January 1989

Real Time Inverse Kinematics With Joint Limits and Spatial Constraints

Jianmin Zhao
University of Pennsylvania

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-09.

Real Time Inverse Kinematics With Joint Limits and Spatial Constraints

Abstract
A configuration of an articulated figure of joints and segments can sometimes be specified as spatial constraints. Constrained parts on the articulated figure are abstracted as end effectors, and the counterparts in the space are abstracted as goals. The goal (constraint) can be as simple as a position, an orientation, a weighted combination of position and orientation, a line, a plane, a direction, and so on, or it could be as complicated as a region in the space. An articulated figure consists of various segments connected together by joints. Each joint has some degrees of freedom which are subject to joint limits and manual adjustment. This paper presents an efficient algorithm to adjust the joint angles subject to joint limits so that the set of end effectors concurrently attempt to achieve their respective goals. Users specify end effectors and goals: the program computes a final configuration in *real time* in the sense that actions appear to take no longer than actual physical activities would. If it is impossible to satisfy all the goals owing to the actual constraints, the program should end up with the best possibility according to the users' assignment of importances to each goal.

Comments
University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-09.

REAL TIME INVERSE KINEMATICS WITH JOINT LIMITS AND SPATIAL CONSTRAINTS

Jianmin Zhao and
Norman I. Badler

MS-CIS-89-09
GRAPHICS LAB 27

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

January 1989

Acknowledgements: This research is partially supported by Lockheed Engineering and Management Services, the Pennsylvania Benjamin Franklin Partnership, NSF grants MCS-82-19196-CER, IST-86-12984, DMC-85-16114, IRI84-10413-AO2 and ARO grants DAA29-84-9-0027, DAAG29-84-K-0061 including participation by the U.S. Army Human Engineering Laboratory.

Real Time Inverse Kinematics with Joint Limits and Spatial Constraints

Jianmin Zhao and **Norman I. Badler**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

January 9, 1989

Abstract

A configuration of an articulated figure of joints and segments can sometimes be specified as spatial constraints. Constrained parts on the articulated figure are abstracted as end effectors, and the counterparts in the space are abstracted as goals. The goal (constraint) can be as simple as a position, an orientation, a weighted combination of position and orientation, a line, a plane, a direction, and so on, or it could be as complicated as a region in the space. An articulated figure consists of various segments connected together by joints. Each joint has some degrees of freedom which are subject to joint limits and manual adjustment. This paper presents an efficient algorithm to adjust the joint angles subject to joint limits so that the set of end effectors concurrently attempt to achieve their respective goals. Users specify end effectors and goals: the program computes a final configuration in *real time* in the sense that actions appear to take no longer than actual physical activities would. If it is impossible to satisfy all the goals owing to the actual constraints, the program should end up with the best possibility according to the users' assignment of importances to each goal.

1 Introduction

The ultimate objective of computer animation is to create various motions using computers. Often, we are given motions or goals of some particular points in the figure, and try to solve the whole motion. The specifications on those particular points are usually called constraints.

Badler et al introduced position constraints [1]. They recursively solved for joint angles of articulated figures to satisfy multiple position constraints. But in that paper, orientation constraints and joint limits were not dealt with. Moreover, the sequential nature of the tree traversal often led to realizable but awkward solutions.

Girard and Maciejewski used pseudo-inverse of Jacobian matrix to solve spatial constraints [8]. The main formula is

$$\Delta\theta = J^{+}\Delta\mathbf{r}$$

where $\Delta\theta$ is the increment of the joint angle vector, $\Delta\mathbf{r}$ is the increment of the spatial vector and J^{+} is the pseudo-inverse of the Jacobian $\partial\mathbf{r}/\partial\theta$. If we use a large step size, the method is actually the well known Newton-Raphson method, which is not globally convergent and often needs some special handling (e. g. hybrid method [10]). Or we may use a sufficiently small step size, which was suggested by Girard and Maciejewski in [8]; but this requires excessive iterations. The inverse operation is usually very expensive (say, $O(n^3)$); and they did not deal with joint limits.

Witkin et al used energy constraints [16]. The energy function is the sum of all constraints including position and orientation constraints. Constraints are satisfied if and only if the energy function is zero. Their method is to find the integration of the differential equation:

$$d\theta(t)/dt = -\nabla E(\theta)$$

where θ is the parameter vector, E is the energy function of θ, and ∇ is the gradient operator. The motion is actually driven by the conservative force which serves as the constraint force derived from ∇E, and is smooth in the sense that the parameter vector θ is a smooth function of time t. The energy function may incorporate the constraints on parameter space (joint angle space), but it treats the parameter limits (joint limits) as penalty functions rather than directly. Although the penalty function method is very effective in dealing with general constraints, we have a much more efficient method to deal with linear constraints

Barzel and Barr used dynamic constraints to solve for the motion [2]. In their approach, deviation functions were introduced such that the constraints are met if and only if the deviation function is zero. They solved for constraint forces such that the deviation function decreases exponentially in terms of time t under external forces and constraint forces. This method gives very attractive motion because it considers not only constraint forces but also external forces such

as gravity. The constraint force is not necessarily a conservative force; actually it is derived from the deviation function. Joints can be accomplished by point-to-point constraints with their approach. But they did not consider joint limits.

In Barzel and Barr's method, they use the parameter τ to control the speed (not computational speed) by which the constraint is to be met. This parameter serves as a weight. So to maintain joints as point-to-point constraints, one must assign very small τ to those constraints. But the time step of the computation will then be dominated by those small τ.

The methods of both Witkin et al and Barzel et al solve the problem in θ–t space, and hence are very expensive comparatively. To be more efficient, one often chooses adaptive steps for time t. But the step of t is mainly decided by the spatial improvement. So if we are given a spatial path (in Barzel and Barr's paper [2], the deviation function actually defines a spatial path in terms of time t), we can split the spatial path into sufficiently small segments and get a smooth spatial trajectory. Moreover, often we just want to know some final configurations in an interactive manner, such as making a human figure reach somewhere, look at something, or get into a constrained environment. In human or robot reach space analysis, we may want to know whether or not some spatial constraints are satisfiable in determining the reachable space. If hundreds of reach points are involved, the speed of the algorithm is very crucial. In key frame animation, we create some key frames and let intermediate pictures be interpolated either using function interpolation techniques [15] or motion interpolation techniques using optimal control theory [3, 17]. In many situations, especially in human animation, joint limits are very important. In this paper, joint limits are not special cases but are fundamental to the procedure. An efficient way to deal with joint limits during inverse kinematic positioning is an important concern.

This paper is devoted to solving for spatial constraints subject to joint limits. We solve the problem in θ space (joint angle space or parameter space). It is very fast. For example, a situation with four concurrent goals, involving sixteen degrees of freedom, is achieved in only 2.6 seconds (see Figure 4).

2 The Method

The basic geometric entity being manipulated is the articulated figure. The data structure of the articulated figure we used is created by the Peabody language developed at Computer Graphics Lab

at University of Pennsylvania [4]. A Peabody figure is composed of segments connected together by joints [1]. Each joint has several degrees of freedom subject to joint limits and users' adjustment. Using graph terminology, the data structure of the Peabody figure is a tree, where segments are nodes and joints are edges. An example of human body model is illustrated in Figure 1.

A spatial constraint involves two parts. The constraint parts on the figure are called the end effectors and their counterparts in space are called the goals. In certain contexts, goals and constraints are synonymous, For example, a position goal is satisfied means that a position constraint is satisfied.

Associated with each goal, there is a non negative potential function P such that it is zero if and only if the goal is satisfied. Since we are only concerned about the spatial constraint, and the spatial position and orientation are determined by a point and two vectors (a coordinate frame has three basis vectors, but we can only place two of them in the space), the potential P is, in general, the function of a position and two unit vectors, say,

$$P = P(\mathbf{r}, \mathbf{v}_1, \mathbf{v}_2)$$

where $\mathbf{r}$ is the position vector, and $\mathbf{v}_1, \mathbf{v}_2$ are two unit vectors. Of course, we cannot place two unit vectors arbitrarily. Their angle must be preserved. We call it the potential because it depends only on spatial variables. To form a constraint, we just plug into P the appropriate variables of the end effector which are in turn the functions of the joint angles, i. e. ,

$$P(\theta) = P(\mathbf{r}(\theta), \mathbf{v}_1(\theta), \mathbf{v}_2(\theta)) \tag{1}$$

where θ is the vector of the joint angles. Suppose we have m constraints, then the overall potential is defined as

$$P(\theta) = \sum_{i=1}^{m} w_i P_i(\theta) \tag{2}$$

where w_i are weights put on the ith constraint, and P_i is the potential associated with the ith constraint. Clearly all the constraints are satisfied if and only if P is zero. In general, constraints are not satisfied simultaneously. Our task is to minimize the potential function subject to joint limits. In most cases, joint limits are described by linear equalities or inequalities, such as lower

[1] Actually, Peabody figures are graph-structured rather than being limited strictly to trees. For this discussion, however, the tree structure of the human or robot model suffices.

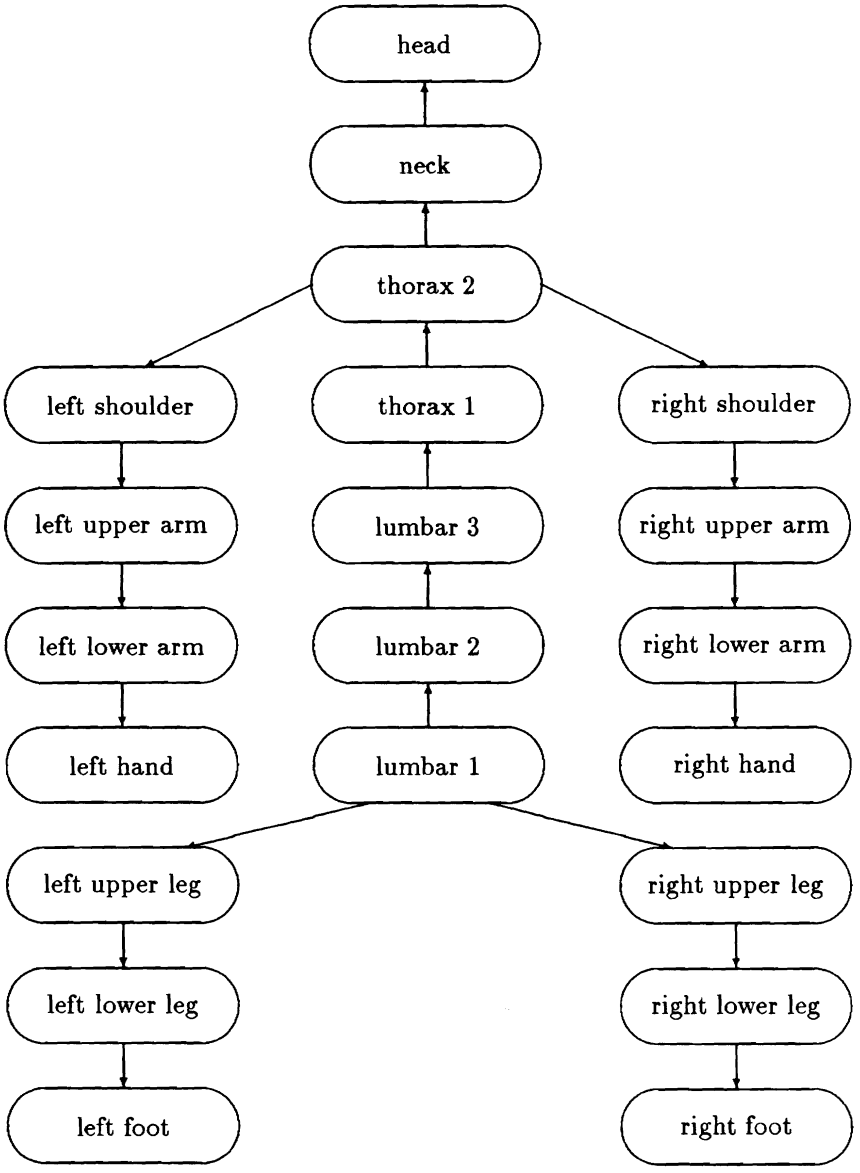

Figure 1: An example of a Peabody human figure model

limits and upper limits. So the technique of nonlinear programming with linear constraints is used. Formally, the problem is

$$\begin{cases} \min P(\theta) \\ s.t. \quad \mathbf{a}_i^T\theta = b_i, i = 1, 2, \ldots, l \\ \qquad \mathbf{a}_i^T\theta \leq b_i, i = l+1, l+2, \ldots, k \end{cases} \tag{3}$$

where $\mathbf{a}_i, i = 1, 2, \ldots, k$ are n-dimensional column vectors. The equality constraints allow for linear relations among the joint angles. The lower limit l_i and upper limit u_i on θ_i contribute to the set of inequality constraints on θ as, respectively:

$$\begin{aligned} -\theta_i &\leq -l_i \\ \theta_i &\leq u_i \end{aligned}$$

The algorithm we adopted to solve this problem is Davidon's variable metric method with the BFGS (Broyden, Fletcher, Goldfarb, Shanno) approximate inverse Hessian matrix update formula (to know its merits, see, e.g. [6, 9, 14]) and Rosen's projection method to handle linear constraints [13, 7]. Under some conditions, the method is superlinear convergent [12] with each iteration of complexity of $O(n^2 + m)$ where n is the total number of joint angles involved, and m is the number of constraints. The method is robust and globally convergent. Of course, generally, it converges to local minima, or rather, Kuhn-Tucker points (constrained stationary points), rather than global minima.

To use the variable metric method, we need to compute the gradient of the objective function $\nabla_\theta P$. From the definition,

$$\nabla_\theta P = \sum_{i=1}^{m} w_i \nabla_\theta P_i \tag{4}$$

The definition of $\nabla_\theta P$ is

$$\nabla_\theta P = \left(\begin{array}{cccc} \frac{\partial P}{\partial \theta_1} & \frac{\partial P}{\partial \theta_2} & \cdots & \frac{\partial P}{\partial \theta_n} \end{array} \right)^T$$

where n is the dimension of the vector θ.

Usually, the number of joint angles involved in the constraint problem n is less than the total number of joint angles in the figure we are dealing with, and the number of joint angles involved in a single constraint is much less than n. So it is both computationally economical and conceptually uniform to treat individual constraints separately and then assemble them together to get $\nabla_\theta P$.

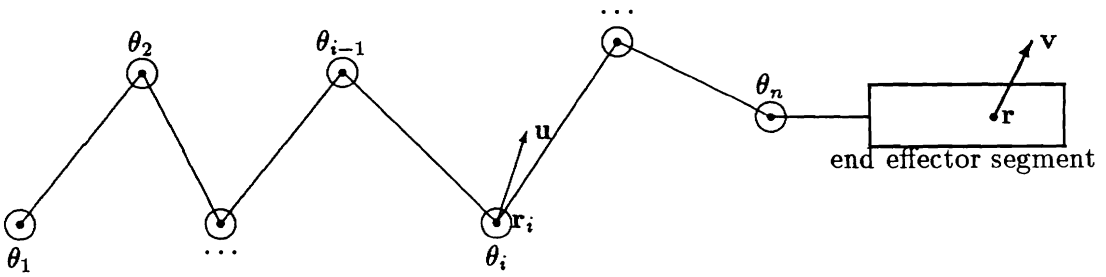

Figure 2: Constraint Chain

3 Single Constraints

Since our data structure of the figure is like a tree (see Figure 1), an end effector of a constraint only depends on those joints which sit along the path from the root of the figure tree to the segment where the end effector belongs [1]. Let's call this path the constraint chain. The constraint chain is illustrated in Figure 2, where a joint with multiple degrees of freedom is separated conceptually into several joints with one degree of freedom. The length of the constraint chain is the total number of joints (or joint angles) along the chain. In Figure 2, it is n. But do not confuse this n with that in the last section. We are only dealing with one constraint in this section. Because all the joints of the human body are revolute joints, we discuss here only revolute joints. But the translational joints can be treated similarly. Let the ith joint angle along the chain be θ_i, the axis of this joint be $\mathbf{u}$ which is a unit vector, the position vector of the end effector be $\mathbf{r}$, the position vector of the ith joint be $\mathbf{r}_i$, and $\mathbf{v}$ be an arbitrary unit vector attached to the end effector segment. $\mathbf{r}$ and $\mathbf{v}$ are functions of the θ_i's. Actually, they can be computed by cascaded multiplication of transformation matrices. It is not hard to see that (see [18])

$$\frac{\partial \mathbf{r}}{\partial \theta_i} = \mathbf{u} \times (\mathbf{r} - \mathbf{r}_i) \tag{5}$$

$$\frac{\partial \mathbf{v}}{\partial \theta_i} = \mathbf{u} \times \mathbf{v} \tag{6}$$

These formulas are useful in deriving the gradient of the potential function.

Let the potential associated with this constraint be $P(\mathbf{r}, \mathbf{v}_1, \mathbf{v}_2)$, and $g(\theta)$ be $\nabla_\theta P$. It is clear

that

$$\begin{aligned} g(\theta) &= \nabla_\theta P \\ &= (\frac{\partial \mathbf{r}}{\partial \theta})^T \nabla_{\mathbf{r}} P + (\frac{\partial \mathbf{v}_1}{\partial \theta})^T \nabla_{\mathbf{v}_1} P + (\frac{\partial \mathbf{v}_2}{\partial \theta})^T \nabla_{\mathbf{v}_2} P \\ &= \begin{pmatrix} \frac{\partial \mathbf{r}}{\partial \theta} \\ \frac{\partial \mathbf{v}_1}{\partial \theta} \\ \frac{\partial \mathbf{v}_2}{\partial \theta} \end{pmatrix}^T \nabla P \end{aligned} \tag{7}$$

where $\frac{\partial \mathbf{r}}{\partial \theta}$ is a 3 by n matrix:

$$\begin{pmatrix} \frac{\partial \mathbf{r}}{\partial \theta_1} & \frac{\partial \mathbf{r}}{\partial \theta_2} & \cdots & \frac{\partial \mathbf{r}}{\partial \theta_n} \end{pmatrix}$$

and similar definition for $\frac{\partial \mathbf{v}_1}{\partial \theta}$ and $\frac{\partial \mathbf{v}_2}{\partial \theta}$, which can be easily computed from (5) and (6), and $\nabla_{\mathbf{r}} P$, $\nabla_{\mathbf{v}_1} P$ and $\nabla_{\mathbf{v}_2} P$ are gradients of P with respect to $\mathbf{r}$, $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively, for example,

$$\nabla_{\mathbf{r}} P = \begin{pmatrix} \frac{\partial P}{\partial \mathbf{r}_x} \\ \frac{\partial P}{\partial \mathbf{r}_y} \\ \frac{\partial P}{\partial \mathbf{r}_z} \end{pmatrix}$$

where $\mathbf{r}_x$ is the x component of the vector $\mathbf{r}$ and similar notation for y and z components, ∇P is the gradient of P with respect to spatial variables, or

$$\nabla P = \begin{pmatrix} \nabla_{\mathbf{r}} P \\ \nabla_{\mathbf{v}_1} P \\ \nabla_{\mathbf{v}_2} P \end{pmatrix}$$

Notice that ∇P is independent of the structure of the articulated figure.

The potential P can be very simple, as will be seen in the following, but, on the other hand, it could be very computationally expensive. For instance, to constrain a portion of the figure to some region, we may create a potential function such that it is zero in the region and enough big outside. But to use our method, we need the gradient of the function. The smooth transition between the two regions often requires some integral, and this integral usually needs comparatively costly numerical treatment. Therefore, in evaluating the cost to compute $P(\theta)$ and $g(\theta)$, we do not count the cost of function P and ∇P. Under this convention and from (5), (6) and (7), we see that $g(\theta)$ is almost as expensive as $P(\theta)$ is, which is dominated by n multiplications of 4 by 4 matrices, or $O(n)$.

Some simple but useful constraints follow.

3.1 Position Goal

The goal is a point in the 3-dimensional space. Let that point be $\mathbf{p}$, and the end effector is also a point which sits on the last segment of the constraint chain. Let this point be $\mathbf{r}$ (see Figure 2). The potential function is:

$$P = (\mathbf{p} - \mathbf{r})^2 \tag{8}$$

and the gradient is:

$$\nabla_{\mathbf{r}} P = 2(\mathbf{r} - \mathbf{p}) \tag{9}$$

$\nabla_{\mathbf{v}_1} P$ and $\nabla_{\mathbf{v}_2} P$ are zero.

3.2 Orientation Goal

The goal is an orthonormal coordinate frame in space. The origin of the frame is irrelevant. Let the goal frame be :

$$\{\mathbf{p}; \mathbf{x}_g, \mathbf{y}_g, \mathbf{z}_g\}$$

where $\mathbf{p}$ is the origin and $\mathbf{x}_g, \mathbf{y}_g, \mathbf{z}_g$ are the orthonormal vectors. Accordingly, the end effector is an orthonormal coordinate frame attached at the last segment of the constraint chain. Let this frame be:

$$\{\mathbf{r}; \mathbf{x}_e, \mathbf{y}_e, \mathbf{z}_e\}$$

The potential function could be:

$$P = (\mathbf{x}_g - \mathbf{x}_e)^2 + (\mathbf{y}_g - \mathbf{y}_e)^2$$

But this function implies that one length unit is as important as about one radian in angle. To enforce one length unit compatible with d degrees in angle, we need to multiply the previous P by c_d such that

$$\frac{1}{c_d} = \frac{2\pi}{360} d$$

i. e. ,

$$c_d = 360/(2\pi d) \tag{10}$$

To be more general, our potential function is, then,

$$P = c_{dx}^2 (\mathbf{x}_g - \mathbf{x}_e)^2 + c_{dy}^2 (\mathbf{y}_g - \mathbf{y}_e)^2 \tag{11}$$

The gradient is:

$$\nabla_{\mathbf{x}_e} P = 2c_{dx}^2(\mathbf{x}_e - \mathbf{x}_g) \tag{12}$$

$$\nabla_{\mathbf{y}_e} P = 2c_{dy}^2(\mathbf{y}_e - \mathbf{y}_g) \tag{13}$$

Some orientation, say y direction, could be suppresed by setting c_{dy} to 0. This is useful, for example, to constrain a person holding a cup of water to keep the cup upward while attaining other constraints.

3.3 Position/Orientation Goals

Position and orientation goals can be treated separately, but sometimes it is convenient to combine them together as a single goal. The goal and end effector are like that in the orientation goal, but the origins of the frames are important here. The potential function for position/orientation goal is:

$$P = w_p(\mathbf{p} - \mathbf{r})^2 + w_o c_{dx}^2(\mathbf{x}_g - \mathbf{x}_e)^2 + w_o c_{dy}^2(\mathbf{y}_g - \mathbf{y}_e)^2 \tag{14}$$

where w_p and w_o are weights put on position and orientation respectively such that

$$w_p + w_o = 1$$

The gradients $\nabla_{\mathbf{r}} P, \nabla_{\mathbf{x}_e} P$ and $\nabla_{\mathbf{y}_e} P$ are obvious from Sections 3.1 and 3.2.

3.4 Direction Goals

The goal is a point in space, say, point $\mathbf{p}$, but the end effector is a vector attached to the end effector segment. Let the starting point of that vector which is fixed on that segment be $\mathbf{r}$, and the vector be $\mathbf{v}$ (see Figure 2). This constraint is to force the vector $\mathbf{v}$ to point toward the point $\mathbf{p}$. This is useful when we want to make a body look in some direction or look at a certain point. The potential function is:

$$P = c_d^2\left(\frac{\mathbf{p} - \mathbf{r}}{\|\mathbf{p} - \mathbf{r}\|} - \mathbf{v}\right)^2 \tag{15}$$

where c_d is defined in (10) and $\|\cdot\|$ is the norm of. The The gradient is:

$$\nabla_{\mathbf{r}} P = 2c_d^2(\|\mathbf{p} - \mathbf{r}\|^2\mathbf{v} - (\mathbf{p} - \mathbf{r})\cdot\mathbf{v}\,(\mathbf{p} - \mathbf{r})) \tag{16}$$

$$\nabla_{\mathbf{v}} P = -2c_d^2\left(\frac{\mathbf{p} - \mathbf{r}}{\|\mathbf{p} - \mathbf{r}\|} - \mathbf{v}\right) \tag{17}$$

3.5 Line Goals

The goal is a line and the end effector is a point $\mathbf{r}$. This constraint forces the point to go to the line. Let the line be defined by point $\mathbf{p}$ and a unit vector ν such that the parametric equation of the line is

$$\mathbf{p} + t\nu$$

The potential function is:

$$P = ((\mathbf{p} - \mathbf{r}) - (\mathbf{p} - \mathbf{r})\cdot\nu\,\nu)^2 \tag{18}$$

and the gradient is:

$$\nabla_{\mathbf{r}} P = 2(\nu\cdot(\mathbf{p} - \mathbf{r})\,\nu - (\mathbf{p} - \mathbf{r})) \tag{19}$$

3.6 Plane Goals

The goal is a plane and the end effector is a point $\mathbf{r}$. This constraint forces the point to go to the plane. Let a point on the plane be $\mathbf{p}$ and the normal of the plane be ν.

The potential function is:

$$P = ((\mathbf{p} - \mathbf{r})\cdot\nu)^2 \tag{20}$$

and the gradient is:

$$\nabla_{\mathbf{r}} P = -2\nu\cdot(\mathbf{p} - \mathbf{r})\,\nu \tag{21}$$

4 Assembly of Local Gradient to Global Gradient

We have dealt with various constraints in Section 3. Of course, the types of constraints are not limited in the Section 3; they are only some examples. The problem now is to assemble all the information about individual constraints to form one constraint.

Suppose we have m constraints. The ith constraint has constraint chain of length n_i and the joint angles

$$\Theta^i = \{\theta_1^i, \theta_2^i, \ldots, \theta_{n_i}^i\} \tag{22}$$

Since constraint chains are from a single figure tree, Θ^i's may overlap with each other. Let

$$\begin{aligned} \Theta &= \bigcup_{i=1}^{m} \Theta^i \\ &= \{\theta_1, \theta_2, \ldots, \theta_n\} \end{aligned} \tag{23}$$

The global index of θ has nothing to do with the topological relation within joint angles, but the local index does. They are numbered from the starting point of the constraint chains to the end effectors. For each constraint, we have a mapping from local index to global index:

$$M^i : \{1, 2, \ldots, n_i\} \longrightarrow \{1, 2, \ldots, n\} \tag{24}$$

such that θ^i_j is $\theta_{M^i(j)}$ in the global index. It is easy to compute $P(\theta)$ from P_i from Equation 2. We just need to take care of $g(\theta) = \nabla_\theta P$. The local gradient for each constraint can be obtained from Section 3. Notice that in that section the derivatives are with respect to local θ_i's contrary to global θ in (4). $g(\theta)$ in (7) is a n_i-dimensional vector while $\nabla_\theta P$ in (4) is a n dimensional vector. Let

$$g^i = (\ g^i_1 \quad g^i_2 \quad \cdots \quad g^i_{n_i}\ )^T$$

be the local gradient of the ith constraint, and

$$g = (\ g_1 \quad g_2 \quad \cdots \quad g_n\ )^T$$

be the global gradient. We can simply assemble g from g^i's as follows:

Step 1. $g_j \leftarrow 0$, for $j = 1, 2, \ldots, n$

Step 2. For $i = 1$ to m do

$g_{M^i(j)} \leftarrow g_{M^i(j)} + w_i g^i_j$, for $j = 1, 2, \ldots, n_i$

4.1 The Algorithm for Nonlinear Programming Problem

We are now ready to solve the problem (3). From Section 3 and Section 4, we can very effectively compute $P(\theta)$ and $g(\theta) = \nabla_\theta P(\theta)$ in $O(n + m)$. There are many algorithms available to solve the problem. Among them, the variable metric method (or conjugate gradient method) is considered most powerful for unconstrained problems with a smooth objective function. Rosen's projection method is very effective in treating linear constraints [13]. Goldfarb combined DFP's method (a variable metric algorithm) [5] with Rosen's projection method [7]. But after that, the variable metric method was much improved. BFGS' improvement has been considered most effective. One of the motives of the improvement is to try to get best conditioning of the approximate inverse Hessian matrix [14]. The algorithm we are presenting here is the combination of the BFGS method and Rosen's projection method. We follow very closely Goldfarb's paper [7].

Without loss of generality, we assume that all the $\mathbf{a}_i$'s in (3) are unit vectors. We say that point θ is feasible if it satisfies all the equalities and inequalities in (3). The ith constraint is said to be active at θ if $\mathbf{a}_i^T\theta = b_i$. So an equality constraint is always active at a feasible point. We assume further that at any point, the $\mathbf{a}_i$'s for active constraints are linearly independent. Let A_q denote a n by q matrix derived from lumping together q vectors from $\mathbf{a}_i$'s, i.e. ,

$$A_q = (\ \mathbf{a}_{i_1}\quad \mathbf{a}_{i_2}\quad \cdots\quad \mathbf{a}_{i_q}\ )$$

In the following description of the algorithm, the superscript i denotes the ith iteration. The algorithm follows.

Step 0. Let θ^0 be a initial feasible point, and H_0^0 a initially chosen n by n positive definite symmetric matrix. Suppose there are q constraints active at point θ^0. A_q is composed of these $\mathbf{a}_i$'s and first l columns of A_q are $\{\mathbf{a}_i : i = 1, 2, \ldots, l\}$. H_q^0 is computed by employing (27) q times. $g^0 = g(\theta^0)$.

Step 1. Given θ^i, g^i and H_q^i, compute $H_q^i g^i$ and

$$\alpha = (A_q^T A_q)^{-1} A_q^T g^i$$

If $H_q^T g^i = 0$ and $\alpha_j \leq 0, j = l+1, l+2, \ldots, q$, then stop. θ^i is a Kuhn-Tucker point.

Step 2. If the algorithm did not terminate at Step 1, either $\|H_q^i g^i\| > \max\{0, \frac{1}{2}\alpha_q a_{qq}^{-1/2}\}$ or $\|H_q^i g^i\| \leq \frac{1}{2}\alpha_q a_{qq}^{-1/2}$, where it is assumed that $\alpha_q a_{qq}^{-1/2} \geq \alpha_i a_{ii}^{-1/2}, i = l+1, \ldots, q-1$ and where a_{ii} is the ith diagonal element of $(A_q^T A_q)^{-1}$. (They are all positive, see [7])

If the former holds, proceed to Step 3.

Otherwise, drop the qth constraint from A_q and obtain H_{q-1}^i from

$$H_{q-1}^i = H_q^i + \frac{P_{q-1}\mathbf{a}_{i_q}\mathbf{a}_{i_q}^T P_{q-1}}{\mathbf{a}_{i_q}^T P_{q-1}\mathbf{a}_{i_q}} \tag{25}$$

where $P_{q-1} = I - A_{q-1}(A_{q-1}^T A_{q-1})^{-1} A_{q-1}^T$ is a projection matrix, $\mathbf{a}_{i_q}$ is the qth column of A_q, and A_{q-1} is the n by $q-1$ matrix got from taking off the qth column from A_q.

Let $q \leftarrow q-1$ and goto Step 1.

Step 3. Let the search direction $s^i = -H_q^i g^i$ and compute

$$\begin{aligned} \lambda_j &= \frac{b_j - \mathbf{a}_j^T \theta^i}{\mathbf{a}_j^T s^i}, j = q+1, q+2, \ldots, k \\ \lambda^i &= \min_j \{\lambda_j > 0\} \end{aligned}$$

Using any line search technique to obtain biggest possible γ^i such that $0 < \gamma^i \leq \min\{1, \lambda^i\}$, and

$$\begin{cases} P(\theta^i + \gamma^i s^i) & \leq P(\theta^i) + \delta_1 \gamma^i (g^i)^T s^i \\ g(\theta^i + \gamma^i s^i)^T s^i & \geq \delta_2 (g^i)^T s^i \end{cases} \tag{26}$$

where δ_1 and δ_2 are positive numbers such that $0 < \delta_1 < \delta_2 < 1$ and $\delta_1 < 0.5$. Let $\theta^{i+1} = \theta^i + \gamma^i s^i$ and $g^{i+1} = g(\theta^{i+1})$.

Step 4. If $\gamma^i = \lambda^i$, add to A_q the $\mathbf{a}_j$ corresponding to the $\min\{\lambda_j\}$ in Step 3. Then compute

$$H_{q+1}^{i+1} = H_q^i - \frac{H_q^i \mathbf{a}_j \mathbf{a}_j^T H_q^i}{\mathbf{a}_j^T H_q^i \mathbf{a}_j} \tag{27}$$

Set $q \leftarrow q+1$ and $i \leftarrow i+1$ and goto Step 1.

Step 5. Otherwise, set $\sigma^i = \gamma^i s^i$ and $y^i = g^{i+1} - g^i$ and update H_q^i as follows:

If $(\sigma^i)^T y^i \geq (y^i)^T H_q^i y^i$ then use the BFGS formula:

$$H_q^{i+1} = H_q^i + ((1 + \frac{(y^i)^T H_q^i y^i}{(\sigma^i)^T y^i}) \sigma^i (\sigma^i)^T - \sigma^i (y^i)^T H_q^i - H_q^i y^i (\sigma^i)^T) / (\sigma^i)^T y^i \tag{28}$$

else use the DFP formula:

$$H_q^{i+1} = H_q^i + \frac{\sigma^i (\sigma^i)^T}{(\sigma^i)^T y^i} - \frac{H_q^i y^i (y^i)^T H_q^i}{(y^i)^T H_q^i y^i} \tag{29}$$

Set $i \leftarrow i+1$ and goto Step 1.

The inexact line search strategy (26) in Step 3 was proposed by Powell [11] and $\delta_1 = 0.0001$ and $\delta_2 = 0.5$ were suggested in [11]. Since s^i is a descent direction, i. e. , $(g^i)^T s^i < 0$, this strategy guarantees that the function value is decreased and $(\sigma^i)^T y^i \geq (1 - \delta_2) |(g^i)^T s^i| > 0$. Because, as we pointed out in Section 3, the gradient $g(\theta)$ is almost as expensive as the function $P(\theta)$, we used cubic Hermite interpolation method in the line search. We feel it is very effective.

The switch between the BFGS formula and the DFP formula was suggested by Fletcher [6].

Notice that all matrix multiplications are performed as a n by n matrix and a vector, or a n by 1 matrix and a 1 by n matrix. For example, matrix multiplication $H_q^i \mathbf{a}_j \mathbf{a}_j^T H_q^i$ can be grouped as $(H_q^i \mathbf{a}_j)(H_q^i \mathbf{a}_j)^T$. The inverse of a matrix might take much time, but, fortunately, for $(A_q^T A_q)^{-1}$, we have a very effective recursive relation of $(A_q^T A_q)^{-1}$ to $(A_{q+1}^T A_{q+1})^{-1}$ and $(A_{q-1}^T A_{q-1})^{-1}$ (see [7] for details). So the complexity of one iteration is $O(n^2)$.

The correctness of the algorithm was proved by Goldfarb in [7] for exact line search in Step 3 and the DFP formula in Step 5. But it is not hard to follow the proof in [7] to show the correctness of our algorithm. Be careful that [7] was for maximum while our algorithm is for minimum. We tried both the BFGS and DFP formula and found that BFGS is really better. Shanno compared them in [14] for many functions, and the results are generally in favor of the BFGS formula.

5 Some Remarks

- We assumed in Section 3 that the constraint chain went from the root of the figure tree to some end effector. It is possible and sometimes useful that the chain goes from a specified joint which is nearer to the root than the end effector is. But then we must take care of those joints which are not in this constraint chain but are in another chain and affect this end effector. We must add those joints to this chain.

- Suppose some joints are active and some joints are inactive, we can add joints to the constraint chain dynamically according to their activeness.

- The obstacle avoidance problem can also be treated here. But we do not look for a path which does not touch the obstacle. Instead, we are concerned about those parts which are pulled by the end effector, since, usually, the end effector is assigned a goal which does not intersect with the obstacle. For example, we may want the hand to get to some place while keeping the elbow away from the obstacle. We can create a potential function around the obstacle and assign this goal to the elbow.

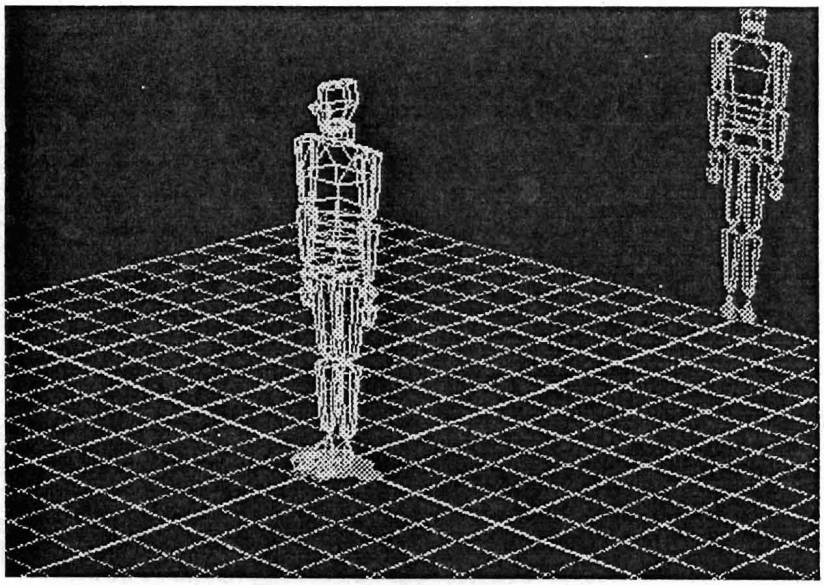

Figure 3: Standing Body

6 Implementation

We have implemented the multiple goal positioning in the Jack interactive environment [4]. By positioning, we mean to satisfy spatial constraints. It is fast enough to be used in an interactive environment. It has been used for simple positioning, or for creating key frames for later interpolation by spline functions or by dynamical simulations.

The examples given here were run using Jack on a Silicon Graphics IRIS 4D/70GT. Figure 3 is an initial configuration. From that position, we use 4 position goals, 2 for elbows and 2 for hands, to get the posture as in Figure 4. Two constraint chains are from shoulders to hands, and another two from shoulders to elbows. It involves 16 degrees of freedom and takes 2.6 seconds.

Figure 5 has two goals for two hands. The goals are on the bar which are not reachable. It involves 31 degrees of freedoms and runs in 13 seconds.

Figure 6 is a person holding a box. To deliver the box to the goal shown on the figure, we used a position/orientation goal to keep the box from tipping upside down. Position and orientation have the same weight. 5 degrees of angle is made as important as 1 unit of length. The result is in Figure 7. It involves 10 degrees of freedom and takes 2 seconds.

In conclusion, this multiple goal achievement algorithm is a significant improvement over other inverse kinematic procedures based on its generality, speed, and fundamental use of joint limits and

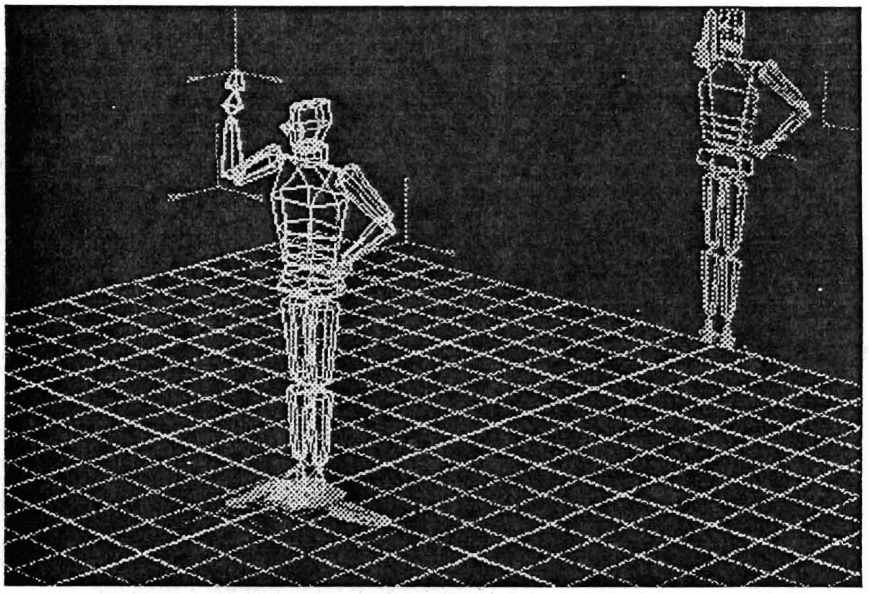

Figure 4: Four position goals: 2.6 seconds

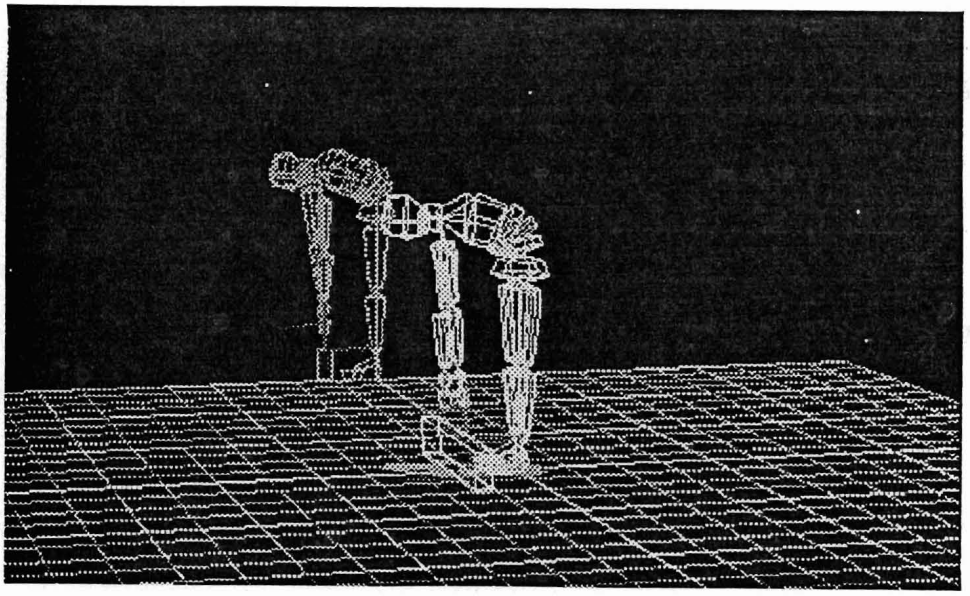

Figure 5: Bending over a bar: 13 seconds

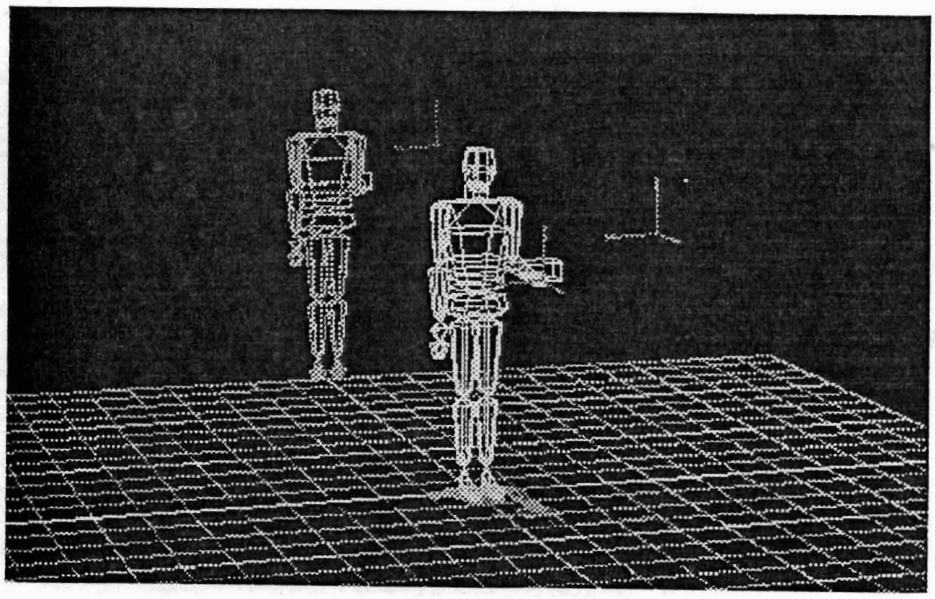

Figure 6: A person holding a box

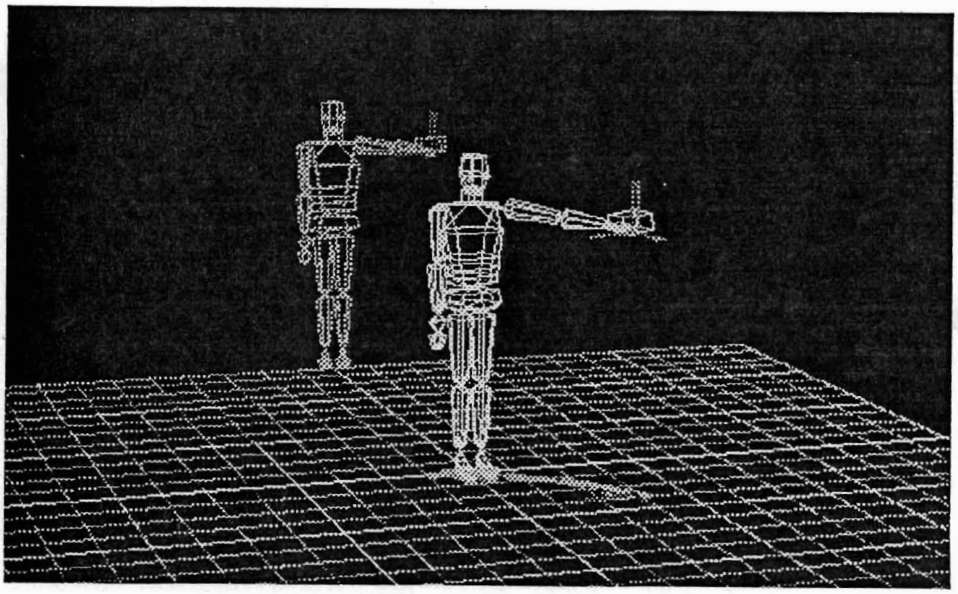

Figure 7: To deliver a box: 2 seconds

spatial constraints. It is a major convenience in the interactive manipulation of articulated figures for positioning, reaching, and viewing tasks.

7 Acknowledgments

This research is partially supported by Lockheed Engineering and Management Services, Pacific Northwest Laboratories B-U0072-A-N, the Pennsylvania Benjamin Franklin Partnership, NASA Grants NAG-2-426 and NGT-50063, NSF CER Grant MCS-82-19196, NSF Grants IST-86-12984 and DMC85-16114, and ARO Grant DAAG29-84-K-0061 including participation by the U.S. Army Human Engineering Laboratory.

References

[1] Norman I. Badler, Kamran H. Manoochehri and Graham Walters, *Articulated Figure Positioning by Multiple Constraints*, IEEE CG&A, June 1987, pp. 28-38

[2] Ronen Barzel and Alan H. Barr, *A Modeling System Based on Dynamic Constraints*, ACM Computer Graphics, **22** (1988), No. 4, pp. 179-188

[3] Lynne Shapiro Brotman and Arun N. Netravali, *Motion Interpolation by Optimal Control*, ACM Computer Graphics, **22** (1988), No. 4, pp. 309-315

[4] Cary B. Phillips and Norman I. Badler, *Jack: A toolkit for manipulating articulated figures*, ACM/SIGGRAPH Symposium on User Interface Software, Banff, Canada, October 1988.

[5] R. Fletcher and M. J. D. Powell, *A Rapidly Convergent Descent Method for Minimization*, Computer J. , **6** (1963), pp. 163-168

[6] R. Fletcher, *A New Approach to Variable Metric Algorithms*, The Computer J. , **13** (1970), pp. 317-322

[7] Donald Goldfarb, *Extension of Davidon's Variable Metric Method to Maximization under Linear Inequality and Equality Constraints*, SIAM J. Appl. Math. **17** (1969), pp. 739-764

[8] Michael Girard and A. A. Maciejewski, *Computational Modeling for the Computer Animation of Legged Figures*, ACM Computer Graphics, **19** (1985), No. 3, pp. 263-270

[9] Donald Goldfarb, *A Family of Variable Metric Methods Derived by Variational Means*, Math. Computation, **24** (1970), pp. 23-26

[10] M. J. D. Powell, *A Hybrid Method for Nonlinear Equations*, in Numerical Methods for Nonlinear Algebraic Equations, Eds. , Philip Rabinowitz, Gordon and Breach Science Publisher, 1970

[11] M. J. D. Powell, *Some Global Convergence Properties of a Variable Metric Algorithm for Minimization without Exact Line Searches*, Nonlinear Programming, SIAM-AMS Proc. Vol. IX (1976), pp. 53-72

[12] Klaus Ritter, *A Variable Metric Method for Linearly Constrained Minimization Problems*, in *Nonlinear Programming 3*, Ed. by O. L. Mangasarian, R. R. Meyer and S. M. Robinson, Academic Press, 1978

[13] J. B. Rosen, *The Gradient Projection Method for Nonlinear Programming, Part* I, *Linear Constraints*, SIAM J. Appl. Math. **8** (1960), pp. 181-217

[14] D. F. Shanno, *Conditioning of Quasi-Newton Methods for Function Minimization*, Math. Computation, **24** (1970), pp. 647-664

[15] S. Steketee and Norman I. Badler, *Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control*, ACM Computer Graphics, **19** (1985), No. 3, pp. 255-262

[16] Andrew Witkin, Kurt Fleischer and Alan Barr, *Energy Constraints on Parameterized Models*, ACM Computer Graphics, **21** (1987), No. 4, pp. 225-232

[17] Andrew Witkin and Michael Kass, *Spacetime Constraints*, ACM Computer Graphics, **22** (1988), No. 4, pp. 159-168

[18] D. E. Whitney, *The Mathematics of Coordinated Control of Prostheses and Manipulators*, J. Dynamic Systems, Measurement, and Control, Transaction ASME, **94** (1972), Series G, pp. 303-309