OPTIMIZATION-BASED DYNAMIC HUMAN MOTION PREDICTION

by

Emily Nicole Horn

A thesis submitted in partial fulfillment
of the requirements for the Master of
Science degree in Mechanical Engineering
in the Graduate College of
The University of Iowa

December 2005

Thesis Supervisor: Professor Karim Abdel-Malek

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the Master's thesis of

Emily Nicole Horn

has been approved by the Examining Committee
for the thesis requirement for the Master of Science
degree in Mechanical Engineering at the December 2005 graduation.

Thesis Committee: ______________________________
Karim Abdel-Malek, Thesis Supervisor

____
Jasbir Arora

____
Jia Lu

To My Husband
Thank you for all your love and patience

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This thesis investigates a formulation for the simulation of the human swinging motion. A swinging motion consists of a human moving his/her torso while maintaining contact with the ground at either both his/her hands or his/her hands and feet. Examples of this type of motion include doing pull ups or shifting weight while on a ladder. The simulation uses optimization-based dynamics to predict the motion of a digital human. Eventually, this simulation could be expanded to predict the motion of climbing a ladder, opening a door, or climbing over an obstacle. The long-term vision is to be able to conduct ingress/egress studies into the cab of a vehicle. When used with digital prototyping, the digital human could provide feedback on handhold and foothold positions or predict the stress on the muscles during the motion. Manufacturers could use this information to improve the ergonomics of a task, to reduce the strain on the body and muscles during a task, and to improve designs in general.

Motivation

Generating realistic human motion has widespread applications: realistic graphics for animation and gaming, ergonomic studies of workstations, task feasibility studies, injury prediction, digital prototyping, and many more. Today, digital humans are commercially available for use in animation and design, but these digital humans are simple mannequins that must be posed by the user or driven by motion capture data. These available programs, in turn, require either an expert user to manipulate the mannequins or a mock-up of the design to use for collecting motion capture data. A digital human that can move naturally without the aid of the user would be an enormous benefit to animators, ergonomic analysts, and designers. Such an intelligent digital human would be an even greater benefit if it could provide feedback during its motion. The design process would become faster, easier, and less costly, since the need for physical prototypes and test subjects could be greatly reduced. Due to the many possible applications, creating a virtual human that can move naturally and provide feedback is the subject of much research, but a realistic and effective virtual human has yet to be developed.

Despite the need for an intelligent digital human, creating such a digital human presents two main difficulties. The first problem is the complexity of the human skeleton. The human body contains intricate joints like the clavicle and spine. To truly study the motion of these joints, a detailed, computationally intensive model is needed. Most often, complex joints like these are modeled by multiple rotational joints, which only approximate gross motion. The vast number of DOFs also makes modeling the human difficult. Inverse kinematic solutions are difficult to implement with a large number of DOFs, so digital humans must be modeled with very few DOFs, or methods other than traditional inverse kinematics must be used. Many human models only predict motion for the upper body (Alexander, 1997; Kim et al., 2004; Mi, 2003; Tolani et al., 2000; Zhang and Chaffin, 2000) or are two-dimensional (2-D) models (Fang and Pollard, 2003; Khang et al., 1989; Rostami and Bessonnet, 2001; Witkin and Kass, 1988). To be truly realistic, the models need to be three-dimensional (3-D) and include whole-body motion.

The second problem with simulating human motion is the redundancy of the human musculoskeletal system. Redundancy in the system results in infinitely many possible motions (or postures) to complete a given task. Thus, the question arises as to why humans use a specific posture or motion. The posture or motion of a body depends upon individual anatomical structure, physiology, personal habits, training, motivation, and environment. Humans not only move differently from one another, but they also move differently depending on the task they are trying to accomplish and the environmental constraints around them. Motion prediction using kinematics can account for physical limitations of the skeleton system and for environmental constraints. Motion prediction with dynamics increases the realism of the motion by also considering the forces involved in the motion. Finally, optimization used in conjunction with dynamics increases the realism further by finding a solution based on an objective function. This objective function, if formulated correctly, encompasses the motivating factors that affect human motion. It can include quantitative measures like joint torque or energy, or more qualitative ideas like discomfort or visibility. Studying which objective functions result in the most realistic motion can provide insight into the motivating reasons of how and why people move. Simulations using optimization coupled with dynamics have been created, but research is still ongoing to develop objective functions that provide consistently realistic results.

The previous two problems dealt with human motion; there is one last problem unique to the motion of swinging. Most of the previously completed simulations dealt with known external forces. The motion of swinging, however, has unknown reaction forces at both the hands and feet of the digital human, and these forces need to be determined.

Objectives

The primary objective of this thesis is to develop a method to predict realistic whole-body motion of a virtual human. To be realistic, the motion needs to appear similar to the motion of a real human. In addition, the motion needs to obey the laws of physics and the physical limitations of the body. Specific objectives are discussed as follows:

a. *To develop a systematic way to represent the joint structure of a human and represent its position and orientation in space*. The digital human used in this thesis is named Santos™ and was created at The University of Iowa in the Virtual Soldier Research Laboratory. Santos™ is modeled with 49 degrees of freedom (DOFs) (not including the hands and eyes). To deal with the large number of

DOFs, the Denavit-Hartenberg method (DH method) is used to relate the joints to one another (Denavit and Hartenberg, 1955). This thesis will expand on previous work done at VSR by increasing motion prediction of the upper body to motion prediction of the whole body. This thesis will add six additional DOFs to describe the position and orientation of Santos^{TM} in space. The swinging simulation will solve for 55 DOFs to describe the motion of Santos^{TM}.

b. *To formulate the swinging motion prediction problem and develop a method for implementing the formulation.* To create a swinging simulation, the motion of the human and the forces necessary to create this motion need to be predicted. This thesis will use optimization-based inverse kinematics to predict the motion of the human. Using optimization overcomes the problem presented by the large number of DOFs and overcomes the problem of redundancy. Previous simulations completed with Santos^{TM} have provided the external forces applied to Santos as input and so computing the generalized forces necessary for the motion was straightforward. The swinging motion, however, involves closed kinematic loops, which means both the reaction forces and generalized forces are unknown. This results in an indeterminant system. This thesis will optimize for the unknown reaction forces and then calculate the generalized forces using inverse dynamics. Finally, since global movement is allowed in this simulation, equilibrium must also be maintained. This will be included as a constraint within the optimization formulation.

c. *Formulate an optimization objective function that results in realistic motion.* The objective function of the simulation is extremely important, because it is responsible for producing realistic motion. In this thesis, two new objective functions are formulated and compared to a function popular with other human simulations. A well-established objective function used in dynamic predictions is the sum of the joint torque squared. The first new objective function normalizes the joint torque before squaring it. By normalizing the joint torque, this function is working under the hypothesis that humans move in such a way to distribute the required torque for the motion to their joints according to the strength of each joint. The second new objective function minimizes the largest normalized joint torque. This function assumes that humans move in such a way to avoid experiencing a large normalized torque at any one joint. The resulting motion from using these three objective functions will be compared.

d. *To provide feedback to the user to evaluate the motion.* Santos™ not only appears to move naturally, but also is a mathematical model and can provide quantitative feedback to the user. For this thesis, the feedback includes the torque at each joint and the maximum torque experienced during the motion. Torque can also be used by down stream programs to calculate muscle activation, muscle strain, fatigue, or energy consumption.

Literature Review

There have been many approaches to predicting human motion. This section will briefly describe previous work in five major areas of research. The first area of study is using motion capture data to predict human motion. Using data from motion capture may be the most realistic way to predict motion, because it uses actual human motions to predict new ones. However, it is extremely limited and resource intensive. A more general approach to predicting motion is to model the human as a linkage chain and mathematically predict motion. The second major area of study uses inverse kinematics to predict human posture to touch a particular target point in space. Inverse kinematics, however, is limited in the number of degrees of freedom it can handle. When combined with optimization this limit is removed; this is the third major area of study: optimization-based kinematics. The fourth area of study uses dynamics to predict motion. By adding dynamics to the formulation, the fidelity increases because it accounts for external forces acting on the body and the strength of the human. Similar to kinematics, however, dynamics by itself is very limiting; when combined with optimization dynamics provides the best platform for human motion prediction. This is the fifth and final area of study that will be discussed: optimization-based dynamics.

Motion Capture

The first approach to simulating human motion is to use actual test data to predict similar motions. This approach addresses one of the disadvantages with predictions that use a mathematical model of the human: “anticipator posture control” (Chaffin et al., 1999). This is the ability of humans to anticipate the dynamics of a motion. For instance, when reaching out to one side a human will lift one of his/her legs in the direction opposite the reach to maintain balance before his/her center of mass is outside the stance. The subject anticipates that he/she will be out of balance. Researchers also claim the motion-capture approach is more credible because it stems from test data.

The first method that uses motion capture data to predict human motion uses a regression model (Chaffin et al., 1999). A large set of data for a particular task is collected with motion capture equipment. The data includes a profile of the joint angles versus time for the movement of each test subject. These joint angles are input to the regression model, which is used to predict the joint angles for a similar task. However, when used to predict the motion of a person reaching for something, the hand does not always reach the target as expected because the joint angles are predicted and inverse kinematics is not used. To solve this problem the regression method was used in conjunction with an optimization-based differential inverse kinematics approach (Zhang and Chaffin, 2000).

Another method that uses motion capture data to predict human motion uses a control system to predict physically viable motion similar to the captured motion (McGuan, 2001). ADAMS (Mechanical Dynamics, Inc.) has a system called LifeMod, which uses motion capture data to simulation a motion. The human model is physics-based so it will not violate environment contacts and it will not violate joint restrictions. The motion and muscle forces to produce the motion are calculated simultaneously by using a control loop. Once the muscle forces are found the simulation can also be driven using forward dynamics.

Another use of control systems and motion capture data uses trajectory tracking to follow the motion capture data and a balance controller to keep the digital human upright in the presence of external disturbances (Zordan and Hodgins, 2002). Contact forces are computed from a physical collision model; the digital human responds automatically to these contact forces and smoothly returns to tracking.

A final method discussed here uses motion capture data to provide motion during underconstrained sections of motion (Playter, 2000). Playter uses a dynamic simulation to predict human motion during running, but supplements the prediction with motion capture data during the flight phase, which is underconstrained.

Kinematics

All of the methods using motion capture data require the creation of libraries of data for each specific task that needs to be predicted. Using a mathematical model to simulate human motion removes this requirement and allows the user to create simulations for which motion capture data has not been taken. In this section, simulations using a mathematical model and kinematics are discussed. For a digital human model, forward kinematics entails finding the position and orientation of a point on the body (usually the fingertip), given the angles of the joints of the body. Alternatively, inverse kinematics entails finding the angles of the joints of the body, given the position and orientation of a point on the body or determining that there is no solution. In general, the inverse kinematic problem is much more complicated, especially for a system like the human body, which is redundant, and involves a large number of DOFs.

The first method uses inverse kinematics to solve for static posture prediction. Since the human skeleton is redundant, all possible solutions are found and then the user explores the solutions using a set of parameters (Tolani et al., 2000). Since human posture can vary so much depending on personal habits, motivation, task, and environment, it is difficult to claim one posture is the *most* realistic. To avoid making this judgment Tolani allows the user to chose the pose that fits the situation. In terms of implementation, this method is analytically intensive and cannot be easily updated for changes in link structure or for additional DOFs. As with all kinematic solutions, it also does not account for forces acting on the body. The benefits of using an analytical formula are that the method is more efficient and reliable, and there are no problems with Jacobian singularities or local minima. When a purely analytical solution cannot be obtained a combination of analytical and numerical techniques is used. Using numerical methods with inverse kinematics can introduce Jacobian singularities and optimization methods introduce problems with local minima. Optimization with kinematics will be discussed in the next section.

Another method to solving inverse kinematics for upper body motion prediction is to reduce the number of DOFs by adding additional constraints according to the object being manipulated by the hand (Kondo, 1991). This reduces the number of DOFs that need to be found using inverse kinematics. From a description of the object to be manipulated, the hand grasp is selected and then the position and orientation of the hand. The position of the hand is used to determine the joint angles for the shoulder and elbow are found using a "sensorimotor transformation model" (Soechting and Flanders, 1989). Then, the orientation of the hand is used to determine the angles in the wrist. Finally, all angles are precisely adjusted to conform the hand to the object.

Optimization with Kinematics

Motion and posture simulation using purely kinematic methods are restricted in the allowable number of DOFs and may produce more than one possible solution. If only one solution is desired, then inverse kinematics can be combined with optimization to find the most realistic posture or motion based on one or more objective functions. Optimization also removes the constraints on how many DOFs the model may have.

Optimization is a numerical method that finds a single solution at which a given function is minimized. The solution is a set of variables called the design variables. These design variables have to satisfy constraints that may be either equality or inequality constraints. The set of all points that satisfy the constraints are called feasible solutions. In general, there will be several points in the feasible set that are local minima for the objective function; these points have objective function values that are lower than all the points immediately surrounding them. The global minimum has the lowest objective function value of all the feasible points and is called the optimal solution.

For a posture prediction problem the design variables are the joint angles of the body. The constraints are the location and possibly orientation of an end effector (usually the fingertip) and the joint ranges of motion. Commonly, optimization-based inverse kinematic methods focus on minimizing some form of discomfort (Jung and Choe, 1996; Abdel-Malek et al., 2004b) or minimizing perturbation from a neutral position (Case et al., 1990). If more than one objective function is used, they can be combined using multi-objective optimization techniques like the objective sum method, the min-max method, or the global criterion method (Yang et al., 2004). Other objective functions that have been tried include minimizing joint displacement, minimizing change in potential energy, minimizing the distance to the target, and maximizing reachability (Abdel-Malek et al., 2004a; Abdel-Malek et al., 2004c; Yang et al., 2004).

The design of workspaces around humans is one of the main uses for kinematic posture prediction with optimization. Optimization can be used not only to find the most realistic posture to reach a point in the workspace, but also to find the best placement of the human within a workspace or the best placement of the target relative to the human (Abdel-Malek et al., 2004a; Abdel-Malek et al., 2004c).

Motion prediction broadens the approach and finds the optimal motion between two target points. For motion prediction, B-spline approximations or polynomials are used to approximate joint displacement with respect to time (Mi, 2003). Also additional objective functions are formulated that deal with the change in joint displacement over time. Examples include minimizing inconsistency, minimizing joint acceleration, and minimizing velocity at the beginning and end of the motion (Abdel-Malek et al., 2004b; Mi, 2003).

Dynamics

For simple reaching motions, natural arm movement may be represented kinematically, and dynamics can be considered as a post-processing step (Kondo, 1991). This hypothesis, however, fails when external forces are applied to the body. For example, the motion of a human carrying a heavy box will be different than the motion of the same human carrying a lighter box. Scenarios including external forces require that dynamics be taken into account when determining the motion. Dynamic motion prediction will be discussed in this section. For a digital human model, forward dynamics entails finding the motion (joint angles over time) of the body given the forces acting on the body. Inverse dynamics entails finding the forces acting on the body given the motion.

Similar to methods using kinematics, if the linkage system is not redundant only one solution exists and can be found in a closed-form (Goel, 1988). However, for human modeling this is not the case. One method for resolving the redundancy of the system and for including the affects of dynamics is to use the generalized inverse of the Jacobian and then minimize the joint torque using a least squares method (Hollerbach and Suh, 1985). In general, the method has stability problems for long motions.

Since all the forces acting on a human during motion are not necessarily known (i.e. ground reaction forces), one method for motion prediction is to combine forward dynamics with a contact force model and a control system (Playter et al., 2001; Layon et al., 2004). The body's actuating forces are simplified to generalized torques acting at each joint, which are calculated from a control system; many suitable control systems have already been developed for use with bipedal robots. To get stable motion, control parameters are tuned depending on the anthropometry, posture, and external forces. One method to model contact forces is a spring-damper system dependent on penetration. The motion resulting from these forces is calculated by the equations of motion that embody Newton's Second Law. Problems with this approach are that the ground reaction forces do not show very good correlation with test data, and the motion, while stable, may not be realistic for humans.

In another study, a feedback controller was used to maintain a vertical posture in the presence of external disturbances while minimizing energy (Khang et al., 1989a; Khang et al., 1989b). This work was used to study the implementation of functional neuromuscular stimulation systems to aid in paraplegic standing.

Optimization with Dynamics

Similar to optimization with kinematics, optimization with dynamics determines the most realistic solution according to some objective function. With the addition of dynamics, however, the solution can now be constrained to obey the laws of physics, and new objective functions can be developed that deal with energy and force. With these additions, motion prediction is more realistic than that computed using only kinematic constraints and objective functions.

Witkin and Kass (1988) were some of the first researchers to optimize for the joint angles over time rather than integrate through time starting at an initial position. They coined the term "spacetime constraints," since the constraints applied to motion through time as well as space. They tried several cost functions and claimed that the cost function used should vary with the task since the "…optimal way to perform a motion, as with any optimization, depends on what you're trying to do" (Witkin, Kass, 1988).

Since the work of Witkin and Kass, many other dynamic models have been developed. Instead of B-spline approximations, polynomials have been used to approximate joint profiles (Ayoub et al., 1998). Also, many different objective functions have been tried: minimizing weighted normalized torques (Ayoub et al., 1998), minimizing torque (Chevallereau and Aoustin, 2001; Rostami and Bossonnet, 2001), minimizing the rate of change of torque (Uno et al., 1989), minimizing weighted joint accelerations (Fang et al., 2003), minimizing energy consumption and visual discomfort (Kim et al., 2004), and minimizing metabolic energy cost (Alexander, 1997) are just a few. One paper compared gait simulations generated using optimization and seven different objective functions, and found that minimizing joint actuating torque was the most realistic for gait (Marshall et al., 1989).

Attempts have also been made to simplify the dynamic formulation by re-writing the Newton-Euler equations of motion in order to allow a linear optimization formulation (Fang et al., 2003). If an optimization problem can be formed with constraints and objective functions that are linear with respect to the design variables, the need for analytic gradients is avoided and the problem finding local minima instead of the global minimum is avoided. The disadvantage of this approach, however, is that joint torque cannot be calculated.

Overview of Thesis

Chapter 2 will provide background concerning the skeletal model for the virtual human as well as the DH-method, which provides a key component of this approach to simulating posture and motion. Chapter 3 will provide background on B-spline approximations, which are used in motion prediction to approximate the joint angle profiles and reaction-force profiles with respect to time. Chapter 4 will formulate the equations of motion for the dynamic simulation using Lagrange's equation and virtual work. Chapter 5 will describe the optimization formulation for dynamic posture prediction. This chapter includes a proof of concept for the simulation, which is a 9-degree of freedom linkage model. This simulation is then expanded to the full Santos™ model, which is a 55-degree of freedom model. Chapter 6 will extend the posture prediction formulation to a new fast method for dynamic motion prediction with both models. Finally, Chapter 7 will be a discussion of the results and future work.

CHAPTER 2

CREATING THE HUMAN MODEL

Engineers in the robotics field developed the Denavit-Hartenberg (DH) method (Denavit and Hartenberg, 1955), but the method is directly applicable to human modeling. The method presents an efficient way to express a vector in one coordinate system in terms of another coordinate system. It is an efficient method because only four parameters are necessary to relate one coordinate system to the previous one in a linkage chain. Due to the large number of degrees-of-freedom (DOFs) in the human body, a systemic method is required to locate coordinate systems on each link in a consistent manner; to describe the position and orientation of a link with respect to any other link; and calculate the position, velocity, and acceleration of a link with respect to the root coordinate system. The DH model fulfills these requirements. To apply the DH method to a virtual human, the body is modeled as a series of links and joints. Each joint has a local coordinate system associated with it and is related to the coordinate system on the previous joint in the chain by means of the DH method.

In this chapter a brief description of the human model used for this thesis (Santos™) will be given followed by an overview of the DH method. Transformation matrices necessary for dynamic simulations using Santos™ will be formulated, as well as kinematic equations to describe the position, velocity, and acceleration of any point on his body. The DH method does not traditionally account for global position and orientation of a linkage chain so transformation matrices to describe the global position and orientation of Santos™ will also be developed. Finally, implementation issues will briefly be discussed.

Human Model

In the model used in this report, one or more theoretical revolute joints represent each physical joint in the human. Prismatic joints, or sliding joints, could have been included, but by only including revolute joints in the human model it simplifies the skeleton, while still providing a very good approximation of gross motion for the human. For instance, a ball and socket joint, like the shoulder, is modeled by three revolute joints located on top of each other and corresponds to three degrees of freedom. Examples of the joints used for the shoulder, elbow, and wrist are shown in Figure 2.1. The joint structure of the digital human model used in this thesis is seen in Figure 2.2; the figure shows the 53 coordinate systems and 49 DOF for the human body. The human model includes 5 chains of joints, one chain for each limb: right arm, left arm, right leg, left leg, and the head.

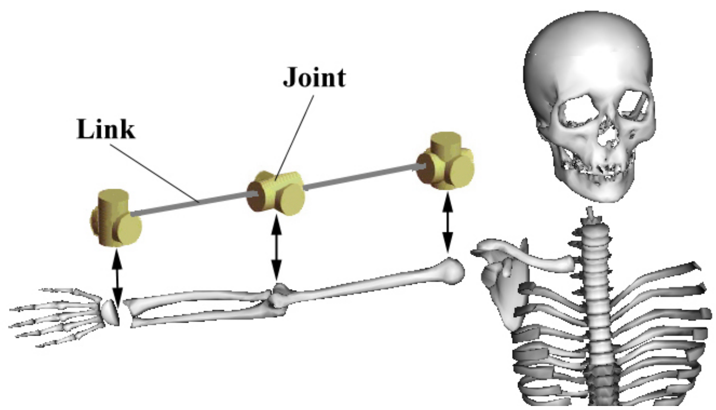

Figure 2.1. A human skeleton showing the physical joints and the corresponding mathematical joints and links.

The link lengths of the body are the $L_{\#}$ parameters shown in Figure 2.2. By changing these parameters, the model can be updated for any anthropometric data. For this report, the anthropometric data is taken from the Santos^{TM} model itself. The 3-D graphic that is used for Santos^{TM} is a body scan of a living person. To make the model move appropriately the link lengths for that human were used. The masses of the links were approximated using published data and assuming a 95th percentile male determined from his height. Test data locating the center of mass location for each link were found in literature (Williams, Lissner, and LeVeau, 1992; Chaffer and Andersson, 1984). The anthropometric data used for Santos^{TM} is given in the Appendix. The link lengths are given in Table A.1, the masses and center of mass information is given in Table A.2, and the mass moments of inertia and the products of inertia are given in Table A.3. For many of the segments, data had to be approximated from the test data available and for body segments with no test data available the information was approximated using cylinders of constant diameter and uniform mass.

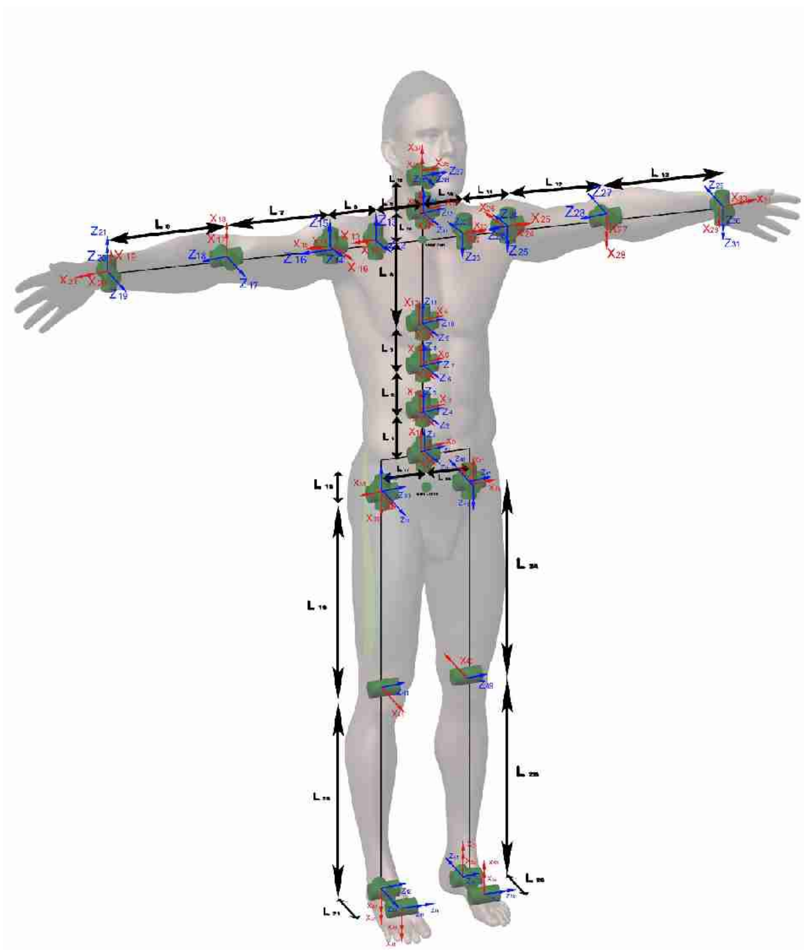

Figure 2.2. The DH model for the virtual human, showing the 53 joints and 49 degrees of freedom for the human body.

Transformation Matrices

In this section the DH method will be explained. As already stated, the DH method is simply a method to easily express a vector in one coordinate system with respect to another coordinate system. This expression is achieved by multiplying the vector by a transformation matrix that relates the two coordinate systems. A 4x4 transformation matrix that includes both rotation and translation is used. Extended vectors are used so that translation is included in the transformation, as well as rotation. The extended vector of $\mathbf{v} = [x\ y\ z]^T$ is $\mathbf{v} = [x\ y\ z\ 1]^T$. To transform a vector from coordinate system B to coordinate system A, the vector is pre-multiplied by the transformation matrix. A general transformation from B to A is given in Equation (2.1). The superscript given before the vector refers to the coordinate system in which the vector is expressed. For example, the vector $\mathbf{v}$ with respect to coordinate system A is ${}^{A}\mathbf{v}$.

$$\underbrace{{}^{A}\mathbf{v}}_{4x1} = \underbrace{\left(\begin{array}{c|c} rotation & translation \\ \underbrace{matrix}_{3x3} & \underbrace{matrix}_{3x1} \\ \hline 0 & 1 \end{array}\right)}_{4x4} \underbrace{{}^{B}\mathbf{v}}_{4x1} \tag{2.1}$$

The DH transformation matrix is a special case of the general transformation matrix given in Equation (2.1). The DH transformation matrix includes rotation and translation and is a function of four parameters: θ, α, d, and a. These parameters correspond to four simultaneous transformations: a rotation of α about the x-axis, a translation of a along the x-axis, a translation of d along the z-axis, and finally a rotation of θ about the z-axis. These parameters and their corresponding transformations are illustrated in Figure 2.3. A special requirement of the DH method is that each coordinate system is orientated such that the local z-axis represents the axis of motion for the joint and the x-axis is perpendicular to the z-axis of the previous coordinate system. These requirements are what enable the transformation matrix to be a function of only four parameters. Equation (2.2) is the transformation matrix from the x_i coordinate system to the x_{i-1} coordinate system, which is referred to as $\mathbf{T}^{(i-1,i)}$.

$$\mathbf{T}^{(i-1,i)} = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

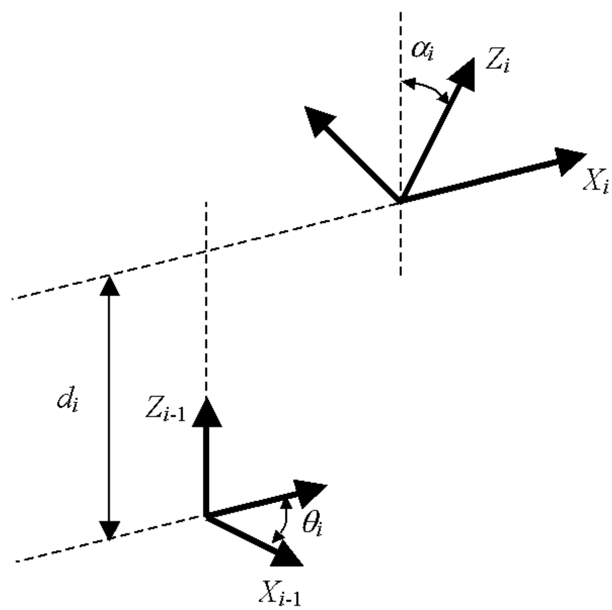

Figure 2.3. The DH parameters for transforming a vector from the x_i coordinate system to the $x_{i\text{-}1}$ coordinate system.

All the information about the orientation of coordinate system x_i with respect to the previous coordinate system is contained in the four DH parameters: θ_i, α_i, a_i, and d_i. For a revolute joint, the last three parameters are constants and represent geometric data for the body, and θ_i is the sum of a constant and the angular joint displacement. For a prismatic joint, the first three parameters are constants and d_i is the sum of a constant parameter and the translational joint displacement. For any given linkage system, the geometry of linkage system can be described by a table listing the values of the four parameters. This table is called the DH table. Tables A.4 through A.8 in the Appendix show the complete DH table for the human model.

By repeated matrix multiplication, a vector in one coordinate system can be transformed to any other coordinate system in the linkage chain. The most common transformation matrices used are those that transform a vector to the base coordinate system, which is referred to as x_0. The base coordinate system is the first coordinate system in the human linkage model. The global coordinate system is a fixed invariant coordinate system outside of the human model. To transform a vector in the x_i coordinate system to the base coordinate system simply pre-multiply by each subsequent transformation matrix for the coordinate systems in that chain, as in Equation (2.3):

$$ {}^{0}\mathbf{v} = \left(\prod_{n=1}^{i} \mathbf{T}^{(n-1,n)} \right) {}^{i}\mathbf{v} \tag{2.3} $$

As shown above, calculating a vector to any point on the body with respect to the base coordinate system is just a matter of matrix multiplication. This is very useful for dynamics simulations. For example, the center of mass of each link can be described as a constant vector with respect to the coordinate system fixed on that link (this is called a local vector). While this vector is constant, the position of the center of gravity with respect to the base coordinate system is changing throughout the motion. At any point in time, the position of the center of mass can be calculated by first calculating the transformation matrices for each coordinate system; then finding the transformation matrices to the base coordinate system; and finally multiplying the local vector to the center of gravity by the transformation matrix.

Inverse Transformation Matrix

Thus far, all the transformations have been from one joint to the previous joint. The DH method can also be used to transform a vector in one coordinate system to the next coordinate system using the inverse of the transformation matrix as shown in Equation (2.4).

$$
\begin{aligned}
{}^{i-1}\mathbf{v} &= \mathbf{T}^{(i-1,i)}\,{}^{i}\mathbf{v} \\
\left[\mathbf{T}^{(i-1,i)}\right]^{-1}\,{}^{i-1}\mathbf{v} &= {}^{i}\mathbf{v} \\
\mathbf{T}^{(i,i-1)}\,{}^{i-1}\mathbf{v} &= {}^{i}\mathbf{v}
\end{aligned}
\tag{2.4}
$$

Normally the inverse of a 4x4 matrix is difficult to find, if one exists. However, since the transformation matrix is homogeneous the inverse is guaranteed to exist and can be found with Equation (2.6) (Fu, Gonzalez, and Lee 1987). Equation (2.6) will be necessary when evaluating the equation of motion, which will be discussed in Chapter 4.

$$
\mathbf{T} = \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{n} & \mathbf{s} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.5}
$$

$$
\mathbf{T}^{-1} = \begin{pmatrix} n_x & n_y & n_z & -\mathbf{n}^T\mathbf{p} \\ s_x & s_y & s_z & -\mathbf{s}^T\mathbf{p} \\ a_x & a_y & a_z & -\mathbf{a}^T\mathbf{p} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.6}
$$

Derivatives of Transformations

The derivatives of transformation matrices are also necessary for evaluating the equation of motion and for calculating gradients in gradient-based optimization. The derivatives are needed with respect to the joint displacements. The joint displacement that locates coordinate system i from the previous coordinate system is q_i; remember this displacement in added to θ_i for a revolute joint and d_i for a prismatic joint. The derivative of a single transformation matrix (i.e. between x_i and x_{i-1}) is calculated by differentiating each entry of the matrix. The derivative of $\mathbf{T}^{(i-1,i)}$ with respect to q_i will be referred to as $d\mathbf{T}_i^{(i-1,i)}$ and is calculated for a revolute joint by Equation (2.7) and for a prismatic joint by Equation (2.8).

$$d\mathbf{T}_i^{(i-1,i)} = \frac{d\mathbf{T}^{(i-1,i)}}{dq_i} = \begin{pmatrix} -\sin\theta_i & -\cos\alpha_i\cos\theta_i & \sin\alpha_i\cos\theta_i & -a_i\sin\theta_i \\ \cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.7}$$

$$d\mathbf{T}_i^{(i-1,i)} = \frac{d\mathbf{T}^{(i-1,i)}}{dq_i} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.8}$$

The second and third derivatives are found similarly. The derivatives of a general transformation matrix can be calculated using the chain rule. If the transformation matrix spans more than one joint, for example $\mathbf{T}^{(0,n)}$, then only the transformation matrix that is a function of q_i is differentiated:

$$d\mathbf{T}_i^{(0,n)} = \frac{\partial\mathbf{T}^{(0,n)}}{\partial q_i} = \mathbf{T}^{(0,1)}\mathbf{T}^{(1,2)}...\mathbf{T}^{(i-2,i-1)}\frac{d\mathbf{T}^{(i-1,i)}}{dq_i}\mathbf{T}^{(i,i+1)}...\mathbf{T}^{(n-1,n)} \tag{2.9}$$

And for the second derivative:

$$d\mathbf{T}_{j,k}^{(0,n)} = \frac{\partial^2\mathbf{T}^{(0,n)}}{\partial q_j\partial q_k} = \mathbf{T}^{(0,1)}\mathbf{T}^{(1,2)}...\frac{d\mathbf{T}^{(j-1,j)}}{dq_j}...\frac{d\mathbf{T}^{(k-1,k)}}{dq_k}...\mathbf{T}^{(n-1,n)} \tag{2.10}$$

For programming purposes it can reduce computation time tremendously by noticing that the commutative property applies for the second derivative, $d\mathbf{T}_{j,k}^{(x,y)} = d\mathbf{T}_{k,j}^{(x,y)}$, and subsequent derivatives.

Global Degrees of Freedom

The DH method is most commonly used in robots where the base coordinate system is fixed to the ground. However, when modeling human motion, the base coordinate system, which is located at Santos'™ hip in Figure 2.2, moves through space as the human moves. To predict movement through space using the mathematical model described thus far, 6 additional global DOFs are necessary. Since global movement has not yet been accomplished using Santos™, these additional DOFs need to be added to the human model. In this section the formulation for global movement of Santos™ is formulated.

As already stated, the DH method is used to describe movement of a linkage system. The joints in a linkage system are constrained to move either along the z axis (for a prismatic joint) or about the z axis (for a revolute joint). In the case of global movement, there are no joints and no constraints. The relationship between the global coordinate system and the base coordinate system could be described with a single transformation matrix that includes 6 DOFs, such as Equation (2.11).

$$T^{(g,0)} = \begin{pmatrix} -\sin\theta_i & -\cos\alpha_i \cos\theta_i & \sin\alpha_i \cos\theta_i & q_0 \\ \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & q_1 \\ 0 & 0 & 0 & q_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.11}$$

However, to find global equilibrium, using the equations of motion developed in Chapter 4, the derivatives of the global transformation matrix with respect to the six global degrees of freedom need to be calculated. As previously discussed, Santos™ only includes revolute joints. This means the formulation for such derivatives already needs to be coded for revolute joints using the DH method. Since this code exists already it is simplest to include the 3 global rotations as 3 additional revolute joints. These are included in the DH table similar to the other joints. It is simplest, however, not to include the 3 translational global DOFs in the DH table.

The three translations could be included in the DH table as prismatic joints. To do so would mean setting up three additional coordinate systems for the prismatic joints with movement along the z-axis of each. However, even though they would be included in the DH table, they would still need to be handled separately from the revolute joints. As explained earlier, for revolute joints the joint displacement, q_i, is added to θ_i in Equation (2.2). For prismatic joints, however, the joint displacement, q_i, is added to d_i in the transformation matrix. The derivatives of the transformation matrices for revolute and prismatic joints are also different.

The three translations could also be included in a non-DH transformation matrix, Equation (2.12). To do so would mean they would still need to be handled separately from the revolute joints because the transformation matrix would calculated differently as would the derivatives of the transformation matrix. The advantage this approach has over prismatic joints is that the transformation matrix is more intuitive than the resultant transformation matrix using the DH table. Finally, the Jacobian, discussed in Chapter 4, is also simplified by using the transformation matrix in Equation (2.12).

$$\mathbf{T}^{(0,1)} = \begin{pmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.12}$$

The purpose of the DH method is to simplify the transformation between two joints in a linkage by reducing the transformation to 4 parameters. The matrix in Equation (2.12) simplifies the three translations a single matrix using only three parameters. If the DH method was used this matrix would be a function of 12 parameters. Using either method yields the same result. Below is an example showing the cumulative transformation matrix from the global coordinate system to the base coordinate system of the human.

$$\mathbf{T}^{(0,1)} = \begin{pmatrix} 1 & 0 & 0 & q_0 \\ 0 & 1 & 0 & q_1 \\ 0 & 0 & 1 & q_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

Equation (2.13) is the transformation matrix not using the DH method; q_0 corresponds to movement along the global x-axis, q_1 is movement along the global y-axis, and q_2 is movement along the global z-axis. Equation (2.14) is the transformation matrix using the

DH method; now q_0 corresponds to movement along the global z-axis, q_1 is movement along the global x-axis, and q_2 is movement along the global y-axis.

$$\mathbf{T}^{(0,1)} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & q_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & q_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & q_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & q_1 \\ 0 & 1 & 0 & q_2 \\ 0 & 0 & 1 & q_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.14)$$

Kinematics

For the swinging simulation, the position and orientation of the hand and feet need to remain constant during the motion. If the joint displacements are known, the DH method can be used to calculate the position and orientation any link in Santos™. It is also necessary to calculate the derivatives of position and orientation with respect to the joint displacements. These derivatives will be used to calculate the gradients of the cost functions and constraints for a gradient-based optimization routine. This section will formulate the equations for position and orientation. It will also develop the equations for velocity and acceleration, which will be used in the future work discussed in Chapter 7.

Calculating Values and Derivatives for Position

To find the global position of a point n on link i pre-multiply the local vector (a local vector is in the x_i coordinate system) pointing to the point, ${}^{i}\mathbf{p}_n$, by the transformation matrix $\mathbf{T}^{(0,i)}$:

$${}^{0}\mathbf{p}_n = \mathbf{T}^{(0,i)}\,{}^{i}\mathbf{p}_n \quad (2.15)$$

The derivative of the position of point n is calculated with respect to each DOF. This means 55 derivatives must be calculated, but many of the degrees of freedom do not affect the position of link i, and those derivatives will be zero. Only the angles that are used to calculate $\mathbf{T}^{(0,i)}$ will result in non-zero derivatives for point n. Also, the current angular velocity and angular acceleration do not affect the current position of point n so the derivative of position with respect to these will also be zero. The derivatives of the position of point n with respect to q_m are:

$$\frac{\partial\,{}^{0}\mathbf{p}_n}{\partial q_m} = d\mathbf{T}_m^{(0,i)}\,{}^{i}\mathbf{p}_n \qquad (\text{ for } \mathrm{m} = 1 \text{ to } 55) \tag{2.16}$$

Calculating Values and Derivatives for Orientation

To find the orientation of the local vector, ${}^{i}\mathbf{p}_n$, with respect to the global coordinate system pre-multiply the local vector by the rotation part of the transformation matrix $\mathbf{T}^{(0,i)}$. As shown in Equation (2.1), the upper left 3x3 matrix within the transformation matrix is the rotation matrix. Instead of extracting the rotation matrix from within the transformation matrix, a simpler method is to modify the extended vector ${}^{i}\mathbf{p}_n$. By changing the 4th entry in the vector from 1 to 0, by Equation (2.17), the translational part of the transformation matrix will not be accounted for and only the rotation matrix will enter into the transformation. Equation (2.18) shows the orientation of the vector ${}^{i}\mathbf{v}$ with respect to the global coordinate system.

$${}^{i}\boldsymbol{\varphi} = {}^{i}\mathbf{v} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.17}$$

$${}^{0}\boldsymbol{\varphi} = \mathbf{T}^{(0,i)}\,{}^{i}\boldsymbol{\varphi} \tag{2.18}$$

The derivative of the orientation of link i is calculated similarly to the position.

$$\frac{\partial\,{}^{0}\boldsymbol{\varphi}}{\partial q_m} = d\mathbf{T}_m^{(0,i)}\,{}^{i}\boldsymbol{\varphi} \qquad (\text{ for } \mathrm{m} = 1 \text{ to } 55) \tag{2.19}$$

Calculating Values and Derivatives for Velocity

To calculate the velocity of point n, differentiate the position with respect to time. First use the chain rule and then use the product rule to differentiate $\boldsymbol{T}^{(0,i)}$ and ${}^{i}\mathbf{p}_n$. Equation (2.20) shows the velocity of point n:

$$
\begin{aligned}
{}^{0}\mathbf{v}_n &= \frac{d\,{}^{0}\mathbf{p}_n}{dt} \\
&= \frac{d}{dt}\left[\mathbf{T}^{(0,i)}\,{}^{i}\mathbf{p}_n\right] \\
&= \sum_{j=1}^{i} \frac{\partial}{\partial q_j}\left[\mathbf{T}^{(0,i)}\,{}^{i}\mathbf{p}_n\right]\frac{dq_j}{dt} \\
&= \left[\sum_{j=1}^{i} d\mathbf{T}_j^{(0,i)}\frac{dq_j}{dt}\right]{}^{i}\mathbf{p}_n
\end{aligned}
\tag{2.20}
$$

The derivatives of velocity vectors are calculated in a manner similar to those of position, except the derivatives with respect to velocity are not all zero. Equation (2.21) shows the derivative with respect to displacement and Equation (2.22) shows the derivative with respect to velocity.

$$
\frac{\partial\,{}^{0}\mathbf{v}_n}{\partial q_m} = \left[\sum_{j=1}^{i} d\mathbf{T}_{j,m}^{(0,i)}\frac{dq_j}{dt}\right]{}^{i}\mathbf{p}_n \qquad (\text{ for } \text{m} = 1 \text{ to } 55) \tag{2.21}
$$

$$
\frac{\partial\,{}^{0}\mathbf{v}_n}{\partial \dot{q}_m} = d\mathbf{T}_m^{(0,i)}\,{}^{i}\mathbf{p}_n \qquad (\text{ for } \text{m} = 1 \text{ to } 55) \tag{2.22}
$$

Calculating Values and Derivatives for Acceleration

Finally, the derivation of acceleration is formulated. Notice that the equation for global acceleration includes the square of angular velocity, which corresponds to centrifugal acceleration. It also includes the product of velocities of different links, which corresponds to Coriolis acceleration.

$$
\begin{aligned}
{}^{0}\mathbf{a}_n &= \frac{d^2\,{}^{0}\mathbf{p}_n}{dt^2} = \frac{d\,{}^{0}\mathbf{v}_n}{dt} \\
&= \frac{d}{dt}\left[\sum_{j=1}^{i} d\mathbf{T}_j^{(0,i)}\frac{dq_j}{dt}\right]{}^{i}\mathbf{p}_n \\
&= \left[\sum_{j=1}^{i}\left[\sum_{k=1}^{i} d\mathbf{T}_{j,k}^{(0,i)}\frac{dq_k}{dt}\frac{dq_j}{dt}\right] + d\mathbf{T}_j^{(0,i)}\frac{d^2q_j}{dt^2}\right]{}^{i}\mathbf{p}_n \\
&= \left[\sum_{j=1}^{i}\left[\sum_{k=j+1}^{i} 2d\mathbf{T}_{j,k}^{(0,i)}\frac{dq_k}{dt}\right]\frac{dq_j}{dt} + d\mathbf{T}_{j,j}^{(0,i)}\left(\frac{dq_j}{dt}\right)^2 + d\mathbf{T}_j^{(0,i)}\frac{d^2q_j}{dt^2}\right]{}^{i}\mathbf{p}_n
\end{aligned}
\tag{2.23}
$$

Note that the summation takes advantage of the fact that the second derivatives of the transformation matrices are commutative. To save time and memory while programming, only slightly more than half the second derivatives of the transformation matrices will be calculated. The derivatives for acceleration include derivatives with respect to displacement, Equation (2.24); velocity, Equation(2.25); and acceleration, Equation(2.26):

$$\frac{\partial\,{}^{0}\mathbf{a}_n}{\partial q_m}=\left[\sum_{j=1}^{i}\left[\sum_{k=j+1}^{i}2d\mathbf{T}_{j,k,m}^{(0,i)}\frac{dq_k}{dt}\right]\frac{dq_j}{dt}+d\mathbf{T}_{j,j,m}^{(0,i)}\left(\frac{dq_j}{dt}\right)^2+d\mathbf{T}_{j,m}^{(0,i)}\frac{d^2q_j}{dt^2}\right]{}^{i}\mathbf{p}_n \tag{2.24}$$

(for m = 1 to 55)

$$\frac{\partial\,{}^{0}\mathbf{a}_n}{\partial \dot{q}_m}=2\left[\sum_{j=1}^{i}d\mathbf{T}_{j,m}^{(0,i)}\frac{dq_j}{dt}\right]{}^{i}\mathbf{p}_n \quad \text{(for m = 1 to 55)} \tag{2.25}$$

$$\frac{\partial\,{}^{0}\mathbf{a}_n}{\partial \ddot{q}_m}=d\mathbf{T}_{m}^{(0,i)}\,{}^{i}\mathbf{p}_n \quad \text{(for m = 1 to 55)} \tag{2.26}$$

Implementation

The human model, the DH method, and the kinematics equations explained in this chapter were implemented in a C code. This section will briefly describe the implementation.

To make the code as general as possible, it was written to be easily updatable for any DH table or joint structure. The DH table itself is input in a matrix, and the link lengths (and other anthropometric data) are read in from a separate input file to allow the user to quickly update the simulation for different anthropometries. This makes the code easily updatable for adding new chains (i.e. fingers and toes) or changing the current joint structure (i.e. adding an additional spine joint).

There are five chains of joints in this human model. For the first chain of joints, the main body and right arm, the joint angle q_i is used to transform the x_i coordinate system to the x_{i-1} coordinate system. For the next chain, however, the indices shift so that the angle q_i is used to transform the x_{i+1} coordinate system to the x_i coordinate system. When a new chain begins there are two coordinate systems fixed to a single link, this dual dependence causes index shifting and complicates the implementation.

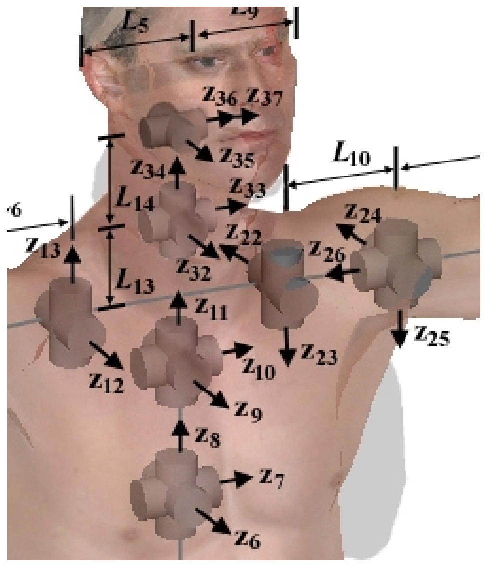

Figure 2.4. A close-up of Santos'™ upper chest showing the branching of his right arm, left arm, and neck.

A good example of this complication is at the last spine joint. The right clavicle, left clavicle, and head all depend on the last joint in the spine as seen in Figure 2.4. All three coordinate systems are fixed to the same link. The last DOF of the spine is q_{12} and it is used in three different transformation matrices: $\mathbf{T}^{(11,12)}$ is for the right clavicle, $\mathbf{T}^{(11,22)}$ is for the left clavicle, and $\mathbf{T}^{(11,32)}$ is for the neck. Equation (2.27) is the next transformation matrix in the left arm chain, $\mathbf{T}^{(22,23)}$, and is a function of q_{22} (not q_{23}). Equation (2.28) is the next transformation matrix in the neck, $\mathbf{T}^{(32,33)}$, and is a function of q_{31}. In short, due to the way Santos™ was originally built, the numbering scheme of the chains is offset by 1 at each branch.

$$\mathbf{T}^{(22,23)} = f(q_{22}, \theta_{23}, \alpha_{23}, a_{23}, d_{23}) \tag{2.27}$$

$$\mathbf{T}^{(32,33)} = f(q_{31}, \theta_{33}, \alpha_{33}, a_{33}, d_{33}) \tag{2.28}$$

For implementation in C, rather than deal with the index shifting, the DH table was modified to drop an index at each branch so that the numbering scheme remains constant. So instead of the first joint in the left arm being q_{22} it is q_{23}, and q_{22} does not exist.

To deal with the order of chains and to make the program easily updatable, a structure was created in C to define each chain's starting joint and link, its ending joint and link, and the chain immediately preceding it. This greatly simplified the code for calculating the DH transformation matrices and helped improve the efficiency of the code for the equation of motion.

CHAPTER 3

B-SPLINE APPROXIMATIONS

The motion of Santos™ is expressed by a set of functions relating his joint displacements to time. These functions are described by B-spline approximations. B-spline approximations will be introduced in this chapter.

A B-spline curve is a piecewise curve that is created from a linear combination of polynomial basis functions called B-splines. Often, B-spline curves are referred to as NURBS (NonUniform Rational B-Splines). Two of the most useful properties of B-spline curves include local control points, and continuity and smoothness. A B-spline curve is completely described by a set of discrete control points or de Boor points. These control points are local, which means that changing one only affects curve segments closest to that point, not the entire curve. The curve is also continuous and smooth depending upon the order of the curve used.

In the human model, B-splines are used to describe the joint angles over time. The control points are used as design variables while optimizing for the best joint displacements to complete the desired motion. Joint displacement versus time should be a smooth, continuous function for all the joints. To ensure this, the joint displacements are represented as functions of time using cubic B-spline curves. Also, a B-spline curve is differentiable, so the angular velocities and accelerations can be easily calculated. In short, the joint displacement profiles require the smoothness and flexibility that a B-spline approximation provides.

B-spline Basis Functions

As previously stated, a B-spline approximation of a function is a linear combination of the basis functions. The coefficients of the basis functions are the control points of the curve. The approximation is defined as:

$$f(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i \tag{3.1}$$

$N_{i,p}(u)$ is the i^{th} basis function of p^{th} degree for u, P_i is the i^{th} control point, and u is a parameter within the range of the knots (the range over which the curve is defined). The curve is of degree p and of order p+1.

Each piecewise curve is defined over an interval; these intervals are defined by a non-decreasing sequence of real numbers called the knot sequence. The knot sequence is written in vector form as $\boldsymbol{U} = \{u_0,\ldots, u_m\}$ and is called the knot vector. The curve made up of two piecewise curves over $[u_i\ u_{i+1}]$ and $[u_{i+1}\ u_{i+2}]$ is continuous, which means that the function exists for all u in the interval $[u_i\ u_{i+2}]$. The curve is also smooth; which means that the derivatives of the function are continuous up to some order in the same interval. Up to p+1 consecutive knots may coincide, but not more.

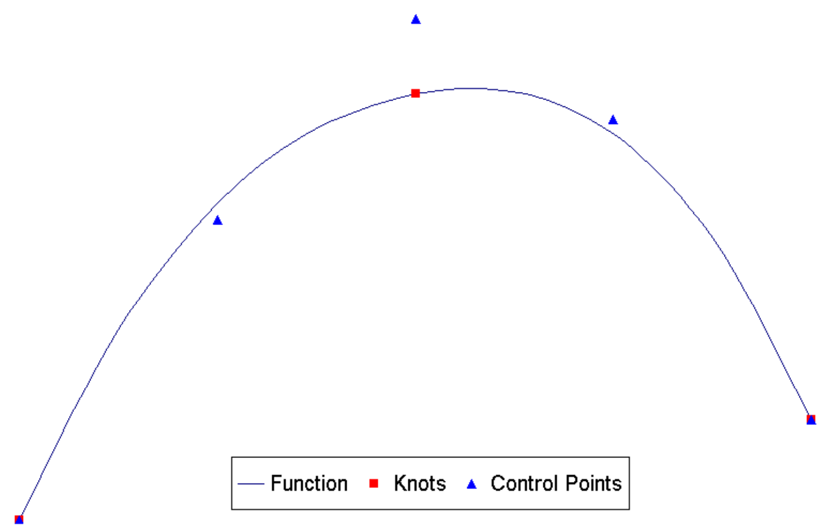

Figure 3.1. A 3rd degree B-spline curve with control points and knots shown.

The number of coincident knots is the multiplicity. If a knot is of multiplicity p+1 then the curve will pass directly through that knot. In the human model, the first and last knots are of multiplicity p+1; this results in the curve passing through the first and last control points. This is called endpoint interpolation. Finally, the degree of the polynomial, p; the number of control points, n; and the number of knots, m, are all related by:

$$m = n + p + 1 \tag{3.2}$$

Figure 3.1 is an example of a 3rd degree B-spline curve. There are 5 control points and 9 knots. Endpoint multiplicity forces the curve to start at the first control point and end at the last control point. Since 4 knots are repeated at the beginning and end of the knot vector, only 3 knots are unique and are shown in Figure 3.1. The entire curve is a composite of two piecewise curves, which meet at the middle knot.

The basis functions form a basis for all piecewise polynomial functions of the desired degree and continuity. There are multiple ways to define the basis functions; in this thesis the recurrence formula will be used because it is easier to implement in a computer program (Piegl and Tiller, 1995). The basis function is given as:

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ N_{i,p}(u) &= \left(\frac{u - u_i}{u_{i+p} - u_i} \right) N_{i,p-1}(u) + \left(\frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} \right) N_{i+1,p-1}(u) \end{aligned} \tag{3.3}$$

The basis span is the number of knot intervals for which a basis function produces a non-zero value. For example, if cubic B-spline curves are used then the basis span is four. This means that a basis function will span four knot intervals; it also means that for the interval $[u_i, u_{i+1}]$ the curve is a function of four nonzero basis functions. However, if endpoint interpolation is used then the first three curve segments are functions of less than four basis functions because of the multiplicity at the endpoints. Over the first interval, $[u_0, u_1]$, the function $N_{0,3}(u)$ is the only non-zero basis function. This also means that the first segment is a function of the first control point only. This reasoning can be used to show that all of the first three curve segments of the approximation are functions of less than four basis functions and control points.

To find a point on a B-spline curve at a fixed u value first find the knot interval in which u lies, then compute the nonzero basis functions using Equation (3.3), and finally apply Equation (3.1).

Derivatives of B-spline Approximations

The parameter u is the independent variable of the function being approximated, Equation (3.1). In the case of joint displacements, u is time. To calculate the velocity and acceleration of point n as formulated earlier, the angular velocity and angular acceleration of each joint is needed. These values are also needed to calculate the equations of motion, which will be developed in Chapter 4.

B-splines are differentiable p times; the $p+1^{\text{th}}$ derivative and higher derivatives are zero. The k^{th} derivative of the curve with respect to u is given in Equation (3.4):

$$f^{(k)}(u) = \sum_{i=0}^{n} N_{i,p}^{(k)}(u) P_i \tag{3.4}$$

The derivatives of the basis functions are given by Equations (3.5) and (3.6):

$$N_{i,p}^{(k)} = \frac{p!}{(p-k)!} \sum_{j=0}^{k} a_{k,j} N_{i+j,p-k} \tag{3.5}$$

$$\begin{aligned} a_{0,0} &= 1 \\ a_{k,0} &= \frac{a_{k-1,0}}{u_{i+p-k+1} - u_i} \\ a_{k,j} &= \frac{a_{k-1,j} - a_{k-1,j-1}}{u_{i+p-k+1} - u_{i+j}} \\ a_{k,k} &= \frac{-a_{k-1,k-1}}{u_{i+p+1} - u_{i+k}} \end{aligned} \tag{3.6}$$

For gradient-based optimization, the derivatives of all the constraints and the objective function with respect to the design variables are required. In Chapter 2, the derivatives of the transformation matrices; of the position, velocity, and acceleration of point n; and the orientation of link i were all formulated with respect to the joint displacements (and time derivatives of joint displacements). However, the joint displacements are not the design variables, the design variables are the control points. So to calculate the derivatives with respect to the control points the chain rule can be used. For example, to find the derivative of position with respect to the control point P_i, the derivative in Equation (2.16) is multiplied by the derivative of q_m with respect to P_i.

The partial derivative of a B-spline approximation with respect to P_i is calculated in Equation (3.7). The partial derivative of a B-spline approximation with respect to u and P_i is calculated in Equation (3.8). Finally, the derivative of a B-spline approximation with respect to P_i and with respect to u, k times, is given in Equation (3.9).

$$\frac{df(u)}{dP_i} = N_{i,p}(u) \tag{3.7}$$

$$\frac{\partial^2 f(u)}{\partial u \partial P_i} = N_{i,p}^{(1)}(u) \tag{3.8}$$

$$\frac{\partial f^{(k)}(u)}{\partial P_i} = N_{i,p}^{(k)}(u) \tag{3.9}$$

Implementation

The joint displacements for both the global DOFs and the joints of the body are functions of time. Each of these functions is approximated by a unique B-spline curve. In addition, the reaction forces at the hands and feet are also functions of time and are approximated using B-splines. The parameter u represents time for the functions, and the knot vector ranges from 0 to the total time required for the motion; the knots are equally spaced. The multiplicity at the start and end enforces endpoint interpolation. This is desired because the initial posture is given and is defined as the first set of control points, which are not optimized. The end posture is approximated using posture prediction and is defined as the last set of control points. Finally, 3rd degree B-splines are used to ensure that acceleration is continuous. The B-spline curve for each joint is calculated as:

$$q_j(t) = \sum_{i=0}^{22} N_{i,3}(t) P_{j,i} \qquad 0 \le t \le t_f \tag{3.10}$$

23 control points are used and 27 knots for both the angle approximations and the reaction force approximations.

CHAPTER 4

INVERSE DYNAMICS

Inverse dynamics involves solving for unknown forces from known displacements, and forward dynamics involves solving for unknown displacements from known forces. With the optimization-based approach used in this thesis, the joint angles of the digital human are found using optimization, and inverse dynamics is then used to calculate the forces necessary for movement. These calculated forces are the forces that must be produced by the muscles of the human in order to perform the simulated movement.

Humans move by actuating muscles, which produce the forces required to move the bones. At each joint in the human musculoskeletal system multiple muscles contribute to a single motion. This means that the exact same movement, for instance completing a bicep curl, can be accomplished multiple times using different muscles. This is muscular redundancy. In this thesis, the muscle forces of the digital human are simplified into generalized torques associated with each joint. This greatly reduces the complexity of the model while still producing accurate results, because it side steps the problem of redundancy in the muscles.

In this chapter, the equation of motion will be formulated in order to compute the generalized joint torque given the joint displacements found with optimization. The generalized torque equals the torque necessary to balance forces due to gravity, inertia, Coriolis and centrifugal acceleration, internal resistance, and interaction with the environment. Virtual work is first used to find the generalized forces associated with the joint stiffness and externally applied forces. Then, the equations of motion for the body, excluding external forces and joint stiffness, are formulated using the Lagrangian formulation. Finally, the equations of motion for the entire system are formulated.

Virtual Work

The principal of virtual work first requires the definition of virtual displacements. A virtual displacement is defined to be an infinitesimal displacement that is consistent with the kinematic constraints imposed on the motion of the system. Time is held fixed during a virtual displacement. Virtual work is then defined to be a force multiplied by a virtual displacement in the direction of the force. The virtual work of an arbitrary force, **F**, which acts at the origin of the x_n frame and a torque, **T**, that acts on body n may be written in terms of the body's virtual displacements $\delta\mathbf{x}$ and virtual rotations $\delta\boldsymbol{\theta}$:

$$
\begin{aligned}
\delta W &= \begin{bmatrix} F_x & F_y & F_z \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} + \begin{bmatrix} T_x & T_y & T_z \end{bmatrix} \begin{bmatrix} \delta\theta_x \\ \delta\theta_y \\ \delta\theta_z \end{bmatrix} \\
\delta W &= \mathbf{F}^T \delta\mathbf{x} + \mathbf{T}^T \delta\boldsymbol{\theta}
\end{aligned}
\tag{4.1}
$$

To ensure that the virtual displacements are consistent with the kinematic constraints of a system, independent generalized coordinates are employed. A set of independent generalized coordinates, also called Lagrangian coordinates, is any set of coordinates that completely specifies the state of the system and satisfies the kinematic constraints of the system. Kinematic constraints impose restrictions on the relative motion between bodies in the system. For example, with a rotational joint the two bodies can only rotate relative to each other so the independent coordinate for the second body is the angle of rotation. Dependent coordinates also describe the state of the system, but do not necessarily satisfy the constraints of the system. If dependent coordinates are used to formulate the equation of motion, then the constraint forces, applied forces, and the equations for the kinematic constraints need to be included in the formulation. By using independent coordinates, it simplifies the problem because only applied forces must be included in the formulation (not the constraint forces). The independent generalized coordinates that are used for this formulation are the joint angles and global degrees of freedom. Finally, it is noted that virtual work is a scalar with units of energy and so the virtual work of a system is conserved regardless of the generalized coordinates used.

The generalized forces of a system are defined as the forces associated with the generalized coordinates. The generalized forces for this model are torques associated with each joint, torques associated with the rotational global degrees of freedom, and forces associated with the translational global degrees of freedom. The virtual work due to joint torque is equal to the torque times a virtual displacement:

$$\delta W_i = \tau_i \delta q_i \tag{4.2}$$

And for the entire system:

$$\delta W = \boldsymbol{\tau}^T \delta \mathbf{q} \tag{4.3}$$

Torque due to Muscle Elasticity

In this section the principal of virtual work will be used to calculate the joint torque necessary to balance joint stiffness due to the muscles. Most muscle models have two main components: contractile elements and elastic elements (Kim, 2004). The contractile elements generate force by contracting, and in this model they are equated to motors producing torque at each joint. The elastic elements contribute to the joint motion in complex ways, and in this model are equated to rotational springs adding stiffness to each joint. This is a very basic model. Damping in the joint is ignored, and the joint stiffness is modeled as being linear with respect to joint displacement. In reality, the joint stiffness varies with joint displacement and tends to be relatively large near the joint limits. Finding accurate values for the joint stiffness coefficients is beyond the scope of this research so approximations are used for the coefficients and are listed in the Appendix in Table A.9 (Kim, 2004).

The stiffness coefficient for joint *i*, K_i, approximates the actual muscle elasticity. The angle of zero displacement for joint *i* is $q_i^{neutral}$; this neutral angle is unique to each joint. The neutral angles of the body are the angles of rest for the joints in the absence of external forces acting on the body. The neutral angles listed in Table A.9 in the Appendix are those for a relaxed person floating in a zero gravity environment (Mount, Whitmore, and Stealey, 2003). Santos^{TM} is shown in this neutral posture in Figure 4.1.

Figure 4.1. Santos^{TM} in the zero gravity neutral position.

The moment associated with the joint stiffness is given in Equation (4.4). The principle of virtual work is then used in Equation (4.5) to find the generalized torque necessary to maintain a non-neutral angle.

$$M_i = K_i(q_i - q_i^{neutral}) \tag{4.4}$$

$$\delta W_i = -K_i\left(q_i - q_i^{neutral}\right)\delta q_i \tag{4.5}$$

Equating Equation (4.2) and Equation (4.5), the torque associated with joint stiffness is:

$$\begin{aligned} \tau_i \delta q_i &= -K_i \left(q_i - q_i^{neutral} \right) \delta q_i \\ \tau_i &= -K_i \left(q_i - q_i^{neutral} \right) \end{aligned} \tag{4.6}$$

Finally, the joint stiffness for each joint is entered into a diagonal square matrix, **K**, and the generalized joint torque is written in vector form as:

$$\boldsymbol{\tau} = -\mathbf{K}\left(\mathbf{q} - \mathbf{q}^{neutral} \right) \tag{4.7}$$

Torque due to External Forces

Next, the torque required to balance an external load will be calculated using the principal of virtual work. In the motion of swinging, the only external loads present are those forces between the hands and feet and the ground. These reaction forces are unknown and so are found with optimization. Once known, the generalized torque associated with them is calculated.

Geometric Jacobian

A Jacobian, in general, is a multidimensional form of a derivative for a system of equations. The first row of the Jacobian consists of the partial derivatives of the first equation with respect to each variable, one partial derivative in each column. The second row of the Jacobian consists of the partial derivatives of the second equation. Analytically the Geometric Jacobian of a linkage system is the derivative of the position of the end-effector with respect to the generalized coordinates. Since time is held fixed this is a map between Cartesian virtual displacements of the end effector and joint virtual displacements:

$$\begin{aligned} \begin{bmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\theta} \end{bmatrix} &= \frac{\partial \mathbf{F}}{\partial \mathbf{q}} \delta \mathbf{q} \\ &= \mathbf{J} \delta \mathbf{q} \end{aligned} \tag{4.8}$$

If the system has n generalized coordinates then the Jacobian is a 6 x n matrix and is a function of the joint displacements at that time instant. Commonly the Jacobian is expressed in terms of velocities. By dividing both sides by the differential time element, the Jacobian becomes a map between the end effector velocities and joint velocities. In Equation (4.9), the Jacobian relates the end-effector linear velocity, $\dot{\mathbf{p}}$, and angular velocity, ω, to the joint velocities, $\dot{\mathbf{q}}$.

$$\begin{gathered}\frac{d}{dt}\begin{bmatrix}\delta\mathbf{x}\\ \delta\boldsymbol{\theta}\end{bmatrix}=\frac{d}{dt}\left(\frac{\partial\mathbf{F}}{\partial\mathbf{q}}\delta\mathbf{q}\right)\\ \begin{bmatrix}\dot{\mathbf{p}}\\ \boldsymbol{\omega}\end{bmatrix}=\mathbf{J}\dot{\mathbf{q}}\end{gathered}\tag{4.9}$$

To implement the Jacobian in code it is much easier to use the following formulation for a revolute joint, which takes advantage of the DH method (Sciavicco and Siciliano, 1996):

$$\mathbf{J}_i=\begin{bmatrix}\mathbf{dT}_i^{0,n}\,{}^n\mathbf{r}_n\\ \mathbf{z}_{i-1}\end{bmatrix}\tag{4.10}$$

$\mathbf{J}_i$ is the ith column of the Jacobian. The orientation of the z-axis of coordinate system x_{i-1} with respect to the global coordinate system is $\mathbf{z}_{i-1}$. The local vector to the point of interest on the end effector is ${}^n\mathbf{r}_n$. The product of a transformation matrix and a position vector is a 4x1 vector, but the extension is dropped and it becomes a 3x1 vector to make $\mathbf{J}_i$ 6x1. For the three global translations, the Jacobian is simply an identity matrix and zeros:

$$\mathbf{J}_{1,2,3}=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\\0&0&0\\0&0&0\\0&0&0\end{pmatrix}\tag{4.11}$$

In addition to finding the velocity of the end effector, the Jacobian is used to find singular configurations, to analyze redundancy, and to map external forces to joint torques. Singularities of the mechanism are locations where the Jacobian becomes a singular matrix; singularities represent configurations at which mobility of the system is reduced. In other words, at a singular configuration it is not possible to impose an arbitrary motion to the end effector. All end effectors have singularities at the boundary of their workspace and most have singularities inside their workspace. If the system is redundant it means that the Jacobian has more columns than rows and infinite solutions exist to Equation (4.9). This is the case with the human model. Finally, the Jacobian can be used to map the external forces applied at the end effector to generalized forces; this will be discussed next.

Generalized Forces due to External Forces

The Jacobian can be used to determine the relationship between the generalized forces applied to the end effector and the generalized forces applied to the joints. For the static case, the generalized torque due to external forces can be calculated by using virtual work. Starting with Equation (4.1), the virtual work of an external force/moment applied to the end effector is the force multiplied by the virtual displacements of the end effector. Let $\mathbf{R}^T = [\mathbf{F}^T \quad \mathbf{T}^T]$:

$$\begin{aligned} \delta W &= \mathbf{F}^T \delta\mathbf{x} + \mathbf{T}^T \delta\boldsymbol{\theta} \\ &= \begin{bmatrix} \mathbf{F}^T & \mathbf{T}^T \end{bmatrix} \begin{bmatrix} \delta\mathbf{x} \\ \delta\boldsymbol{\theta} \end{bmatrix} \\ &= \mathbf{R}^T \begin{bmatrix} \delta\mathbf{x} \\ \delta\boldsymbol{\theta} \end{bmatrix} \end{aligned} \tag{4.12}$$

Then substitute in Equation (4.8) to relate the virtual displacements of the end effector to the joint virtual displacements:

$$\delta W = \mathbf{R}^T \mathbf{J} \delta\mathbf{q} \tag{4.13}$$

Since virtual work must be conserved, the virtual work due to joint torque, Equation (4.3), and the virtual work due to the external forces are equal for all virtual displacements of the generalized coordinates:

$$\boldsymbol{\tau}^T \delta \mathbf{q} = \mathbf{R}^T \mathbf{J} \delta \mathbf{q}$$
$$\boldsymbol{\tau}^T = \mathbf{R}^T \mathbf{J} \tag{4.14}$$
$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{R}$$

The individual joint torque can be calculated by:

$$\tau_i = \mathbf{J}_i^T \mathbf{R} \tag{4.15}$$

This can be extended to the general case when multiple end effectors exist. The Jacobian is different for each end effector. If a force is applied to end effector *m* then the joint torque vector due to the external load is:

$$\boldsymbol{\tau} = \left(\mathbf{J}^m\right)^T \mathbf{R}_m \tag{4.16}$$

Finally, using the principle of superposition the total joint torque due to external forces applied at each end effector is calculated in Equation (4.17).

$$\boldsymbol{\tau} = \sum_{m=1}^{n_{end}} \left(\mathbf{J}^m\right)^T \mathbf{R}_m \tag{4.17}$$

Lagrangian Formulation of the Equation of Motion

In this section the equations of motion for the human will be formulated. These equations will not take into account external forces or joint stiffness. Lagrange's equation is used for the formulation.

The Lagrangian formulation of the equation of motion is an energy balance approach to dynamics as opposed to the traditional Newton-Euler formulation, which is a force balance approach. Like virtual work, the Lagrangian formulation does not require consideration of the constraint forces, only consideration of the scalar energy quantities. The final equation of motion for any system, with the same set of generalized coordinates, will be the same regardless of the formulation used. The Lagrangian formulation begins with a scalar quantity called the Lagrangian. The Lagrangian, *L,* is defined as the total kinetic energy of the system, *K*, minus the total potential energy, *P*:

$$L = K - P \tag{4.18}$$

The equation of motion is then formed by differentiating the Lagrangian. Below is the equation of motion for one joint, the unknown torque, τ_i, is conveniently separated on the right side of the equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right)-\frac{\partial L}{\partial q_i}=\tau_i \tag{4.19}$$

The equation of motion for the system is:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}\right)^T-\left(\frac{\partial L}{\partial \mathbf{q}}\right)^T=\boldsymbol{\tau} \tag{4.20}$$

Potential Energy

The potential energy for each link is simply the dot product of the global location of the center of gravity of the link, ${}^0\boldsymbol{cg}_j$, and the gravity vector, $\mathbf{g}$ (Fu, Gonzalez, and Lee, 1987). The potential energy, P_j, of each link is:

$$\begin{aligned} P_j &= -m_j \mathbf{g}^T\, {}^0\mathbf{cg}_j \\ &= -m_j \mathbf{g}^T \mathbf{T}^{0,j}\, {}^j\mathbf{cg}_j \end{aligned} \tag{4.21}$$

A constant, P_{ref}, may be added to the right hand side of the equation such that the minimum value of P_i is zero. However, since the potential energy is differentiated with respect to $\mathbf{q}$ this constant will not enter into the equation of motion and so is arbitrary. The total potential energy for the system is the sum of the potential energy for all the links:

$$P=\sum_{j=1}^{n_l} P_j = \sum_{j=1}^{n_l} -m_j \mathbf{g}^T \mathbf{T}^{0,j}\, {}^j\mathbf{cg}_j \tag{4.22}$$

Kinetic Energy

Calculating the kinetic energy of a link is more involved. The kinetic energy of a differential mass in link j is (Fu, Gonzalez, and Lee, 1987):

$$dK_j = \frac{1}{2} Trace(\mathbf{v}_j \mathbf{v}_j^T) dm$$
$$= \frac{1}{2} Trace\left[\left(\sum_{p=0}^{j} \dot{q}_p \mathbf{dT}_p^{0,j} \, {}^j\mathbf{r} \right) \left(\sum_{r=1}^{j} \dot{q}_r \mathbf{dT}_r^{0,j} \, {}^j\mathbf{r} \right)^T \right] \text{dm} \tag{4.23}$$
$$= \frac{1}{2} Trace\left[\sum_{p=0}^{j} \sum_{r=0}^{j} \mathbf{dT}_p^{0,j} \, {}^j\mathbf{r} \, {}^j\mathbf{r}^T \left(\mathbf{dT}_r^{0,j} \right)^T \dot{q}_p \dot{q}_r \right] \text{dm}$$

Integrate to obtain the kinetic energy of the entire link:

$$K_j = \int dK_j$$
$$= \frac{1}{2} Trace\left[\sum_{p=0}^{j} \sum_{r=0}^{j} \mathbf{dT}_p^{0,j} \left(\int {}^j\mathbf{r} \, {}^j\mathbf{r}^T dm \right) \left(\mathbf{dT}_r^{0,j} \right)^T \dot{q}_p \dot{q}_r \right] \tag{4.24}$$
$$= \frac{1}{2} \sum_{p=0}^{j} \sum_{r=0}^{j} Trace\left[\mathbf{dT}_p^{0,j} \mathbf{M}_j (\mathbf{dT}_r^{0,j})^T \right] \dot{q}_p \dot{q}_r$$

$$\mathbf{M}_j = \begin{bmatrix} \frac{-I_{xx\,j} + I_{yy\,j} + I_{zz\,j}}{2} & I_{xy\,j} & I_{xz\,j} & m_j \bar{x}_j \\ I_{xy\,j} & \frac{I_{xx\,j} - I_{yy\,j} + I_{zz\,j}}{2} & I_{yz\,j} & m_j \bar{y}_j \\ I_{xz\,j} & I_{yz\,j} & \frac{I_{xx\,j} + I_{yy\,j} - I_{zz\,j}}{2} & m_j \bar{z}_j \\ m_j \bar{x}_j & m_j \bar{y}_j & m_j \bar{z}_j & m_j \end{bmatrix} \tag{4.25}$$

Equation (4.25) shows the mass matrix for link *j*. The mass of link *j* is m_j and $[\bar{x}_j, \bar{y}_j, \bar{z}_j]$ are the components of ${}^j\mathbf{cg}_j$, the local vector to the center of mass of link *j*. The mass moments of inertia and the products of inertia are given as $I_{xx\,j}, \ldots I_{xy\,j}, \ldots$ and are about the local coordinate system. Since the location of the center of mass and the mass moments of inertia are known for each link, the mass distribution is completely characterized; and since $\mathbf{M}_j$ is expressed with respect to the local coordinate frame, it does not change during the motion of the system.

Finally summing the kinetic energy for each link, the total kinetic energy of the system is:

$$K = \sum_{j=1}^{n_l} K_j = \frac{1}{2} \sum_{j=1}^{n_l} \sum_{p=0}^{j} \sum_{r=0}^{j} Trace\left[\mathbf{dT}_p^{0,j} \mathbf{M}_j (\mathbf{dT}_r^{0,j})^T \right] \dot{q}_p \dot{q}_r \tag{4.26}$$

Lagrange's Equation

The kinetic energy and the potential energy are subtracted to obtain the Lagrangian term:

$$L = \frac{1}{2}\sum_{j=1}^{n_l}\sum_{p=0}^{j}\sum_{r=0}^{j} Trace\left[\mathbf{dT}_p^{0,j}\mathbf{M}_j(\mathbf{dT}_r^{0,j})^T\right]\dot{q}_p\dot{q}_r + \sum_{j=1}^{n_l} m_j\mathbf{g}^T\mathbf{T}^{0,j\ j}\mathbf{cg}_j \tag{4.27}$$

Lagrange's equation is formed by differentiating the Lagrangian with respect to the generalized coordinates and time; the final equations of motion are obtained for each joint (Fu, Gonzalez, and Lee, 1987):

$$\begin{aligned}\tau_i = &\sum_{j=i}^{n_l}\sum_{k=0}^{j} Trace\left[\mathbf{dT}_k^{0,j}\mathbf{M}_j(\mathbf{dT}_i^{0,j})^T\right]\ddot{q}_k \\ &+\sum_{j=i}^{n_l}\sum_{k=0}^{j}\sum_{m=0}^{j} Trace\left[\mathbf{dT}_{k,m}^{0,j}\mathbf{M}_j(\mathbf{dT}_i^{0,j})^T\right]\dot{q}_k\dot{q}_m \\ &-\sum_{j=i}^{n_l} m_j\mathbf{g}^T\mathbf{dT}_i^{0,j}\mathbf{cg}_j\end{aligned} \tag{4.28}$$

In matrix form:

$$\boldsymbol{\tau} = \mathbf{D\ddot{q}} + \mathbf{h} + \mathbf{c} \tag{4.29}$$

The inertial acceleration force term is $\mathbf{D\ddot{q}}$, the Coriolis and centrifugal force term is **h**, and the gravity force term is **c**.

Formulating the Equations of Motion for the System

Using the principle of superposition the equations of motion for the entire system is obtained by summing the torque due to muscle elasticity, the torque due to the reaction forces, and the torque due to motion and gravity. As a result, the generalized force associated with joint *i* is:

$$\begin{aligned}\tau_i = &\sum_{j=i}^{n_l}\sum_{k=1}^{j} Trace\left[\mathbf{dT}_k^{0,j}\mathbf{M}_j(\mathbf{dT}_i^{0,j})^T\right]\ddot{q}_k + \sum_{j=i}^{n_l}\sum_{k=1}^{j}\sum_{m=1}^{j} Trace\left[\mathbf{dT}_{k,m}^{0,j}\mathbf{M}_j(\mathbf{dT}_i^{0,j})^T\right]\dot{q}_k\dot{q}_m \\ &-\sum_{j=i}^{n_l} m_j\mathbf{g}^T\mathbf{dT}_i^{0,j}\mathbf{cg}_j + \sum_{m=1}^{n_{end}}(\mathbf{J}_i^m)^T\mathbf{R}_m - K_i\left(q_i - q_i^{neutral}\right)\end{aligned} \tag{4.30}$$

The generalized forces for the human can be written in matrix form as the governing equation of motion for the system:

$$\underbrace{\boldsymbol{\tau}}_{\substack{\text{generalized}\\\text{forces}}} = \underbrace{\mathbf{D}\ddot{\mathbf{q}}}_{\text{inertia term}} + \underbrace{\mathbf{h}}_{\substack{\text{Coriolis \&}\\\text{centrifugal term}}} + \underbrace{\mathbf{c}}_{\text{gravity term}} + \underbrace{\boldsymbol{\tau}_R}_{\text{reaction term}} + \underbrace{\boldsymbol{\tau}_K}_{\text{joint stiffness term}} \tag{4.31}$$

This equation calculates the generalized forces necessary to create the motion of the virtual human. However, to guarantee that the human is in dynamic equilibrium the generalized forces associated with the global degrees of freedom need to be zero. This is because these joints are theoretical and have no muscles available to produce a torque. If they are not equal to zero it is equivalent to some outside agent applying a force on the system.

CHAPTER 5

DYNAMIC POSTURE PREDICTION

As an intermediate step before formulating dynamic motion prediction and as a critical tool for ergonomic design and analysis, a new formulation for dynamic posture prediction, which includes whole-body motion, is presented and coded. Dynamic posture prediction entails predicting a posture with external forces acting on the body. The formulations for posture prediction and motion prediction are similar, and formulating dynamic posture prediction serves as a test case for formulating dynamic motion prediction. In this chapter, the optimization formulation for posture prediction is discussed. Then, this approach is implemented with a preliminary 9 DOF model. Finally, the approach is implemented with Santos^{TM}. Results for each model are discussed.

Optimization Formulation

In this section, the optimization formulation for posture prediction is discussed. First, the design variables are presented, followed by the objective function, and finally the constraints. This formulation is explained in terms of an example problem where Santos^{TM} is standing while holding onto a bar as shown in Figure 5.1. This simulation will find the most realistic posture for him to assume while keeping his hands on the bar and his feet on the ground. The posture is found by solving the following optimization problem:

Find: *Joint displacements, global degrees of freedom, and reaction forces*

To minimize: *Objective function*

Subject to: *Joint limits*

Reaction force limits

Joint torque limits

Global static equilibrium

Hand and foot locations

Hand and foot orientations

Hip location (optional)

Figure 5.1. Santos^{TM} standing flat-footed while holding onto a stationary bar.

Design Variables

The design variables are unknown variables for which the optimization routine is solving. With this problem, the design variables are the joint displacements, the 6 global degrees of freedom, and the reaction forces at the hands and feet. The reason that the reaction forces are included as design variables is because they cannot be calculated by the equation of motion. Postures like that in Figure 5.1 are closed kinematic loops. Each joint has an unknown actuating torque and unknown joint displacement. In addition, there are 6 unknown reaction force components at each end effector. This results in an indeterminate system of equations because there are more unknowns than equations. To solve the system for all the unknowns, the reaction forces and moments at each end effector are used as design variables.

Objective Function

As mentioned earlier, the human system is a redundant system, which means that multiple solutions are possible for a given task and given set of constraints. With optimization, the objective function is responsible for driving the solution towards a realistic one. Four different objective functions are tried with dynamic posture prediction. They are compared for realism, computational speed, and robustness. Each function is described in this section.

Objective Function 1

The first objective function is one typically used with posture prediction. It is a very simple objective, but usually yields reasonably realistic results. The objective function is based on the idea that humans tend to gravitate toward a comfortable, relaxed posture. The function is a measure of how fair from this posture the human has moved.

Figure 5.2. Santos™ standing in his relaxed posture.

This relaxed posture is not the same as the neutral posture used for joint stiffness in Chapter 4. The purpose of the relaxed posture is to drive Santos™ towards a posture that is relaxed and comfortable in a gravity-environment. In this simulation, the relaxed posture is Santos™ standing with feet shoulder width apart and arms at his side, see Figure 5.2. Equation (5.1) is the first objective function; it is the difference between the joint angle and the relaxed joint angle. This difference for each joint is squared before summing all the differences to avoid sign issues. In vector form the dot product of the difference is used.

$$Obj_1 = (\mathbf{q} - \mathbf{q}^{relaxed})^T (\mathbf{q} - \mathbf{q}^{relaxed}) \tag{5.1}$$

Objective Function 2

The second objective function is the sum of the joint torque squared. For gait simulations minimizing joint torque has been shown to produce realistic results (Chevallereau and Aoustin, 2001; Rostami and Bossonnet, 2001). Equation (5.2) is used as the second objective function; it is the sum of the square of torque, or the dot product of the torque vector. Ideally, the sum of the absolute values of the joint torques would yield the lowest joint torque for the system, but using the dot product avoids discontinuities in the objective function, which would occur with using absolute value.

$$Obj_2 = \boldsymbol{\tau}(\mathbf{q},\mathbf{R})^T \boldsymbol{\tau}(\mathbf{q},\mathbf{R}) \tag{5.2}$$

Objective Function 3

The third objective function normalizes joint torque before taking the dot product. By normalizing torque, the third objective function takes into account that some joints naturally support more load, i.e. the shoulder is stronger than the wrist. The torque limits for each joint are used to normalize the torque. The positive and negative torque limits for a joint are not necessarily symmetric (Gonzalez et al., 2002). For instance, for a particular man the maximum amount of torque he can generate by curling his arm (the black arrow in Figure 5.3) is not the same as the maximum amount of torque he can generate by extending his arm in the same position (the white arrow). So for the elbow joint, the limit on maximum positive torque is different than the limit on maximum negative torque. These maximum torque limits have not been well researched, instead data for the maximum strength of individual muscles are much more common. Approximations were used for Santos'™ maximum strength and are listed in Table A.10 in the Appendix.

Figure 5.3. Santos™ performing a bicep curl.

By using the torque limits to normalize the torque, if the joint torque is within its limits then the normalized joint torque will always be between 0 and 1. This normalized torque, however, is discontinuous. The normalized torque is given in Equation (5.3):

$$\tau_i^{norm} = \frac{\tau_i(\mathbf{q},\mathbf{R})}{\tau_i^{limit}} \qquad \text{where } \tau_i^{limit} = \begin{cases} \tau_i^{upper} & \text{for } \tau_i \geq 0 \\ \tau_i^{lower} & \text{for } \tau_i < 0 \end{cases} \tag{5.3}$$

The discontinuity in Equation (5.3) occurs at τ_i equal to zero and is clearly seen in Figure 5.4. The graph in Figure 5.4 is for a joint with a minimum limit of –3 Nm and a maximum limit of 10 Nm. This discontinuity will cause problems with the optimization routine since the derivative at zero is undefined. Mathematically, this can be written as:

$$f'(0)^{-} \neq f'(0)^{+} \qquad \text{where } f = \tau_i^{norm} \tag{5.4}$$

To solve this problem the normalized torque can be squared. When squared, the function is piecewise continuous at zero up to the 1st derivative. This means that the derivative of the curve on the right is equal to the curve on the left:

$$f'(0)^{-} = f'(0)^{+} \qquad \text{where } f = \left(\tau_i^{norm}\right)^2 \tag{5.5}$$

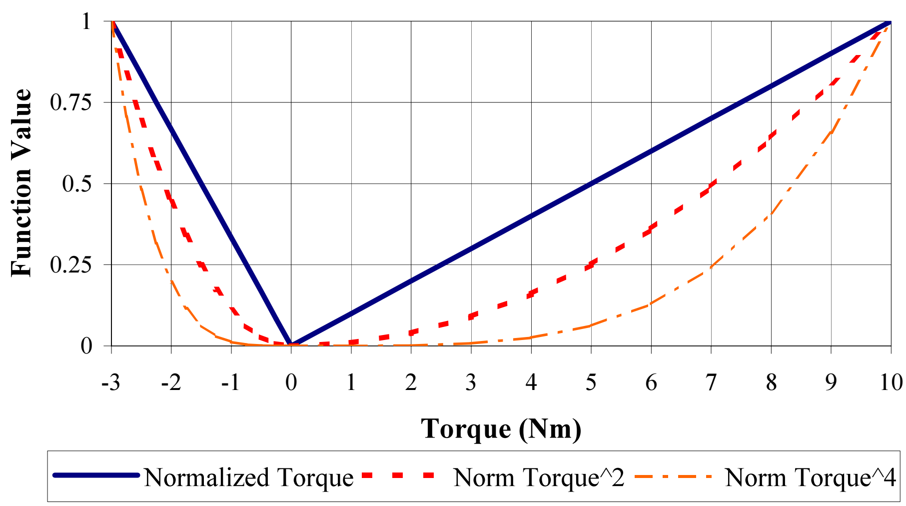

Figure 5.4. A graph of normalized torque, normalized torque squared, and normalized torque raised to the 4th power showing the discontinuity in normalized torque.

The normalized torque must be raised to the 4th power (see Figure 5.4) to ensure that the Hessian (the second derivative) is a smooth function, but this causes problems with non-unique minima. Non-unique minima occur when a function is at its minimum value for different values of the independent variable. In other words, the function is approximately zero for many values of joint torque. In Figure 5.4 this corresponds to the range [-0.5, 2.5] where τ_i^4 is almost zero. So for this optimization routine, the third objective function is the square of the normalized torque and is given in Equation (5.6):

$$Obj_3 = \sum_{i=1}^{n_{DOF}} \left(\tau_i^{norm}(\mathbf{q}, \mathbf{R}) \right)^2 \tag{5.6}$$

Objective Function 4

The final objective function is based on the idea that humans assume postures that do not require a single joint to carry a much larger load than the other joints. This idea was conceived after examining a report that claims humans assume postures that minimize maximum joint discomfort rather than postures that minimize the sum of joint discomfort or average discomfort (Zacher and Bubb, 2004). The report also shows test data indicating that joint discomfort is proportional to joint torque. The goal for the last objective function is to not allow any single joint to tolerate a normalized torque that is much larger than the normalized torque experienced by the other joints. In order to minimize the maximum joint torque, a variation of the global criterion (L_p-norm) is used, based on work by Marler (2001). The consequent objective function is shown as follows:

$$Obj_4 = \left(\sum_{i=1}^{n_{DOF}} \left(\left(\tau_i^{norm}(\mathbf{q}, \mathbf{R}) \right)^2 + 1 \right)^p \right)^{1/p} \tag{5.7}$$

As p approaches infinity Equation (5.7) approximates the min-max formulation or L_∞-norm, with which the largest component of the sum is minimized. For this formulation, p=100. One is added to the normalized torque so that each component of the sum ranges from 1 to 2 instead of from 0 to 1, thus avoiding computational difficulties with large values for p. Including the root in Equation (5.7) typically helps to avoid computational difficulties and theoretically is necessary to approximate the min-max formulation.

Constraints

The constraints of the posture prediction problem include physical limitations of the body (i.e. range of motion and strength), as well as constraints on the type of motion (i.e. maintaining contact with the ground). In this section, the constraints for the optimization routine are discussed.

Joint Displacement Limits

The first two constraints are constraints on the range of motion of the human. For each joint, an upper and a lower displacement limit is enforced to ensure that the digital human does not assume a physically infeasible posture, like hyper-extending his elbow. These theoretical limits approximate the physical limitations that exist due to mechanical constraints of the human skeleton system and due to the soft tissue restricting motion. For instance, the elbow cannot hyper-extend because of the geometry of the joint; however, when curling, the elbow reaches its displacement limit when the forearm collides with the bicep and cannot move further. The joint displacement limits for each joint are listed in Table A.10 in the Appendix. The six global degrees of freedom do not have limits. The joint displacement limits are written as inequality constraints and are listed in Equation (5.8) and Equation (5.9).

$$g_1 = \mathbf{q}^{lower} - \mathbf{q} \leq \boldsymbol{\varepsilon} \tag{5.8}$$

$$g_2 = \mathbf{q} - \mathbf{q}^{upper} \leq \boldsymbol{\varepsilon} \tag{5.9}$$

Reaction Force Limits

The next two constraints are constraints on the magnitude of the reaction forces at the end effectors, which for the swinging motion are the hands and feet. The limits of the reaction forces at the hands and feet depend on the strength of the human and the geometry of the joint between the end effector and the ground. Equations (5.10) and

(5.11) are the constraints imposed on the reaction forces. The limits, $\mathbf{R}^{lower}$ and $\mathbf{R}^{upper}$, are different for each end effector.

$$g_3 = \mathbf{R}^{lower} - \mathbf{R} \le \boldsymbol{\varepsilon} \tag{5.10}$$

$$g_4 = \mathbf{R} - \mathbf{R}^{upper} \le \boldsymbol{\varepsilon} \tag{5.11}$$

The reaction forces at the hands are limited by the geometry of the joint between the hands and the ground. For example, if the digital human is holding onto a bar, as in Figure 5.1, then the connection between the hands and the ground can either be modeled as a pin joint, allowing the hands to spin on the bar, or as a fixed support, allowing a moment between the hands and the bar. In the first scenario, the moment about the bar is zero. In the second scenario, the moment would be limited by the strength of the hand and the friction between the hand and the support. For this simulation, the hands will be considered pin joints, so the moment about the bar will be zero, but there will be moments normal to the bar.

The reaction forces at the hands are also limited by the strength of the hands. For instance, if Santos™ is pulling on the bar in Figure 5.1 then the reaction force is limited by the strength of his fingers. If pushing against the bar the force is not limited at all by the strength of the hand because only the palm of the hand is contacting the bar, Figure 5.5. The reaction force is limited by how hard Santos™ can push, but this limitation will be incorporated in the limits on the torque individual joints can produce. If just the interface between the bar and the hand is examined, the pushing force would be limited by Santos'™ pain threshold or the strength of the bones in the palm. Another important distinction is that the "pushing" and "pulling" directions (see Figure 5.5) are with respect to the hand's local coordinate system and not the global frame.

For this simulation, the reaction forces are given with respect to the global coordinate system. The reaction force limits are also chosen to be symmetric. The grip strengths for Santos™ are taken from Mathiowetz (1985) and are used as the limits on reaction force; his right hand can grip 540 N and his left hand can grip 490 N. Developing precise limits and using local vectors for reaction forces are included in Chapter 7 as future work.

a. Pushing b. Pulling

Figure 5.5. Example of the hand pushing (a) and pulling (b) and the reaction forces associated with the hand.

The reaction forces at the feet are mainly limited by the geometry of the joint between the feet and the floor. The ground can push up on the feet, but the ground cannot pull down on the feet. Similar to the reaction force on the palm, the vertical force that the floor applies to the feet is limited by a human's pain threshold or strength of the bones of the feet. The forces in the other two directions (tangent to the ground) and the moment normal to the ground depend on friction between the foot and the ground. Finally, the two moments tangent to the ground are zero. A moment cannot exist between the bottom of the foot and the ground unless the foot is stuck to the ground or hooked under something. Figure 5.6 shows the allowable reaction forces between the foot and the ground.

One complication to the reaction forces at the feet is the location of the forces. As a human leans forward or leans back the point of application of the reaction forces changes significantly compared to the length of the foot. These details are not taken into account in the model, but are in included as recommendations for future work. For this simulation, the force is applied to the ball of the foot.

Figure 5.6. The reaction forces between the feet and the ground.

Joint Torque Limits

As already discussed in the section on objective functions, each joint has limits on the torque it can produce in the positive and negative directions. The torque limits listed in Table A.11 represent the strength of the human and cannot be exceeded. The torque limits are written as inequality constraints and are the 5th and 6th constraints:

$$g_5 = \boldsymbol{\tau}^{lower} - \boldsymbol{\tau}(\mathbf{q}, \mathbf{R}) \le \boldsymbol{\varepsilon} \tag{5.12}$$

$$g_6 = \boldsymbol{\tau}(\mathbf{q}, \mathbf{R}) - \boldsymbol{\tau}^{upper} \le \boldsymbol{\varepsilon} \tag{5.13}$$

Global Static Equilibrium

For dynamic posture prediction the body must be in static equilibrium. *Dynamic* posture prediction is defined as including external forces acting on the body. *Static* equilibrium refers to the fact that the body is not moving. In short, the resultant force/moment acting on the system must be equal to zero, because the body is stationary.

The reaction forces at the hands and feet must balance the resultant force due to gravity and the joint stiffness.

In the equations of motion developed in Chapter 4, there are six generalized forces associated with the six global degrees of freedom. All the other generalized forces in the model are associated with physical joints of the body and correspond to forces that the muscles would produce to complete a given motion. The global generalized forces, however, are not associated with any physical joint. Instead of corresponding to forces produced by the muscles, these global forces are equal to the six components of the resultant force due to gravity, joint stiffness, and the reaction forces. To be in static equilibrium these generalized forces must equal zero. This is the next constraint, and is the first equality constraint:

$$g_7 = \boldsymbol{\tau}_{global}(\mathbf{q}, \mathbf{R}) = 0 \tag{5.14}$$

While conceptually an equality constraint, g_7 is coded as an inequality constraint using very small limits. This is done to avoid computational difficulties. It is written as:

$$\begin{aligned} g_{7a} &= \boldsymbol{\tau}_{global}(\mathbf{q}, \mathbf{R}) \le \varepsilon \\ g_{7b} &= \boldsymbol{\tau}_{global}(\mathbf{q}, \mathbf{R}) \ge -\varepsilon \end{aligned} \tag{5.15}$$

<u>End Effector Position</u>

The posture predicted in this simulation requires that the locations of the hands and feet of the digital human be fixed. To constrain the location of an end effector, the global location of the end effector is calculated and compared with the desired location of the end effector. Equation (5.16) is the distance between the global location of the end effector, ${}^{g}p_{end}$, and the desired global location of the end effector, ${}^{g}p_{ideal}$. Constraint 8 forces this distance to be zero.

$$g_8 = \left\| {}^{g}\mathbf{p}_{end}(\mathbf{q}) - {}^{g}\mathbf{p}_{ideal} \right\| = 0 \tag{5.16}$$

Again this equality constraint is coded as an inequality constraint:

$$g_{8a} = \left\| {}^{g}\mathbf{p}_{end}(\mathbf{q}) - {}^{g}\mathbf{p}_{ideal} \right\| \le \varepsilon \quad (5.17)$$

Figure 5.7. A hand grasping a bar, the point used for the position constraint is shown.

For the feet, a point on the ball of each foot is constrained. With the hand, however, the constrained point may not be on the hand. For instance, in Figure 5.7 the point that is constrained to remain fixed is not on the hand, but is at the center of the grasp. This allows the hand to rotate about the bar, while keeping the grasp centered on the bar. As the shape of the hand grasp changes the point should change. The location of the point relative to the local coordinate system of the hand is provided to the program.

End Effector Orientation

The location of a point on or near the hands and feet are fixed, but the hands and feet can still rotate about this point. To be realistic, this rotation should be constrained depending upon the geometry of the joint between the end effector and the ground. For instance, if Santos™ is holding onto a bar, Figure 5.7, his hands can rotate about the bar. In terms of vectors this means that the local vector through his grasp should be parallel to the bar. To fix the orientation of an end effector, the global orientation of the end effector is found and compared with the desired orientation. Equation (5.18) is the difference between the end effector orientation, ${}^{g}\varphi_{end}$, and the desired orientation, ${}^{g}\varphi_{ideal}$.

The orientation of the end effector is found using the equations developed in Chapter 2. Constraint 9 forces this distance to be zero:

$$g_9 = \left\| {}^{g}\boldsymbol{\varphi}_{end}(\mathbf{q}) - {}^{g}\boldsymbol{\varphi}_{ideal} \right\| = 0 \tag{5.18}$$

As an inequality constraint it is written as:

$$g_{9a} = \left\| {}^{g}\boldsymbol{\varphi}_{end}(\mathbf{q}) - {}^{g}\boldsymbol{\varphi}_{ideal} \right\| \le \varepsilon \tag{5.19}$$

<u>Distance to Target Point</u>

The last constraint is a constraint on the position of some point on the body other than the end effectors. This constraint is optional, and is used to drive the posture to reach some particular target. For instance, if the simulation is of pull-ups then the point on Santos^{TM} could be his chin, which would be constrained to reach a point above the pull-up bar. For this simulation, the point on the body will always be the hip, the base coordinate system of Santos^{TM}. In the future, this point will be updatable by the user and will change according to the task. Mathematically, this constraint is similar to those constraining the location of the end effectors:

$$g_{10} = \left\| {}^{g}\mathbf{p}_{hip}(\mathbf{q}) - {}^{g}\mathbf{p}_{target} \right\| = 0 \tag{5.20}$$

Finally, as an inequality constraint it is written as:

$$g_{10a} = \left\| {}^{g}\mathbf{p}_{hip}(\mathbf{q}) - {}^{g}\mathbf{p}_{target} \right\| \le \varepsilon \tag{5.21}$$

<u>Implementation</u>

The first linkage model coded was a 9 DOF model. This model was programmed first because it is a much simpler problem to debug and check than the 55 DOF Santos^{TM} model. Once the 9 DOF model was working, then the code was expanded to the 55 DOF human model. In this section, both models and their results will be discussed. The pseudo-code for dynamic posture prediction is briefly described as follows:

1. Read in input: an initial posture for Santos™, a location of the target point, estimated time for the motion, and all anthropometric data.
2. Optimization
    a. Guess at joint displacements and reaction forces.
    b. Calculate generalized forces associated with joints and global DOFs.
    c. Calculate constraints and cost function.
    d. If the current guess satisfies optimality conditions, stop;
       if not, go to step a.
3. Write out output: joint displacements and global degrees of freedom for final posture, joint torques, and reaction forces.

9 Degree of Freedom Model

The motivation for completing a 9 DOF model before attempting the 55 DOF was that the smaller model could be checked in part by hand calculations. This would make debugging both the formulation and the code easier. The DH structure of the 9 DOF model was created to resemble that of the 55 DOF model, so increasing the number of DOF did not result in many additional debugging issues.

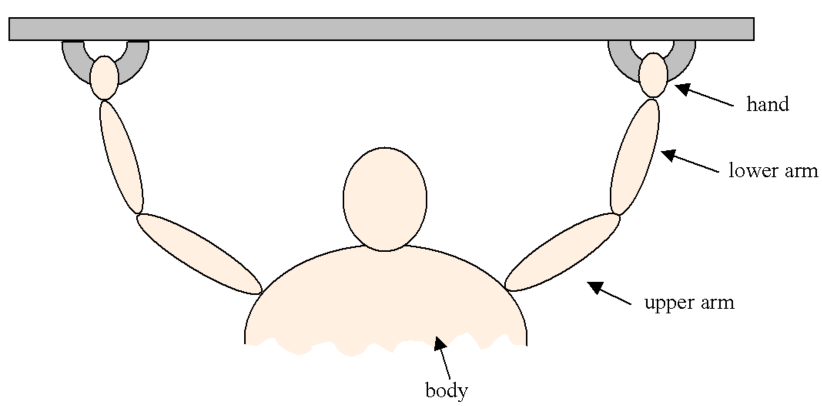

Figure 5.8. A model of the upper body of a human holding onto two supports.

<u>Mathematical Model</u>

In Figure 5.8 the simplified human model is shown. It consists of 7 linkages and 6 joints. This system is planar and only has 3 global DOFs. In the optimization problem the two hands are constrained to position, not orientation.

Table 5.1. The DH Table for the 9 DOF Linkage Model.

Description	Coordinate System	θ	d	α	a
Base	z_1	--	--	--	--
R shoulder	z_2	q_2	0	0	6
R upper arm	z_3	q_3	0	0	8
R lower arm	z_3	q_4	0	0	6
R hand	z_4	q_5	0	0	3
L shoulder	z_5	q_2	0	0	-6
L upper arm	z_6	q_6	0	0	-8
L lower arm	z_7	q_7	0	0	-6
L hand	z_8	q_8	0	0	-3

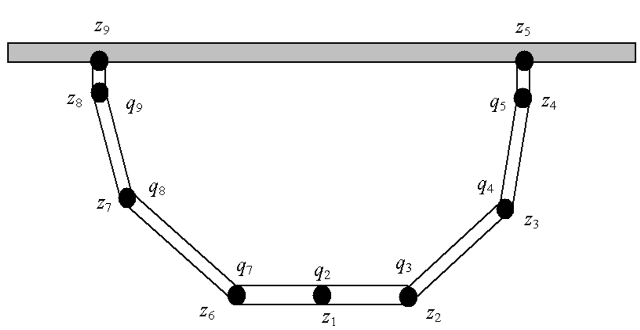

Figure 5.9. The upper body model and ground, shown as a closed-chain linkage system using DH notation.

The model is shown as a 9 bar linkage in Figure 5.9 using DH notation. There are 2 translational DOFs (not shown), which locate z_1 to the global coordinate system and 7 rotational DOFs. The model is set up to have two chains. Table 5.1 shows the DH table for the linkage; q_2 affects the position (and velocity and acceleration) of both the right and left shoulders.

Results

Many postures were found using the 9 DOF model for all four objective functions and for various optimization settings. For this thesis a handful of postures are shown. For all the trials, SNOPT™ Version 6.0 was used for the optimization. SNOPT™ uses a Sequential Quadratic Programming (SQP) method; within SNOPT™, the npOpt interface was used. For these examples, all settings were left at the default values except the "Verify level" and "Derivative level" settings were changed to use finite difference to calculate the gradients. The results are plotted using Mathematica®.

Figure 5.10 shows the initial posture used for all the trials. This initial posture is the initial guess for the optimization routine. It also provides the position of the end effectors to the program. The initial position for z_1 (see Figure 5.9) is at (0, 0, 0), the right end effector is at (0.06, 0.17, 0), and the left end effector is at (-0.06, 0.17, 0). The target for trial 1 is shown as the pink point in Figure 5.10 is located at (0, 0.06, 0). The resulting postures for all four objective functions are also illustrated in Figure 5.10. All constraints are met for each solution. Some of the optimization statistics are listed in Table 5.2 for all of the trials. Note that none of the simulations found the optimum solution, instead the optimization routine exited with the message that the current solution could not be improved.

The target for trial 2 is on the right side and is located at (0.04, 0.06, 0). Figure 5.11 shows the initial posture and target, and the resulting postures for all four objective functions. All four predictions meet the constraints. Figure 5.12 shows trial 3, the target point is moved to the left side and is located at (-0.02, 0.03, 0). All four predictions meet the constraints. Finally, for trial 4 the target is moved outside of the reachable space; it is located at (0.15, 0.06, 0). Figure 5.12 shows the results. The final posture for Obj_1 does not meet the static equilibrium constraint, g_7, or the distance to target constraint, g_{10}. The other three final postures meet all constraints except the distance to target constraint.

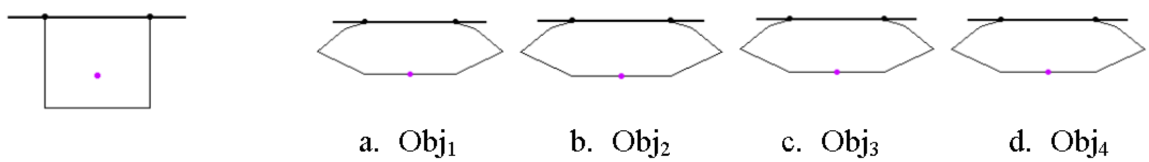

Figure 5.10. Trial 1 posture results for the 9 DOF model; the initial posture and target is shown first, followed by the final posture for each objective function.

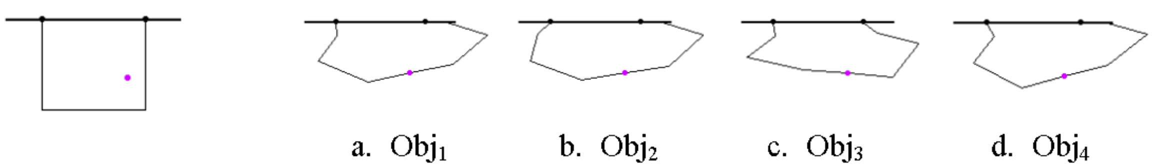

Figure 5.11. Trial 2 posture results for the 9 DOF model.

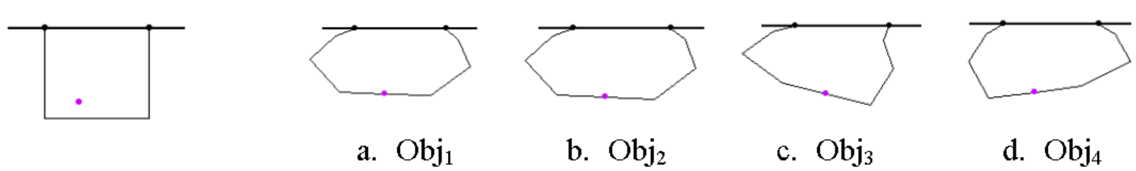

Figure 5.12. Trial 3 posture results for the 9 DOF model.

Figure 5.13. Trial 4 posture results for the 9 DOF model.

Discussion

Dynamic posture prediction for the 9 DOF model encountered no major problems. The only noticeable problem is that the optimal solution is not found, instead the optimization routine runs until the solution can no longer be improved. This could be due to inaccurate gradient information. By adding analytical gradients to the formulation it will improve the performance of the routine and may solve this problem. For some trials, the optimal solution is found by increasing the interval used for the finite difference calculations. However, no single value for this interval produces an optimal solution for all trials and so it was left at its default value. The constraint violations differed for each objective function and trial, see Table 5.2, but all runs meet the constraints for the feasible trials. The number of iterations is on the same order of magnitude for each objective function, but Obj_2 requires the least number of iterations for the feasible problems. Obj_1, however, requires less time to find a solution because the equations of motion are not calculated in the objective function. Calculating the equations of motion slow down the routine considerably.

The 9 DOF model is not a human and is not 3D so it is impossible to judge which posture is the most realistic. However, the behavior of the four objective functions can be compared. Table 5.3 compares the average absolute torque, the maximum absolute torque, the average normalized torque, and the maximum normalized torque for each posture. Un-symmetric joint torque limits are used and the discontinuity in the Hessian for Obj_3 and Obj_4 causes no problems, although none of the joint torques is near zero.

Table 5.2. Optimization Routine Statistics for Posture Prediction using the 9 DOF Model.

	Iterations	Major Iterations	Constraint Violations	Objective Value	Exit
Trial 1					
Obj_1	1647	897	1.1E-08	6.18	Cannot be improved
Obj_2	**560**	**110**	2.2E-08	3.57E+02	Cannot be improved
Obj_3	886	190	2.1E-10	3.20E-01	Cannot be improved
Obj_4	691	260	8.1E-10	1.09	Cannot be improved
Trial 2					
Obj_1	2013	1000	2.8E-04	6.69	Cannot be improved
Obj_2	**843**	**181**	1.7E-07	3.73E+02	Cannot be improved
Obj_3	1835	1000	1.9E-04	2.70	Cannot be improved
Obj_4	1043	466	4.2E-10	1.13	Cannot be improved
Trial 3					
Obj_1	988	175	2.6E-12	4.26	Cannot be improved
Obj_2	**682**	**95**	1.3E-07	1.23E+02	Cannot be improved
Obj_3	1358	476	3.5E-10	1.50E-01	Cannot be improved
Obj_4	906	392	4.4E-10	1.05	Cannot be improved
Trial 4					
Obj_1	979	283	3.2E-02	5.19	Cannot be improved
Obj_2	1007	571	3.2E-02	1.67E+03	Cannot be improved
Obj_3	2643	901	3.2E-02	2.37	Cannot be improved
Obj_4	**822**	**230**	3.2E-02	1.67	Cannot be improved

Table 5.3. Summary of Posture Results for the 9 DOF Model.

	Ave. Absolute Torque	Max. Absolute Torque	at Joint	Ave. Normalized Torque	Max. Normalized Torque	at Joint
<u>Trial 1</u>						
Obj_1	27.00	40.00	q_3	0.954	1.000	q_3
Obj_2	7.27	10.91	q_4	0.333	0.553	q_5
Obj_3	**6.71**	12.69	q_4	**0.195**	0.353	q_4
Obj_4	7.58	**10.20**	q_3	0.264	**0.281**	q_4
<u>Trial 2</u>						
Obj_1	17.00	35.45	q_7	0.636	1.000	q_9
Obj_2	**6.61**	13.62	q_4	0.317	1.000	q_5
Obj_3	17.62	37.75	q_7	0.570	0.993	q_7
Obj_4	8.88	**12.47**	q_3	**0.286**	**0.345**	q_4
<u>Trial 3</u>						
Obj_1	15.91	31.52	q_7	0.568	1.000	q_5
Obj_2	**3.83**	**7.01**	q_8	0.182	0.608	q_9
Obj_3	4.14	10.64	q_8	**0.120**	0.295	q_8
Obj_4	4.67	7.23	q_3	0.152	**0.190**	q_3
<u>Trial 4</u>						
Obj_1	16.99	39.89	q_3	0.68	1.00	q_9
Obj_2	**15.15**	**26.09**	q_4	0.64	1.00	q_9
Obj_3	17.79	36.00	q_4	**0.59**	1.00	q_4
Obj_4	17.24	29.72	q_7	0.65	**0.81**	q_4

In trial 1 and trial 4 the postures predicted using all four objective functions are nearly identical. In trial 2, the posture using Obj_1 is extremely similar to the posture using Obj_4 and close to Obj_2, with only the left elbow in an inverted position. In trial 3, the posture using Obj_1 is very similar to that of Obj_2, but different than the other two results. Overall, all four objective functions result in postures that meet the constraints of the problem and while similar, do vary from one another.

As stated earlier, Obj_1 is a commonly used objective function in kinematic posture prediction programs. The reason for including it in this study is to compare the posture predicted using a non-dynamic objective function with dynamic objective functions. The major difference between Obj_1 and the other objectives is the resulting reaction forces. The total reaction force for each trial and objective function is 726 N in the positive y direction. This is equal to the weight of the linkage. Depending on the objective function, however, the force and moment is distributed to the end effectors differently. For instance, the reaction forces are not included in Obj_1 at all. So for trial 1, for which the final posture of all four objective functions is extremely similar, the reaction forces/moments at each end effector are much higher for Obj_1 than the other functions, see Table 5.4. As a result, the joint torque for Obj_1 is much higher than for the three other objective functions. In summary, while using a kinematic objective function produces similar postures as the dynamic objective functions, the torque results are not similar.

The goal of Obj_2 is to find the optimum posture that minimizes the torque on the joints. Obj_2 has the lowest values for average absolute torque for the last three trials. For the first trial, however, the average absolute torque for Obj_3 is lower. This is surprising because the goal of Obj_2 is to minimize torque. However, the objective function is torque squared, so it does not guarantee that the average absolute torque will be minimized. Ideally, the posture would be optimized for the lowest sum of absolute torque, but as stated earlier using average absolute torque as the objective function would introduce a discontinuity in the objective function and the gradients at certain points would be undefined. In short, Obj_2 is meant to approximate the sum of the joint torque, but this approximation does not always result in the lowest average torque.

The purpose of Obj_3 is to minimize normalized torque. It has the lowest average normalized torque for all the trials, except trial 2. Similar to Obj_2 this objective would ideally be the sum of the normalized torque, but to avoid a discontinuity, the normalized torque is squared. This approximation does not always result in the lowest possible normalized torque.

Table 5.4. Reaction Forces for Trial 1 for the 9 DOF Model.

	Left Hand			Right Hand		
	F_x	F_y	M_z	F_x	F_y	M_z
Obj_1	549	363	30	-549	363	-30
Obj_2	32	363	18	-32	363	-18
Obj_3	-37	363	13	37	363	-12
Obj_4	-50	363	12	50	363	-14

Obj_4 has the lowest values for maximum normalized torque, which is to be expected, because this is the min-max formulation. It also has the lowest values for absolute torque in trials 1 and 2, while Obj_2 is the lowest for trials 3 and 4. In Obj_4 and Obj_3 normalized torque is used in the objective function instead of torque. This results in the stronger joints carrying more load than the weaker ones. The average absolute torque on the shoulders is 29.6 Nm, 7.7 Nm, 18.4 Nm, and 12.8 Nm for objective functions 1, 2, 3, and 4 respectively. The respective values for average absolute torque on the wrists are

9.3 Nm, 6.4 Nm, 2.8 Nm, and 3.6 Nm. The torque on the wrist and shoulder for Obj_2 is about the same even though the shoulder is much stronger than the wrist. For Obj_3 and Obj_4 the torque on the shoulder is greater than the torque on the wrist. So Obj_3 and Obj_4 succeed in distributing the force to the stronger joints.

55 Degree of Freedom Model

In this section, the results for dynamic posture prediction of the 55 DOF model, discussed in Chapter 2, will be demonstrated for several trials. First the results will be presented, followed by a discussion.

Results

SNOPT™ Version 6.0 was again used for the optimization; all settings were left at the default values, except the "Verify level" and "Derivative level" were changed to use finite difference to calculate the gradients and the maximum number of iterations was increased to 100,000 for most runs.

Several postures were found using Santos™ for all the objective functions. Figure 5.14 shows the initial posture for the first trial and the results for each of the four objective functions. The initial hip position is at (0, 0, 0). The target is not shown because it is inside Santos™, located at (0, 0, 0). All constraints are met for each solution. The optimization statistics for all the trials are listed in Table 5.5. The initial posture and results for trial 2 are shown in Figure 5.15. The target is shown by the red ball in Figure 5.15 and is located at (0, -25, 0). All constraints are met for each solution. The initial posture and results for trial 3 are shown in Figure 5.16. The target is located at (0, 0, 20) and is again depicted by the red ball. All constraints are met for each solution. Finally, Figure 5.17 shows the initial posture and results for trial 4. The target is located at (-10, -10, -10) and all constraints are met for each solution.

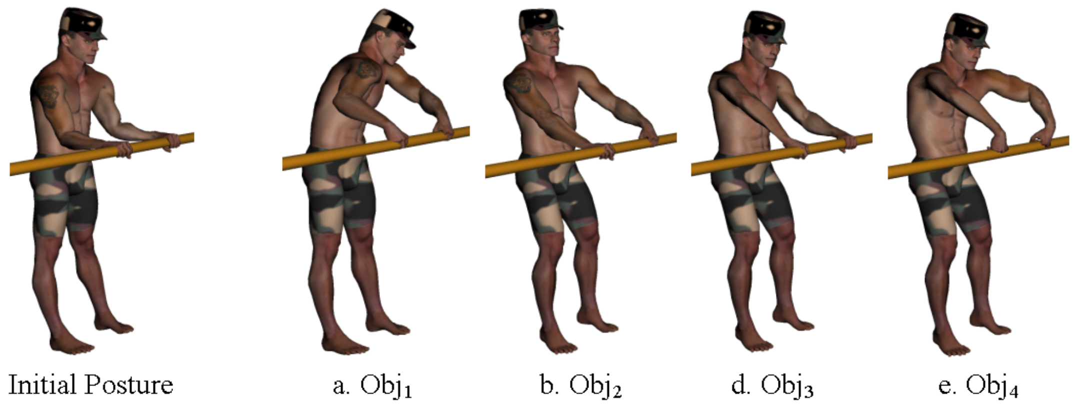

Figure 5.14. Trial 1 results for Santos™.

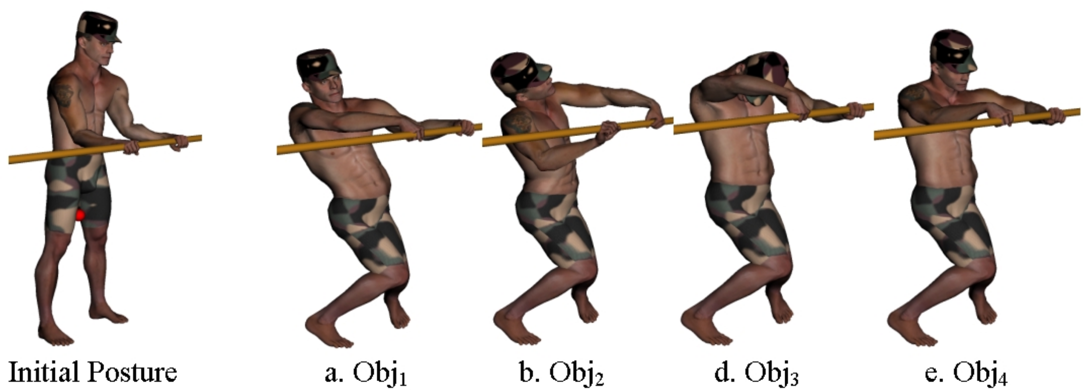

Figure 5.15. Trial 2 results for Santos™.

Discussion

Three major problems surfaced while solving the optimization problem for the Santos™ model. The first problem is the time the program requires to solve the problem. Depending on the CPU speed and the number of processes running on the computer the program took 20 to 150+ hours to find a solution. Plans to modify the program to be more computationally efficient are included in the future work section in Chapter 7. The two improvements that will have the largest impact are adding analytical gradients and combining the objective function and constraint calculations into one sub-routine.

Initial Posture a. Obj_1 b. Obj_2 d. Obj_3 e. Obj_4

Figure 5.16. Trial 3 results for Santos™.

Initial Posture a. Obj_1 b. Obj_2 d. Obj_3 e. Obj_4

Figure 5.17. Trial 4 results for Santos™.

Adding analytical gradients will decrease the amount of time by first eliminating the need to use finite difference to approximate the gradients; finite difference is extremely computationally intensive. Analytical gradients will also decrease the time by producing a better search direction than finite difference for the optimization routine. This will allow the optimization routine to find the optimal solution in less iterations. Combining the objective function and constraints into one sub-routine will decrease the required time by allowing the program to only calculate the equation of motion once per iteration. Calculating the equation of motion is the most time consuming portion of the code. The time required to find a solution is simply a function of the number of iterations and the time per iteration. So by decreasing both the number of iterations required and the time per iteration the total time for the program will be decreased significantly. Obj_1 requires the least number of iterations for most of the trials, see Table 5.5. Obj_2 requires more, but typically less than Obj_3 or Obj_4. Obj_1 also does not require the equation of motion to be calculated and so is much faster than the other objective functions.

The second issue is that the optimization routine cannot solve the problem when the joint torque limits, constraints g_5 and g_6, are included in the formulation. When these limits are included, the optimization routine exits claiming that the constraints cannot be satisfied. Different values for the LU factorization tolerance were tried, but did not improve the performance of the program. Several different methods of scaling the design variables and/or the constraints were also attempted, but did not solve the problem. The joint torque constraints were removed from the program for all of the simulations using Santos™. The reason the optimization routine could not solve the problem with the torque limits included is unclear, because for several trials the routine found a solution that predicted all the joint torques well within their limits. For these same trials the optimization routine would exit claiming the constraints could not be satisfied if the torque limits were included. This seems contradictory.

Using analytical gradients instead of finite difference to approximate the gradients may also solve this issue. Another possible improvement to the formulation is to add the joint torques as design variables and then add equality constraints to ensure that the joint torques and joint displacements satisfy the equations of motion. Theoretically this does not increase the feasible space for the problem, but it will simplify the gradients of the dynamic objective functions and constraints and thus may help the solution process. A final improvement is to examine the anthropometric data in more detail. It may be that the torque limits for Santos™ are too restrictive. Even if the torque limits are not included, by optimizing for normalized torques, it increases the likely hood that the torques will be within their limits even if not constrained. If a torque is not within its limits it will be larger than the other torques when normalized and will dominate the objective function.

A third issue that arose with the Santos™ model is the posture of the head and neck. Several of the final postures predicted Santos'™ head hanging down as in Figure 5.18. This could be a result of the objective function chosen or it could be a result of the optimization routine not successfully finding the optimal solution. Regardless, however, it raises the question of what motivates head posture. Intuitively, head posture is not determined by joint torque. The position of the head during human movement is determined by vision. Humans tend to keep their heads up to enable them to see their environment.

A validation study of the results for dynamic posture prediction has not been completed. For this thesis, validation is by inspection only. As with the 9 DOF model, the behavior of the four objective functions is also compared. Table 5.6 compares the average absolute torque, the maximum absolute torque, the average normalized torque, and the maximum normalized torque for each posture.

Figure 5.18. Dynamic posture prediction resulting in Santos'™ head hanging down.

Table 5.5. Optimization Routine Statistics for Posture Prediction using Santos™.

	Iterations	Major Iterations	Constraint Violations	Objective Value	Exit
Trial 1					
Obj_1	**5237**	**1792**	1.9E-07	2.17E+01	Cannot be improved
Obj_2	10000	3082	8.3E-05	4.79E+03	Iteration limit reached
Obj_3	24441	8172	5.5E-06	7.91E-01	Cannot be improved
Obj_4	12395	7693	3.9E-06	5.42E+01	Cannot be improved
Trial 2					
Obj_1	**3194**	**461**	2.7E-08	3.29E+01	Cannot be improved
Obj_2[1]	23607+	1342+	9.2E-05	7.32E+04	Cannot be improved
Obj_3[1]	43404+	16095+	5.9E-06	5.93	Cannot be improved
Obj_4[1,2]	23221+	4700+	5.6E-06	6.07E+01	Optimal
Trial 3					
Obj_1	**3828**	958	5.6E-05	2.77E+01	Cannot be improved
Obj_2	5609	**948**	4.3E-06	3.87E+03	Cannot be improved
Obj_3	12619	4488	1.3E-05	1.53E+01	Cannot be improved
Obj_4[2]	5072	534	1.4E-05	5.61E+01	Optimal
Trial 4					
Obj_1	4877	1778	3.9E-05	2.16E+01	Cannot be improved
Obj_2	**4756**	**186**	7.9E-05	1.29E+04	Cannot be improved
Obj_3	4943	657	2.1E-05	4.21	Cannot be improved
Obj_4[2]	4514	635	4.2E-06	5.84E+01	Optimal

[1]Runs were restarted after the iteration limit was reached.

[2]Optimality tolerance was increased from 1.0E-6 to 1.0E-3 to decrease computation time.

Table 5.6. Summary of Posture Results for Santos™.

	Ave. Absolute Torque	Max. Absolute Torque	at Joint	Ave. Normalized Torque	Max. Normalized Torque	at Joint
<u>Trial 1</u>						
Obj_1	10.19	53.44	q_{47}	0.174	1.174	q_{49}
Obj_2	4.09	44.48	q_{40}	0.064	0.556	q_{49}
Obj_3	4.12	44.86	q_{40}	0.056	0.561	q_{49}
Obj_4	6.49	49.21	q_{47}	0.082	0.615	q_{49}
<u>Trial 2</u>						
Obj_1	15.13	131.12	q_{39}	0.246	1.639	q_{49}
Obj_2	24.17	148.11	q_{46}	0.424	2.740	q_{49}
Obj_3	11.54	123.01	q_{46}	0.156	1.538	q_{46}
Obj4	12.59	127.50	q_{39}	0.194	1.594	q_{49}
<u>Trial 3</u>						
Obj_1	23.99	115.88	q_2	0.440	3.968	q_{49}
Obj_2	5.00	37.83	q_{47}	0.100	0.473	q_{49}
Obj_3	19.65	94.41	q_2	0.330	2.302	q_{49}
Obj4	12.76	39.88	q_{28}	0.185	0.798	q_{49}
<u>Trial 4</u>						
Obj_1	21.53	82.85	q_{43}	0.297	1.128	q_{49}
Obj_2	9.43	56.87	q_{39}	0.211	1.524	q_{49}
Obj_3	18.69	78.08	q_{43}	0.230	0.781	q_{43}
Obj_4	19.78	82.75	q_{43}	0.241	0.849	q_{49}

Obj_1 produces realistic results, even though it does not consider the joint torques at all. The disadvantage of this objective, however, is that since the reaction forces are not included in the objective function they are not optimized. The reaction forces are still in global equilibrium with the body, but they sometimes result in very unrealistic loading of the joints. Table 5.7 shows the reaction forces for each of the objective functions for the left hand for trial 1. Notice that the reactions for Obj_1 are much higher than the reaction forces for the other objective functions. These higher reaction forces result in higher average torque on the joints.

Table 5.7. Reaction Forces for Santos'™ Left Hand for Trial 1.

	F_x (N)	F_y (N)	F_z (N)	M_x (Nm)	M_y (Nm)	M_z (Nm)
Obj1	549	363	30	-549	363	-30
Obj2	32	363	18	-32	363	-18
Obj3	-37	363	13	37	363	-12
Obj4	-50	363	12	50	363	-14

Overall, Obj_2 produced the most realistic postures, the lowest average torques, and the lowest maximum torques. The postures for Obj_3 and Obj_4 are not as realistic as Obj_2 and Obj_4 takes much longer to solve than the other functions. For some initial conditions, Obj_4 has problems producing a solution even when the joint torque limits are not included. The constraint violations for the dynamic objective functions were all close to one other.

Obj_2 has lower average normalized torque than Obj_3 for two trials and lower maximum normalized torque than Obj_4 for two trials. This is unexpected and may be due to the routine exiting before the optimal solution is found. The optimization routine exits with the message that the current solution cannot be improved. The optimality conditions are not met and so it is uncertain whether the resulting posture is optimal or not. Another explanation is that the global optimal solution was not found. SNOPT™, like all optimization routines that use a SQP method, does not guarantee that the global optimal solution is found, it finds a local optimal solution.

Another interesting observation for all the objective functions is that the postures predicted for trials 1, 2, and 3 are not symmetric about the plane perpendicular to the bar (the Sagittal plane of Santos™). For these trials, the target is located on this plane and so intuitively, the solution is expected to be approximately symmetric. There do exist differences in geometry between the left and right sides of Santos™, which would account for slight differences in the posture of his left and right sides, but not the differences seen in the results. Another cause may be the optimization routine exiting before the optimal solution was found, as just discussed, or the optimization routine finding a local, not global, solution. The best example of this un-symmetric posture is trial 2. Even Obj_2 did not produce a realistic posture for this target.

Conclusions

The posture results for the 9 DOF model show that the formulation for dynamic posture prediction works. The posture results for the 55 DOF model, however, show that there are still some issues to be addressed in the formulation or code. The addition of the 6 global DOF to Santos™ allows him to move his base coordinate system while properly locating his hands and feet in the environment. The joint angles for his entire body are predicted, and the postures are in force equilibrium.

The more complicated 55 DOF Santos™ model is not as stable as the 9 DOF linkage model. Adding analytical gradients may increase the stability and help the optimization routine find an optimal solution while including joint torque limits in the formulation. Gradients may also help the optimization routine find an optimal solution, instead of exiting prematurely.

Once the stability of the code is improved, more research needs to be conducted into what objective function performs the best for dynamic posture prediction. Obj_2 produced the most realistic looking posture, but Obj_3 seemed to distribute the torque to the joints the most realistically. Combining joint torque with other functions should also be researched to improve the posture of the head and neck. Discomfort, vision, or joint displacement would be good candidates for use in combination with one of the joint torque functions.

CHAPTER 6
DYNAMIC MOTION PREDICTION

The swinging motion is defined in Chapter 1 as a motion that consists of a human moving his/her torso while maintaining contact with the ground at either both his/her hands or his/her hands and feet. Dynamic motion prediction entails predicting a swinging motion and the reaction forces at the hands and feet. The formulation of the motion prediction problem is very similar to the formulation presented in Chapter 5 for dynamic posture prediction. Instead of a single posture, however, the optimization routine solves for the time profile of the joint angles, global degrees of freedom, and reaction forces. These time profiles are approximated by B-spline curves. In this chapter, the optimization formulation for motion prediction will be discussed. Then, the results of the 9 DOF model, introduced in Chapter 5, will be shown. Finally, the results for Santos^{TM} will be shown and discussed.

Optimization Formulation

In this section, the optimization formulation for motion prediction is presented. Given an initial posture and a target point, this simulation will find the most realistic motion that enables Santos^{TM} to reach the target while keeping his hands and feet stationary. The first step in the program is to use optimization to find the reaction forces at the initial posture. This is accomplished by using the posture prediction code formulated in Chapter 5, but only optimizing the reaction forces, not the joint angles. Next, the posture prediction program is used to find the end posture. For both of these preliminary optimization problems any of the objective functions discussed in Chapter 5 could be used. Finally, another optimization routine is used to calculate the motion from the initial posture to the end posture. The motion is found by solving the following optimization problem:

Find: *Control points for curves approximating Joint displacements, global degrees of freedom, and reaction forces*

To minimize: *Objective function*

Subject to: *Joint limits*

Reaction force limits

Joint torque limits

Global dynamic equilibrium

Hand and foot locations

Hand and foot orientations

Initial and final velocities

Initial and final accelerations

First the design variables will be discussed, followed by the objective functions, and finally the constraints.

Design Variables

For motion prediction, the design variables are slightly different than for posture prediction. Instead of the joint angles at a single instant in time, the optimization routine finds the function of angle versus time for each joint. B-spline curves, discussed in Chapter 3, are used to approximate these functions. The control points of the B-spline curves become the design variables. As discussed in Chapter 3, given a knot vector and the order of the curve, the control points will completely describe the function, including the derivatives. Since the entire curve is optimized, this also guarantees that the joint displacement functions are continuous and smooth, which is necessary to produce smooth motion. The reaction forces and global degrees of freedom are also represented by B-spline approximations.

Since the initial and final postures are found using posture prediction, the initial and final control points are given. By using endpoint interpolation, the first and last control points for a joint angle curve are equal to the first and last angles of the joint. So these control points are not optimized. The same is true for the control points describing reaction forces and global degrees of freedom. For the equations used in this chapter, *nc* is the number of control points, $\mathbf{P}_q$ are the control points for joint displacement and global degrees of freedom, and $\mathbf{P}_R$ are the control points for reaction forces.

Objective Functions

The first four objective functions used for motion prediction are similar to those used for posture prediction in Chapter 5. In addition, there is a fifth objective function added for motion prediction that will also be discussed in this section.

Objective Function 1

The first objective function for motion prediction is the sum of joint displacement over time. Joint displacement from the relaxed posture is used because it is a very simple objective to code, because joint displacement produced realistic postures in Chapter 5, and because joint displacement was faster than the other objective functions in Chapter 5. Using a kinematic objective function in motion prediction does, however, disregard the affect of dynamics. The function measures the difference between the current posture and the relaxed posture, shown in Figure 5.2, and tries to minimize this difference. Equation (6.1) is the objective function. The difference between the joint angle and the relaxed angle is squared to avoid sign issues.

$$Obj_1 = \sum_{t=0}^{t_{end}} \left(\mathbf{q}(\mathbf{P}_q, t) - \mathbf{q}^{relaxed}\right)^T \left(\mathbf{q}(\mathbf{P}_q, t) - \mathbf{q}^{relaxed}\right) \tag{6.1}$$

Ideally, this objective function would be the integral of the joint displacement over time. Instead, the angle differences are found at certain time steps throughout the motion and these values are summed. It approximates an integral.

Objective Function 2

The second objective function used with motion prediction is very similar to Obj_2 in Chapter 5. It is the sum of joint torque squared over time. Equation (6.2) is the objective function.

$$Obj_2 = \sum_{t=0}^{t_{end}} \boldsymbol{\tau}(\mathbf{P}_q, \mathbf{P}_R, t)^T \boldsymbol{\tau}(\mathbf{P}_q, \mathbf{P}_R, t) \tag{6.2}$$

Objective Function 3

The third objective function normalizes joint torque before taking the dot product, similar to the corresponding objective function in Chapter 5. By normalizing torque, the third objective function takes into account that some joints naturally support more load. The torque limits for each joint are used to normalize the torque. Equation (6.3) is the objective function.

$$Obj_3 = \sum_{t=0}^{t_{end}} \sum_{i=1}^{n_{DOF}} \left(\tau_i^{norm}(\mathbf{P}_q, \mathbf{P}_R, t) \right)^2 \tag{6.3}$$

Objective Function 4

The fourth objective function used with motion prediction is similar to Obj_4 in Chapter 5. It is the min-max formulation for normalized joint torque over the entire motion. This assumes that the largest normalized torque experienced during the motion governs the motion. The optimization routine will try to minimize this maximum normalized torque. Equation (6.4) is the objective function. It is similar to Obj_4 in Chapter 5, however, now the summation is over all the degrees of freedom and over time so that the maximum torque at any one time step is minimized. The value of p for motion prediction is the same as that for posture prediction, p=100.

$$Obj_6 = \left(\sum_{t=0}^{t_{end}} \sum_{i=1}^{n_{DOF}} \left(\left(\tau_i^{norm}(\mathbf{P}_q, \mathbf{P}_R, t) \right)^2 + 1 \right)^p \right)^{1/p} \tag{6.4}$$

Objective Function 5

The last objective function used for motion prediction is a constant. This objective function does not affect the motion or the reaction forces; it is used to find the closest feasible solution to the initial guess. Sequential Quadratic Programming (SQP) uses the Lagrange Function to move toward a feasible solution while minimizing the objective. If a constant is provided as the objective, the Lagrange Function will consist only of the constraints and will provide a search direction towards the feasible space. There are existing methods specifically designed to find feasible solutions for a given problem. However, conducting research into these methods is beyond the scope of this work.

The initial guess for the motion is a linear interpolation between the initial posture and the final posture. The last objective function simply takes this initial guess and finds the closest solution that satisfies the constraints. This approach assumes that humans move in a direct and smooth way to reach a final posture and that the joint torque at this final posture governs the movement more than the joint torque at intermediate postures. The last objective function is a constant and is given by Equation (6.5).

$$Obj_5 = c \tag{6.5}$$

Since the design variables are not included in the objective function it could produce very undesirable and unpredictable results. However, it will produce a feasible solution. This function is included because of the instability of the Santos^{TM} model. By using a constant, it simplifies the formulation and allows a feasible solution to be found.

Constraints

Most of the constraints for posture prediction also apply to the motion prediction formulation. Instead of being calculated for one posture, however, they are calculated for each time step. The number of time steps calculated for a motion is not necessarily related to the number of control points or knots used for the B-spline approximations, it is simply the number of frames at which the constraints are enforced. For example, if a motion takes 4 seconds to complete and 5 time steps are desired, then the initial posture occurs at 0 seconds and the final posture occurs at 4 seconds. In-between the initial and final posture the constraints on the motion are checked at 1 second, 2 seconds, and 3 seconds. This section will briefly revisit the constraints discussed in Chapter 5 and introduce four new constraints: initial joint velocity, final joint velocity, initial joint acceleration, and final joint acceleration.

Joint Angle Limits

The first constraint is on the control points describing the joint angles and is very similar to the first constraint described in Chapter 5. The constraint limits the control points for joint i, $P_{i,j}$, to be within the joint angle limits for joint i:

$$g_1 = q_i^{lower} - P_{i,j} \le \varepsilon \qquad \text{for j = 3, nc-3} \tag{6.6}$$

$$g_2 = P_{i,j} - q_i^{upper} \le \varepsilon \qquad \text{for j = 3, nc-3} \tag{6.7}$$

The reason the first three and last three control points are not included in the constraint is explained later in this section. The limits are the same as those used for constraining the joints in posture prediction, represented by q_i^{lower} and q_i^{upper} respectively. However, using these limits for constraining the control points is an approximation because the control points do not equal the joint angles.

Some experimentation was done to constrain the joint values rather than the control points. If a particular joint angle is at its limit, as in Figure 6.1, then the control point could be slightly outside of the joint angle limit. So by constraining the joint angle instead of the control point, the joints are allowed to utilize their full mobility. However, the difference in the available range of motion between constraining the control points versus constraining joint angles is very small. So in reality, constraining the control points does not reduce the joint range significantly and it costs time since the constraints are applied to the joints at each time step rather than limits on the design variables. Also, constraining the joint angles only affects the joint profile at peaks, such as the one shown in Figure 6.1, and so for the majority of the motions it has no effect. Finally, it is important to remember that the chosen joint limits are only approximate and so a difference of this magnitude could easily be lost in the accuracy of the selected limits.

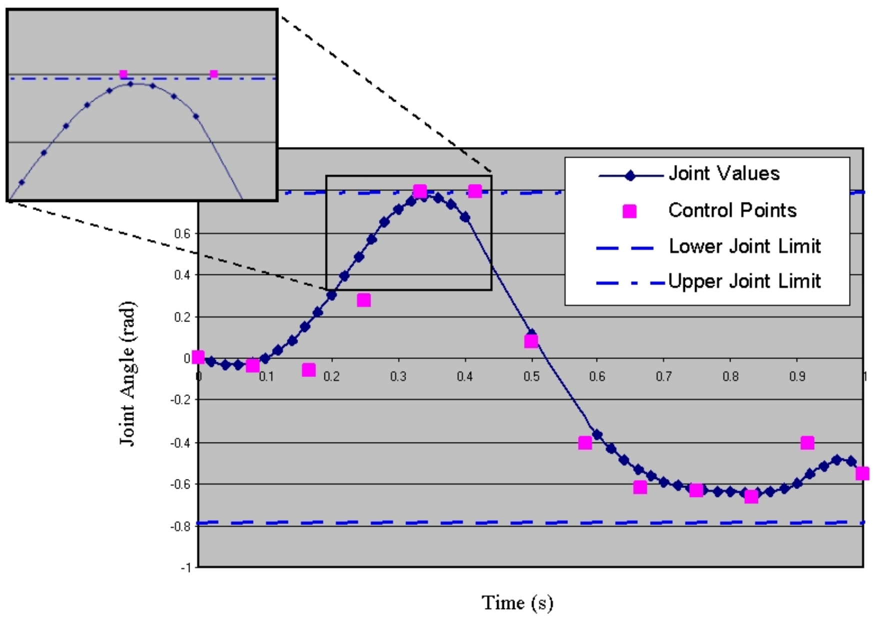

Figure 6.1. A joint profile for the shoulder, using constraints on joint angle rather than control points.

Reaction Force Limits

The reaction force limits, Equation (6.8) and Equation (6.9), are the same as those for posture prediction. In motion prediction they are applied at each time step. The same rational used for restricting the control points describing the joint angles is used for reaction forces. This means that the control points are constrained and not the reaction forces.

$$g_3 = R_m^{lower} - P_{m,j} \le \varepsilon \qquad \text{for j = 3, nc-3} \tag{6.8}$$

$$g_4 = P_{m,j} - R_m^{upper} \le \varepsilon \qquad \text{for j = 3, nc-3} \tag{6.9}$$

Joint Torque Limits

The joint torque limits are also the same as those for posture prediction. In motion prediction, however, the motion of the body also contributes to joint torque. In other words, the Coriolis and centrifugal term and the inertia term in Equation (4.31) are no longer zero as they were for posture prediction. This complicates the equation of motion and increases the computation time for the program tremendously. Constraints 5 and 6 are given in Equation (6.10) and Equation (6.11).

$$g_5 = \boldsymbol{\tau}^{lower} - \boldsymbol{\tau}(\mathbf{P}_q, \mathbf{P}_R, t) \le \varepsilon \qquad \text{for t = 0, t}_{\text{end}} \tag{6.10}$$

$$g_6 = \boldsymbol{\tau}(\mathbf{P}_q, \mathbf{P}_R, t) - \boldsymbol{\tau}^{upper} \le \varepsilon \qquad \text{for t = 0, t}_{\text{end}} \tag{6.11}$$

Global Dynamic Equilibrium

The motion of the body must obey the equations of motion. This means the resultant force and moment of the body must be zero. As with the joint torques, now the motion of the entire body plans a role in the force balance making it a dynamic equilibrium and no longer static.

$$g_7 = \boldsymbol{\tau}_{global}(\mathbf{P}_q, \mathbf{P}_R, t) = 0 \qquad \text{for t = 0, t}_{\text{end}} \tag{6.12}$$

Also, similar to postures prediction, this equality constraint is implemented as two inequality constraints. This decreases the accuracy of the solution, but facilitates the solution process. The two inequalities are written as:

$$g_{7a} = \boldsymbol{\tau}_{global}(\mathbf{P}_q, \mathbf{P}_R, t) \le \varepsilon \qquad \text{for t = 0, t}_{\text{end}} \tag{6.13}$$

$$g_{7b} = \boldsymbol{\tau}_{global}(\mathbf{P}_q, \mathbf{P}_R, t) \ge -\varepsilon \qquad \text{for t = 0, t}_{\text{end}} \tag{6.14}$$

<u>End Effector Position and Orientation</u>

The four end effectors must maintain their positions and orientations throughout the motion. Ideally these constraints are equalities, Equations (6.15) and (6.17), but for implementation they are written as inequality constraints, Equations (6.16) and (6.18).

$$g_8 = \left\| {}^g\mathbf{p}_{end}(\mathbf{P}_q, t) - {}^g\mathbf{p}_{ideal} \right\| = 0 \qquad \text{for t = 0, t}_{\text{end}} \tag{6.15}$$

$$g_{8a} = \left\| {}^g\mathbf{p}_{end}(\mathbf{P}_q, t) - {}^g\mathbf{p}_{ideal} \right\| \le \varepsilon \qquad \text{for t = 0, t}_{\text{end}} \tag{6.16}$$

$$g_9 = \left\| {}^g\boldsymbol{\varphi}_{end}(\mathbf{P}_q, t) - {}^g\boldsymbol{\varphi}_{ideal} \right\| = 0 \qquad \text{for t = 0, t}_{\text{end}} \tag{6.17}$$

$$g_{9a} = \left\| {}^g\boldsymbol{\varphi}_{end}(\mathbf{P}_q, t) - {}^g\boldsymbol{\varphi}_{ideal} \right\| \le \varepsilon \qquad \text{for t = 0, t}_{\text{end}} \tag{6.18}$$

<u>Initial and Final Motion</u>

This simulation is one complete motion and so the body should begin at rest and end at rest. In the future, this program will be used is to simulate one stage of a multi-stage motion, like climbing a ladder. In that situation, the initial motion of the body may be prescribed to match the final motion of the previous stage. The end motion may not be constrained or may have defined goals for velocity and acceleration. For now, however, the initial and final velocities of the joints and the initial and final accelerations of the joints are constrained to zero:

$$g_7 = \dot{\mathbf{q}}_{initial}(\mathbf{P}_q, t) = 0 \qquad \text{for t = 0} \tag{6.19}$$

$$g_8 = \dot{\mathbf{q}}_{final}(\mathbf{P}_q, t) = 0 \qquad \text{for t = t}_{\text{end}} \tag{6.20}$$

$$g_9 = \ddot{\mathbf{q}}_{initial}(\mathbf{P}_q, t) = 0 \qquad \text{for t = 0} \tag{6.21}$$

$$g_{10} = \ddot{\mathbf{q}}_{final}(\mathbf{P}_q, t) = 0 \qquad \text{for t = t}_{\text{end}} \tag{6.22}$$

Within the code, these constraints are implemented outside of the optimization routine. Since the functions describing the joint angles are represented by B-spline approximations, the first three control points and the last three control points for each joint and the global degrees of freedom are set equal to the initial and final posture respectively. This ensures that the initial and final velocity and acceleration of the joints are zero without adding more constraints to the optimization routine. It does, however, decrease the degrees of freedom of the optimization problem by decreasing the number of design variables because these six control points are then not included as design variables for the problem. This change in the formulation was made after initial difficulties were encountered while running the 9 DOF simulation.

Implementation

The pseudo-code for dynamic motion prediction is briefly described as follows:

1. Read in input: an initial posture for Santos^{TM}, a location of the target point, estimated time for the motion, and all anthropometric data.
2. Optimization of reaction forces at initial posture
    a. Guess at reaction forces.
    b. Calculate generalized forces associated with joints and global DOFs.
    c. Calculate constraints and cost function.
    d. If the current guess satisfies optimality conditions, stop;
       if not, go to step a.
3. Optimization of final posture and reaction forces
    a. Guess at joint displacements and reaction forces.
    b. Calculate generalized forces associated with joints and global DOFs.
    c. Calculate constraints and cost function.
    d. If the current guess satisfies optimality conditions, stop;
       if not, go to step a.
4. Optimization of motion

a. Guess at control points for joint displacements and reaction forces by using a linear interpolation between the initial and final solutions.
b. Calculate generalized forces associated with joints and global DOFs at each time step.
c. Calculate constraints and cost function.
d. If the current guess satisfies optimality conditions, stop; if not, go to step a.

5. Write out output: joint displacements and global degrees of freedom for final posture, joint torques, and reaction forces.

In this section, the results of the 9 DOF model will be examined, followed by the results of the 55 DOF Santos™ model. Many trials were completed with the 9 DOF model, but due to computational difficulties only two trials were completed using Santos™.

9 Degree of Freedom Model

Results

Eight example trials are shown using the 9 DOF model. SNOPT™ Version 6.0 was used for the optimization; all settings were left at the default values except the "Verify level" and "Derivative level" settings were changed to use finite difference to calculate the gradients. The results are plotted using Mathematica®.

For the first 6 trials, 12 control points and 16 knots were used for the 3rd degree B-spline approximations of the motion and reaction forces. Since the initial three and final three control points are used to constrain the initial and final accelerations, only 6 control points are optimized. The objective function used for the initial and final postures in this formulation is the min-max formulation for normalized torque (Obj_4 in Chapter 5). Since the posture of the 9 DOF model did not vary greatly between the different objective functions, this objective function was chosen arbitrarily and is used for all the 9 DOF model trials, regardless of the objective function used for the motion optimization. This ensures that the initial and final postures are the same for all trials and so the difference in motion is only due to the objective function used for the motion optimization routine. In this chapter, only the motion between two postures is studied so the initial and final postures should be the same for all objective functions.

Each trial was completed for all 5 objective functions. Figure 6.2 shows the motions for the first trial. The red point is the initial starting point and the blue point is the target. For trial 1, the starting point is located at (0, 0, 0) and the target is located at (0.12, 0.08, 0). To aid in visualizing the movement, the linkage is drawn darker as it nears the target. The total time allowed for the motion is 30 seconds and the constraints are checked at two points during the motion. In all, there are four time steps: the initial posture, two postures during the motion, and the final posture. Figure 6.5 through 6.9 show the normalized torque at each joint. A best-fit line is drawn through the four data points to aid in visualization. In Figure 6.6, Figure 6.7, and Figure 6.8 the normalized torque appears to drop below zero between the 10 and 20 seconds. This is a result of the best-fit line; mathematically, it is impossible for the normalized torque to fall below zero.

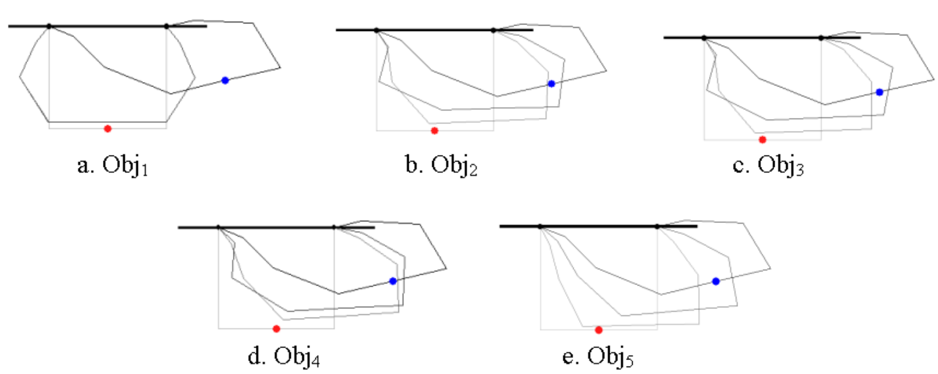

Figure 6.2. Motion results for trial 1, the red point is the starting point and the blue point is the target.

The second motion has the same hand positions as trial 1, but the starting point is located at (-0.02, 0.03, 0) and the target is located at (0.04, 0.06, 0). The total time allowed for the motion is 30 seconds and the constraints are checked at two points during the motion. Figure 6.3 shows the resulting motion for all the objective functions and Figures 6.10 through 6.14 show the normalized torque.

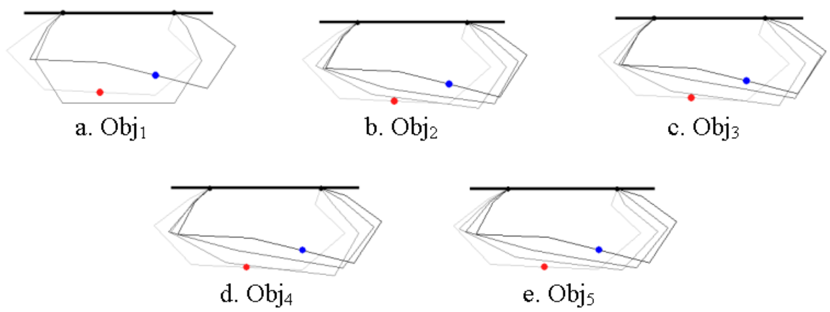

Figure 6.3. Motion results for trial 2.

The starting position for trial 3 is (-0.1, 0.06, 0) and the target position is (0.1, 0.06, 0). The time allowed for the motion is 30 seconds and the constraints are checked at two points during each motion. Figure 6.4 shows the motion for trial 3 and the normalized joint torques are shown in Figures 6.15 through 6.19.

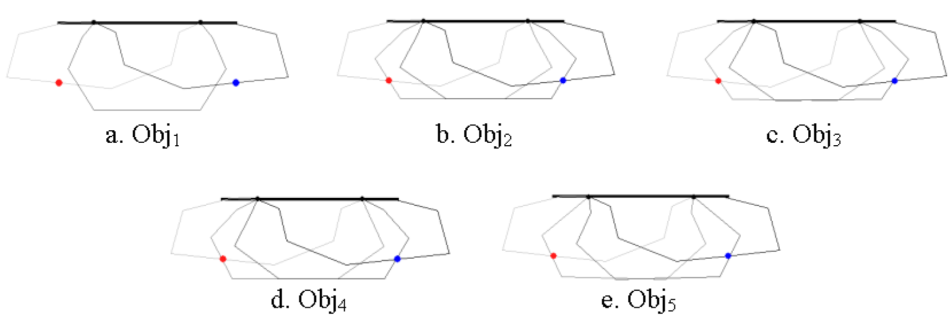

Figure 6.4. Motion results for trial 3.

The fourth, fifth, and sixth trial motions share the same starting point and target as trial 3. The time allowed for the motion is different for the simulations, but the constraints are still enforced at two points during each motion. For the third trial 30 seconds are allowed for the motion, for the fourth trial 5 seconds are allowed, for the fifth trial 100 seconds are allowed, and finally for the sixth trial 300 seconds are allowed. Figure 6.20 shows the motion for trials 3, 4, 5, and 6 for Obj_4. Since Obj_1 and Obj_5 do not deal with dynamics, changing the time allowed for the motion does not change the resulting motion, so these results are not shown. The differences between the results for Obj_2 and Obj_3 and the results for Obj_4 are imperceptible to the eye and so these results are not shown. The normalized joint torque for joint q_3 is shown for each trial in Figure 6.21. The joint q_3 was chosen as an example, the other joints show similar trends.

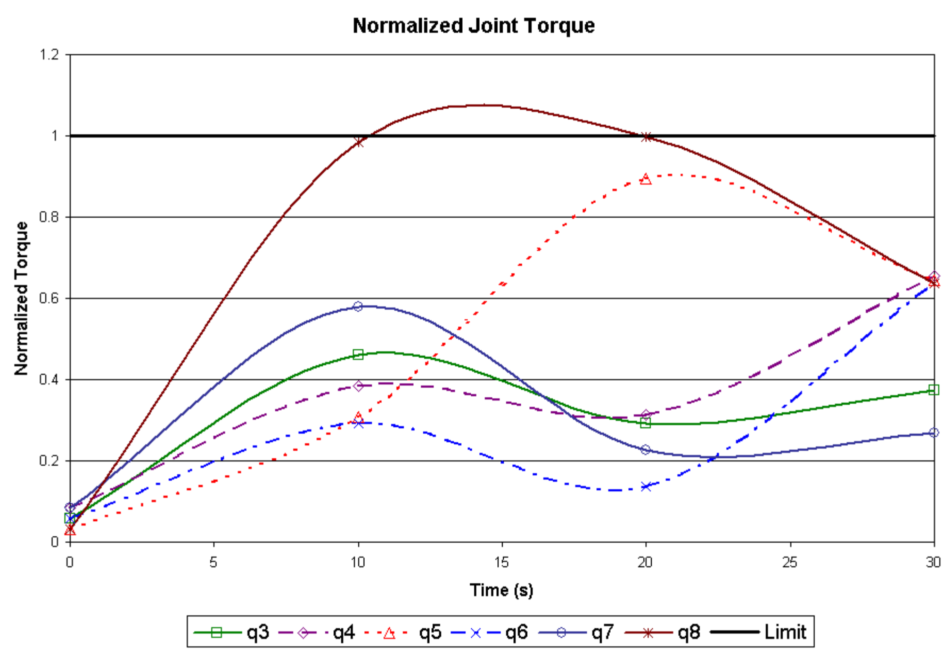

Figure 6.5. Trial 1 normalized joint torque for Obj_1.

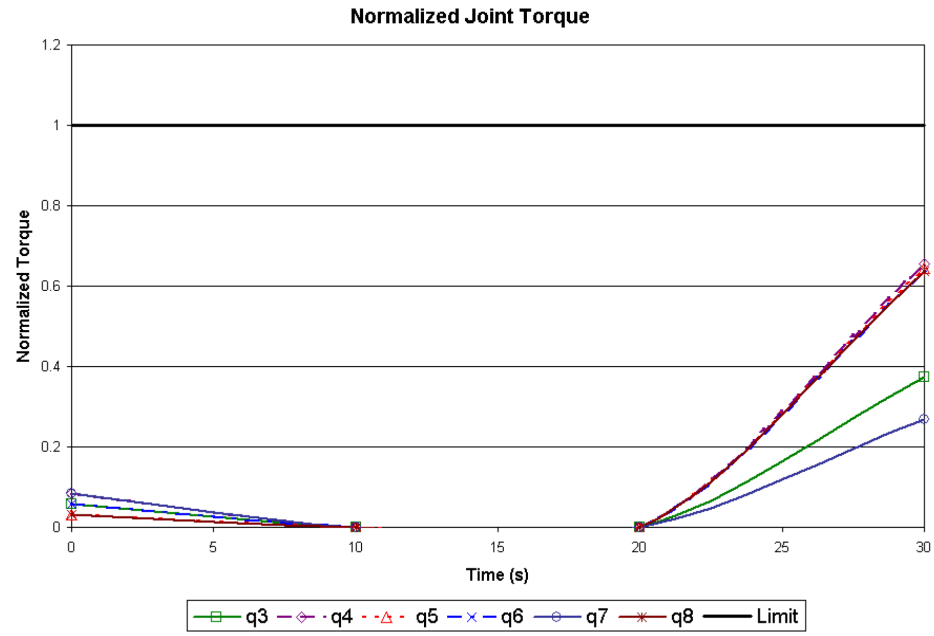

Figure 6.6. Trial 1 normalized joint torque for Obj_2.

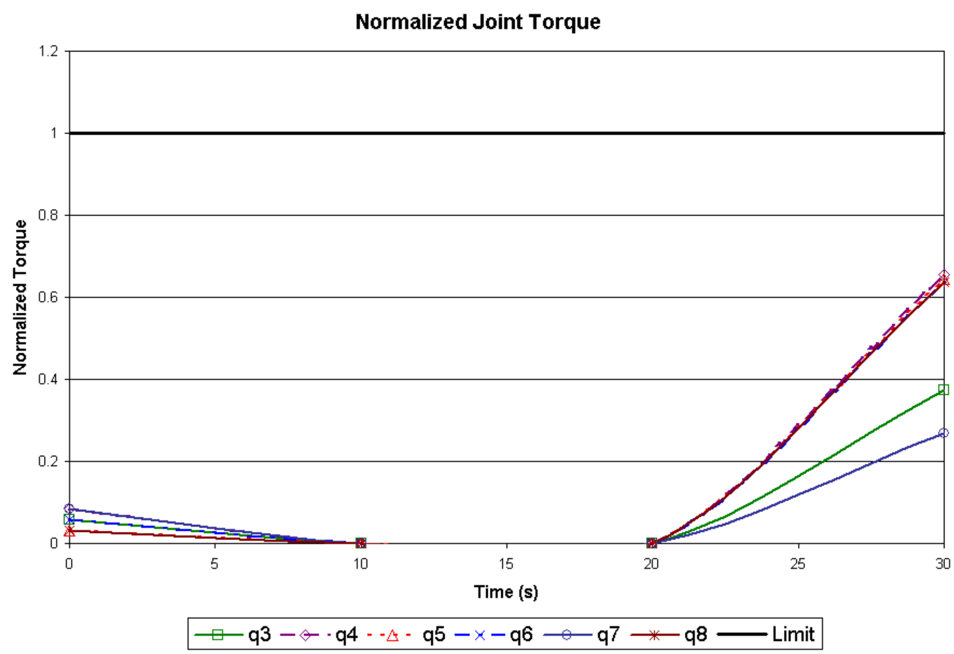

Figure 6.7. Trial 1 normalized joint torque for Obj_3.

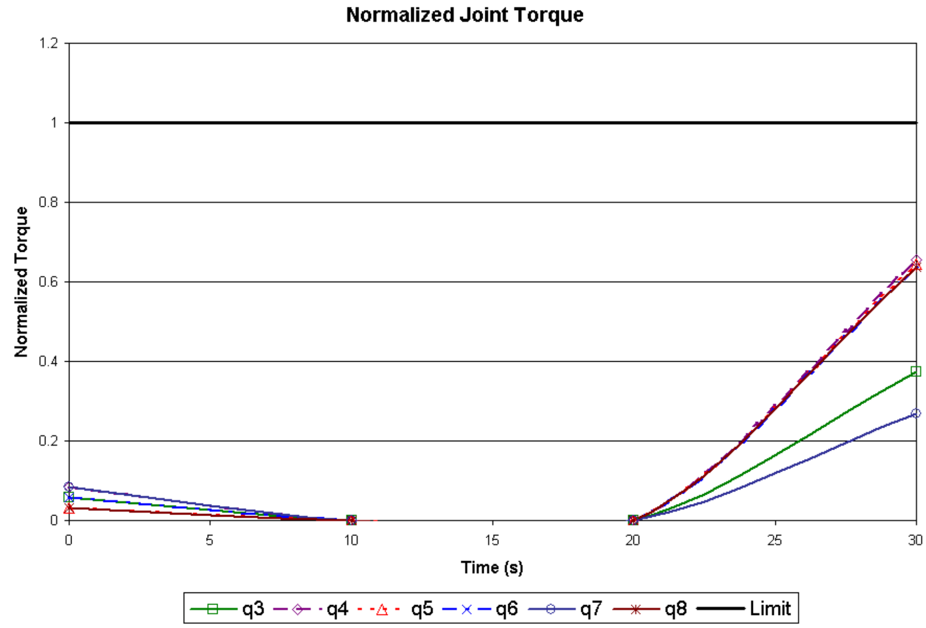

Figure 6.8. Trial 1 normalized joint torque for Obj_4.

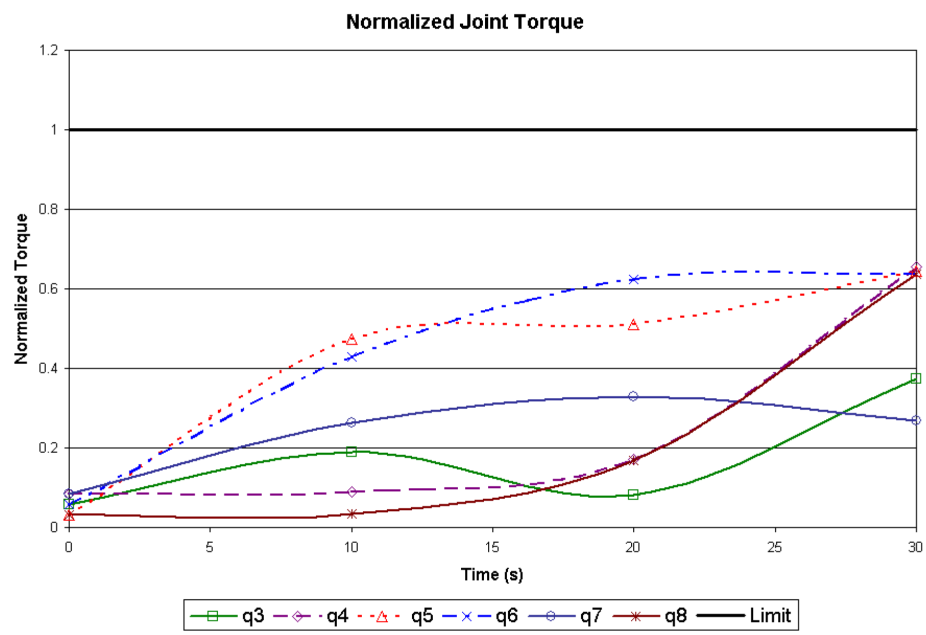

Figure 6.9. Trial 1 normalized joint torque for Obj_5.

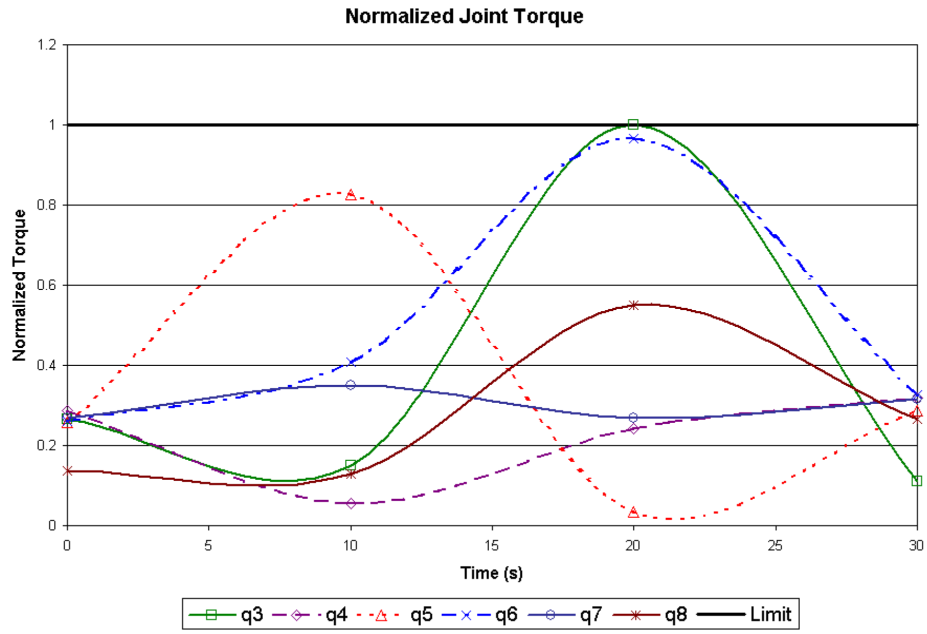

Figure 6.10. Trial 2 normalized joint torque for Obj_1.

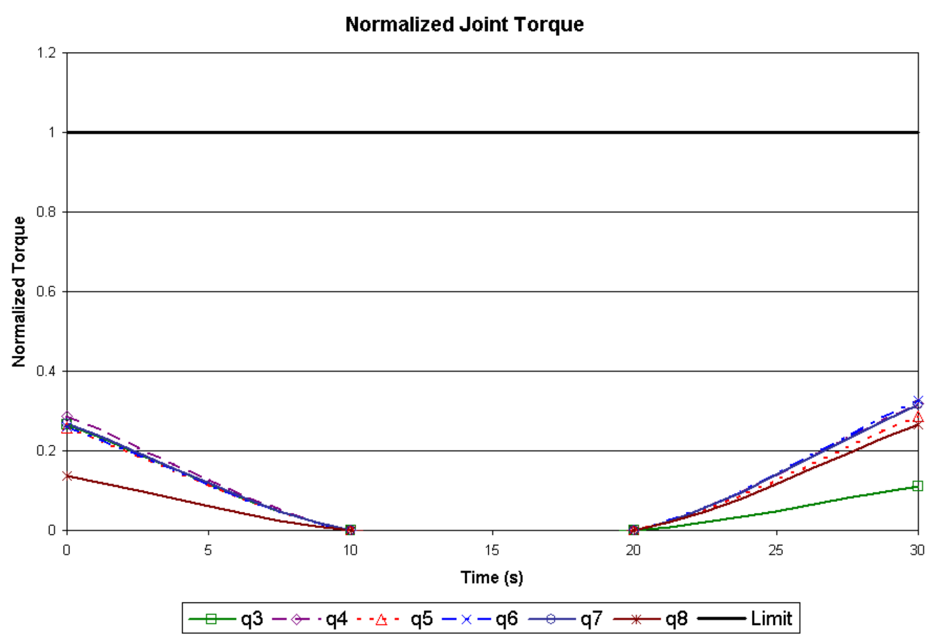

Figure 6.11. Trial 2 normalized joint torque for Obj_2.

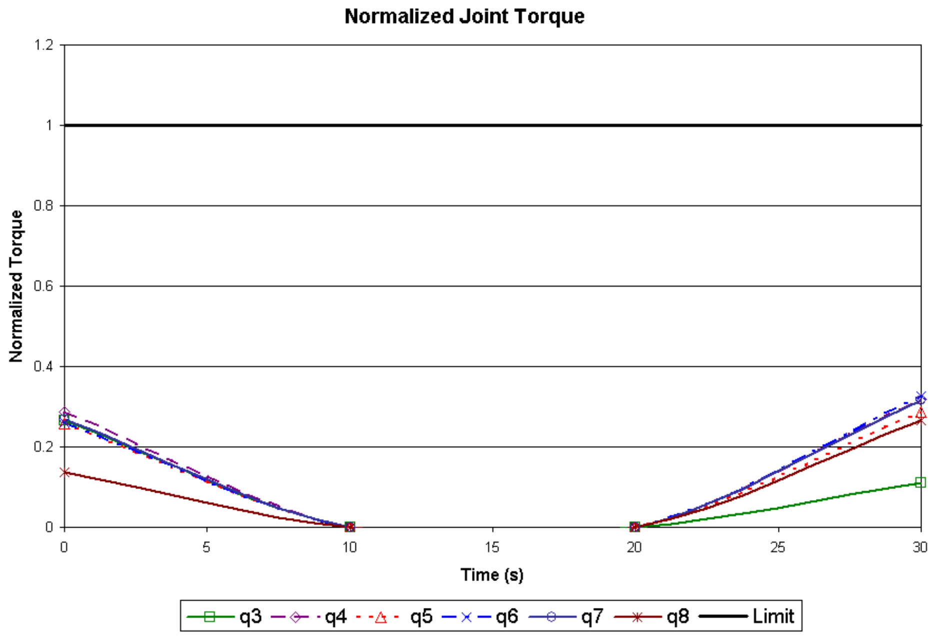

Figure 6.12. Trial 2 normalized joint torque for Obj_3.

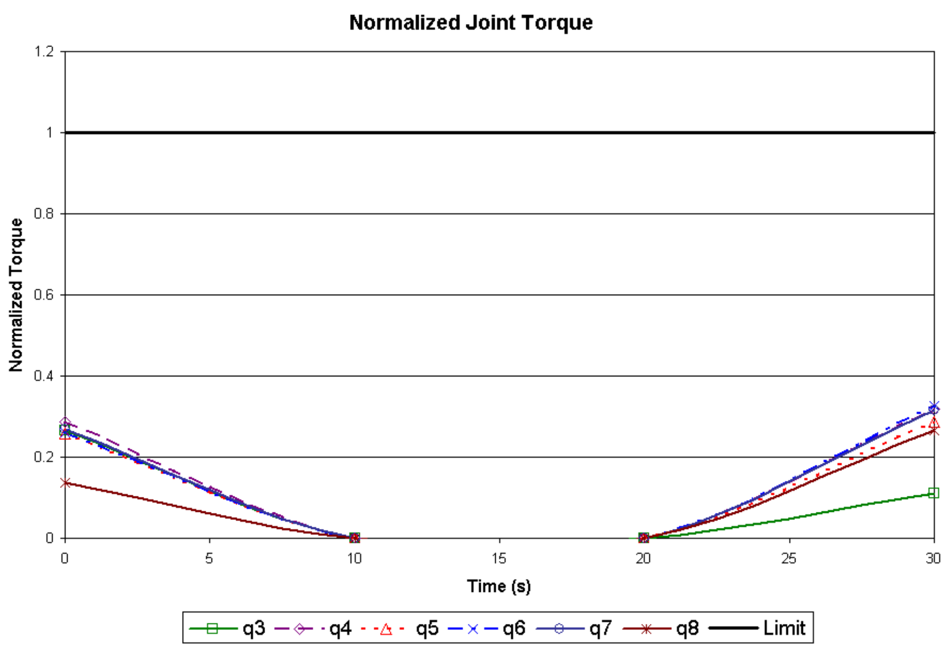

Figure 6.13. Trial 2 normalized joint torque for Obj_4.

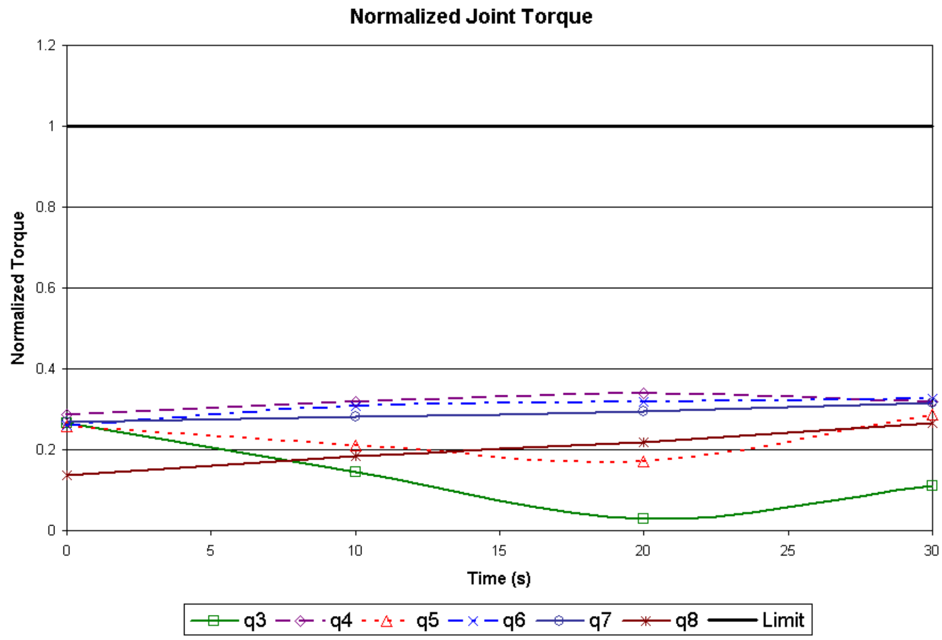

Figure 6.14. Trial 2 normalized joint torque for Obj_5.

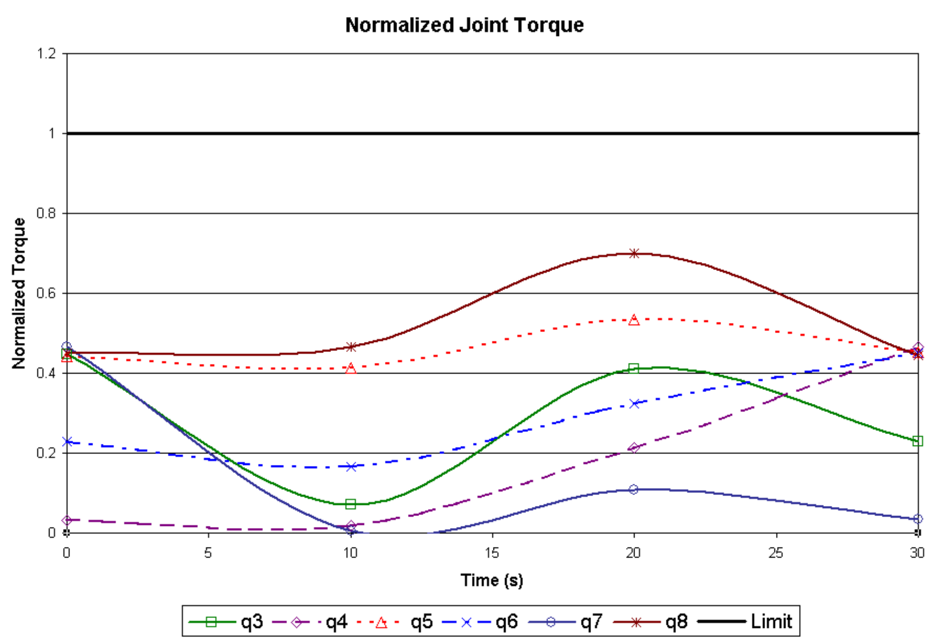

Figure 6.15. Trial 3 normalized joint torque for Obj_1.

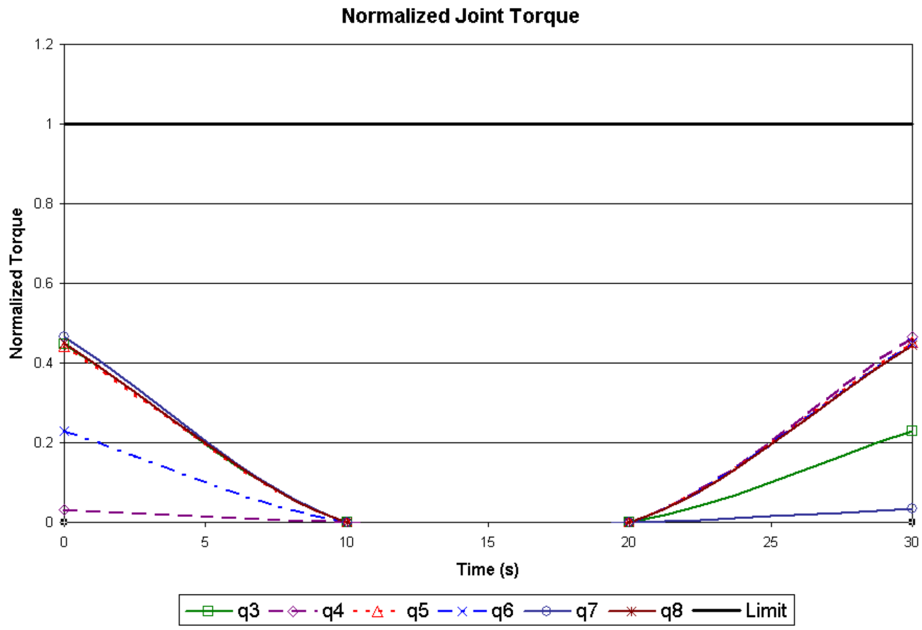

Figure 6.16. Trial 3 normalized joint torque for Obj_2.

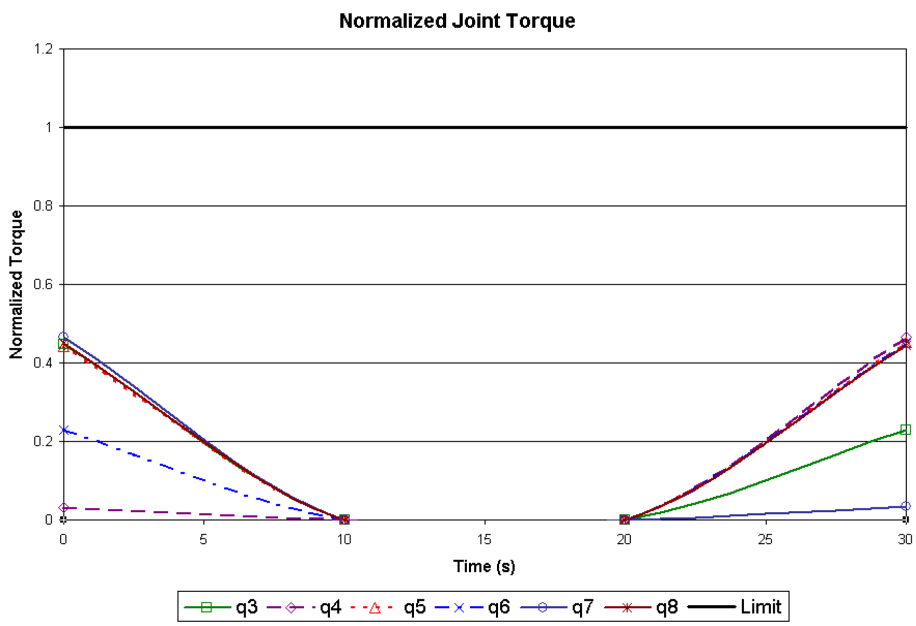

Figure 6.17. Trial 3 normalized joint torque for Obj_3.

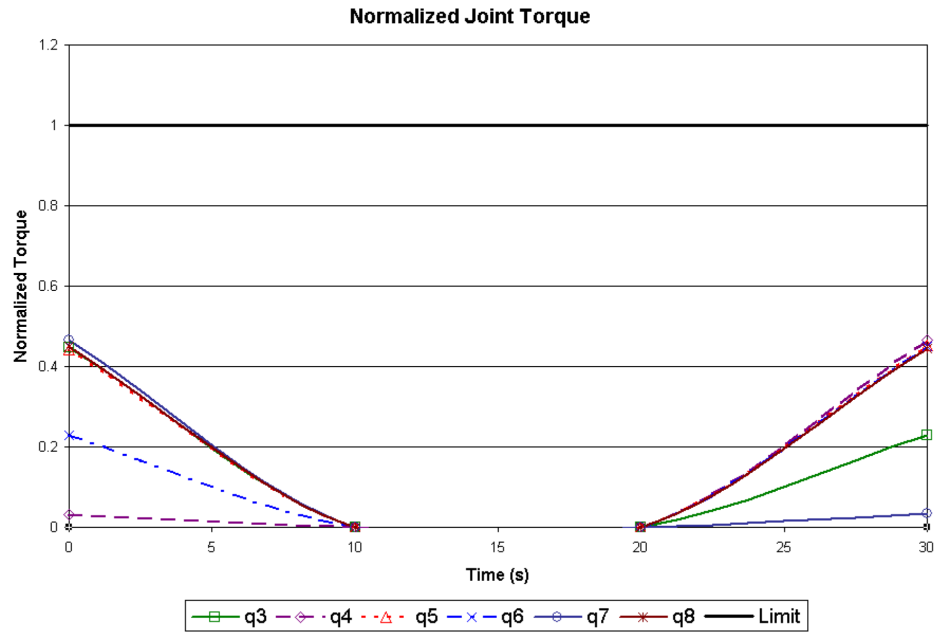

Figure 6.18. Trial 3 normalized joint torque for Obj_4.

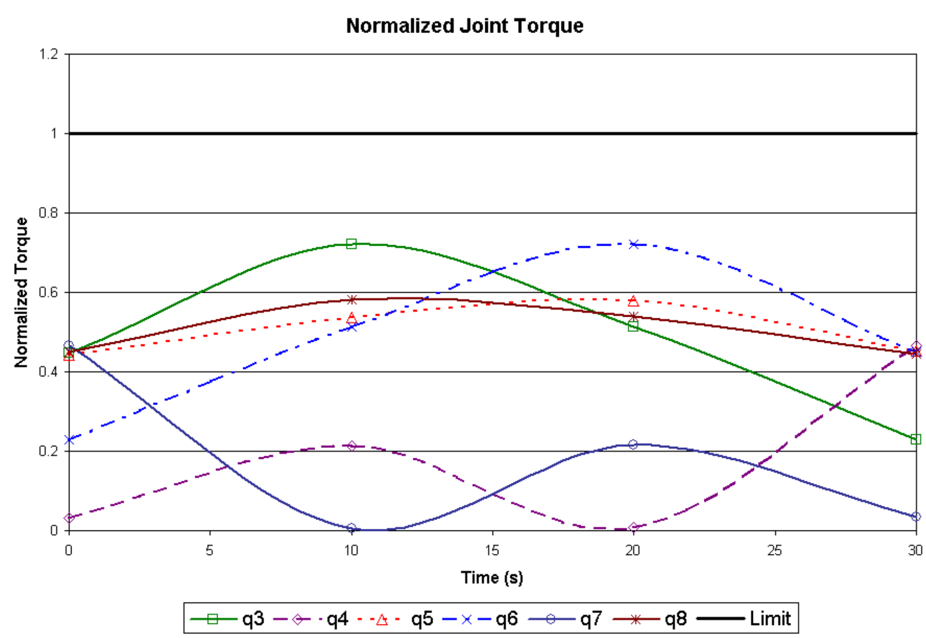

Figure 6.19. Trial 3 normalized joint torque for Obj_5.

Trials 7 and 8 also share the same starting point and target as trial 3. The time allowed for motion is held constant at 30 seconds, but the number of time steps at which is different for the simulations. Four time steps are used for trial 3, 3 time steps are used for trial 7, and 5 time steps are used for trial 8. Figure 6.22 shows the motion for the trials and Figure 6.23 shows the normalized joint torque for q_3. The results for the other joints and objective functions are similar.

a. Trial 3 b. Trial 4 c. Trial 5 d. Trial 6

Figure 6.20. The motion results for Obj_4 when total time for the motion is varied.

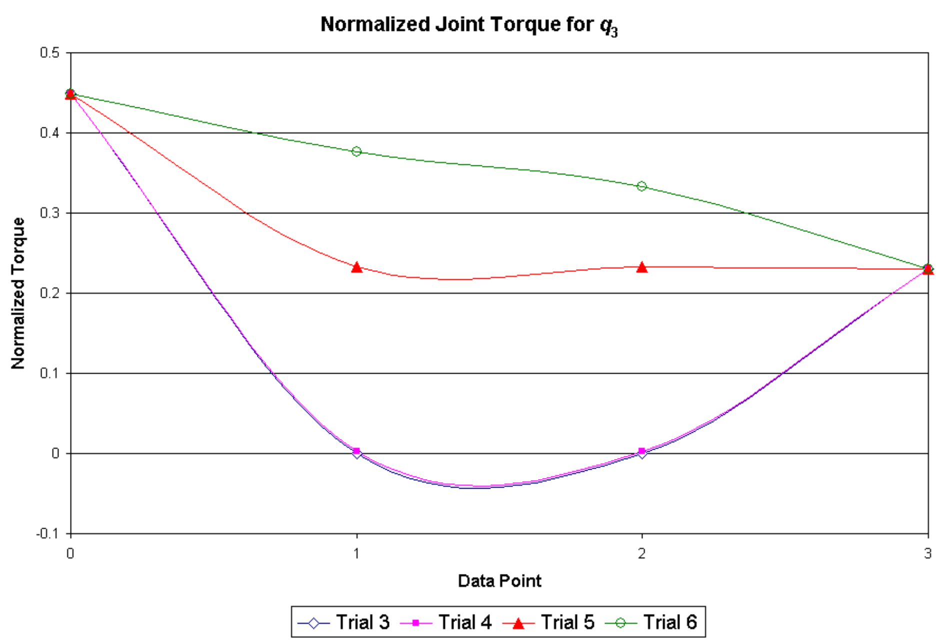

Figure 6.21. Normalized torque on joint q_3 for trials 3, 4, 5, and 6 for Obj_4.

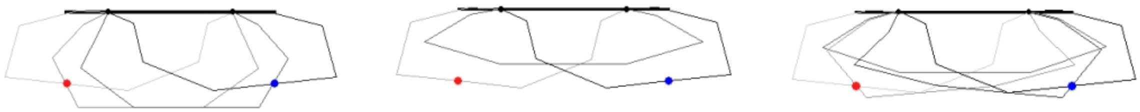

Figure 6.22. The motion results for Obj_4 for varying number of time steps.

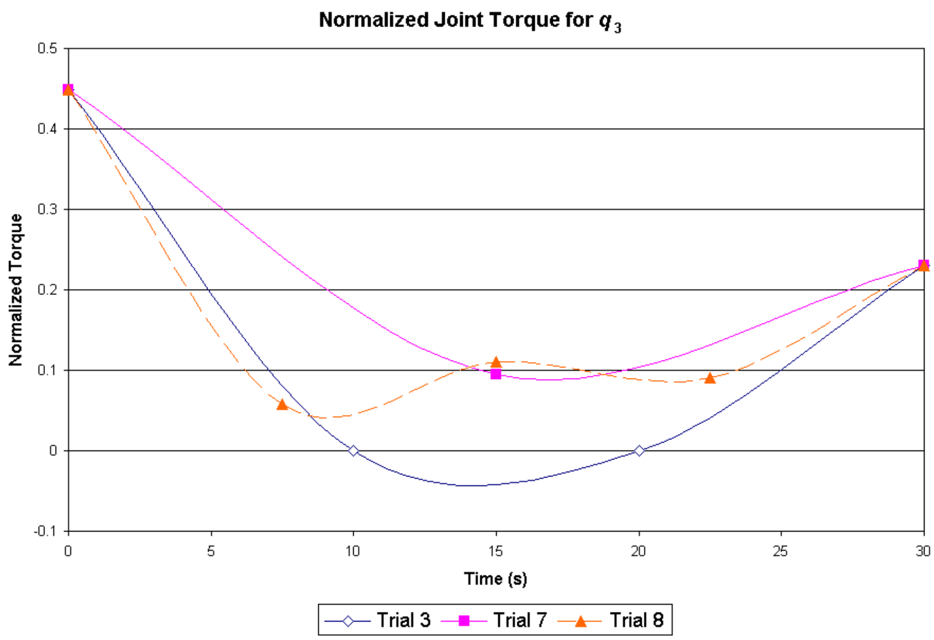

Figure 6.23. Normalized torque on joint q_3 for trials 3, 7, and 8 for Obj_4.

Discussion

As discussed in Chapter 5, Obj_1 is not a dynamic objective function, so it did not result in particularly low torques for posture prediction or low reaction forces. For motion prediction, however, the reaction forces did not vary much between the five objective functions. The sum of the reaction forces over the entire motion is shown in Table 6.1. Even though the reaction forces were similar, the joint torque varied greatly between Obj_1 and the other four objective functions. The difference in joint torque resulted from differences in the motions, not differences in the reaction forces. Obj_1 resulted in high joint torques because its predicted accelerations were high. In Figure 6.2, Figure 6.3, and Figure 6.4 the motion for Obj_1 appears as though it only consists of three time steps because the two intermediate positions are the same for each trial. The linkage accelerates to this intermediate position and then stays there until accelerating to the final posture. These high accelerations are unrealistic.

Table 6.1. Total Reaction Forces for Trial 1.

	Left Hand			Right Hand		
	F_x	F_y	M_z	F_x	F_y	M_z
Obj_1	-791.381	1824.079	48.025	791.506	1079.197	45.458
Obj_2	-791.510	1823.753	48.687	791.503	1078.923	44.533
Obj_3	-791.514	1823.762	48.670	791.499	1078.924	44.645
Obj_4	-791.511	1823.762	48.543	791.501	1078.924	44.671
Obj_5	-791.500	1823.725	52.616	791.499	1078.913	45.022

Obj_5 is not dynamic-based either, but it results in a more believable solution than Obj_1 because the motion is smooth. The resulting torque on the joints is also not minimized, but the joint torque is a smooth function and does not vary greatly from the joint torque at the initial and final postures. Simulations using Obj_5 are also extremely fast compared to the other objective functions. Not only is the equation of motion not included in the objective function, but the objective is already minimized. Table 6.2 shows that it requires fewer iterations than the other functions.

Table 6.2. Optimization Routine Statistics for 9 DOF Motion Trials.

	Iterations	Major Iterations	Constraint Violations	Objective Value	Exit
Trial 1					
Obj_1	1161	106	2.4E-06	7.52	constraints cannot be satisfied
Obj_2	1004	285	1.2E-14	2.16E-12	optimal
Obj_3	1516	243	2.9E-11	3.04E-12	optimal
Obj_4	1044	282	4.1E-08	1.03	optimal
Obj_5	124	14	6.7E-05	0	optimal
Trial 2					
Obj_1	3516	721	5.8E-07	7.52	cannot be improved
Obj_2	506	68	1.7E-13	7.01E-14	optimal
Obj_3	368	44	2.0E-12	8.06E-13	optimal
Obj_4	634	109	1.2E-08	1.03	optimal
Obj_5	181	18	4.4E-07	0	optimal
Trial 3					
Obj_1	1131	83	1.5E-06	7.52	cannot be improved
Obj_2	557	80	3.7E-14	3.38E-13	optimal
Obj_3	394	56	7.1E-12	1.07E-12	Optimal
Obj_4	591	90	3.6E-09	1.03	Optimal
Obj_5	147	19	8.2E-07	0	Optimal

The motion and normalized torque for each trial of Obj_2, Obj_3, and Obj_4 were extremely similar. At first glance, Obj_2, Obj_3, and Obj_4 seem to offer the best solutions.

The joint torque appears to be close to zero between the second and third time step and the overall motion appears smooth. This, however, is an illusion created by only reporting the results at the constrained time steps. Figure 6.24 shows the torque for joint q_3 for trial 3 of Obj_4 sampled at more time steps than those used to enforce the optimization constraints. Points depict the constrained time steps, and sampling the B-spline approximation of joint angle and reaction force at 40 time steps generates the curve. The function of joint torque oscillated greatly and the normalized torque violates the torque limit by over 20%. It is only at the two intermediate time steps that the torque was minimized. If the motion of the linkage is examined for all 40 time steps there are two distinct moments when the linkage does not move smoothly, see Figure 6.25. These moments correspond to the constrained time steps.

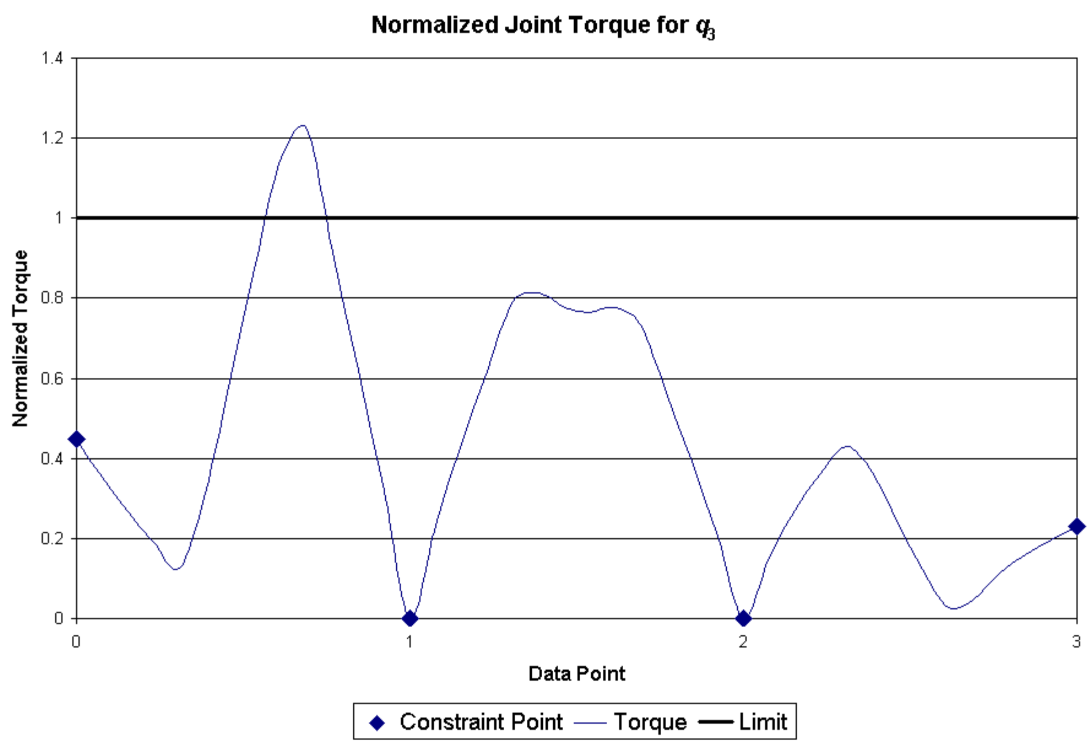

Figure 6.24. Joint torque for joint q_3 for trial 3 and objective function Obj_4.

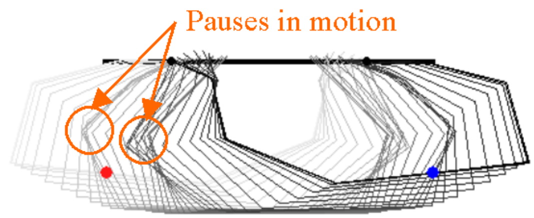

Figure 6.25. Trial 3 of Obj_4 sampled at 40 time steps.

This issue with sampling rate also helps explain Figure 6.21. The joint torque should decrease as the time allowed for the motion increases because the joint velocities and accelerations decrease. Figure 6.21, however, shows the exact opposite. As the time allowed is increased the joint torque increases. If more time steps are examined, as in Figure 6.26, the average torque for trial 6 is lower than trial 3. This is as expected.

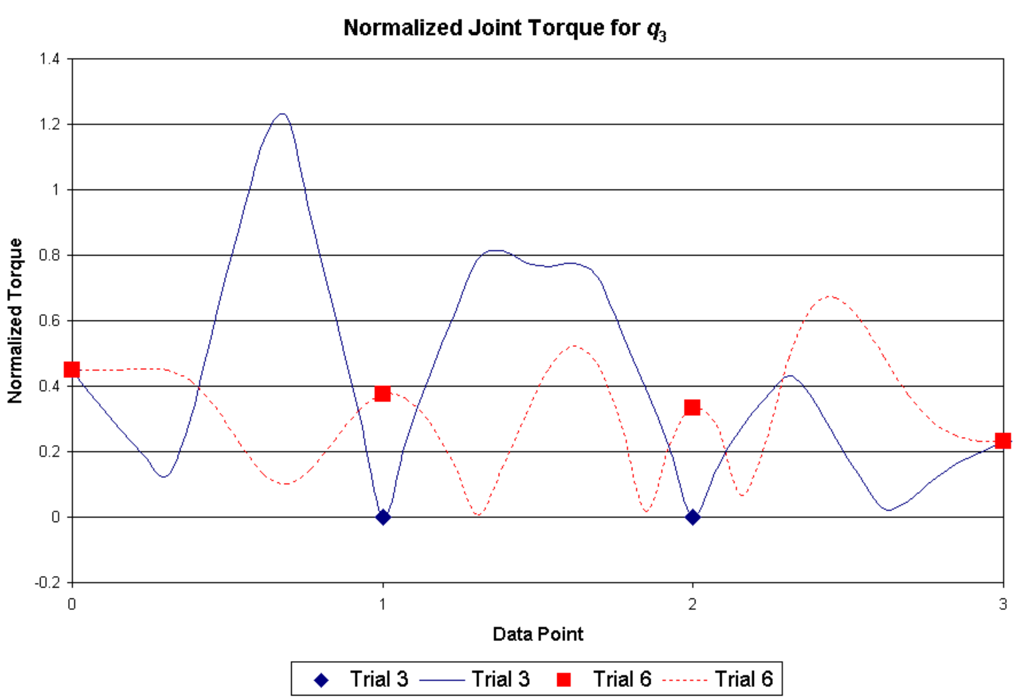

Figure 6.26. The joint torque for joint q_3 for trials 3 and 6 and Obj_4.

Trials 7 and 8 altered the number of time steps at which the optimization constraints are enforced. This changes the shape of the function considerably, Figure 6.23. However, if the number of control points is increased to greater than 5 the optimization routine exits with the message that the constraints cannot be satisfied. Since it is unknown whether the motion is indeed feasible, the amount of time allowed for the motion is increased to 30,000,000 seconds. This drives the velocities, accelerations, and joint torques almost to zero and still the optimization routine cannot solve the problem. If, however, the dynamic constraints are removed (constraints g_5, g_6, g_7, and g_8) and Obj_5 is used, then the solution can be found for at least 40 time steps. This solution, however, is not ideal because it removes the constraint of dynamic equilibrium, which is necessary to produce realistic reaction forces and motion. More research is needed to improve the stability of this code.

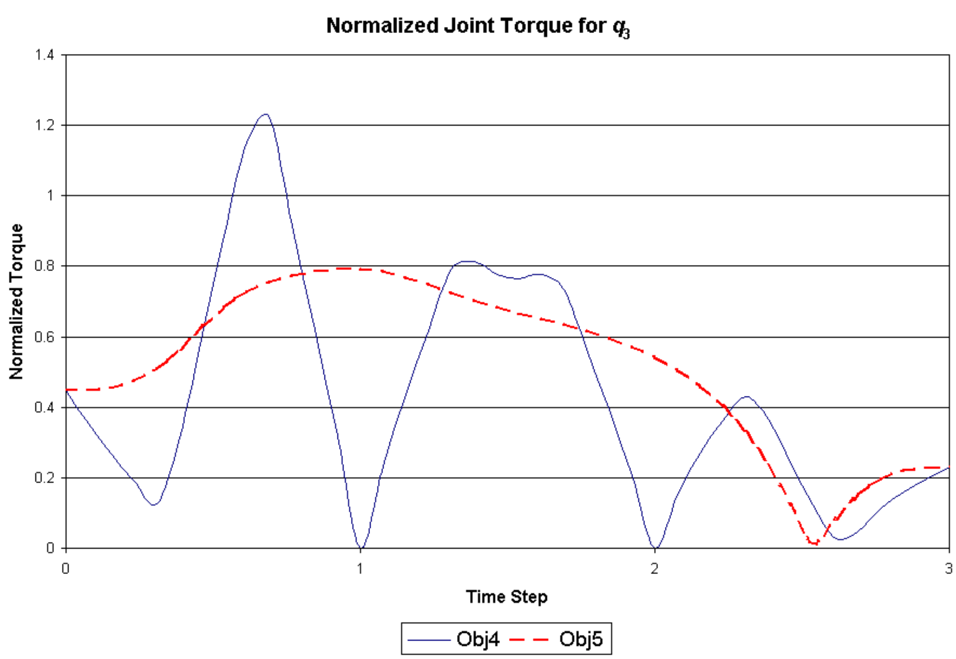

Figure 6.27. Joint torque for joint q_3 for Obj_4 and Obj_5 for trial 3.

Finally, if the torque curves are sampled at the higher rate of 40 time steps, Obj$_5$ appears to be a good candidate for predicting realistic motion. Even though the motion is the result of a simple linear interpolation between the initial and final postures, by not minimizing the torque at particular time steps the end motion and joint torque is smoother than the dynamic-based objective functions. Figure 6.27 shows the torque on q_3 for Obj$_4$ and Obj$_5$.

55 Degree of Freedom Model

Results

Motion prediction for the 55 DOF Santos™ model proved to be very unstable. The problems encountered made it impossible to find a feasible motion using the formulation outlined in this chapter. Two trials were completed, however, by removing the constraints dealing with dynamics, g_5, g_6, g_7, and g_8. Obj$_5$ was used as the objective function. Figure 6.28 illustrates the solution for the motion between the initial and final postures used for trial 3 in Chapter 5. Figure 6.29 illustrates the solution for the motion between the initial and final postures used for trial 4 in Chapter 5.

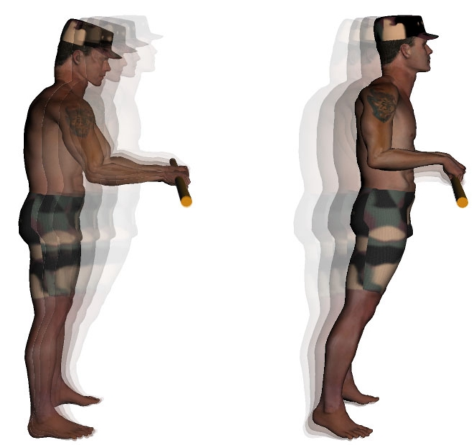

Figure 6.28. Santos™ leaning toward the bar.

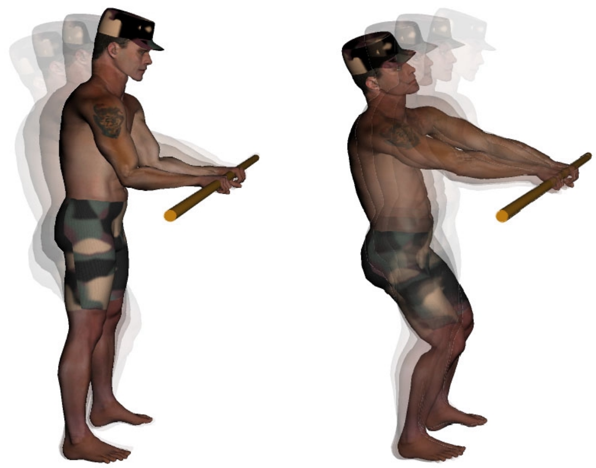

Figure 6.29. Santos™ squatting back and to his right.

Discussion

The motion of Santos™ resulting from this modified formulation appears realistic and the optimization routine can find a solution quickly if the final posture has already been found. The reaction forces and motion of his body, however, do not satisfy dynamic equilibrium. Using this modified formulation, the simulation can also solve for more time steps than is possible with the 9 DOF model. The simulations shown in Figures 6.28 and 6.29 contain 4 time steps, but simulations using up to 7 time steps were also completed.

The instability of the code is not due to the size of the optimization problem. SNOPT™ is capable of handling problems with thousands of design variables and constraints. One possible source of trouble is the lack of analytical gradients. Without accurate gradient information, the optimization routine must use finite difference, which only provides approximate gradients. As already discussed with posture prediction, analytical gradients may help the optimization routine find a feasible solution while including all the constraints. Adding joint torque as a design variable might also help by simplifying the gradients of the constraints and dynamic objective functions.

Another possible source of trouble is inaccurate input data. If the torque limits are too restrictive or the joint stiffness constants are too high, the problem may be infeasible. Most of the anthropometric data is based on estimations; some of the estimations are based on statistical data, but some are based on educated guesses.

A possible solution to solving the current problem is to add an additional optimization routine to find the reaction forces. This approach is similar to what is currently done for the initial posture. Given the initial posture, the reaction forces are found that meet all the constraints and minimize joint torque. Given the motion, an optimization routine could be used to solve for the time profiles of the reaction forces that meet all the constraints. By breaking the motion prediction problem into two optimization routines, it simplifies the problem and may make it easier to find a feasible solution.

Conclusions

The results for the 9 DOF model show that a solution can be found using this formulation. It is, however, not very stable. The number of time steps that the optimization routine can successfully solve limits even the 9 DOF model. The 55 DOF model is even more limited because the dynamic constraints cannot be included.

One positive conclusion that can be made from the motion prediction results is that a linear interpolation between the initial and final postures results in a realistic motion for short movements of the type explored here. The realism of the motion seems to depend more on the initial and final postures than the motion between these postures. In addition, finding the feasible solution closest to the linear interpolation is very quick compared to the dynamic-based objective functions.

Another interesting conclusion is that the realism of the solution is heavily dependent on the number of time steps used in the optimization. The dynamic objective functions produced poor results for only 4 time steps. If the number of time steps can be increased, however, the quality of the solution may increase. However, even with more time steps, the solution may not be more realistic than the linear interpolation and will certainly be slower. For the realism and the amount of time required, using Obj_5 offers the best solution.

CHAPTER 7

FUTURE WORK AND SUMMARY

There are two main issues, which need to be addressed to improve the fidelity of the swinging simulation. The first issue is the correctness of the input. The accuracy of this program, like any other, depends on the accuracy of the input. The anthropometric data for this simulation are estimated, and while some estimations are accurate, others can be improved upon greatly. The second issue is that of adding different objective functions to produce more realistic motion.

The code can also be improved in many ways to improve its current performance and to enhance its capabilities. First, the stability of the code should be improved so that the optimization routine can find a solution when all the constraints are included. Then the efficiency of the code should be improved, which will help decrease the amount of time required to find a solution. Another issue is the generality of the code. To be able to handle slightly different tasks the code should be more general.

Finally, this thesis has presented a formulation and code to predict human motion, but has not validated any of the predictions. Validation must be completed to develop a truly realistic simulation. In this chapter, each of these areas of improvement and future work will be discussed. Conclusions will follow.

Anthropometric Input

All the anthropometric data used for this simulation were estimations. The only accurate way to measure anthropometric data like link mass, mass moments of inertia, and even link lengths is to dissect the human. This is impossible for a model of a living person, so statistical data established using cadavers are used. Even these data are not exactly accurate because once dead, the weight and mass distribution of the body changes as fluid circulation stops and other living processes cease.

As stated earlier, the link lengths used in this report are taken from the 3-D body scan of the human used for Santos™. These link lengths are estimated based on an animator's perception of where Santos' ™ joints appear to be located. The limits for joint displacement are also estimations based on the model of Santos™. Both of these estimations are well tested and appear to provide realistic results. The link masses are based on data for a 95th percential male (Williams, Lissner, and LeVeau, 1992; Chaffer and Andersson, 1984). While there are data available for the mass of body segments, the body segments used in the cadaver studies do not necessary correspond to Santos' ™ body segments. For instance, one mass is given for the chest of a cadaver, but Santos™ has 6 body segments that comprise his upper body, so deciding how to divide the mass assigned to "chest" results in approximations. The same is true for the mass moments of inertia and the location of the center of mass for each body segment. For this thesis, the mass moments of inertia and the center of mass locations are estimated by approximating Santos'™ body segments as cylinders of evenly distributed mass.

Other anthropometric data are estimated because statistical data do not exist. Joint torque limits and joint stiffness coefficients have not been the subject of enough research to develop statistical databases. Most biomechanic research is focused on determining maximum muscle strength, not generalized torque for the joints. Some data do exist, but it is not available for all joints or for humans of different statues (Anderson and Pandy 1999; Gonzalez et al., 2002). The joint torque limits and joint stiffness coefficients greatly affect the motion, and so finding accurate values for these would improve the model greatly.

All of these approximations reduce the accuracy of the model. The input approximations should be improved to enhance the realism of the model and to eliminate the input as a source of error in the output. While exact values may never be found, better approximations are entirely possible.

Objective Functions

More research needs to be conducted into what the best objective function is to model realistic swinging motion. For non-dynamic posture prediction already implemented in Santos^{TM} there are multiple objective functions from which to choose: effort, discomfort, joint displacement, potential energy, visual acuity, and visual displacement. These objective functions can be used individually or combined to predict posture.

For dynamic motion prediction, some possible objective functions to try include vision, energy consumption, balance, and discomfort. Vision and discomfort would both be similar to the functions used for non-dynamic posture prediction. Vision would help raise the fidelity of the simulation during climbing because a human looks to where he/she is climbing or looks at his/her hands when reaching for a new handhold.

Santos^{TM} currently uses energy consumption for upper body dynamic motion prediction (Kim, 2004). Energy consumption is related to torque, but also accounts for muscle strain and heat energy. Since it also includes heat energy loss it can take into account the surrounding temperature and clothing.

Finally balance is very important for predicting realistic lower body postures. Equilibrium is enforced in the swinging simulation, which requires that the reaction forces at the hands and feet balance the resultant forces of the body. However, some postures feel more balanced than others. This is dependent on the width of the stance and the force distribution between the hands and feet. Incorporating the idea of balance might also improve the fidelity of the simulation.

Improving the Robustness of the Code

Several changes could improve the robustness of the code and hopefully reduce the computation time. The first improvement has already been discussed several times. Adding analytical gradients of the constraints and objective function will improve the accuracy of the solution and may help the optimization routine find an optimal solution for problems it currently cannot solve. Gradients may also reduce the time because finite difference would no longer be used to approximate the gradients.

Adding the joint torque as design variables may also help the routine find an optimal solution. If the joint torques are added as design variables, then an equal number of equality constraints need to be added to ensure the equations of motion are satisfied for each joint. Theoretically this does not increase the feasible space for the problem, but it will simplify the gradients of the dynamic objective functions and constraints and thus may help the solution process.

To help decrease the computation time, the constraint and objective function routines can be combined within the optimization code. Currently there is one subroutine for the constraints and one subroutine for the objective function. Within the optimization software being used for this simulation, there is an option of combining these subroutines. This would greatly reduce the time because the equations of motion would only be calculated once to find both the constraints and the objective function.

Another optimization tool that could be used to increase the efficiency of the program is a sparse matrix solver. This type of optimization routine is used for problems with sparse matrices (lots of the derivatives are zero). It takes advantage of the zeros by not including them in the calculations thus decreasing the number of operations performed by the computer.

Generalizing the Task

To handle slightly different tasks and to ready the program for use with climbing simulations, improvements need to be made that will generalize the code. In this section, several possible improvements will be discussed in detail.

Orientation and Position of the End Effectors

The first improvement would be to allow the user to interactively constrain the position and orientation of the end effectors. Currently, the positions of the end effectors are set by the initial posture read into the program. Also the vector from the local coordinate system to the point of interest, the vector shown in Figure 7.1, is hard coded. The vector used for orientation, a vector along the bar in Figure 7.1, is also hard coded. If this program is used to simulate a different task, both the orientation vector and the position vector may change.

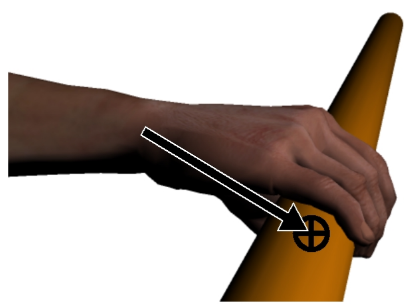

Figure 7.1. The hand holding onto a bar and the vector from the local coordinate system to the point used for the position constraint.

Allowing the end effectors to be constrained to a surface or line, rather than just a point, is another valuable improvement. If this program is expanded to simulate opening doors, one of the hands should be constrained to move along the path of the handle as the door opens. This will be achieved by creating an equation to describe the path of the door handle (an arc). Then x, y, and z components of the end effector location will be constrained to satisfy this equation. This would not constrain the hand to be any particular location at a particular time, unlike path prediction currently used in Santos™ (Mi, 2003).

Reaction Forces

An improvement to the reaction force constraint would be to find realistic limits and give the limits for the hand in the local coordinate system of the hand. The reaction force limits for the hand are currently given by symmetric limits based on hand strength. As explained in Chapter 5, this is not the case in real life. For instance, a man can push with his palm and generate a much larger reaction force than pulling. To implement these non-symmetric limits, however, they need to be expressed in the local coordinate system of the hand. Currently in the swinging simulation the reaction forces are optimized with respect to the global frame. This change will be made by multiplying the reaction forces by the inverse transformation matrix to transform them from global vectors to local vectors.

The reaction force limits can also be improved by relating the force limits to the friction coefficient between the end effector and the ground. For instance at the foot, the normal force between the foot and ground and a known friction coefficient should determine the limit on the reaction forces tangent to the ground. This, however, means that the limit would be changing during the optimization routine, which if the constraint is active may cause difficulties for the optimization routine.

Changing the point of application of the reaction force would also increase the realism of the simulation. The point of application of the reaction force at the feet should change with the motion. A simple way to incorporate this would be to add two more end effectors to the routine in include the heel of the foot. Then the optimization routine would also determine how to distribute the force between the heel and the ball of the foot. As Santos™ leans forward the reaction force on the toe would increase and the reaction force on the heel would decrease. As he leans back the opposite would occur.

Finally, the reaction force limits depend on the task being simulated. For the example of opening a door, the hand opening the door cannot have a reaction force along the path of the door (assuming the force it takes to open the door is small).

Updatable Target Point

Another simple update that would greatly increase the generality of the simulation is to add the ability to input the point on Santos' ™ body that moves to the target. Currently the Santos'™ hip moves to the target during the motion. If this was updatable, the user could specify whether he/she wanted the hip to touch the target point or the chin or any other point on the body. Creating a simulation for pull-ups is a good example because then the chin is the point of interest on the body. For a pull-up, the chin should be above the bar at the end of the motion. Other improvements would be to include intermediate points. For instance if the entire pull-up is simulated then the target point at the end of the motion would be the same as the initial position and the intermediate target point could be a point above the bar. A final improvement would be to be able to define a target plane or line rather than just a point.

General External Forces

The reaction forces at the end effectors are the only allowed external forces in this simulation. By adding the ability to consider other external forces, weight due to clothing or backpacks could be also considered. These forces may greatly affect the motion.

Collision Avoidance

Collision avoidance provides an important tool for realistically modeling a human's interaction with the environment. Collision avoidance will be added to the optimization formulation as an additional constraint that would prohibit collision with the environment and with Santos'™ own body. This tool will be crucial, once the swinging simulation is expanded to predict the motion of climbing a ladder. When climbing, the legs and arms of the human cannot pass through the ladder.

For non-dynamic posture prediction, currently implemented in Santos™, collision avoidance is used to prevent self-avoidance. Figure 7.2 shows the current posture prediction with and without self-avoidance. Self-avoidance is achieved by using spheres of various sizes to represent different body parts. The Cartesian coordinates of the center of a sphere i is $\mathbf{x}_i$ and its radius is R_i. The constraint in Equation (7.1) ensures that any two spheres (sphere i and sphere j) do not intersect (Yang et al., 2006).

$$\left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2 \geq \left(R_i + R_j \right)^2 \tag{7.1}$$

Figure 7.2. Santos™ predicting posture to a point without and with collision avoidance.

Validation

The formulation proposed in this thesis is a theory for dynamic motion prediction and needs to be validated. Three stages of validation need to be completed. First the joint angles of the posture and motion predictions must be validated. Validation of posture and motion is accomplished using motion capture. Motion capture is a method of recording a live motion by tracking a number of key points on the body, and then combining them into a single 3D representation of the motion. The motion of real people will be recorded performing various tasks and then compared to the simulation results of Santos™ performing the same tasks.

Secondly, the reaction forces at the hands and feet need to be validated. This can accomplished using force plates in the floor to record the vertical force of the feet and handholds with force instrumentation to record the forces at the hands. Validating the moments and directional components of the reactions may prove to be difficult.

Lastly, the generalized forces at the joints must be validated. This poses a significant problem. As already stated in the discussion of joint torque limits, biomechanical researchers have not been interested in measuring joint torque and so no well-established methodology exists. As an initial check, the joint torques could be compared to other simulation packages. For example LifeMod® calculates the generalized forces at each joint during a motion. The motion can either be created using motion capture data or the joint angles from a simulation can be used. As a final validation this check would not be adequate because it is not using human test data and is trusting code written by some one else. However, it may help point out problems with the simulation before more expensive validation can be completed.

Summary

The problem of dynamic motion prediction is a complex one. No single simulation has been developed to accurately predict 3-D human motion using both a large number of DOF and dynamics. This simulation attempted to solve one specific type of dynamic motion, swinging. The problem was formulated using optimization-based dynamics and implemented in a C code. The simulation works well for the 9 DOF model, but is unstable when used for the 55 DOF Santos™ model. However, kinematic motion prediction between two dynamic postures was successfully implemented and produces realistic results.

Regardless of the problems the simulation experiences, it still adds important new capabilities to Santos™. It enhances Santos™ by increasing his degrees of freedom, by adding global position and orientation, by allowing closed kinematic loops, and by formulating three new dynamic objective functions. The program provides tools for dynamic posture prediction and motion prediction.

This code is only the initial step towards simulating more complex dynamic motions like climbing or opening doors. The approaches created for this simulation, like the method for modeling global DOF, can aid in the development of simulations for other dynamic motions, like walking or running. Dynamics must be considered when modeling human posture or motion for the results to be convincing. The work in this thesis increases Santos'™ dynamic capabilities and puts him one step closer to being truly lifelike.

APPENDIX

Table A.1. Link Lengths for Santos™.

Link	Link Length (cm)	Link	Link Length (cm)
L_1	5.4055	L_{14}	23.2267
L_2	5.2288	L_{15}	4.1972
L_3	5.7638	L_{16}	8.5694
L_4	16.7023	L_{17}	8.0344
L_5	1.6650	L_{18}	8.6463
L_6	5.5782	L_{19}	39.2444
L_7	14.6783	L_{20}	38.6933
L_8	26.08	L_{21}	13.0644
L_9	23.2267	L_{22}	8.0344
L_{10}	1.6650	L_{23}	8.6463
L_{11}	5.5791	L_{24}	39.2444
L_{12}	14.6783	L_{25}	38.6933
L_{13}	26.08	L_{26}	13.0644

Table A.2. Link Mass Information for Santos™.

Body Segment	Mass (kg)	Local CG Vector (m)	Body Segment	Mass (kg)	Local CG Vector (m)
Spine 1	4.8	(0, - 0.027, 0)	R Hip	6.6	(-0.04, -0.043, 0)
Spine 2	4.8	(0, -0.026, 0)	R Up Leg	9.7	(0, -0.196, 0)
Spine 3	4.8	(0, -0.029, 0)	R Lo Leg	4.5	(-0.193, 0, 0)
R Chest	6.0	(-0.008, 0.084, 0)	R Foot	1.4	(0, 0.065, 0)
R Clavicle	0.5	(0, -0.028, 0)	L Hip	6.6	(-0.04, -0.043, 0)
R Clavicle	0.85	(-0.073, 0, 0)	L Up Leg	9.7	(0, 0.196, 0)
R Up Arm	2.8	(0, -0.130, 0)	L Lo Leg	4.5	(0.193, 0, 0)
R Lo Arm	1.6	(0, -0.116, 0)	L Foot	1.4	(0, -0.065, 0)
R Hand	0.6	(0.08, 0, 0)			
L Chest	6.0	(-0.008, 0.084, 0)			
L Clavicle	0.5	(0, 0.028, 0)			
L Clavicle	0.85	(-0.073, 0, 0)			
L Up Arm	2.8	(0, 0.130, 0)			
L Lo Arm	1.6	(0, 0.116, 0)			
L Hand	0.6	(0.08, 0, 0)			
Mid Chest	6.5	(0, -0.104, 0)			
Neck	2.2	(0, -0.043, 0)			
Head	6.1	(0.08, 0, 0)			

Table A.3. Mass Moments of Inertia for Santos™.

Body Segment	Ixx	Iyy (kg m^2)	Izz	Body Segment	Ixx	Iyy (kg m^2)	Izz
Spine 1	0.0489	0.0239	0.0274	R Hip	0.0335	0.0289	0.0412
Spine 2	0.0487	0.0238	0.0271	R Up Leg	0.4008	0.1381	0.5116
Spine 3	0.0494	0.0240	0.0280	R Lo Leg	0.0046	0.2269	0.2269
R Chest	0.0560	0.0216	0.0634	R Foot	0.0064	0.0022	0.0082
R Clavicle	0.0016	0.0007	0.0011	L Hip	0.0335	0.0289	0.0412
R Clavicle	0.0031	0.0076	0.0076	L Up Leg	0.4008	0.1381	0.5116
R Up Arm	0.0518	0.0180	0.0656	L Lo Leg	0.0046	0.2269	0.2269
R Lo Arm	0.0223	0.0076	0.0291	L Foot	0.0064	0.0022	0.0082
R Hand	0.0000	0.0038	0.0038				
L Chest	0.0560	0.0216	0.0634				
L Clavicle	0.0016	0.0007	0.0011				
L Clavicle	0.0031	0.0076	0.0076				
L Up Arm	0.0518	0.0180	0.0656				
L Lo Arm	0.0223	0.0076	0.0291				
L Hand	0.0000	0.0038	0.0038				
Mid Chest	0.0918	0.0341	0.1050				
Neck	0.0150	0.0068	0.0109				
Head	0.0000	0.0098	0.0098				

Table A.4. The DH Table for the Spine.

Description	Coordinate System	θ	d	α	a
Spine 1a	z_0	$\pi / 2$		$\pi / 2$	
Spine 1b	z_1	$q_1 + \pi / 2$	0	$\pi / 2$	0
Spine 1c	z_2	$q_2 + \pi / 2$	0	$\pi / 2$	0
Spine 2a	z_3	$q_3 + \pi / 2$	L_1	$\pi / 2$	0
Spine 2b	z_4	$q_4 + \pi / 2$	0	$\pi / 2$	0
Spine 2c	z_5	$q_5 + \pi / 2$	0	$\pi / 2$	0
Spine 3a	z_6	$q_6 + \pi / 2$	L_2	$\pi / 2$	0
Spine 3b	z_7	$q_7 + \pi / 2$	0	$\pi / 2$	0
Spine 3c	z_8	$q_8 + \pi / 2$	0	$\pi / 2$	0
Spine 4a	z_9	$q_9 + \pi / 2$	L_3	$\pi / 2$	0
Spine 4b	z_{10}	$q_{10} + \pi / 2$	0	$\pi / 2$	0
Spine 4c	z_{11}	$q_{11} + \pi / 2$	0	$\pi / 2$	0

Table A.5. The DH Table for the Right Arm.

Description	Coordinate System	θ	d	α	a
R clavicle a	z_{12}	$q_{12} - \pi / 2$	L_4	$- \pi / 2$	L_5
R clavicle b	z_{13}	q_{13}	L_{25}	$\pi / 2$	0
R shoulder a	z_{14}	q_{14}	0	$- \pi / 2$	L_6
R shoulder b	z_{15}	q_{15}	0	$\pi / 2$	0
R shoulder c	z_{16}	$q_{16} + \pi / 2$	0	$\pi / 2$	0
R elbow a	z_{17}	$q_{17} + \pi / 2$	L_7	$\pi / 2$	0
R elbow b	z_{18}	q_{18}	0	$- \pi / 2$	0
R wrist a	z_{19}	q_{19}	L_8	$\pi / 2$	0
R wrist b	z_{20}	$q_{20} + \pi / 2$	0	$\pi / 2$	0
R wrist c	z_{21}	q_{21}	0	0	0

Table A.6. The DH Table for the Left Arm.

Description	Coordinate System	θ	d	α	a
L clavicle a	z_{22}	$q_{12} + \pi / 2$	L_4	$- \pi / 2$	L_9
L clavicle b	z_{23}	q_{22}	$-L_{26}$	$- \pi / 2$	0
L shoulder a	z_{24}	q_{23}	0	$\pi / 2$	L_{10}
L shoulder b	z_{25}	q_{24}	0	$- \pi / 2$	0
L shoulder c	z_{26}	$q_{25} - \pi / 2$	0	$\pi / 2$	0
L elbow a	z_{27}	$q_{26} + \pi / 2$	$-L_{11}$	$\pi / 2$	0
L elbow b	z_{28}	q_{27}	0	$- \pi / 2$	0
L wrist a	z_{29}	q_{28}	$-L_{12}$	$\pi / 2$	0
L wrist b	z_{30}	$q_{29} - \pi / 2$	0	$- \pi / 2$	0
L wrist c	z_{31}	q_{30}	0	0	0

Table A.7. The DH Table for the Neck and Head.

Description	Coordinate System	θ	d	α	a
Neck a	z_{32}	$q_{12} + \pi / 2$	L_4+L_{13}	$\pi / 2$	0
Neck b	z_{33}	$q_{31} + \pi / 2$	0	$\pi / 2$	0
Neck c	z_{34}	$q_{32} + \pi / 2$	0	$\pi / 2$	0
Head a	z_{35}	$q_{33} + \pi / 2$	L_{14}	$\pi / 2$	0
Head b	z_{36}	$q_{34} + \pi / 2$	0	$\pi / 2$	0
Head c	z_{37}	q_{35}	0	0	0

Table A.8. The DH Table for the Legs.

Description	Coordinate System	θ	d	α	a
R Hip a	z_{38}	$-\pi/2$	$-L_{16}$	$\pi/2$	L_{15}
R Hip b	z_{39}	$q_{36}+\pi/2$	0	$-\pi/2$	0
R Hip c	z_{40}	$q_{37}-\pi/2$	0	$\pi/2$	0
R knee	z_{41}	q_{38}	L_{17}	$-\pi/2$	0
R ankle a	z_{42}	$q_{39}+\pi/2$	0	0	$-L_{18}$
R ankle b	z_{43}	q_{40}	0	$\pi/2$	0
R foot a	z_{44}	q_{41}	$-L_{19}$	$-\pi/2$	0
R foot b	z_{45}	q_{42}	0	0	0
L Hip a	z_{46}	$\pi/2$	$-L_{21}$	$-\pi/2$	L_{20}
L Hip b	z_{47}	$q_{43}-\pi/2$	0	$-\pi/2$	0
L Hip c	z_{48}	$q_{44}-\pi/2$	0	$\pi/2$	0
L knee	z_{49}	q_{45}	L_{22}	$-\pi/2$	0
L ankle a	z_{50}	$q_{46}+\pi/2$	0	0	$-L_{23}$
L ankle b	z_{51}	q_{47}	0	$\pi/2$	0
L foot a	z_{52}	q_{48}	$-L_{24}$	$-\pi/2$	0
L foot b	z_{53}	q_{49}	0	0	0

Table A.9. Joint Stiffness Coefficients and Neutral Posture for Santos™.

Joint	K (Nm/rad)	$q^{neutral}$ (rad)	Joint	K (Nm/rad)	$q^{neutral}$ (rad)
q_1	10	0	q_{26}	0.7	-2.356
q_2	1	0.349	q_{27}	0.1	-1.606
q_3	1	0	q_{28}	0.1	-0.855
q_4	10	0	q_{29}	0.1	0
q_5	1	0.175	q_{30}	0.1	0
q_6	1	0	q_{31}	1	0
q_7	10	0	q_{32}	1	0.175
q_8	1	0.175	q_{33}	1	0
q_9	1	0	q_{34}	1	0
q_{10}	10	0	q_{35}	1	0
q_{11}	1	0.873	q_{36}	30	0.175
q_{12}	1	0	q_{37}	1	-0.419
q_{13}	30	0	q_{38}	1	-0.384
q_{14}	0.7	-0.524	q_{39}	0.1	0.960
q_{15}	2	0.890	q_{40}	0.1	0.367
q_{16}	2	0.943	q_{41}	0.1	0
q_{17}	0.7	-2.356	q_{42}	5	0
q_{18}	0.1	-1.606	q_{43}	30	-0.175
q_{19}	0.1	-0.855	q_{44}	1	-0.419
q_{20}	0.1	0	q_{45}	1	-0.384
q_{21}	0.1	0	q_{46}	0.1	0.960
q_{22}	30	0	q_{47}	0.1	0.367

Table A.9. Continued

q_{23}	0.7	-0.524	q_{48}	0.1	0
q_{24}	2	0.890	q_{49}	5	0
q_{25}	2	0.943			

Table A.10. Joint Displacement Limits for Santos™.

Joint	Joint Displacement (rad) Upper Limit	Lower Limit	Joint	Joint Displacement (rad) Upper Limit	Lower Limit
q_1	-0.1222	0.1222	q_{26}	-2.2689	0.0000
q_2	-0.1222	0.0873	q_{27}	-2.4435	-0.3491
q_3	-0.1745	0.1745	q_{28}	-1.5708	1.5708
q_4	-0.1222	0.1222	q_{29}	-0.1745	0.5236
q_5	-0.1222	0.0873	q_{30}	-0.7854	0.7854
q_6	-0.1745	0.1745	q_{31}	-0.1745	0.1745
q_7	-0.1222	0.1222	q_{32}	0.0000	1.5708
q_8	-0.1222	0.0873	q_{33}	-1.3963	1.3963
q_9	-0.1745	0.1745	q_{34}	-0.1745	0.1745
q_{10}	-0.1222	0.1222	q_{35}	-0.3491	0.1745
q_{11}	-0.1222	0.0873	q_{36}	-0.3491	1.0472
q_{12}	-0.2618	0.2618	q_{37}	-0.6109	2.0944
q_{13}	-0.6109	0.0000	q_{38}	-0.1745	0.7854
q_{14}	-0.9599	0.0000	q_{39}	0.1745	2.3562
q_{15}	0.0000	2.0944	q_{40}	0.0000	1.0472
q_{16}	-0.2618	1.7453	q_{41}	-0.1745	0.6981
q_{17}	-2.2689	0.0000	q_{42}	-0.7854	0.0000
q_{18}	-2.4435	-0.3491	q_{43}	-0.3491	1.0472
q_{19}	-1.5708	1.5708	q_{44}	-0.6109	2.0944
q_{20}	-0.1745	0.5236	q_{45}	-0.1745	0.7854
q_{21}	-0.7854	0.7854	q_{46}	0.1745	2.3562
q_{22}	-0.6109	0.0000	q_{47}	0.0000	1.0472

Table A.10. Continued

q_{23}	-0.9599	0.0000	q_{48}	-0.1745	0.6981
q_{24}	0.0000	2.0944	q_{49}	-0.7854	0.0000
q_{25}	-0.2618	1.7453			

Table A.11. Joint Torque Limits for Santos™.

Joint	Joint Torque (Nm) Upper Limit	Joint Torque (Nm) Lower Limit	Joint	Joint Torque (Nm) Upper Limit	Joint Torque (Nm) Lower Limit
q_1	-100	100	q_{26}	-60	60
q_2	-100	100	q_{27}	-50	50
q_3	-100	100	q_{28}	-50	50
q_4	-100	100	q_{29}	-10	10
q_5	-100	100	q_{30}	-10	10
q_6	-100	100	q_{31}	-50	50
q_7	-100	100	q_{32}	-50	50
q_8	-100	100	q_{33}	-50	50
q_9	-100	100	q_{34}	-50	50
q_{10}	-100	100	q_{35}	-50	50
q_{11}	-100	100	q_{36}	-100	100
q_{12}	-100	100	q_{37}	-100	100
q_{13}	-80	80	q_{38}	-100	100
q_{14}	-80	80	q_{39}	-80	80
q_{15}	-60	60	q_{40}	-80	80
q_{16}	-60	60	q_{41}	-40	40
q_{17}	-60	60	q_{42}	-40	40
q_{18}	-50	50	q_{43}	-100	100
q_{19}	-50	50	q_{44}	-100	100
q_{20}	-10	10	q_{45}	-100	100
q_{21}	-10	10	q_{46}	-80	80
q_{22}	-80	80	q_{47}	-80	80

Table A.11. Continued

q_{23}	-80	80	q_{48}	-40	40
q_{24}	-60	60	q_{49}	-40	40
q_{25}	-60	60			

REFERENCES

Abdel-Malek, K., Mi, Z., Yang, J., and Nebel, K. (2004a) 'Optimization-Based Layout Design', *4th International Simposium on Robotics and Automation ISRA*, Queretaro, Mexico, 25-27 August 2004.

Abdel-Malek, K., Yang, J., Mi, Z., Patel, V., and Nebel, K. (2004b) 'Human Upper Body Motion Prediction', *Conference on Applied Simulation and Modeling (ASM)*, Rhodes, Greece, 28-30 June 2004.

Abdel-Malek, K., Yu, W., Yang, J., and Nebel, K. (2004c) 'A mathematical method for ergonomic-based design: placement', *International Journal of Industrial Ergonomics*, 34(5), 375-394.

Alexander, R. (1997) 'A minimum energy cost hypothesis for human arm trajectories', *Biological Cybernetics*, 76, 97-105.

Anderson, F., and Pandy, M. (1999) 'A Dyanmic Optimization Solution for Vertical Jumping in Three Dimensions', *Computer Methods in Biomechanics and Biomedical Engineering*, 2, 201-231.

Ayoub M., Woldstad J., Lin C., and Bernard T. (1998) 'A Model to Predict Human Motion During Lifting', *SAE Digital Human Modeling for Design and Engineering Conference*, Dayton, Ohio, 28-29 April 1998.

Case, K.J., Porter, M., and Bonney, M.C. (1990) 'SAMMIE: A man and workplace modeling system', *Computer-Aided Ergonomics*, 31-56.

Chaffin, D., and Andersson, G. (1984) *Occupational biomechanics*, 3rd ed., New York: Wiley-Interscience Publication.

Chaffin, D., Faraway, J., and Zhang, X. (1999) 'Simulating Reach Motions', *SAE Digital Human Modeling for Design and Engineering International Conference*, The Hague, The Netherlands, 18-20 May 1999.

Denavit, J., and Hartenberg, R. S. (1955) 'A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices', *Journal of Applied Mechanics*, 22, 215-221.

Fang, A. and Pollard, N. (2003) 'Efficient Synthesis of Physically Valid Human Motion', *ACM Transactions on Graphics (TOG)*, 22(3), 417-426.

Fu, K.S., Gonzalez, R.C., Lee, C.S.G. (1987) *Robotics: Control, Sensing, Vision, and Intelligence*, New York: McGraw-Hill.

Goel, P.K. (1988) 'The Inverse Kinematics Solution, Close-form Dynamics and Simulation of AdeptOne Industrial Robot', *IEEE International Conference on Robotics and Automation*, Philadelphia, PA, 24-29 April 1988.

Gonzalez, L.J., Maida, J.C., Miles, E.H., Rajulu, S.L., and Pandy, A.K. (2002) 'Work and fatigue characteristics of unsuited and suited humans during isolated isokinetic joint motions', *Ergonomics*, 45(7), 484-500.

Jung, E., and Choe, J. (1996) 'Human reach posture prediction based on psychophysical discomfort', *International Journal of Industrial Ergonomics*, 18, 173-179.

Khang, G. and Zajac, F. (1989a) 'Paraplegic Standing Controlled by Functional Neuromuscular Stimulation: Part I – Computer Model and Control-System Design', *IEEE Transactions on Biomedical Engineering*, 36(9), September, 873-884.

Khang, G. and Zajac, F. (1989b) 'Paraplegic Standing Controlled by Functional Neuromuscular Stimulation: Part II – Computer Simulation Studies', *IEEE Transactions on Biomedical Engineering*, 36(9), September, 885-894.

Kim, J.H. (2004) 'Human Performance Measures', *VSR End of Year Report 2004*, VSR-04.02.

Kim, J., Abdel-Malek, K., Mi, Z., and Nebel, K. (2004) 'Layout Design using an Optimization-Based Human Energy Consumption Formulation', *SAE Digital Human Modeling for Design and Engineering Symposium*, Rochester, MI, June 2004.

Kondo, K. (1991) 'Inverse Kinematics of a Human Arm', *Journal of Robotic Systems*, 8(2), 115-175.

Layon, A., De Lasa, M., Playter, R., Talebi, S., LaFiandra, M., Piscitelle, L., and Obusek, J. (2004) 'Validation of a Load Carriage Simulation', *SAE Digital Human Modeling for Design and Engineering Symposium*, Rochester, MI, June 2004.

Mathiowetz, V., Kashman, N., Volland, G., Weber, K., Dowe, M., and Rogers, S. (1985) 'Grip and Pinch Strength: Normative Data for Adults', *Arch. Phys. Med. Rehabil.*, 66, 69-72

Marler, T. (2004) *A Study of Multi-Objective Optimization Methods for Engineering Applications*, Dissertation (PhD), The University of Iowa.

Marshall, R.N., Wood, G.A., and Jennings, L.S. (1989) 'Performance objectives in human movement: a review and application to the stance phase of normal walking', *Human Movement Science*, 8, 571-594.

McGuan, S. (2001) ‘Human Modeling – From Bubblemen to Skeletons’, *SAE Digital Human Modeling for Design and Engineering Conference*, Arlington, Virginia, 26-28 June 2001.

Mi, Z. (2003) *Task-based Prediction of Upper Body Motion*, Dissertation (PhD.), The University of Iowa.

Mount, F., Whitmore, M., and Stealey, S. (2003) ‘Evaluation of Neutral Body Posture on Shuttle Mission STS-57 (SPACEHAB-1)’, *NASA Technical Memorandum*, 2003-104805, February 2003.

Piegl, L., and Tiller, W. (1995) *The NURBS Book*, Berlin: Springer-Verlag.

Playter, R., Blank, W., Jewell, D., Cornelius, N., Pratt, J., and Obusek, J. (2001) ‘Validating a Physics-Based Simulation of a Load Carriage Task’, *SAE Digital Human Modeling for Design and Engineering Conference*, Arlington, Virginia, 26-28 June 2001.

Playter, R. (2000) ‘Physics-based simulation of running using motion capture’, *Course notes for SIGGRAPH 2000*, ACM SIGGRAPH.

Rostami, M. and Bessonnet, G. (2001) ‘Sagittal gait of a biped robot during the single support phase. Part 2: optimal motion’, *Robotica*, 19, 241-253.

Sciavicco, L. and Siciliano, B. (1996) *Modeling and Control of Robot Manipulators*, New York: McGraw-Hill.

Shabana, A.A. (2001) *Computational Dynamics*, 2nd ed., New York, NY: John Wiley & Sons, Inc.

Soechting, J.F. and Flanders, M. (1989) ‘Errors in pointing are due to approximations in sensorimotor transformations’, *Journal of Neurophysiology*, 62(2), 595-608.

Tolani, D., Goswami, A., and Badler, N. (2000) ‘Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs’, *Graphical Models*, 62, 353-388.

Uno, Y., Kawato, M., and Suzuki, R. (1989) ‘Formation and control of optimum trajectory in human multijoint arm movement: minimum torque-change model’, *Biological Cybernetics*, 61,89-101.

LeVeau B.F. (1992) *Williams and Lissner’s Biomechanics of Human Motion*, 3rd ed., Philadelphia, PA: W.B. Saunders Company.

Witkin, A. and Kass, M. (1988) ‘Spacetime Constraints’, *Computer Graphics*, 22(4), August 1988, 159-168.

Yang, J., Marler, T., Beck, S., Kim, J., Wang, Q., Zhou, X., Pena Pitarch, E., Farrell, K., Patrick, A., Sinokrot, T., Potratz, J., Abdel-Malek, K., Arora, J., and Nebel, K. (2006) 'New Capabilities for the Virtual-Human Santos^{TM}', *SAE 2006 World Congress*, 3-6 April 2006, Cobo Center, Detroit, Michigan (submitted).

Yang, J., Marler, R. T., Kim, H., Arora, J. S., and Abdel-Malek, K. (2004) 'Multi-objective Optimization for Upper Body Posture Prediction', *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, 30-1 August 2004.

Zacher, I. and Bubb, H. (2004) 'Strength Based Discomfort Model of Posture and Movement', *SAE Digital Human Modeling for Design and Engineering Symposium*, Rochester, Michigan, June 2004.

Zhang, X. and Chaffin, D. (2000) 'A three-dimensional dynamic posture prediction model for simulation in-vehicle seated reaching movements: development and validation', *Ergonomics*, 43(9), 1 September, 1314-1330.

Zordan, V. and Hodgins, J. (2002) 'Motion capture-driven simulations that hit and react', *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, San Antonio, Texas, 2002, New York, New York: ACM Press, 89-96.