. Since the activation matrix can be precomputed and the number of splines are usually much lower than the number of mesh nodes, the number of splines has negligible effect on the overall time spent.

As can be seen from all these timings the systems runs interactively for even moderately large tetrahedral meshes in our c++ implementation.

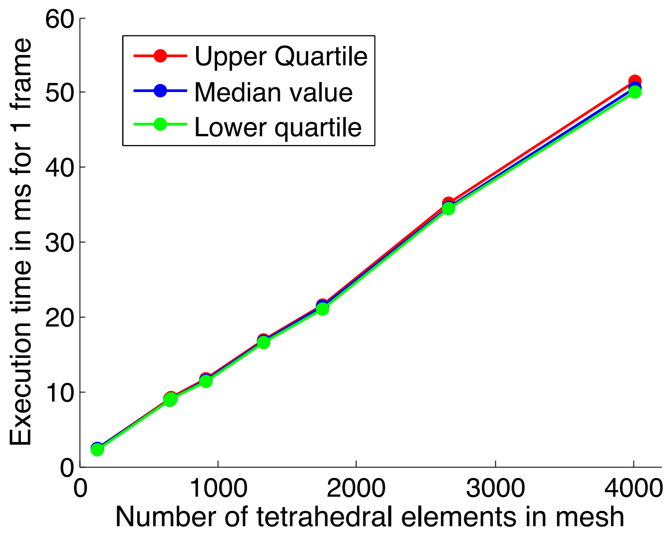


Figure 38: The timings for contraction of a single tetrahedral mesh cylinder with varying mesh resolution. a single spline was fitted to the mesh and contracted for 1000 frames. The plot shows the median and upper and lower quartiles for different mesh resolutions

Further refinements might give even better speed, but since the activation spline method takes up such a comparatively small amount of the time we have not investigated this further. It would seem that With our method the limiting factor is always going to be the passive mesh simulation and possibly the collision detection algorithm. It would seem that the time is better spent optimizing these than tweaking the already fast activation method.

22 Conclusion and Future Work

We have presented a novel approach for describing and modelling the activation forces of deformable models, based on 1D splines. We have demonstrated the method in several scenarios, both 2D and 3D, to show how it is possible to make animations using the actuation splines. No explicit control scheme is present, but it is still possible to explicitly control posing due to the straight forward and intuitive relation between the spline and the mesh.

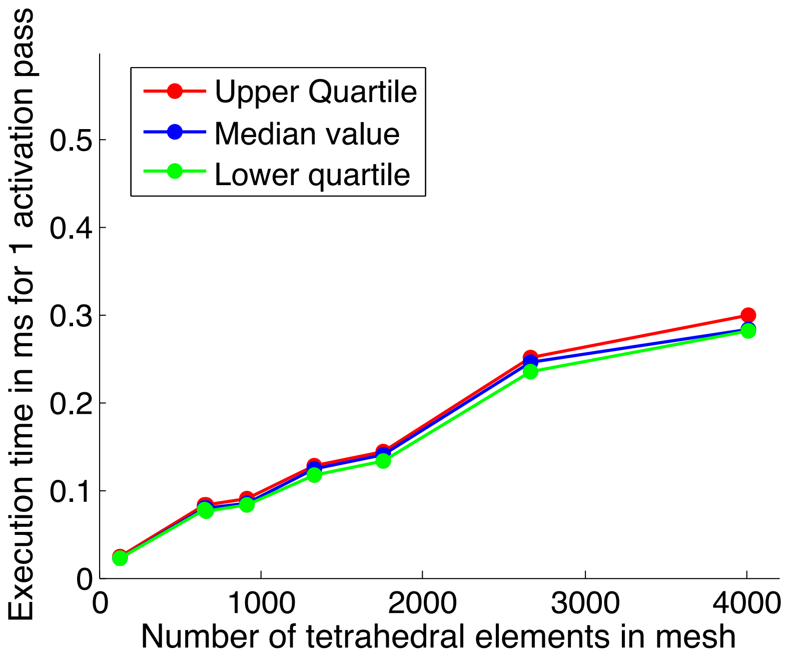


Figure 39: The timings for activation part of simulation. The plot shows the median and upper and lower quartiles for different mesh resolutions. Notice that the time never exceeds $0.4ms/fr$

As we have demonstrated, our method presents an attractive compromise between precision and realism on the one hand, and speed and generality on the other. Activation splines are an intuitive way of extending the simple line-actuator to a deformable contraction spline. These follow the deformation of the surrounding mesh, and are influenced by collisions and attachments to other bodies.

A number of promising avenues for extending the presented ideas are obvious. Regarding the muscle simulation case, the current version uses a hand modelled skeleton. It would be interesting to use real scanned data and actual measured parameters to further refine the model and show its applicability outside the animation world.

Control using key framing, or similar techniques, would make the system easier to use for animators. We speculate that short horizon space-time optimization or inverse dynamics schemes might be possible ideas for this.

Our work focused on allowing animators artistic freedom to create activation splines in. However, to assist animators one could consider creating tools that could automate some of the spline creation work. We speculate that medial surface representations or curve skeletons could be used to

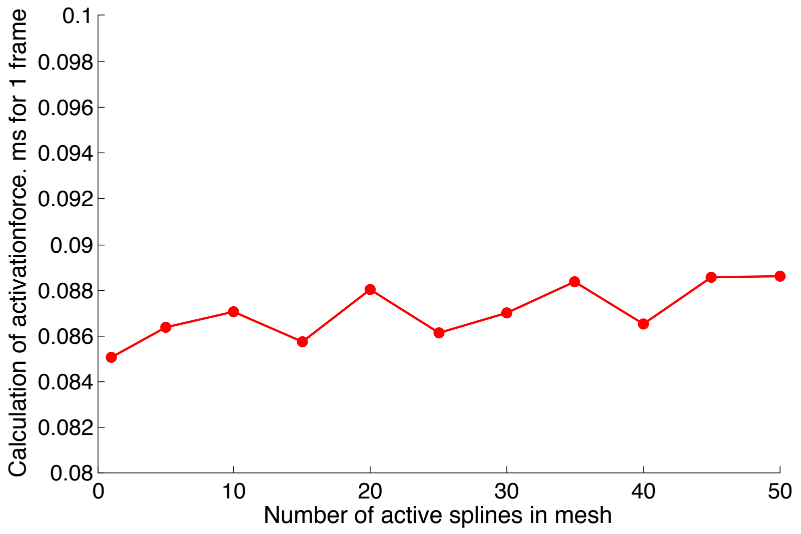


Figure 40: The timings for running simulations with varying number of splines. As it can be seen the number of splines have very little effect on the simulation time.

seed 1D splines [98]. Even with our simple interface it was comparatively straight forward to rig the activation splines and their influence regions but with more time this could be refined considerably, making the activation splines a valuable tool for animation and simulation.

Part IV
Optimal control using space time optimization

Control of synthesized motion is something I have not yet touched upon. Well that may not be entirely true, the motion planning in itself, is an example of a simple form of control. It does not take forces into account though.

Therefore, some other scheme is needed to get from the purely kinematic motion to a dynamic simulation. In short we need something that can calculate the forces from a kinematically driven motion. If we where talking about rigid body dynamics, a simple inverse dynamics scheme might suffice. Using that, we could plan the motion, with the inverse kinematics method from section 3. Then, we could calculate the necessary forces, using an off the shelf inverse dynamics solver such as the one used in the Anybody system [99] or in opensim [100].

When it comes to controlling the motion of deformable objects however, there is no such easy road. Equally important is the fact that inverse dynamics only works if there is a pose available in every time step of the simulation, which makes optimizing motions hard. We wanted something that could both, handle deformable objects, and calculate the forces necessary, for a given motion, with only keyframes as the input. Space Time optimization does exactly that.

The following section describes our first ideas and our results for simple spring connected particles. The theory extends easily to deal with more complex structures and our first promising results makes us believe this is indeed a feasible way of solving the problem of optimal control for deformable objects.

The research in this last part of my thesis is not published yet. The plan is to publish this as soon as we have sufficient results to be sure the method works robustly. Our initial results are good enough for us to believe this is the case.

23 Introduction to Space Time Optimization

In the following we will introduce the Adjoint method for space time optimization of deformable bodies.

The space time optimization problem is, by design, stated as an optimization problem and hence our challenge is that of finding a computational cheap way of computing the gradient of the objective function. For this we will use the Adjoint method. This makes it possible for us to reduce the number of unknowns from n^2 to n. Further we will take advantage of the sparse block structure of the resulting equations to yield an $\mathcal{O}(n)$ algorithm for computing the gradient. We present our modification of the adjoint method for solving the space time optimization problem for deformable bodies the method may be extended to spline activated soft-bodies as the ones described in the previous part, by constraining the possible forces to the subspace given by the force splines, and incorporating the activation mapping in the calculation of the active component of the gradient.

24 Previous Work

In this section we will give a short introduction to space time optimization. this is by no means a full description of the many methods developed for control of articulated mechanisms. we only seek to give the reader a background for appreciating the following.

In 1988 Witkin & Kass [81] introduced space–time constraints to the graphics community for articulated rigid bodies. Here motion control was solved as an optimization problem. The gradient of their objective function was found using automatic numerical differentiation. Witkin & Kass never experienced problems with discontinuities as their simple examples allowed for analytical solutions (they never used the Adjoint method) and contact was modelled as explicit constraints. Later, McNamara et. al. [101] applied the Adjoint method for optimal control of a fluid solver. McNamara et. al. did have discontinuities when dealing with their level set surface and did report some problems from the collision of a clay blob. Wojtan et. al. [79] used the Adjoint method for particle systems and cloth. They did report that the collisions could be a problem but showed examples that handled the cases. The Adjoint method was applied for deformable models in [80]. The authors used per vertex control forces. However, they applied a reduced coordinate method to the deformable models and they created a technique for obtaining a good starting iterate. The work goes into detail on how to create a reduction basis that actually contain the deformation modes of the key frames. Collisions and impact are dealt with using bilateral contraints this is similar to what Witkin & Kass did [81].

25 The Space Time Constraint Problem

We will now explain how the Adjoint problem is related to the space time constraint problem. Assume we are given a state vector of some system $\mathbf{q} \in \mathbb{R}^m$. For now we will abstract over the details of the specific system and keep things on a general abstract level. Later we will explain explicitly how our state vector is defined. Using some forward simulation routine we advance the state from some time t to the time $t + \Delta t$. We write this abstractly as

$$\mathbf{q}^{t+\Delta t} = \mathbf{F}^t(\mathbf{q}^t, \alpha^t) \tag{100}$$

where $\alpha^t \in \mathbb{R}^s$ is the value of the control parameter vector for the t^{th} simulation step. The function $\mathbf{F}(\cdot,\cdot)$ is the actual forward simulation that is being run. However, for all purposes we can just think of this as a function that takes a given input state and input control parameter vector and produces a resulting final output state.

If we are given the initial state of the system $\mathbf{q}^0$ as initial conditions and then after N simulation steps we will have computed $\mathbf{q}^N$. Introducing matrix vector notation we may stack all the elements from (100) and write one system of equations as

$$\underbrace{\begin{bmatrix} \mathbf{q}^N \\ \vdots \\ \mathbf{q}^2 \\ \mathbf{q}^1 \end{bmatrix}}_{\mathbf{Q}} = \underbrace{\begin{bmatrix} \mathbf{F}^{N-1}(\mathbf{q}^{N-1}, \alpha^{N-1}) \\ \vdots \\ \mathbf{F}^1(\mathbf{q}^1, \alpha^1) \\ \mathbf{F}^0(\mathbf{q}^0, \alpha^0) \end{bmatrix}}_{\mathbf{F}(\mathbf{Q},\mathbf{a})} \tag{101}$$

where $\mathbf{Q} \in \mathbb{R}^n$ with $n = N\,m$, $\mathbf{F} : \mathbb{R}^n \times \mathbb{R}^k \mapsto \mathbb{R}^n$ with $k = N\,s$ and we have defined the stacked control parameter vector $\mathbf{a} \in \mathbb{R}^k$ as

$$\mathbf{a} = \left[\left(\alpha^{N-1}\right)^T \quad \dots \quad \left(\alpha^0\right)^T \right]^T \tag{102}$$

Definition 25.1 *The simplest form of the space-time constraint problem can now be stated as the problem of finding a solution* $\mathbf{a}^*$ *for the control parameter vector* $\mathbf{a}$ *such that, given the initial state* $\mathbf{q}^0$*, after* N *simulation steps,* $\mathbf{a}^*$ *will produce a motion such that, a given wanted final state* $\mathbf{q}_{goal}$ *is reached.*

In principle there may be many solutions for this problem hence to make the problem solvable we write the problem mathematically as a nonlinear optimization problem

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} \phi(\mathbf{Q}, \mathbf{a}) \tag{103}$$

such that $\mathbf{Q} - \mathbf{F}(\mathbf{Q}, \mathbf{a}) = \mathbf{0}$ and where

$$\phi(\mathbf{Q}, \mathbf{a}) \equiv \frac{1}{2} \parallel \mathbf{a} \parallel^2 + \frac{\mu}{2} \parallel \mathbf{q}^N - \mathbf{q}_{\text{goal}} \parallel^2 \tag{104}$$

where $\mu \in \mathbb{R}_+$ is a user-specified penalty parameter. This is a penalty method that seeks to minimize the control effort given by the first term, subject to the penalty constraint (the second term) of meeting the prescribed goal state after N simulation steps. The parameter μ can be used as a tradeoff parameter between minimizing the effort or reaching the goal state. To effectively solve such a minimization problem we are interested in being able to compute the gradient $\frac{d\phi}{d\mathbf{a}}$. This is the computational problem we need to solve.

The specific choice of objective function (104) is based on current practice in computer graphics [81, 101, 79, 80]. In principle one can alter and change the objective function depending on specific needs.It may seem strange that the penalty term is associated with the state term and not the control parameter term. The answer is obvious as the goal state may be a physically unrealisable state due to the fact that it can be a user specified input(e.g. a pose specified by an artist). It is therefore quite possible that we are in the situation where no solution for $\mathbf{a}$ exist. A penalty formulation allow us to solve the problem even in this case.

One generalization of the simple space time optimization problem may be to introduce multiple goal states along the motion path (also known as key frames). This would change the objective function into

$$\phi_G(\mathbf{Q}, \mathbf{a}) \equiv \frac{1}{2} \parallel \mathbf{a} \parallel^2 + \frac{\mu}{2} \sum_{i=1}^{N} w_i \parallel \mathbf{q}^i - \mathbf{q}_{\text{goal},i} \parallel^2 \tag{105}$$

Here we have introduced the weights $w_i \in [0..1]$. These can be used to control the importance of the individual key frames. In particular using $w_N = 1$ and $w_i = 0$ for all $i \neq N$ we recover the simple form in (104).

Similar optimization formulations exist in other fields such as image processing. Here people often use 1-norms and a kind of regularization of the $\mathbf{a}$-term that prevents it from changing too much from frame to frame.

This leads to the following formulation.

$$
\begin{aligned}
\phi_I(\mathbf{Q}, \mathbf{a}) \equiv \| \ \mathbf{a} \ \|_1^2 + \frac{\beta}{2} \sum_{j=2}^{k} \left| \mathbf{a}^k - \mathbf{a}^{k-1} \right| \\
+ \frac{\mu}{2} \sum_{i=1}^{N} w_i \ \| \ \mathbf{q}^i - \mathbf{q}_{\text{goal},i} \ \|^2
\end{aligned}
\tag{106}
$$

The 1-norm may be a numerical advantage as it often favours sparse solutions. The rate limited constraint can be controlled through the weight parameter $\beta \in \mathbb{R}_+$. The rate limitation has some physical interpretation imagine $\mathbf{a}$ representing the power input for an engine, or the activation of a muscle. In real life both muscles and motors have a certain maximal activation speed. For the muscle, this is the time for the nerve signals to reach the muscle fibres, and the chemical process to take place. In the motor it is the time needed to accelerate.

From an optimization viewpoint the term is attractive as it couples the components of $\mathbf{a}$. Intuitively speaking the extra term reduces the possible search space of $\mathbf{a}$ making the optimization problem “easier” to solve.

25.1 The Adjoint Method

We will start by introducing the Adjoint problem on an abstract level. We want to compute the value of $\mathbf{g}^T\mathbf{B}$ for a known $\mathbf{g} \in \mathbb{R}^n$ and an unknown $\mathbf{B} \in \mathbb{R}^{n \times k}$ such that the relation $\mathbf{AB} = \mathbf{C}$ always hold for a given known $\mathbf{A}, \in \mathbb{R}^{n \times n}$ and $\mathbf{C} \in \mathbb{R}^{n \times k}$. The difficulty here is that $\mathbf{B}$ is a matrix of $n \times k$ unknowns. We write the problem as

$$
\mathbf{g}^T\mathbf{B} \quad \text{such that} \quad \mathbf{AB} = \mathbf{C} \tag{107}
$$

The Adjoint problem is then given by

$$
\mathbf{r}^T\mathbf{C} \quad \text{such that} \quad \mathbf{A}^T\mathbf{r} = \mathbf{g} \tag{108}
$$

where $\mathbf{r} \in \mathbb{R}^n$ is the unknown Adjoint vector. Observe that

$$
\mathbf{r}^T\mathbf{C} = \mathbf{r}^T\mathbf{AB} = \left(\mathbf{A}^T\mathbf{r}\right)^T \mathbf{B} = \mathbf{g}^T\mathbf{B} \tag{109}
$$

Thus, rather than solving for $\mathbf{B}$ and compute $\mathbf{g}^T\mathbf{B}$ we can solve for $\mathbf{r}$,reducing the number of unknowns from n^2 to n unknowns – and compute $\mathbf{r}^T\mathbf{C}$ instead.

25.2 Computing the Gradient using the Adjoint Method

From the chain rule we get that

$$\frac{d\phi}{d\mathbf{a}} = \frac{\partial\phi}{\partial\mathbf{Q}}\frac{d\mathbf{Q}}{d\mathbf{a}} + \frac{\partial\phi}{\partial\mathbf{a}} \tag{110}$$

The first term $\frac{\partial\phi}{\partial\mathbf{Q}}\frac{d\mathbf{Q}}{d\mathbf{a}}$ is the tricky part as the part $\frac{d\mathbf{Q}}{d\mathbf{a}}$ is very difficult to compute analytically for large complex systems. The second term $\frac{\partial\phi}{\partial\mathbf{a}}$ can usually be obtained analytically with little difficulty and as such pose no real difficulty. For our specific objective function (104) we find that

$$\frac{\partial\phi}{\partial\mathbf{a}} = \mathbf{a} \tag{111}$$

We know $\mathbf{Q} - \mathbf{F}(\mathbf{Q},\mathbf{a}) = \mathbf{0}$ always hold for all values of $\mathbf{Q}$ and $\mathbf{a}$. Thus, the gradient must always be $\mathbf{0}$

$$\frac{d}{d\mathbf{a}}\left(\mathbf{Q} - \mathbf{F}(\mathbf{Q},\mathbf{a})\right) = \mathbf{0} \tag{112}$$

This can be rewritten as

$$\left(\mathbf{I} - \frac{\partial\mathbf{F}}{\partial\mathbf{Q}}\right)\frac{d\mathbf{Q}}{d\mathbf{a}} = \frac{\partial\mathbf{F}}{\partial\mathbf{a}} \tag{113}$$

The problem of computing the gradient can now be stated as the problem of computing

$$\frac{d\phi}{d\mathbf{a}} = \frac{\partial\phi}{\partial\mathbf{Q}}\frac{d\mathbf{Q}}{d\mathbf{a}} + \frac{\partial\phi}{\partial\mathbf{a}} \tag{114}$$

such that

$$\left(\mathbf{I} - \frac{\partial\mathbf{F}}{\partial\mathbf{Q}}\right)\frac{d\mathbf{Q}}{d\mathbf{a}} = \frac{\partial\mathbf{F}}{\partial\mathbf{a}} \tag{115}$$

It seems we have obtained nothing but making the problem more complicated but Defining:

$$\begin{aligned}
\mathbf{g}^T &= \frac{\partial\phi}{\partial\mathbf{Q}} && (116)\\
\mathbf{A} &= \left(\mathbf{I} - \frac{\partial\mathbf{F}}{\partial\mathbf{Q}}\right) && (117)\\
\mathbf{B} &= \frac{d\mathbf{Q}}{d\mathbf{a}} && (118)\\
\mathbf{C} &= \frac{\partial\mathbf{F}}{\partial\mathbf{a}} && (119)\\
& && (120)
\end{aligned}$$

we have

$$\frac{d\phi}{d\mathbf{a}} = \mathbf{g}^T\mathbf{B} + \frac{\partial \phi}{\partial \mathbf{a}} \quad \text{such that} \quad \mathbf{AB} = \mathbf{C} \tag{121}$$

From this we immediately recognize that we can apply the Adjoint method and instead solve the Adjoint problem of computing

$$\mathbf{r}^T\mathbf{C} \quad \text{such that} \quad \mathbf{A}^T\mathbf{r} = \mathbf{g} \tag{122}$$

which is

$$\mathbf{r}^T \frac{\partial \mathbf{F}}{\partial \mathbf{a}} \quad \text{such that} \quad \left(\mathbf{I} - \frac{\partial \mathbf{F}}{\partial \mathbf{Q}}\right)^T \mathbf{r} = \frac{\partial \phi}{\partial \mathbf{Q}}^T. \tag{123}$$

Afterwards we may compute the gradient as

$$\frac{d\phi}{d\mathbf{a}} = \mathbf{r}^T \frac{\partial \mathbf{F}}{\partial \mathbf{a}} + \frac{\partial \phi}{\partial \mathbf{a}} \tag{124}$$

We may now summarize the computational method for computing the gradient

Step 1: Calculate and assemble $\frac{\partial \mathbf{F}}{\partial \mathbf{a}}$, $\frac{\partial \mathbf{F}}{\partial \mathbf{Q}}$, $\frac{\partial \phi}{\partial \mathbf{Q}}$, and $\frac{\partial \phi}{\partial \mathbf{a}}$.

Step 2: From the right hand side of (123) solve for $\mathbf{r}$ such that

$$\mathbf{r} = \frac{\partial \mathbf{F}}{\partial \mathbf{Q}}^T \mathbf{r} + \frac{\partial \phi}{\partial \mathbf{Q}}^T \tag{125}$$

Step 3: Finally compute the gradient

$$\frac{d\phi}{d\mathbf{a}} = \mathbf{r}^T \frac{\partial \mathbf{F}}{\partial \mathbf{a}} + \frac{\partial \phi}{\partial \mathbf{a}} \tag{126}$$

This is the general recipe which we may now apply some optimization for where we take advantage of sparsity patterns. Note that $\mathbf{r}$ depends on itself, but bear with us a moment then we will explain how this can be.

25.3 Exploiting the Sparse Block Matrix Patterns

For our specific choice of (104) we find that

$$\frac{\partial \phi}{\partial \mathbf{q}^i} = \begin{cases} \mu \left(\mathbf{q}^N - \mathbf{q}_{\text{goal}}\right)^T & \text{if } i = N \\ \mathbf{0}^T & \text{otherwise} \end{cases} \tag{127}$$

Thus

$$\begin{aligned}\frac{\partial \phi}{\partial \mathbf{Q}} &= \begin{bmatrix} \frac{\partial \phi}{\partial \mathbf{q}^N} & \frac{\partial \phi}{\partial \mathbf{q}^{N-1}} & \cdots & \frac{\partial \phi}{\partial \mathbf{q}^1} \end{bmatrix} \\ &= \begin{bmatrix} \mu \left(\mathbf{q}^N - \mathbf{q}_{\text{goal}}\right)^T & \mathbf{0}^T & \cdots & \mathbf{0}^T \end{bmatrix}\end{aligned} \tag{128}$$

and $\frac{\partial \phi}{\partial \mathbf{a}} = \mathbf{a}$. Further, we observe from (101) that the partial derivative of the forward simulation with respect to the state has the block structure

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} = \begin{bmatrix} \mathbf{0} & \frac{\partial \mathbf{F}^{N-1}}{\partial \mathbf{q}^{N-1}} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{F}^{N-2}}{\partial \mathbf{q}^{N-2}} & \ldots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \frac{\partial \mathbf{F}^{1}}{\partial \mathbf{q}^{1}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{129}$$

Notice that only the first upper band blocks are non-zero. The partial derivative with respect to the control parameters give the block diagonal structure

$$\frac{\partial \mathbf{F}}{\partial \mathbf{a}} = \begin{bmatrix} \frac{\partial \mathbf{F}^{N-1}}{\partial \alpha^{N-1}} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{F}^{N-2}}{\partial \alpha^{N-2}} & \ldots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \frac{\partial \mathbf{F}^{1}}{\partial \alpha^{1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \frac{\partial \mathbf{F}^{0}}{\partial \alpha^{0}} \end{bmatrix} \tag{130}$$

The very sparse block structure of $\frac{\partial \mathbf{F}}{\partial \mathbf{Q}}$ and $\frac{\partial \mathbf{F}}{\partial \mathbf{a}}$ allow us to solve (125) very efficiently. Defining the block notation of the Adjoint vector

$$\mathbf{r} = \begin{bmatrix} \left(\mathbf{r}^N\right)^T & \cdots & \left(\mathbf{r}^1\right)^T \end{bmatrix}^T \tag{131}$$

with $\mathbf{r}^i \in \mathbb{R}^m$ for all $i = [1..N]$. We compute the solution of (125) by initially setting $\mathbf{r}^N = \frac{\partial \phi}{\partial \mathbf{q}^N}^T$ and then incrementally compute

$$\mathbf{r}^{i-1} = \frac{\partial \mathbf{F}}{\partial \mathbf{q}^i}^T \mathbf{r}^i + \frac{\partial \phi}{\partial \mathbf{q}^{i-1}}^T \quad \text{for } i = N-1 \text{ to } 2 \tag{132}$$

The final ingredients we are missing in our framework is closed form solutions for computing $\frac{\partial \mathbf{F}^i}{\partial \mathbf{q}^i}$ and $\frac{\partial \mathbf{F}^i}{\partial \alpha^i}$. Here we must consider the specific case of the deformable model simulation.

25.4 Deformable Models

After appropriate spatial discretization (using for instance the finite element method) the governing equations of a deformable model is given by the ordinary differential system of equations

$$\mathbf{M}\dot{\mathbf{u}} + \mathbf{C}\mathbf{u} + \mathbf{k}(\mathbf{x} - \mathbf{x}_0) = \mathbf{f}_{\text{ext}} + \mathbf{f}(\alpha) \tag{133a}$$

$$\dot{\mathbf{x}} = \mathbf{u} \tag{133b}$$

where $\mathbf{M} \in \mathbb{R}^{3V \times 3V}$ is the mass matrix and V is the number of vertices in the spatial computational mesh, $\mathbf{C}^{3V \times 3V}$ is the damping matrix, $\mathbf{x} \in \mathbb{R}^{3V}$ is the spatial coordinate vector and $\mathbf{u} \in \mathbb{R}^{3V}$ is the spatial velocity vector, $\mathbf{x}_0 = \mathbf{x}(0)$ is the initial state of the deformable model. Hence $\mathbf{x} - \mathbf{x}_0$ is the current displacement field and $\mathbf{k}(\mathbf{x} - \mathbf{x}_0) : \mathbb{R}^{3V} \mapsto \mathbb{R}^{3V}$ is the elastic force function that computes the spatial elastic forces given the current displacement field as argument. Further, the system is subject to external forces such as gravity etc. given by $\mathbf{f}_{\text{ext}} \in \mathbb{R}^{3V}$. The last term $\mathbf{f}(\alpha) : \mathbb{R}^s \mapsto \mathbb{R}^{3V}$ is the control forces. These are a function of the control parameters $\alpha \in \mathbb{R}^s$. Time discretization of the ordinary differential equations yields the discrete velocity update

$$\begin{aligned}\mathbf{u}^{t+1} &= \left(\mathbf{I} - \Delta t \mathbf{M}^{-1}\mathbf{C}\right)\mathbf{u}^t \\ &\quad + \Delta t \mathbf{M}^{-1}\left(\mathbf{f}_{\text{ext}} + \mathbf{f}(\alpha^t) - \mathbf{k}(\mathbf{x}^t - \mathbf{x}_0)\right)\end{aligned} \tag{134}$$

and the position update

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{u}^{t+1}. \tag{135}$$

From the velocity and position updates, (134) and (135), we have that the partial derivatives of the forward simulation function are given by

$$\frac{\partial \mathbf{x}^{t+1}}{\partial \mathbf{x}^t} = \mathbf{I} \tag{136a}$$

$$\frac{\partial \mathbf{x}^{t+1}}{\partial \mathbf{u}^t} = \Delta t \frac{\partial \mathbf{u}^{t+1}}{\partial \mathbf{u}^t} \tag{136b}$$

$$\frac{\partial \mathbf{u}^{t+1}}{\partial \mathbf{x}^t} = \Delta t \mathbf{M}^{-1}\mathbf{K} \tag{136c}$$

$$\frac{\partial \mathbf{u}^{t+1}}{\partial \mathbf{u}^t} = \mathbf{I} + \Delta t \mathbf{M}^{-1}\mathbf{C} \tag{136d}$$

and

$$\frac{\partial \mathbf{x}^{t+1}}{\partial \alpha^t} = \mathbf{0} \tag{137a}$$

$$\frac{\partial \mathbf{u}^{t+1}}{\partial \alpha^t} = \Delta t \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \alpha^t} \tag{137b}$$

where $\mathbf{K} = \frac{\partial \mathbf{k}}{\partial \mathbf{x}}$ is the stiffness tangent matrix. We Define the state space vector $\mathbf{q}^t \in \mathbb{R}^m$ with $m = 6V$ as

$$\mathbf{q}^t = \begin{bmatrix} \mathbf{x}^t \\ \mathbf{u}^t \end{bmatrix} \tag{138}$$

This results in the partial derivatives

$$\frac{\partial \mathbf{F}^i}{\partial \mathbf{q}^i} = \begin{bmatrix} \mathbf{I} & \Delta t \mathbf{I} + \Delta t^2 \mathbf{M}^{-1}\mathbf{C} \\ \Delta t \mathbf{M}^{-1}\mathbf{K} & (\mathbf{I} + \Delta t \mathbf{M}^{-1}\mathbf{C}) \end{bmatrix} \tag{139a}$$

$$\frac{\partial \mathbf{F}^i}{\partial \alpha^i} = \begin{bmatrix} \Delta t^2 \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \alpha^t} \\ \Delta t \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \alpha^t} \end{bmatrix} \tag{139b}$$

We observe from this, that we really only need to compute the tangent stiffness matrix $\mathbf{K}$ and the derivative of the control force $\frac{\partial \mathbf{f}}{\partial \alpha}$.

Notice that for linear elasticity the tangent stiffness matrix is constant and needs not be re-computed. However, for nonlinear elasticity the stiffness matrix would depend on the current displacement field and must be re-computed for each simulation step. This requires us to store all the states $\mathbf{q}^0$,$\mathbf{q}^1$, $\ldots$, $\mathbf{q}^N$. We may compute the tangent stiffness matrix for general hyper elastic materials using a closed form solution (See Erleben [102, 103, 104, 105, 106]). The control force can be quite complicated. However, the most simple model would simply be a nodal force. Hence

$$\mathbf{f}(\alpha) = \alpha \tag{140}$$

and one would find $\frac{\partial \mathbf{f}}{\partial \alpha} = \mathbf{I}$. For the muscle splines described in the previous part we could replace $\mathbf{f}(\alpha)$ in (140) with the spline activation and force distribution function.

Observe that to compute ϕ, the forward simulation is performed and $\mathbf{q}^1$,. . .,$\mathbf{q}^N$ are stored. Once this is done it is straightforward to evaluate the value of the objective function.

In order to compute the gradient of the objective function, compute the Adjoint vectors $\mathbf{r}^N$, . . .,$\mathbf{r}^1$. Notice these are computed in reverse order of the forward simulation. For the i^{th} step of the Adjoint problem we need the $\mathbf{q}^i$ state to calculate the partial derivatives. This is a potential drawback, as we need sufficient storage, to store the state for all simulation steps. Another potential drawback is the costly computation of the tangent stiffness matrix. Finally, from previous work it is well known that the Adjoint method has difficulties dealing with discontinuities in the state function. This, obviously occurs during collisions and contacts. Some scheme is needed to handle this if the method is to be effective beyond short horizon planning.

26 A Simple 2D Particle Example

Consider the equation of motion of a single 2D particle in a gravitational field

$$m\ddot{\mathbf{r}} = \mathbf{f}_{\text{ext}} + \mathbf{f}(\alpha) \tag{141}$$

where m is the particle mass and $\mathbf{r} = (x, y)^T$ is the particle position, $\mathbf{f}_{\text{ext}} = (0, -mg)$ is the gravitational force with the gravitational acceleration given by $g \approx 9.82$ $[m/s^2]$. The control force is given by $\mathbf{f}(\alpha) = (\alpha_x, \alpha_y)$.

Introducing the particle velocity $\dot{\mathbf{r}} = \mathbf{v}$ and using the usual time-discretization we obtain the velocity and update rules

$$\mathbf{r}^{t+1} = \mathbf{r}^t + \Delta t \mathbf{v}^{t+1}, \tag{142a}$$

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \frac{\Delta t}{m}\left(\mathbf{f}_{\text{ext}} + \alpha^t\right), \tag{142b}$$

where Δt is the time step size. This defines our forward simulation function $\mathbf{F}^t(\mathbf{q}^t, \alpha^t)$ with the state given by

$$\mathbf{q}^t = \begin{bmatrix} \mathbf{r}^t \\ \mathbf{v}^t \end{bmatrix}. \tag{143}$$

We may now compute the partial derivatives $\frac{\partial \mathbf{F}^t}{\partial \mathbf{q}^t}$ and $\frac{\partial \mathbf{F}^t}{\partial \alpha^t}$ as

$$\frac{\partial \mathbf{F}^t}{\partial \mathbf{q}^t} = \begin{bmatrix} \frac{\partial \mathbf{r}^{t+1}}{\partial \mathbf{r}^t} & \frac{\partial \mathbf{r}^{t+1}}{\partial \mathbf{v}^t} \\ \frac{\partial \mathbf{v}^{t+1}}{\partial \mathbf{r}^t} & \frac{\partial \mathbf{v}^{t+1}}{\partial \mathbf{v}^t} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{2\times 2} & \Delta t \mathbf{I}_{2\times 2} \\ \mathbf{0} & \mathbf{I}_{2\times 2} \end{bmatrix} \tag{144}$$

and

$$\frac{\partial \mathbf{F}^t}{\partial \alpha^t} = \begin{bmatrix} \frac{\partial \mathbf{r}^{t+1}}{\partial \alpha^t} \\ \frac{\partial \mathbf{v}^{t+1}}{\partial \alpha^t} \end{bmatrix} = \begin{bmatrix} \frac{\Delta t^2}{m} \mathbf{I}_{2\times 2} \\ \frac{\Delta t}{m} \mathbf{I}_{2\times 2} \end{bmatrix} \tag{145}$$

It may be instructive to compare this to the general deformable model formulas using $\mathbf{C} = \mathbf{K} = \mathbf{0}$ and $\mathbf{M} = m$

27 A 2 Particle Example in 2D

Building on the example from the previous section, we extend our system to have two particles label A and B. Now let us consider adding a single spring between the two particles. this simple system is the foundation of more complicated systems since every mass spring system can be disassembled into this simple relation.

Defining $\mathbf{e} = \mathbf{r}_A - \mathbf{r}_B$ then the spring force on particle A is

$$\mathbf{k}_A = -k\left(\| \mathbf{e} \| - l_0\right)\frac{\mathbf{e}}{\| \mathbf{e} \|} - c\left(\frac{\mathbf{e}\mathbf{e}^T}{\| \mathbf{e} \|^2}\right)(\mathbf{v}_A - \mathbf{v}_B) \tag{146}$$

The force on particle B is by symmetry $\mathbf{k}_B = -\mathbf{k}_A$. By the product rule we have

$$\begin{aligned}\frac{\partial \mathbf{k}_A}{\partial \mathbf{r}_A} = & -k\frac{\mathbf{e}}{\| \mathbf{e} \|}\left(\frac{\partial\left(\| \mathbf{e} \|\right)}{\partial \mathbf{r}_A}\right) \\ & - k\left(\| \mathbf{e} \| - l_0\right)\left(\frac{\partial\left(\frac{\mathbf{e}}{\|\mathbf{e}\|}\right)}{\partial \mathbf{r}_A}\right) \\ & - c\left(\frac{\partial\left(\frac{\mathbf{e}}{\|\mathbf{e}\|}\right)}{\partial \mathbf{r}_A}\right)\frac{\mathbf{e}^T(\mathbf{v}_A - \mathbf{v}_B)}{\| \mathbf{e} \|} \\ & - c\frac{\mathbf{e}}{\| \mathbf{e} \|}(\mathbf{v}_A - \mathbf{v}_B)^T\left(\frac{\partial\left(\frac{\mathbf{e}^T}{\|\mathbf{e}\|}\right)}{\partial \mathbf{r}_A}\right)\end{aligned}$$

Defining the auxiliary variables

$$\begin{aligned}\mathbf{h}^T &= \frac{\partial\left(\| \mathbf{e} \|\right)}{\partial \mathbf{r}_A} = \frac{\mathbf{e}^T}{\| \mathbf{e} \|} \\ \mathbf{H} &= \frac{\partial\left(\frac{\mathbf{e}}{\|\mathbf{e}\|}\right)}{\partial \mathbf{r}_A} = \frac{1}{\| \mathbf{e} \|}\mathbf{I}_{2\times 2} - \frac{\mathbf{e}\mathbf{e}^T}{\| \mathbf{e} \|^3}\end{aligned}$$

We may rewrite the equation

$$\begin{aligned}\frac{\partial \mathbf{k}_A}{\partial \mathbf{r}_A} = & -\frac{\mathbf{e}}{\| \mathbf{e} \|}\left(k\mathbf{h}^T + c(\mathbf{v}_A - \mathbf{v}_B)^T\mathbf{H}\right) \\ & - \left(k\left(\| \mathbf{e} \| - l_0\right) + c\frac{\mathbf{e}^T(\mathbf{v}_A - \mathbf{v}_B)}{\| \mathbf{e} \|}\right)\mathbf{H}\end{aligned}$$

and $\frac{\partial \mathbf{k}_A}{\partial \mathbf{r}_B} = -\frac{\partial \mathbf{k}_A}{\partial \mathbf{r}_A}$. Now the stiffness matrix for our 2 particle mass spring system would be

$$\mathbf{K} = \begin{bmatrix}\frac{\partial \mathbf{k}_A}{\partial \mathbf{r}_A} & \frac{\partial \mathbf{k}_A}{\partial \mathbf{r}_B} \\ \frac{\partial \mathbf{k}_B}{\partial \mathbf{r}_A} & \frac{\partial \mathbf{k}_B}{\partial \mathbf{r}_B}\end{bmatrix}$$

The damping terms are

$$\frac{\partial \mathbf{k}_A}{\partial \mathbf{v}_A} = -c\frac{\mathbf{e}\mathbf{e}^T}{\| \mathbf{e} \|^2}$$

and $\frac{\partial \mathbf{k}_A}{\partial \mathbf{v}_B} = -\frac{\partial \mathbf{k}_A}{\partial \mathbf{v}_A}$ so the damping matrix is

$$\mathbf{C} = \begin{bmatrix} \frac{\partial \mathbf{k}_A}{\partial \mathbf{v}_A} & \frac{\partial \mathbf{k}_A}{\partial \mathbf{v}_B} \\ \frac{\partial \mathbf{k}_B}{\partial \mathbf{v}_A} & \frac{\partial \mathbf{k}_B}{\partial \mathbf{v}_B} \end{bmatrix}$$

and the mass matrix is

$$\mathbf{C} = \begin{bmatrix} m_A\mathbf{I} & \mathbf{0} \\ \mathbf{0} & m_B\mathbf{I} \end{bmatrix}$$

where m_A and m_B are the respective particle masses.

Observe that if one extends to a mass spring system with several particles and several springs. Then one would assemble a global stiffness matrix and damping matrix by looping over all springs and compute the above local spring element matrices and accumulate their values into a system global matrix.

28 Results

Since this research is still very much ongoing, we have less results to show that was the case for the other parts of this thesis. We have however made a simple setup with 2 particles connected by a spring as described in section 27. To test the method we shoot each of the particles in a given direction, and let them travel for a fixed number of timesteps under the influence of a given force and the spring force from their connection. For each of the time-steps we measure the position and velocity of each of the particles we then make a perturbed solution which we use as a starting guess for the optimization. Our objective function is the least squares difference of velocity and position for every fourth frame.

We then run the optimization routine with the expectation of getting a solution which is close to the original. We use the inbuilt matlab solver fminunc, with the gradient which we have calculated using the results from above. The optimization took around 10-15 seconds in most cases. Figure 41 show the results of our test.

As it can be seen the result is almost identical to the ground truth.

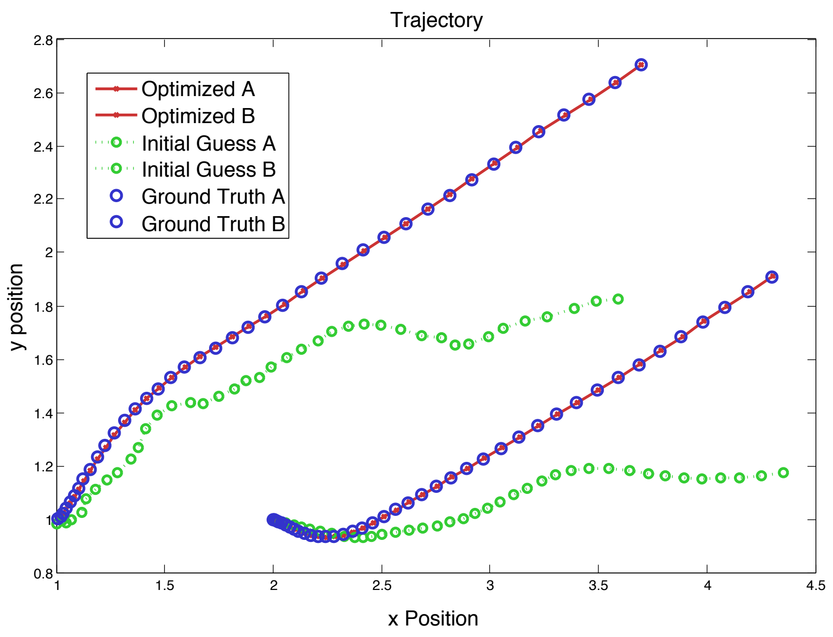


Figure 41: The result of testing the adjoint method on two particles connected with a spring.

29 Conclusion on space time optimization

A lot of work still remains for this method to be feasible but the preliminary result are very promising. We have shown that we can set up the relation of the passive deformable mesh and shown a simple 2D example to verify the correctness of the solution. We have sketched the modification necessary, to incorporate active contraction of the mesh along a fibre direction. What we are still missing is the actual implementation of the general method in 3D for larger, more complex systems and the process of validating the method based on the results

Part V

Concluding remarks

This concludes my research work for this PhD project in the following, I will try to wrap up and conclude on the project as a whole. Three years is a long time and much have happened. During these years I have learned many things I didn't even know was possible (some of them weren't when I started out.) Most of my research have been published but more ideas keep popping up, so if possible, I expect to keep doing this for the next many years.

30 Conclusion

Initially I set out to do Interactive Human motion. Our common goal in the HUMIM group, was a system which could help people in physiotherapeutic rehabilitation. As exciting a goal as this was it did not fully encompass what I wanted to do. My interests are in the simulation of human motion, not in vision and machine learning, even though these subject are interesting in their own right. My goal was to be able to simulate motion of humans but even when I started I knew that to make a full simulation system from scratch in three years would be an impossible task. Even more so, as I had to learn (and to some extend invent) all the theory behind it first. My more realistic goal was the following

- develop a robust and intuitive inverse kinematics system for motion planning and kinematic analysis.
- investigate methods and develop a model for joint constraints which could describe the highly non convex form of human joint limits.
- develop a muscle model which included deformation without loosing the interactive quality.
- investigate ideas for controlling the motion of complex deformable bodies in space and time.

I believe I have succeeded in at least coming up with some solutions for each of these points.

I believe that the methods and models that we have developed are usable and applicable and I hope that People will use them, not just as citations in their own papers but as actual tools in their projects, be that research, industrial, artistic or something completely different.

31 Future Work

Human motion simulation is a huge area. I have so far only scratched the surface of the many possibilities for contributing to this exciting field of research. With the knowledge I have acquired through my three years of PhD studies, I expect to be able to continue contributing to the Human Motion simulation research. Some of the topics I find most appealing in the light of my previous work are:
Developing the control scheme further so it becomes possible to drive the spline activation using kinematic data.

Applying the spline activation in a biomechanical muscle model thus, hopefully making better analysis of muscle activation analysis possible.
Combine all the techniques I have developed in a motion analysis framework. Such a framework might be able to do synthesized or optimized motion from temporally sparse samples, instead of relying, as most current tools do, on full motion captured kinematic data

32 Epilogue

Life is just a party, and parties weren't meant to last!

Prince

All things must have an end. And so this is it, for my PhD studies. This Thesis effectively concludes three exciting years of research. I can honestly say that I have had fun all the way. I feel privileged to have been able to follow my whims and ideas and to explore what I found interesting. The only Problem with this is, I want more. I cannot now imagine a life where I do not have the opportunity to investigate ideas, and invent solutions for problems I encounter and find interesting.

I am convinced that I will continue doing research , either in Academia if the opportunity presents itself, in an R & D department in a private company, or just as something i do on the side. Conviction got me this far, i am sure it will get me further. It is My hope that my collaborators and colleagues are prepared to indulge me in the future as they have done until now.

with this I conclude.

References

[1] Blender. The blender foundation. web page. http://www.blender.org.

[2] the Ogre team. Ogre3d. web page. http://www.ogre3d.org.

[3] Morten Engell-Nørregård and Kenny Erleben. A projected back-tracking line-search for constrained interactive inverse kinematics. *Computers & Graphics*, 35(2):288 – 298, 2011.

[4] M. Engell-Nørregård, S. Hauberg, J. Lapuyade, K. Erleben, and K.S. Pedersen. Interactive inverse kinematics for human motion estimation.

[5] Autodesk. Autodesk. web page. http://usa.autodesk.com.

[6] Søren Hauberg, Jerome Lapuyade, Morten Engell-Nørregård, Kenny Erleben, and Kim Steenstrup Pedersen. Three dimensional monocular human motion analysis in end-effector space. In Daniel Cremers et al., editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, pages 235–248. Springer, August 2009.

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

[8] Michael Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 263–270, New York, NY, USA, 1985. ACM Press.

[9] T. C. Hsia and Z. Y. Guo. New inverse kinematic algorithms for redundant robots. *Journal of Robotic Systems*, 8(1):117–132, 1991.

[10] Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Unpublished survey, april 2004.

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

[12] Anthony A. Maciejewski. Motion simulation: Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Comput. Graph. Appl.*, 10(3):63–71, 1990.

[13] Pasquale Chiacchio and Bruno Siciliano. A closed-loop jacobian transpose scheme for solving the inverse kinematics of nonredundant and redundant wrists. *Journal of Robotic Systems*, 6(5):601–630, 1989.

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

[15] Chris Wellman. Inverse kinematics and geometric constraints for articulated figure manipulation. Master's thesis, Simon Fraser University, 1993.

[16] Edmond S. L. Ho, Taku Komura, and Rynson W. H. Lau. Computing inverse kinematics with linear programming. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 163–166, New York, NY, USA, 2005. ACM.

[17] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popovi&#263;. Mesh-based inverse kinematics. *ACM Trans. Graph.*, 24(3):488–495, 2005.

[18] Hadi Moradi and Sukhan Lee. Joint limit analysis and elbow movement minimization for redundant manipulators using closed form method. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *Advances in Intelligent Computing, International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings, Part II*, volume 3645 of *Lecture Notes in Computer Science*, pages 423–432. Springer, 2005.

[19] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521, 2004.

[20] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics (TOG)*, 22(3):417–426, 2003.

[21] C. Karen Liu, Aaron Hertzmann, and Zoran Popovi&#263;. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph.*, 24(3):1071–1081, 2005.

[22] Walter Maurel and Daniel Thalmann. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics*, 24(2):203–218, 2000.

[23] Lorna Herda, Raquel Urtasun, Pascal Fua, and Andrew J. Hanson. Automatic determination of shoulder joint limits using quaternion field boundaries. *I. J. Robotic Res.*, 22(6):419–438, 2003.

[24] Michael Meredith and Steve Maddock. Using a half-jacobian for real-time inverse kinematics. In *Proceedings of The 5th International Conference on Computer Games: Artificial Intelligence, Design and Education (CGADIDE*, pages 81–88, Reading, United Kingdom, November 2004.

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

[26] Wei Shao and Victor Ng-Thow-Hing. A general joint component framework for realistic articulation in human characters. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 11–18, New York, NY, USA, 2003. ACM.

[27] OpenTissueBoard. Opentissue. web page, 2010. http://www.opentissue.org.

[28] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[29] Yousef Saad. *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, Philadelpha, PA, 2003.

[30] J. B. Rosen. The gradient projection method for nonlinear programming. part i. linear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 8(1):181–217, 1960.

[31] J. B. Rosen. The gradient projection method for nonlinear programming. part ii. nonlinear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):514–532, 1961.

[32] Michael Meredith and Steve Maddock. Adapting motion capture data using weighted real-time inverse kinematics. *Comput. Entertain.*, 3(1):5–5, 2005.

[33] Se-Young Oh, David Orin, and Michael Bach. An inverse kinematic solution for kinematically redundant robot manipulators. *Journal of Robotic Systems*, 1(3):235–249, 1984.

[34] Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *journal of graphics tools*, 10(3):37–49, 2005.

[35] Ladislav Kavan and Jiri Zara. Spherical blend skinning: A real-time deformation of articulated models. In *2005 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 9–16. ACM Press, April 2005.

[36] Matthew Mason. *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, Massachusetts, London, England, August 2001. Intelligent Robotics and Autonomous Agents Series, ISBN 0-262-13396-2.

[37] CGgenie. Cg genie. web page, 2010. http://cgenie.com/.

[38] Zlib. Zlib-libpng license. web page, 2010. http://opensource.org/licenses/zlib-license.php.

[39] Morten Engell-Nørregård and Kenny Erleben. A Projected Non-linear Conjugate Gradient Method for Interactive Inverse Kinematics. In *MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, 2009.

[40] Morten Engell-Nørregård. A comparative analysis of selected optimization solvers, 2008.

[41] Søren Hauberg. *Spatial Models of Human Motion*. PhD thesis, 2011.

[42] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *ICRA*, pages 2641–2648, 2006.

[43] Ronald Poppe. Vision-based human motion analysis: An overview. *Computer Vision Image Understing*, 108(1-2):4–18, 2007.

[44] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian Process Dynamical Models for Human Motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):283–298, 2008.

[45] Raquel Urtasun, David J. Fleet, and Pascal Fua. 3D People Tracking with Gaussian Process Dynamical Models. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society , Conference on Computer Vision and Pattern Recognition*, pages 238–245, Washington, DC, USA, 2006. IEEE Computer Society.

[46] Zhengdong Lu, Miguel Carreira-Perpinan, and Cristian Sminchisescu. People Tracking with the Laplacian Eigenmaps Latent Variable Model. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1705–1712. MIT Press, Cambridge, MA, 2008.

[47] Cristian Sminchisescu and Bill Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking. In *In IEEE International Conference on Computer Vision and Pattern Recognition*, pages 69–76, 2003.

[48] M. Engell-Nørregård, S. Niebe, and K. Erleben. A joint-constraint model for human joints using signed distance-fields. *Multibody System Dynamics*, pages 1–13, 2012.

[49] M. Engell-Nørregård, S. Niebe, and K. Erleben. Local joint–limits using distance field cones in euler angle space. In *Computer graphics international*, 2010.

[50] *CMU Graphics Lab Motion Capture Database*, http://mocap.cs.cmu.edu/, 2009. Carnegie Mellon University.

[51] Morten Engell-Nørregård, Sarah Niebe, and Kenny Erleben. Local Joint–Limits using Distance Field Cones in Euler Angle Space.

[52] Joo Kim, Jingzhou Yang, and Karim Abdel-Malek. A novel formulation for determining joint constraint loads during optimal dynamic motion of redundant manipulators in dh representation. *Multibody System Dynamics*, 19:427–451, 2008. 10.1007/s11044-007-9100-4.

[53] Hiroyuki Sugiyama and Hiroki Yamashita. Spatial joint constraints for the absolute nodal coordinate formulation using the non-generalized intermediate coordinates. *Multibody System Dynamics*, 26:15–36, 2011. 10.1007/s11044-010-9236-5.

[54] Darwin 3d. Acclaim fileformat. web page. http://www.darwin3d.com/gamedev/.

[55] Morten Engell-Nørregård and Kenny Erleben. Estimation of Joint types and Joint Limits from Motion capture data. In *WSCG 2009: 17-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2009.

[56] A. E. Engin and S. T. Tümer. Three-dimensional kinematic modelling of the human shoulder complex—part i: Physical model and determination of joint sinus cones. *Journal of Biomechanical Engineering*, 111(2):107–112, 1989.

[57] James U. Korein. *A Geometric Investigation of reach*. MIT Press Cambridge, 1985.

[58] Sung-Hee Lee and Demetri Terzopoulos. Spline joints for multibody dynamics. *ACM Trans. Graph.*, 27(3):1–8, 2008.

[59] Lorna Herda, Raquel Urtasun, and Pascal Fua. Hierarchical implicit surface joint limits for human body tracking. *Computer Vision and Image Understanding*, 99(2):189–209, 2005.

[60] A. Bærentzen. Robust generation of signed distance fields from triangle meshes. In D. Fellner, T. Moller, and S. Spencer, editors, *Fourth International Workshop on Volume Graphics, 2005.*, pages 167–239, jun 2005.

[61] Erik Dam, Martin Koch, and Martin Lillholm. Quaternions, interpolation and animation. Technical report, University of Copenhagen, 1998.

[62] Jakob Andreas Bærentzen and Niels Jorgen Christensen. Interactive modelling of shapes using the level-set method. *International Journal of Shaping Modeling*, 8(2):79–97, 2002.

[63] A. V. Hill. The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 126(843):pp. 136–195, 1938.

[64] FE Zajac, EL Topp, and PJ Stevenson. A dimensionless musculotendon model. *Proceedings IEEE Engineering in Medicine and Biology*, pages 26–31, 1986.

[65] Alan H. Barr. Global and local deformations of solid primitives. *SIGGRAPH Comput. Graph.*, 18:21–30, January 1984.

[66] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20:151–160, August 1986.

[67] Andrew Witkin, Kurt Fleischer, and Alan Barr. Energy constraints on parameterized models. *SIGGRAPH Comput. Graph.*, 21:225–232, August 1987.

[68] Ronen Barzel and Alan H. Barr. A modeling system based on dynamic constraints. *SIGGRAPH Comput. Graph.*, 22:179–188, June 1988.

[69] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. *SIGGRAPH Comput. Graph.*, 26:309–312, July 1992.

[70] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. *SIGGRAPH Comput. Graph.*, 23:207–214, July 1989.

[71] Doug L. James and Dinesh K. Pai. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. Graph.*, 21:582–585, July 2002.

[72] A. Aubel and D. Thalmann. Efficient muscle shape deformation. In *Deformable avatars: IFIP TC5/WG5. 10 DEFORM'2000 Workshop, November 29-30, 2000, Geneva, Switzerland and AVATARS'2000 Workshop, November 30-December 1, 2000, Lausanne, Switzerland*, page 132. Springer Netherlands, 2001.

[73] K.S. Lee and G. Ashraf. Simplified Muscle Dynamics for Appealing Real-Time Skin Deformation. In *International Conference on Computer Graphics and Virtual Reality*, 2007.

[74] Gavin S. P. Miller. The motion dynamics of snakes and worms. *SIGGRAPH Comput. Graph.*, 22(4):169–173, June 1988.

[75] Radek Grzeszczuk and Demetri Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 63–70, New York, NY, USA, 1995. ACM.

[76] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, July 1997.

[77] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. *SIGGRAPH Comput. Graph.*, 23:243–252, July 1989.

[78] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 43–50, New York, NY, USA, 1994. ACM.

[79] Chris Wojtan, Peter J. Mucha, and Greg Turk. Keyframe control of complex particle systems using the adjoint method. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 15–23, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[80] Jernej Barbič, Marco da Silva, and Jovan Popović. Deformable object animation using reduced optimal control. *ACM Trans. Graph.*, 28(3):53:1–53:9, July 2009.

[81] Andrew Witkin and Michael Kass. Spacetime constraints. *SIGGRAPH Comput. Graph.*, 22:159–168, June 1988.

[82] M. Viceconti, D. Testi, F. Taddei, S. Martelli, G. J. Clapworthy, and S. V. Jan. Biomechanics modeling of the musculoskeletal apparatus: Status and key issues. *Proceedings of the Ieee*, 94(4):725–739, April 2006.

[83] Marcus G. Pandy and Thomas P. Andriacchi. Muscle and joint function in human locomotion. *Annual Review of Biomedical Engineering*, 12(1):401–433, 2010.

[84] Qing hong Zhu, Yan Chen, and Arie E. Kaufman. Real-time biomechanically-based muscle volume deformation using fem. *Comput. Graph. Forum*, 17(3):275–284, 1998.

[85] C W J Oomens, M Maenhout, C H van Oijen, M R Drost, and F P Baaijens. Finite element modelling of contracting skeletal muscle. *Philosophical Transactions. Royal Society B*, 358(1437), 2003.

[86] Xianlian Zhou and Jia Lu. Nurbs-based galerkin method and application to skeletal muscle modeling. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, SPM '05, pages 71–78, New York, NY, USA, 2005. ACM.

[87] J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 68–74, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[88] Joseph Teran, Eftychios Sifakis, Silvia S. Blemker, Victor Ng-Thow-Hing, Cynthia Lau, and Ronald Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics*, 11:317–328, 2005.

[89] L. P. Nedel and D. Thalmann. Real time muscle deformations using mass-spring systems. In *Proceedings of the Computer Graphics International 1998*, CGI '98, pages 156–, Washington, DC, USA, 1998. IEEE Computer Society.

[90] Shinjiro Sueda, Andrew Kaufman, and Dinesh K. Pai. Musculotendon simulation for hand animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 27(3), 2008.

[91] Dinesh K. Pai, Shinjiro Sueda, and Qi Wei. Fast physically based musculoskeletal simulation. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[92] Brian A. Garner and Marcus G. Pandy. The obstacle-set method for representing muscle paths in musculoskeletal models. "*Computer Methods in Biomechanics and Biomedical Engineering*", 3, 2000.

[93] D. Baraff and A. Witkin. Partitioned dynamics. Technical Report CMU-RI-TR-97-33, Robotics Institute, Carnegie Mellon University, 1997.

[94] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, GI '04, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[95] Matthias Teschner, Bruno Heidelberger, Matthias Muller, and Markus Gross. A versatile and robust model for geometrically complex deformable solids. In *Proceedings of the Computer Graphics International*, pages 312–319, Washington, DC, USA, 2004. IEEE Computer Society.

[96] Kenny Erleben, Jon Sporring, Knud Henriksen, and Henrik Dohlman. *Physics-based Animation (Graphics Series)*. Charles River Media, Inc., Rockland, MA, USA, 2005.

[97] W.L. Briggs, V.E. Henson, and S.F. McCormick. *A multigrid tutorial*. Miscellaneous Bks. Society for Industrial and Applied Mathematics, 2000.

[98] B. Gilles and N. Magnenat-Thalmann. Musculoskeletal mri segmentation using multi-resolution simplex meshes with medial representations. *Medical Image Anaylsis*, pages 291–302, June 2010.

[99] J. Rasmussen, M. Damsgaard, E. Surma, S.T. Christensen, M. de Zee, and V. Vondrak. Anybody-a software system for ergonomic optimization. In *Fifth World Congress on Structural and Multidisciplinary Optimization, 2003*, 2003.

[100] S.L. Delp, F.C. Anderson, A.S. Arnold, P. Loan, A. Habib, C.T. John, E. Guendelman, and D.G. Thelen. Opensim: open-source software to create and analyze dynamic simulations of movement. *Biomedical Engineering, IEEE Transactions on*, 54(11):1940–1950, 2007.

[101] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Trans. Graph.*, 23:449–456, August 2004.

[102] Kenny Erleben. Hyper elasticity for small and large deformations based on a lagrangian formulation. Technical Report 2011/8, Department of Computer Science, University of Copenhagen, Denmark, December 2011.

[103] Kenny Erleben. The tangent stiffness matrix for for the implicit nonlinear finite element method. Technical Report 2011/9, Department of Computer Science, University of Copenhagen, Denmark, December 2011.

[104] Kenny Erleben. Preliminaries on tensor notation and continuum mechanics. Technical Report 2011/7, Department of Computer Science, University of Copenhagen, Denmark, December 2011.

[105] Kenny Erleben. Implementation tricks and tips for finite element modeling of hyper elastic materials. Technical Report 2011/10, Department of Computer Science, University of Copenhagen, Denmark, December 2011.

[106] Kenny Erleben. hyper-sim. Published online at code.google.com/p/hyper-sim/, November 2011. Open source project for simulation methods for hyper elastic materials in physics-based animation.