Graphical Models **62,** 353–388 (2000)
doi:10.1006/gmod.2000.0528, available online at http://www.idealibrary.com on IDEAL®

Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs

Deepak Tolani, Ambarish Goswami, and Norman I. Badler

Computer and Information Science Department, University of Pennsylvania, Philadelphia, Pennsylvania 19104-6389

Received August 6, 1999; accepted May 30, 2000; published online August 15, 2000

In this paper we develop a set of inverse kinematics algorithms suitable for an anthropomorphic arm or leg. We use a combination of analytical and numerical methods to solve generalized inverse kinematics problems including position, orientation, and aiming constraints. Our combination of analytical and numerical methods results in faster and more reliable algorithms than conventional inverse Jacobian and optimization-based techniques. Additionally, unlike conventional numerical algorithms, our methods allow the user to interactively explore all possible solutions using an intuitive set of parameters that define the redundancy of the system. © 2000 Academic Press

Key Words: inverse kinematics; real-time IK; human arm kinematics; analytical algorithms.

1. INTRODUCTION

Inverse kinematics plays a key role in the computer animation and simulation of articulated figures. Often these figures contain more than a hundred degrees of freedom, making it infeasible (or at best tedious) for the animator to manipulate every joint to control the figure's posture. With the assistance of an inverse kinematics algorithm, the animator merely gives the desired location of certain chosen points on the body and relies on the algorithm to automatically compute a set of joint angles that satisfy the end-effector constraints. Another important use of inverse kinematics occurs in motion capture applications where the positions and orientations of sensors on a live subject are used to drive the animation of a computer model. In this case, inverse kinematics is used to find joint angle trajectories that interpolate the sensor data. Finally, inverse kinematics can also be used in task feasibility studies in which a virtual agent and environment are used to simulate the performance of a real-life task, such as an assembly line operation or the workspace analysis of a cockpit. In these applications, inverse kinematics is useful in determining which objects in the environment are reachable.

1524-0703/00 $35.00

Most computer animation systems have adopted inverse kinematics techniques from robotics. In these approaches, an inverse kinematics problem is cast into a system of nonlinear equations or an optimization problem which can be solved using an iterative numerical algorithm. Because most inverse kinematics algorithms were originally designed to meet the requirements of robotics, their straightforward application to computer animation frequently leads to problems. It is instructive to highlight some of the difficulties:

1. In robotics, inverse kinematics tasks only involve constraining the position and orientation of the terminal segment or the end effector. In computer animation, other types of constraints have to be considered. Some examples include constraining selected points on nonterminal segments, aiming the end effector, keeping the figure balanced, and avoiding collisions. It is not always easy to incorporate these constraints into a conventional inverse kinematics formulation. To make matters worse, multiple and possibly conflicting constraints can be simultaneously active, and the system is usually underdetermined or overdetermined.
2. Few robots have more than six joints. On the other hand, a virtual human model may have over 100 degrees of freedom. Traditional inverse kinematics algorithms can break down or become unacceptably slow for the highly redundant systems that occur in computer animation.
3. In most conventional robots, the joints are independent and the joint limits are simple linear inequality constraints. On the other hand, in a human skeleton many of the joints are coupled because they may easily form closed loops or because they move simultaneously when a single muscle contracts. Thus in complex joint systems, the number of degrees of freedom can be less than the number of joint variables. In these cases, it is useful to find ways of parameterizing the kinematics other than with joint variables.
4. In computer graphics applications, there is often a considerable amount of error in both the kinematic model and the end-effector trajectory. For example, in a motion capture application there will be errors in measuring the sensor locations and mismatches between the human subject and the computer model. Similarly, in an interactive application the user may only be interested in specifying a crude estimate of the motion for the end effector and to have the animation system determine a feasible trajectory. Errors are, in general, less critical in computer graphics applications.

We propose an inverse kinematics toolkit satisfying the special needs of computer graphics. Important features of our approach include:

1. Our toolkit generalizes inverse kinematics constraints to include problems such as aiming and partial orientation constraints.
2. Our focus is on analytical methods rather than numerical ones. Analytical methods are generally more efficient and reliable than their numerical counterparts, but require special kinematic structure. Whenever possible we will subdivide the joint structure of the body into kinematic units for which analytical solutions can be derived and partition an inverse kinematics problem into subproblems for each of these units. When a purely analytical solution cannot be obtained we will use a combination of analytical and numerical techniques to achieve the greatest possible speed and reliability.
3. In cases where the inverse kinematics problem is underconstrained, we use an intuitive set of parameters to encode the extra degrees of freedom. The user can interactively adjust the parameters to explore the space of solutions and to choose the solution best suited for the application.

4. Obviously, there is a tradeoff between obtaining an accurate kinematic model of the body and limiting computational expense. One of our goals is to find a suitable balance between these two objectives so that visually acceptable results can be obtained in real time.

The work focuses on 7 DOF, fully revolute, open kinematic chains with two spherical joints connected by a single revolute joint. The primary interest of this work is on the human arm: the revolute joint models the elbow and the two spherical joints model the shoulder and the wrist. The kinematic structure of the human leg is remarkably similar to that of the arm and the same chain may be used to model the leg. In the leg model the spherical joints are the hip and the ankle, and the revolute joint is the knee.[1] We present examples of both arm and leg animations.

This work does not attempt to design a model suitable for simulating dynamics. Unlike, for example, the European CHARM project [15], we are not concerned with the development of a complex biomechanical model for the human arm but only with workspace analysis and interactive posturing applications.

Additionally, this work does not attempt to address the problem of generating "realistic" joint trajectories. Instead, we seek to develop tools that allow a user to investigate all kinematically feasible solutions and to select the solution most suitable for his/her needs. For interactive applications, we provide a set of intuitive parameters for inspecting all possible postures. For applications where an optimization criterion is employed, we provide a way of characterizing the set of solutions using the lowest possible number of variables allowing the user to express an objective function and its derivatives in terms of these variables. The responsibility of generating realistic simulations stays with the animator.

2. TRADITIONAL INVERSE KINEMATICS ALGORITHMS

2.1. Problem Definition

Let $f: q \in \Re^n \rightarrow SE(3)$ represent the forward kinematics map of a kinematic chain. In other words, given the values of n joint variables f returns the position and orientation of the end effector. The inverse kinematics problem can be stated as follows: given $\mathbf{G} \in SE(3)$, find $\mathbf{q} \in \Re^n$ such that $f(\mathbf{q}) = \mathbf{G}$ or determine that no solution is possible. If homogeneous matrices are used the problem assumes the form:

Find $\mathbf{q}$ such that

$$\prod_{i=1}^{n} \mathbf{A}_i(q_i) = \mathbf{G},$$

where

$$\mathbf{A}_i(q_i), \mathbf{G} \in SE(3)$$

$$\mathbf{A}_i(q_i) = \begin{bmatrix} \mathbf{R}(q_i) & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}.$$

Since $\mathbf{G}$ defines six constraints, the problem is well posed only if the number of independent joint variables is equal to 6. If $\dim(\mathbf{q}) < 6$, the problem is overconstrained and in

[1] Strictly speaking, the knee is not revolute: it has a small but nontrivial sliding component. The approximation as a revolute joint will suffice for graphical purposes.

TABLE 1
Number of Analytic Solutions for 6-Degree-of-Freedom Systems

n	Upper bound on solutions
<6	0
>6	∞
6R, 5RP	16
4R2P, 6R with S joint	8
3R3P	2

general no solution is possible. Conversely, if $\dim(\mathbf{q}) > 6$, the problem is said to be underconstrained or to possess *redundant* degrees of freedom. In this case, an infinite number of solutions may exist. In a redundant system, for a fixed posture of the end effector the joints of the robot are still free to move along a constraint surface in joint space. This surface is called the *self-motion manifold*. Redundant systems are useful because the extra degrees of freedom can be used for optimizing a cost function, avoiding collision, or keeping clear from joint limits or Jacobian singularities.

2.1.1. *Theoretical Results*

For 6-degree-of-freedom systems, precise upper bounds on the number of solutions have been established. Let R denote a revolute (rotating) joint and P denote a prismatic (translating) joint. For a general 6R or 5RP manipulator, there are at most 16 possible solutions. For a 4R2P or a 6R manipulator with a spherical joint, the number of possible solutions drops to 8. Finally, a 3R3P system can have at most 2 solutions. See Table 1 for a summary of available results.

2.1.2. *Taxonomy of Inverse Kinematics Algorithms*

Broadly speaking, inverse kinematics algorithms can be characterized as *analytical* or *numerical*. Analytical methods are said to be *complete* since they find all possible solutions. Analytical methods can be further subdivided into *closed-form* and *algebraic-elimination-based* methods. In a closed-form method, the solution to the joint variables can be directly expressed as a set of closed-form equations. In general, closed-form solutions can only be obtained for 6-degree-of-freedom systems with special kinematic structure. Methods based on algebraic elimination express the joint variables as solutions to a system of multivariable polynomial equations, or alternatively express a single joint variable as the solution to a very-high-degree polynomial and determine the other joint variables using closed-form computations. Since the degree of these polynomials will be greater than 4, algebraic-elimination-based methods still require the use of numerical subroutines. However, since numerical methods exist for solving all the roots of a polynomial equation, the algebraic elimination methods are still classified as analytical in nature.

In contrast to the analytical methods, numerical approaches iteratively converge to a single solution based on an initial guess. In general, analytical methods are preferable to their numerical counterparts because analytical methods yield all solutions and are computationally faster and more reliable. The primary advantage of numerical algorithms is they can be used in cases where the system is ill-posed. There are three popular numerical methods

used in solving inverse kinematics problems. The simplest method is the straightforward application of the Newton–Raphson algorithm for solving systems of nonlinear equations. Alternatively, the inverse kinematics problem can be converted into a differential equation in terms of $\mathbf{q}$ and $\dot{\mathbf{q}}$. Finally, the third category of numerical algorithms is based on recasting the inverse kinematics problem into a nonlinear optimization problem.

2.2. Numerical Algorithms

In the following we list the frequently used numerical inverse kinematics algorithms:

1. Newton–Raphson methods: The solution to an inverse kinematics problem is the roots to the system of nonlinear equations

$$\mathbf{F}(\mathbf{q}) \equiv \mathbf{f}(\mathbf{q}) - \mathbf{g} = \mathbf{0}.$$

Here $\mathbf{f}$ is the forward kinematics map, $\mathbf{g}$ is the desired position and orientation of the end effector, and $\mathbf{q}$ is the joint angle vector. Because Newton–Raphson methods use a first-order approximation to the original equations, convergence can be slow when the equations are highly nonlinear. Moreover, near the vicinity of a singularity the inverse of the Jacobian is ill-conditioned and may cause the algorithm to fail.

2. Pieper's methods: Pieper [17] was one of the first authors to adapt the Newton–Raphson method for solving inverse kinematics equations. Pieper derived two alternative methods with different interpretations of the forward kinematics mapping function $\mathbf{f}(\mathbf{q})$. In the first method, $\mathbf{f}(\mathbf{q})$ is viewed as a homogeneous transformation. In the second method an alternative version of the Newton–Raphson method is used in which the forward kinematics map is viewed as a screw motion instead of a homogeneous transformation.

3. Methods based on differential equations–resolved motion rate control [26]: The joint velocities can then be integrated from 0 to t_f to produce the joint angles corresponding to the solution

$$\mathbf{q}(t_f) = \mathbf{q}_0 + \int_0^{t_f} \dot{\mathbf{q}}(t)\, dt. \tag{1}$$

This technique is sometimes called *resolved motion rate control*. If a fixed value of Δt and a first-order integration scheme are used, this method is virtually identical to the Newton–Raphson method. However, it is possible to utilize more accurate and robust integration techniques. In particular, Cheng and Gupta [4] have proposed a modified predictor–corrector algorithm for performing the joint velocity integration. They have demonstrated that their method is more efficient and stable than the best Newton–Raphson methods.

4. For redundant manipulators, $\mathbf{J}$ is not square, but both the Newton–Raphson and differential-equations-based approaches can be extended to redundant manipulators using $\mathbf{J}^+$, the pseudo-inverse [9], or the weighted pseudo-inverse in place of $\mathbf{J}^{-1}$ [8, 16].

5. Control-theory-based methods [21]: These are based on casting the differential equation into a control problem. Suppose $\mathbf{e} = \mathbf{x}_d - \mathbf{x}$ is the error between the desired and the current position and orientation. Selecting $\dot{\mathbf{q}} = \mathbf{J}_a^+(\mathbf{x}_d + \mathbf{Ke})$ yields the linear system $\dot{\mathbf{e}} + \mathbf{Ke} = \mathbf{0}$. $\mathbf{J}_a$ is the analytical Jacobian with respect to the Euler angles. Suitable values for $\mathbf{K}$ can then be chosen ensure covergence and to weight the units of orientation relative to position.

6. Methods based on $\mathbf{J}^T$ [21]: Interpreting a tiny displacement in the joint vector as a torque and the error vector as a force suggests the update law

$$\dot{\mathbf{q}} = \mathbf{J}^T \mathbf{K} \mathbf{e}.$$

This equation has the physical interpretation of a generalized spring of stiffness constant $\mathbf{K}$ that pulls the end effector toward the goal state. This approach is computationally inexpensive and does not require inversion of the Jacobian. On the other hand, if $\mathbf{Ke}$ lies in $N(\mathbf{J}^T)$, $\dot{\mathbf{q}}$ will be zero and no further progress toward the goal state can be made.

7. Optimization approaches: Inverse kinematics can also be regarded as a nonlinear optimization problem. Let $\mathbf{f}(\mathbf{q})$ denote the forward kinematics map from joint space to Cartesian space, and let $\mathbf{x}_{\text{goal}}$ denote the desired end-effector position. Consider the scalar potential function defined by

$$P(\mathbf{q}) = (\mathbf{f}(\mathbf{q}) - \mathbf{x}_{\text{goal}})^T \cdot (\mathbf{f}(\mathbf{q}) - \mathbf{x}_{\text{goal}}). \tag{2}$$

Clearly the function is nonnegative and has a global minimum for any $\mathbf{q}^*$ that solves the corresponding inverse kinematics problem. Examples of optimization-based approaches in computer graphics include Badler *et al*. [2] and Zhao and Badler [27].

2.3. Other IK Techniques and HAL Chains

In general, closed-form solutions can only be obtained for kinematic chains with special structure. Pieper found closed-form techniques for 6-degree-of-freedom manipulators when any three consecutive joint axes intersect at a common point or any three joints are prismatic. For a completely arbitrary 6-degree-of-freedom manipulator a closed-form solution is not possible and other analytical techniques must be used. Raghavan and Roth [19] developed a method based on dialytic elimination for finding all solutions to an arbitrary 6R mechnism. Their method reduces an inverse kinematics problem to finding the roots of a 16-degree polynomial. The roots of this polynomial correspond to the solution of one of the joint variables. The other variables can be computed by solving simple linear systems. The numerical properties of the algorithm was recently improved by Manocha and Canny [14] who recast the root finding problem into a generalized eigenvalue problem.

There has been significant interest in the research community to study the inverse kinematics of what is called a *human-arm-like* chain or a HAL chain. Korein [12] was one of the first to do a principled exploration of the geometry of human arm. Hollerbach [6] listed the three singularities (the shoulder, the elbow, and the wrist) of a 6R manipulator and investigated how best to add an additional revolute joint to this manipulator. The HAL chain was found to best satisfy the requirement of simplicity of inverse kinematics, and the wrist-partitioning algorithm [7] was used to solve it. The inverse kiematics methods developed by Asano [1] and later by Koga *et al.* [10] and Kondo [11] were motivated by grasping rules. Asano presented two rules, 1DOF hand-redundancy and 2DOF hand-redundancy hand postures, and derived inverse kinematics solutions for the human arm based on the minimum wrist muscle load. Kondo's work is based on the observation that the arm posture parameters are approximately linearly related to the spherical coordinates at the shoulder. The linear mapping model was obtained from biomechanical analysis [22]. The end-effector errors remaining after the linear transformation were corrected by a constrained optimization technique. Building on the earlier work of Hemami [5], Rieseler *et al.* [20] solved some of the prototypical trigonometric equations encountered in the inverse

kinematics of 7 DOF manipulatiors, including the HAL geometry. Wang and Kazerounian [25] obtained explicit symbolic formulation for the Jacobian matrix and its null space for a HAL manipulator. In two recent papers Wang [23] and Wang and Verriest [24] present an inverse kinematics scheme based on minimum joint velocity norm. The scheme incorporates nonlinear joint limits and a detailed three-dimensional description of the shoulder model. More recently Lee and Shin [13a] employed a hybrid numerical analytical approach in their inverse kinematics scheme. The analytical component of the limb IK is similar to ours [22a].

2.4. Goals in *Jack*

Jack [3] permits the user to specify a wide variety of goals for the end effector. Multiple and disjunctive goals are also permitted. Some of these goals and their corresponding potential functions are enumerated below.

1. Position goals: In a position goal, the user wants to position the end effector at $\mathbf{r}_{\text{goal}} \in R^3$ and is unconcerned about the final orientation. This can be achieved by minimizing the potential function

$$P(\mathbf{q}) = \|(\mathbf{r}_{\text{goal}} - \mathbf{r}_e)\|.$$

2. Orientation goals: In an orientation goal, the user is only interested in orienting the end effector so that $\mathbf{x}_e$ and $\mathbf{y}_e$ point in the same direction as $\mathbf{x}_g$, $\mathbf{y}_g$. The corresponding potential function is

$$P(\mathbf{q}) = \|(\mathbf{x}_g - \mathbf{x}_e)\| + \|(\mathbf{y}_g - \mathbf{y}_e)\|.$$

3. Position and orientation goals: A position and orientation goal corresponds to the conventional inverse kinematics problem. A suitable potential function is

$$P(\mathbf{q}) = w_p\|(\mathbf{r}_{\text{goal}} - \mathbf{r}_e)\| + w_o c_{dx}^2\|(\mathbf{x}_g - \mathbf{x}_e)\| + w_o c_{dy}^2\|(\mathbf{y}_g - \mathbf{y}_e)\|,$$

where c_{dx} and c_{dy} are used to scale the units of rotation relative to translation and w_p and w_o are weights that adjust the relative importance of the translation goal with respect to the rotation goal.

4. Aiming at goals: It is sometimes necessary to "aim" a line on the end effector at a point $\mathbf{p}$. Let the line on the end effector be represented by the point $\mathbf{r}_e$ and a vector $\mathbf{v}$ written in terms of $\mathbf{x}_e$, $\mathbf{y}_e$, and $\mathbf{x}_e \times \mathbf{y}_e$. The goal is achieved when $\frac{\mathbf{p}-\mathbf{r}_e}{\|\mathbf{p}-\mathbf{r}_e\|} = \mathbf{v}$ and the corresponding potential function is defined as

$$P(\mathbf{q}) = \left\| \frac{\mathbf{p} - \mathbf{r}_e}{\|\mathbf{p} - \mathbf{r}_e\|} - \mathbf{v} \right\|.$$

5. Plane goal: In a plane goal, the user wants the position of the end effector to lie on a plane specified by a point and a normal vector $(\mathbf{p}, \hat{\mathbf{n}})$. The condition for the point being on the plane is

$$(\mathbf{p} - \mathbf{r}_e) \cdot \hat{\mathbf{n}} = 0,$$

which is captured by the potential function

$$P(\mathbf{q}) = ((\mathbf{p} - \mathbf{r}_e) \cdot \hat{\mathbf{n}})^2.$$

Multiple goals are handled by combining individual goals into a global potential function of the form

$$G(\mathbf{q}) = \sum w_i P_i(\mathbf{q}),$$

where w_i are weight factors. Moreover, goals can also be satisfied disjunctively. In this case the goal function is defined as

$$G(\mathbf{q}) = \min_i \{P_i(\mathbf{q})\}.$$

2.5. Quality Criteria

We will now attempt to define a set of criteria for assessing the strengths and shortcomings of an inverse kinematics algorithm.

- *Efficiency:* Since inverse kinematics must often be performed in real time, efficiency or speed is of paramount concern.
- *Reliability:* An algorithm is reliable if it can consistently find a solution when one exists and if it correctly detects instances of the problem that are unsolvable.
- *Completeness:* Additionally, in some applications it is desirable to find the entire set of solutions; algorithms satisfying this requirement are termed complete.
- *Stability:* Numerical stability refers to the algorithm's robustness when degenerate or ill conditioned cases arise.
- *Generality:* Finally, each algorithm will be evaluated on its generality, that is whether it can be adapted to redundant manipulators or other ill-posed problems where an infinite number of solutions may exist.

Efficiency is difficult to determine because it depends upon the quality of the implementation. Additionally, some algorithms may perform well on certain types of problems and poorly on others. In general, we would expect the Newton–Raphson methods to be the slowest and the analytical algorithms to be the fastest.

Not all of the algorithms are reliable. For example, the Jacobian transpose method and the optimization-based approaches can stop in local minima. Numerical instability can also prevent an algorithm from converging to a solution, and almost all of the numerical routines suffer from poor reliability near singularities of the Jacobian. Analytical methods do not suffer from singularities in the Jacobian, but can be susceptible to numerical problems unless special precautions are taken.

The most significant advantage of numerical algorithms is that they can be generalized to accommodate additional constraints and objective functions, whereas the analytical approaches are restricted to 6-degree-of-freedom systems. Optimization methods in particular provide a convenient framework for incorporating a wide variety of criteria.

3. AN ANALYTICAL ALGORITHM FOR A 7-DOF LIMB

In this section we present the inverse kinematics algorithm for the human arm. The kinematic chain of our interest contains seven joint variables; i.e., it has one redundant degree of freedom. Our algorithm uses the extra of freedom to avoid joint limits or to place the elbow as close as possible to a desired position. Our algorithm is purely analytical

and has no problems with Jacobian singularities or local minima. We also give empirical evidence to demonstrate that our algorithm is more efficient and reliable than numerical approaches.

3.1. Forward Kinematics

Consider the kinematic chain shown in Fig. 1. Let

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{R}_1(\theta_1, \theta_2, \theta_3) & 0 \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

denote the rotation matrix from the proximal to the distal site of S_1 as a function of θ_1, θ_2, and θ_3. Similarly,

$$\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_2(\theta_5, \theta_6, \theta_7) & 0 \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

represents the rotation matrix from the proximal to the distal site of S_2 and

$$\mathbf{T}_y = \begin{bmatrix} \mathbf{R}_y(\theta_4) & 0 \\ 0\ 0\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_4 & 0 & s\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ -s\theta_4 & 0 & c\theta_4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

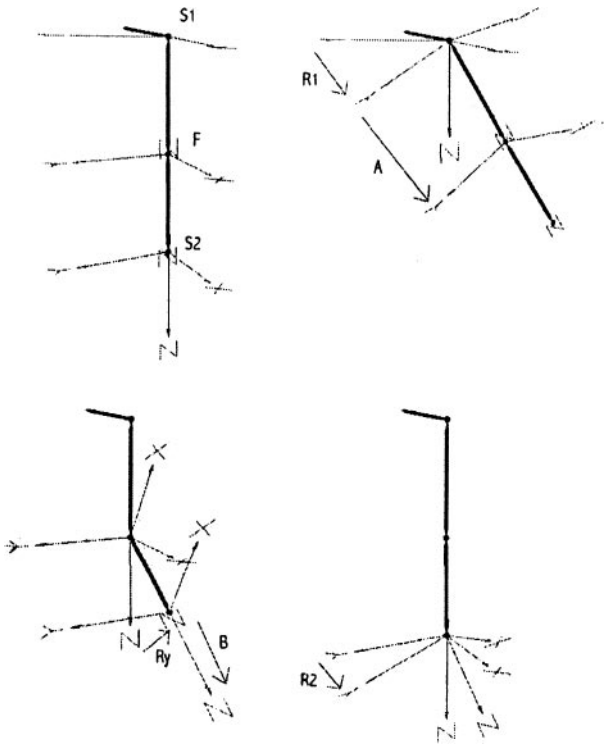

FIG. 1. A 7-degree-of-freedom limb.

is the rotation produced by revolute joint F. Without loss of generality, we have assumed that the coordinate frames about the revolute joint have been defined so that rotation produced by θ_4 is about the local y axis of the proximal frame of F. Finally,

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_a & \mathbf{t}_a \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

and

$$\mathbf{B} = \begin{bmatrix} \mathbf{R}_b & \mathbf{t}_b \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

are the constant transformation matrices from the distal frame of S_1 to the proximal frame of F, and the distal frame of F to the proximal frame of S_2.

3.2. Inverse Kinematics

To solve the inverse kinematics problem for a desired goal

$$\mathbf{G} = \begin{bmatrix} \mathbf{R}_g & \mathbf{t}_g \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

we must solve the equation

$$\mathbf{T}_1\mathbf{A}\mathbf{T}_y\mathbf{B}\mathbf{T}_2 = \mathbf{G} \tag{3}$$

for the unknowns $\mathbf{R}_1$, $\mathbf{R}_2$, and θ_4. The values of θ_1, θ_2, θ_3, θ_5, θ_6, and θ_7 are then determined by extracting the Euler angles from the rotation matrices $\mathbf{R}_1$ and $\mathbf{R}_2$.

3.2.1. *Solving for θ_4*

Since θ_4 is the only joint variable that affects the distance of S_2, relative to S_1, θ_4 may be computed independently. If the normal vector of the plane containing S_1, S_2, and F is parallel to the axis of rotation of F, then θ_4 can be computed trivially using the law of cosines. We consider the more general case where the axis of rotation of the revolute joint is not necessarily parallel to the normal. We first note that the position of S_2 relative to S_1 does not depend on $\mathbf{R}_2$ and is given by

$$\mathbf{T}_1\mathbf{A}\mathbf{T}_y\mathbf{B}\mathbf{T}_2[0, 0, 0, 1]^T = \mathbf{R}_1\mathbf{R}_a\mathbf{R}_y\mathbf{t}_b + \mathbf{R}_1\mathbf{t}_a. \tag{4}$$

Taking the dot product of Eq. (4) with itself and setting it equal to the square of the distance of the goal gives

$$2\mathbf{t}_a^T\,\mathbf{R}_a\mathbf{R}_y\mathbf{t}_b = \mathbf{t}_g^T\mathbf{t}_g - \mathbf{t}_a^T\mathbf{t}_a - \mathbf{t}_b^T\mathbf{t}_b, \tag{5}$$

which is a trigonometric equation of the form $a\cos(\theta_4) + b\sin(\theta_4) = c$ and can be solved using straightforward trigonometric methods. In general there are two solutions to (5), but only one solution is physically realizable because of joint limits on the knee or elbow.

3.2.2. *Characterizing the Extra Degree of Freedom*

Because Eq. (3) has seven unknowns but only six constraints, the system has one degree of redundancy. A simple physical interpretation of the redundant degree of freedom is based on the observation that if the wrist is held fixed, the elbow is still free to swivel about a circular arc whose normal vector is parallel to the axis from the shoulder to the wrist.

The workspace of this mechanism was first systematically analyzed by Korein [12]. Korein observed that the first two shoulder joints along with their joint limits restrict the tip of the elbow to lie on a spherical polygon. By interesecting the elbow circle with the polygon it is possible to determine the legal elbow configurations as a function of the joint limits of the first two joints. Additionally, the twist induced by the third joint also restricts the elbow to lie on a circular arc. It is also possible to determine the restrictions on the elbow position as a function of the wrist joints. By taking the intersection of all sets of valid elbow arcs, Korein derived the restrictions on the elbow position induced by the joint limits.

Our approach to solving this problem is based on the same observation as Korein's. Korein's algorithm is derived from a geometric analysis of the problem. By contrast, our method is purely algebraic and gives an exlicit formula for the joint angles and their derivatives as a function of the swivel angle. This is an advantage when an objective function is used to select an appropriate value of ϕ since it is often necessary to express the objective function in terms of the joint angles.

In Fig. 2, $\mathbf{s}$, $\mathbf{e}$, and $\mathbf{t}_g$ define the positions of the shoulder, the elbow, and the goal location of the wrist. The origin (0, 0, 0) is coincident with the shoulder position. The scalars L_1, L_2, and L_3 denote the lengths of the upper arm, lower arm, and the distance from the goal position to the shoulder. The origin of the coordinate system is taken as $\mathbf{s}$. As the swivel angle ϕ varies, the elbow traces an arc of a circle lying on a plane whose normal is parallel to the wrist-to-shoulder axis. To mathematically describe the circle we first define the normal vector of the plane by the unit vector in the direction from the shoulder to the wrist,

$$\hat{\mathbf{n}} = \frac{\mathbf{t}_g}{\|\mathbf{t}_g\|}.$$

Additionally, we need two unit vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ that form a local coordinate system for the plane containing the circle. We set $\hat{\mathbf{u}}$ to be the projection of an arbitrary axis $\hat{\mathbf{a}}$ (corresponding

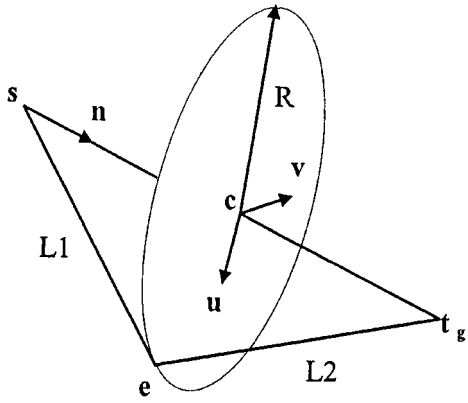

FIG. 2. For a given goal, the elbow is free to move on a circle.

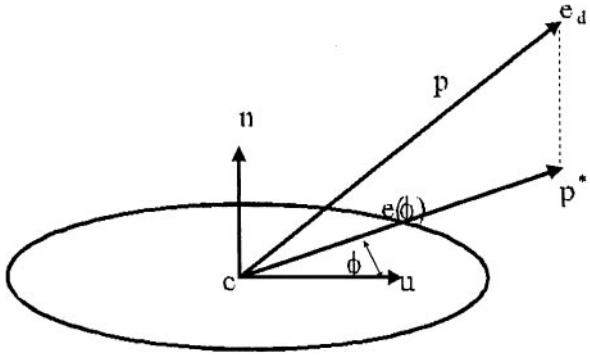

FIG. 3. Finding the elbow position that is the closest possible to a desired position.

to $\phi = 0$), selected by the user, onto the plane of the circle

$$\hat{\mathbf{u}} = \frac{\hat{\mathbf{a}} - (\hat{\mathbf{a}} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}}{\|\hat{\mathbf{a}} - (\hat{\mathbf{a}} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}\|}$$

and set $\hat{\mathbf{v}} = \hat{\mathbf{n}} \times \hat{\mathbf{u}}$. The center of the circle $\mathbf{c}$ and its radius R can be computed from simple trigonometry:

$$\cos(\alpha) = \frac{L_3^2 + L_1^2 - L_2^2}{\|\mathbf{t}_g\| L_1}$$

$$\mathbf{c} = \cos(\alpha) L_1 \hat{\mathbf{n}}$$

$$R = \sin(\alpha) L_1.$$

Finally the equation of the elbow position is given by

$$\mathbf{e}(\phi) = \mathbf{c} + R(\cos(\phi)\hat{\mathbf{u}} + \sin(\phi)\hat{\mathbf{v}}). \tag{6}$$

The swivel angle ϕ gives a useful way of characterizing the space of solutions. The swivel angle has a meaningful physical interpretation to the user and for a given value of ϕ, Eq. (6) can be evaluated to give an additional constraint on the position of the elbow so that the solution to the inverse kinematics problem is uniquely defined.

Sometimes the user wants to give a desired position of the elbow $\mathbf{e}_d$ instead of a swivel angle; in this case, it is a simple matter to compute the corresponding ϕ that minimizes $\|\mathbf{e}_d - \mathbf{e}(\phi)\|$. Define the vectors $\mathbf{p} = (\mathbf{e}_d - \mathbf{c})$ and $\mathbf{p}^* = \mathbf{p} - (\mathbf{p} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}$. Note that $\mathbf{p}^*$ is merely the projection of $\mathbf{p}$ onto the plane containing the elbow circle. As shown in Fig. 3 the value of ϕ that minimizes $\|\mathbf{e}_d - \mathbf{e}(\phi)\|$ is the angle between $\mathbf{p}^*$ and $\hat{\mathbf{u}}$. Since $\sin(\phi) = \frac{\|\mathbf{p}^* \times \hat{\mathbf{u}}\|}{\|\mathbf{p}^*\|}$ and $\cos(\phi) = \frac{\mathbf{p}^* \cdot \hat{\mathbf{u}}}{\|\mathbf{p}^*\|}$, $\phi = a\tan 2(\|\mathbf{p}^* \times \hat{\mathbf{u}}\|, \mathbf{p}^* \cdot \hat{\mathbf{u}})$.

3.2.3. *Solving for* $\mathbf{R}_1$ *and* $\mathbf{R}_2$

In this section we show how to compute $\mathbf{R}_1$ and $\mathbf{R}_2$ efficiently given the values of ϕ and θ_4. Figure 4 shows the initial configuration of the arm corresponding to an elbow rotation of θ_4 and the goal configuration for a desired swivel angle ϕ. As before, we have taken the position of the shoulder joint $\mathbf{s}$ as the origin. $\mathbf{e}$, $\mathbf{e}_g(\phi)$, $\mathbf{w}$, $\mathbf{w}_g$ denote the positions of

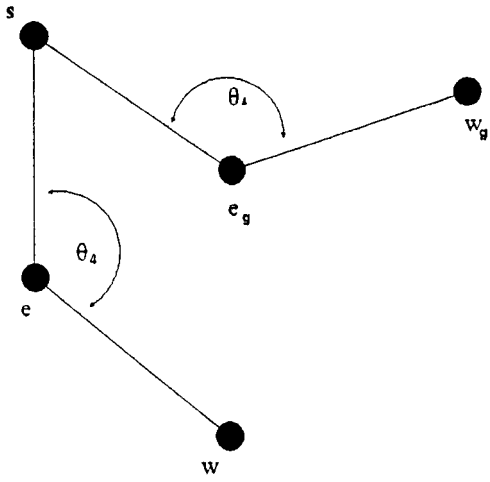

FIG. 4. Calculating R_1.

the elbow and wrist in the rest and goal configurations. Assuming that the elbow is not completely outstretched, the shoulder, elbow, and wrist locations form a reference triangle. The rotation matrix $\mathbf{R}_1$ is merely the rigid body transformation of $\triangle\mathbf{sew}$ onto $\triangle\mathbf{se}_{\mathrm{g}}(\phi)\mathbf{w}_{\mathrm{g}}$. To find $\mathbf{R}_1$ we first define a local coordinate system $\left[\begin{smallmatrix}\hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} & \mathbf{0}\\ 0 & 0 & 0 & 1\end{smallmatrix}\right]$ associated with $\triangle\mathbf{sew}$ by

$$\hat{\mathbf{x}} = \frac{\mathbf{e}}{\|\mathbf{e}\|}$$

$$\hat{\mathbf{y}} = \frac{\mathbf{w} - (\mathbf{w}\cdot\hat{\mathbf{x}})\hat{\mathbf{x}}}{\|\mathbf{w} - (\mathbf{w}\cdot\hat{\mathbf{x}})\hat{\mathbf{x}}\|}$$

$$\hat{\mathbf{z}} = \hat{\mathbf{x}} \times \hat{\mathbf{y}}.$$

Analogously we define the coordinate system $\left[\begin{smallmatrix}\hat{\mathbf{x}}_{\mathrm{g}}(\phi) & \hat{\mathbf{y}}_{\mathrm{g}}(\phi) & \hat{\mathbf{z}}_{\mathrm{g}}(\phi) & \mathbf{0}\\ 0 & 0 & 0 & 1\end{smallmatrix}\right]$ associated with $\triangle\mathbf{se}_{\mathrm{g}}(\phi)\mathbf{w}_{\mathrm{g}}$. $\mathbf{R}_1$ is then given by

$$\mathbf{R}_1 = \begin{bmatrix} \hat{\mathbf{x}}_{\mathrm{g}}(\phi) & \hat{\mathbf{y}}_{\mathrm{g}}(\phi) & \hat{\mathbf{z}}_{\mathrm{g}}(\phi) & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix}^T. \tag{7}$$

Finally, $\mathbf{R}_2$ is obtained by rearranging Eq. (3)

$$\mathbf{T}_2 = (\mathbf{T}_1\mathbf{A}\mathbf{T}_y\mathbf{B})^{-1}\mathbf{G}$$

$$\mathbf{R}_2 = (\mathbf{R}_1\mathbf{R}_{\mathrm{a}}\mathbf{R}_y\mathbf{R}_{\mathrm{b}})^{-1}\mathbf{R}_{\mathrm{g}}.$$

It is straightforward to extend the algorithm to problems that involve only position constraints. In this case the wrist angles are chosen by the user and only $\mathbf{R}_1$ and θ_4 are computed.

3.2.4. *Joint Limits*

In practice, it is necessary to consider joint limits to ensure plausible looking solutions. In this section, we describe an analytical algorithm that computes all possible values of ϕ that satisfy the joint limits. The algorithm also determines when a solution is not possible.

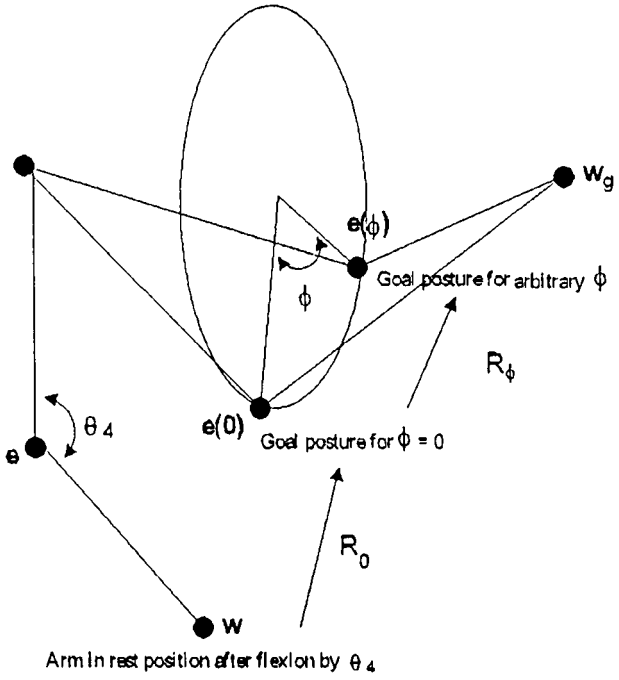

FIG. 5. Decomposing $\mathbf{R}_1$ into $\mathbf{R}_0$ and $\mathbf{R}_\phi$.

We will first derive the relationship between the joint variables $\theta_1, \theta_2, \theta_3, \theta_5, \theta_6, \theta_7$ and the swivel angle ϕ and show how this relationship can be used to determine the valid ranges of ϕ for each joint.

Computing the valid ranges for $\theta_1, \theta_2, \theta_3$. As shown in Fig. 5 $\mathbf{R}_1$ can be expressed as a sequence of two rotations $\mathbf{R}_1 = \mathbf{R}_0\mathbf{R}_\phi$, where $\mathbf{R}_0$ is the rotation matrix that moves the elbow to $\mathbf{e}(0)$ from the rest position by setting $\phi = 0$ in Eq. (7) and $\mathbf{R}_\phi$ is the rotation matrix for a rotation of θ about the swivel axis vector $\hat{\mathbf{n}}$. The joint variables $\theta_1, \theta_2, \theta_3$ are the Euler angles of $\mathbf{R}_1$. Irrespective of the Euler angle convention chosen, expanding the equation $\mathbf{R}_1 = \mathbf{R}_0\mathbf{R}_\phi$ into its scalar components yields one of two possible systems of equations of the form

$$\begin{aligned}
\sin(\theta_i) &= f_1(\phi)\\
\cos(\theta_i)\cos(\theta_j) &= f_2(\phi)\\
\cos(\theta_i)\sin(\theta_j) &= f_3(\phi)\\
\cos(\theta_i)\cos(\theta_k) &= f_4(\phi)\\
\cos(\theta_i)\sin(\theta_k) &= f_5(\phi)
\end{aligned}$$

or

$$\begin{aligned}
\cos(\theta_i) &= f_1(\phi)\\
\sin(\theta_i)\cos(\theta_j) &= f_2(\phi)\\
\sin(\theta_i)\sin(\theta_j) &= f_3(\phi)\\
\sin(\theta_i)\cos(\theta_k) &= f_4(\phi)\\
\sin(\theta_i)\sin(\theta_k) &= f_5(\phi),
\end{aligned}$$

where i, j, and k are any permutation of the sequence $\{1, 2, 3\}$, $f_l = \alpha_l \sin(\phi) + \beta_l \cos(\phi) + \gamma_l$, and α_l, β_l, and γ_l are constants that depend upon $\mathbf{R}_0$. Without loss of generality, we will assume that we have a set of equations of the form

$$\begin{aligned}
\sin(\theta_1) &= f_1(\phi) \\
\cos(\theta_1)\cos(\theta_2) &= f_2(\phi) \\
\cos(\theta_1)\sin(\theta_2) &= f_3(\phi) \\
\cos(\theta_1)\cos(\theta_3) &= f_4(\phi) \\
\cos(\theta_1)\sin(\theta_3) &= f_5(\phi).
\end{aligned}$$

The other cases can be handled using a straightforward generalization of the techniques that we are about to describe. We first consider how to determine the valid ranges of ϕ that satisfy the joint limits for θ_1. There are two families of solutions for θ_1 of the equation $\sin(\theta_1) = k$ ($|k| \leq 1$). Family one corresponds to values of θ_1 that lie in the range $(-\frac{\pi}{2}, \frac{\pi}{2})$ and family two corresponds to values that lie in the range $(\frac{\pi}{2}, \frac{3\pi}{2})$. To distinguish between these two families we will use the notation θ_{1_1} to indicate a value of θ_1 that belongs to the first family and θ_{1_2} to denote a value of θ_1 in the second family. Obviously, within each family the relationship between $\sin(\theta_1)$ and θ_1 is monotonic. Given a set of joint limits $\theta_{1\,\min} < \theta < \theta_{1\,\max}$, we can find a corresponding set of valid joint limit ranges for each family. There will be at most two such ranges. For example, if $\theta_{\min} = \frac{\pi}{4}$ and $\theta_{\max} = \frac{7\pi}{4}$, then any value of θ in family two satisfies the joint limits, but solutions in family one are restricted to lie in the interval $(\frac{\pi}{4}, \frac{\pi}{2})$ or in the interval $(\frac{3\pi}{2}, \frac{7\pi}{4})$ as illustrated in Fig. 6.

Suppose (a, b) is a valid range for θ_1 in family i. Because the relationship between θ_{1_i} and $\sin(\theta_{1_i})$ is monotonic in a given family i, if $\theta_{1_i} \in (a, b)$, then we must have

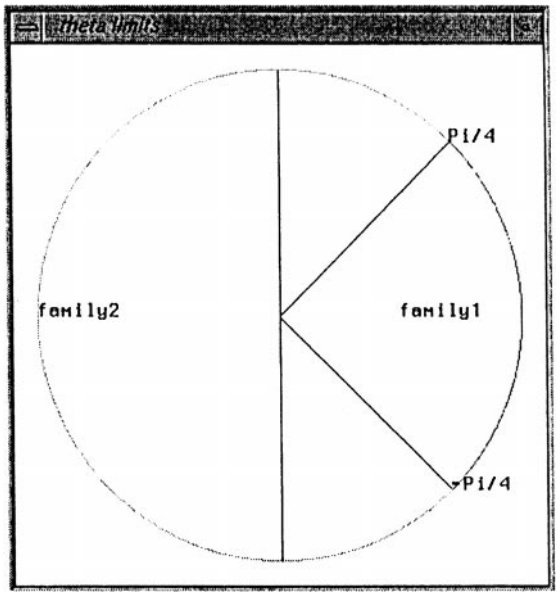

FIG. 6. Joint limits on θ_1.

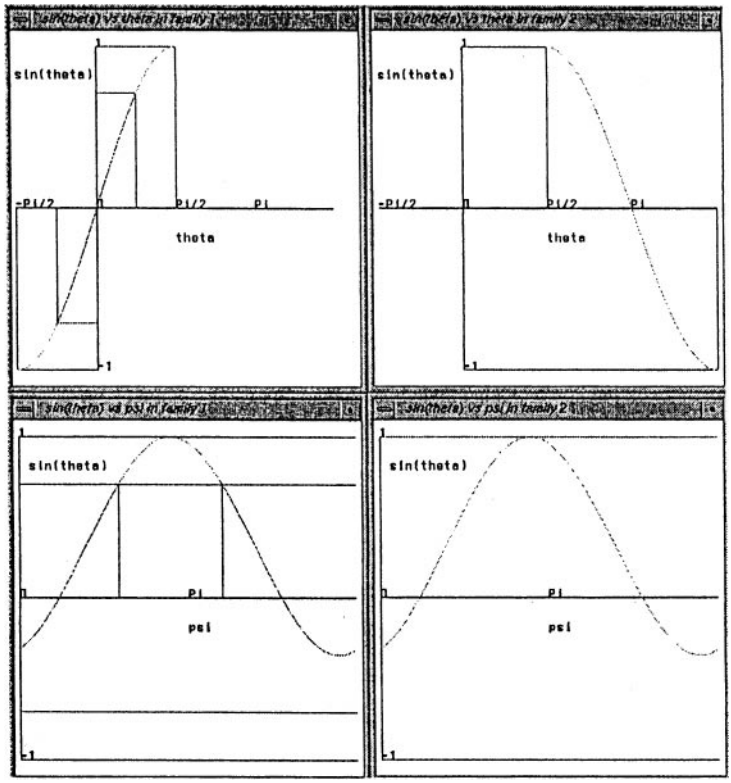

FIG. 7. Finding the values of ϕ that satisfy the joint limits of θ_1.

$\sin(\theta_{1_i}) \in (\sin(a), \sin(b))$. From Eq. (8), the relationship between $\sin(\theta_1)$ and ϕ is given by the equation

$$\sin(\theta_1) = \alpha_1 \sin(\phi) + \beta_1 \cos(\phi) + \gamma_1.$$

As shown in Fig. 7, to find the corresponding ranges of ϕ that yield values of θ_1 in the range (a, b) we intersect the $\alpha_1 \sin(\phi) + \beta_1 \cos(\phi) + \gamma_1$ curve with the straight line segments $\sin(\theta) = a$ and $\sin(\theta) = b$ and determine if the sections of the curve lying between two consecutive intersection points are in the range $(\sin(a), \sin(b))$. This test can be accomplished by checking the sign of the derivative of the curve to see if the function is increasing or decreasing.

The valid ranges of ϕ for θ_1 are stored in two sets

$$\Pi_{1_1} = \{(a, b) \mid \theta_{1\min} < \theta_{1_1}(\phi) < \theta_{1\max}, a < \phi < b\}$$
$$\Pi_{1_2} = \{(a, b) \mid \theta_{1\min} < \theta_{1_2}(\phi) < \theta_{1\max}, a < \phi < b\}.$$

As with θ_1, there are two families of solutions for θ_2 corresponding to whether θ_1 is in the range $(-\frac{\pi}{2}, \frac{\pi}{2})$ or $(\frac{\pi}{2}, \frac{3\pi}{2})$.

$$\theta_{2_1}(\phi) = a \tan 2(f_3, f_2) \qquad -\frac{\pi}{2} < \theta_1 < \frac{\pi}{2}$$
$$\theta_{2_2}(\phi) = a \tan 2(-f_3, -f_2) \qquad \frac{\pi}{2} < \theta_1 < \frac{3\pi}{2}.$$

To determine the valid set of ϕ such that θ_2 lies within the joint limits, we compute the intersections of the θ_{2_i} curves with $\theta_{2\,\min}$ and $\theta_{2\,\max}$ and use these intersection points to partition the curves into piecewise components. If a component lies within $\theta_{2\,\min}$ and $\theta_{2\,\max}$, then the corresponding ϕ interval is valid. To determine the intersections of the θ_{2_1} and θ_{2_2} curves with $\theta_{2\,\min}$ and $\theta_{2\,\max}$ we note that since $\tan(\alpha) = \tan(\alpha + \pi)$ the solution of ϕ for the equation

$$\frac{f_2(\phi)}{f_3(\phi)} = \tan(\alpha) \tag{8}$$

implies that either

$$a\tan 2(f_2,\ f_3) = \alpha$$

or

$$a\tan 2(f_2,\ f_3) = \alpha + \pi \Rightarrow a\tan 2(-f_2, -f_3) = \alpha.$$

Thus to compute the desired intersection points, we only need solve Eq. (8) with $\theta_{2\,\min}$ and $\theta_{2\,\max}$ substituted for α. We then determine if the intersection point corresponds to family one or family two by checking if $\theta_{2_1}(\phi) = \alpha$.

The algorithm for determining the valid ranges of ϕ proceeds as follows. For each curve $\theta_{2_i(\phi)}$ compute the associated intersection points and store them into a sorted sequence along with the end points 0 and 2π. For each set of adjacent intersection points (ϕ_j, ϕ_{j+1}) in the sequence determine if the corresponding curve segment $\theta_{2_i}(\phi)_{\phi_j<\phi<\phi_{j+1}}$ lies within the range $\theta_{2\,\min}$ and $\theta_{2\,\max}$. This test can be accomplished by checking the derivative of θ_{2_i} or by evaluating the curve at a randomly chosen point inside the interval (ϕ_i, ϕ_{i+1}). Also, because an angle can wrap around from 0 to 2π the algorithm merges two intervals of the form $(0, \phi_i)$ and $(\phi_i, 2\pi)$ into a single interval. As with θ_1, the valid intervals are stored in two sets Π_{2_1} and Π_{2_2}. The analysis of θ_3 is identical to that of θ_2.

Computing the valid joint ranges for θ_5, θ_6, θ_7. Recalling $\mathbf{R}_2 = (\mathbf{R}_1\mathbf{R}_a\mathbf{R}_y\mathbf{R}_b)^{-1}\mathbf{R}_g$ and substituting $\mathbf{R}_1 = \mathbf{R}_0\mathbf{R}_\phi$ gives us a way of expressing the Euler angles of $\mathbf{R}_2$ as a function of ϕ. The valid ranges of θ_5, θ_6, and θ_7 can be determined using the same techniques described in the previous section.

Selecting a suitable value of ϕ. After we apply the procedure described above, there will be 12 intervals $\Pi_{i_j\ i=1,2,3,5,6,7;j=1,2}$. Let A_1 and A_2 represent the set of valid ϕ intervals for families 1 and 2 that satisfy the joint limits for θ_1, θ_2, and θ_3:

$$A_1 = \bigcap_{i=1}^{3} \Pi_{i_1}$$

$$A_2 = \bigcap_{i=1}^{3} \Pi_{i_2}.$$

Analogously, define B_1 and B_2 for joints θ_5, θ_6, and θ_7:

$$B_1 = \bigcap_{i=5}^{7} \Pi_{i_1}$$

$$B_2 = \bigcap_{i=5}^{7} \Pi_{i_2}.$$

V, the set of valid psi intervals that satisfy all six sets of joint limits, is given by

$$V = (A_1 \cap B_1) \cup (A_1 \cap B_2) \cup (A_2 \cap B_1) \cup (A_2 \cap B_2).$$

Given V there are at least two useful ways of selecting an appropriate value of ϕ. The first method is to find the largest interval in V and to select its midpoint. This will tend to keep the arm in a posture that maximizes clearance from the joint limits. Another possibility is to choose the value of ϕ that lies in a valid interval and that is closest to a desired swivel angle ϕ_d.

More generally, we can find the value of ϕ that minimizes an arbitrary objective function $f(\theta_1, \ldots, \theta_7)$. In order to find a minimizing value for f, we need to solve $\frac{d}{d\phi} f = 0$. Applying the chain rule gives

$$\frac{d}{d\phi} f(\theta_1(\phi), \ldots, \theta_7(\phi)) = \left(\frac{d\theta_1}{d\phi}, \ldots, \frac{d\theta_7}{d\phi}\right) \nabla_\theta f.$$

If the relationship between θ and ϕ is of the form $\sin(\theta) = e(\phi)$, where $e(\phi) = \alpha \cos(\phi) + \beta \sin(\phi) + \gamma$, the derivatives $\frac{d\theta}{d\phi}$ can be computed as

$$\frac{d}{d\phi} \sin(\theta) = \frac{d}{d\phi} e(\phi)$$

$$\cos(\theta) \frac{d\theta}{d\phi} = e'(\phi)$$

$$\frac{d\theta}{d\phi} = \frac{e'(\phi)}{\cos(\theta)} = \pm \frac{e'(\phi)}{\sqrt{1 - \sin^2(\theta)}} = \pm \frac{e'(\phi)}{\sqrt{1 - e^2(\phi)}}.$$

For family 1,

$$-\frac{\pi}{2} < \theta < \frac{\pi}{2} \Rightarrow \cos(\theta) > 0 \Rightarrow \frac{d\theta}{d\phi} = \frac{e'(\phi)}{\sqrt{1 - e^2(\phi)}}.$$

For family 2,

$$\cos(\theta) < 0 \Rightarrow \frac{d\theta}{d\phi} = -\frac{e'(\phi)}{\sqrt{1 - e^2(\phi)}}.$$

If the relationship between θ and ϕ is of the form

$$\begin{aligned} \sin(\theta)\cos(\psi) &= e_1(\phi) \\ \cos(\theta)\cos(\psi) &= e_2(\phi) \\ \sin(\psi) &= e_3(\phi) \\ e_i(\phi) &= \alpha_i \cos(\phi) + \beta_i \sin(\phi) + \gamma_i, \end{aligned}$$

where ψ is another joint variable, the derivative $\frac{d\theta}{d\phi}$ is slightly more complicated. We first note that this system of equations implies that

$$\theta = \tan^{-1}\left(\frac{e_1(\phi)}{e_2(\phi)}\right) \quad \text{or} \quad \theta = \tan^{-1}\left(\frac{e_1(\phi)}{e_2(\phi)}\right) + \pi.$$

Taking the derivative of either equation gives

$$\frac{d\theta}{d\phi} = \frac{e_1' e_2 - e_1 e_2'}{e_1^2 + e_2^2} = \frac{e_1' e_2 - e_1 e_2'}{\cos^2(\psi)} = \frac{e_1' e_2 - e_1 e_2'}{\sqrt{1 - e_3^2}}.$$

Note that in this case the derivative does not depend upon what family θ belongs to.

Since f can be arbitrarily complex, there will in general be no closed-form solution to $\frac{d}{d\phi} f = 0$, and a numerical method must be used instead. However, this approach still has a number of significant advantages compared to the approach of minimizing with respect to the joint variables $\theta_1, \ldots, \theta_7$ as in Zhao and Badler [27], for example. In traditional optimization methods, there are two ways of solving for the minimum of f while satisfying the inverse kinematics and joint limit constraints. In the approach favored in [27], the problem is cast as

$$\min(w_1 f(\theta_1, \ldots, \theta_7) + w_2(\|\mathbf{g}(\theta_1, \ldots, \theta_7) - \mathbf{g}_\mathrm{d}\|^2)$$

subject to

$$\theta_{i\,\min} < \theta_i < \theta_{i\,\max}; \quad i = 1 \ldots 7, \tag{9}$$

where $\mathbf{g}(\theta_1, \ldots, \theta_7)$ is the forward kinematics mapping function, $\mathbf{g}_\mathrm{d}$ is the desired end-effector position and orientation, and w_1 and w_2 are weights that rank the importance of minimizing the objective function relative to satisfying the inverse kinematics constraint. In this formulation, the optimization problem is in a seven-dimensional space and because the inverse kinematics task is merely part of an augmented objective function there is no guarantee that the inverse kinematics constraint will be satisfied. Another traditional approach is to pose the problem as

$$\min f(\theta_1, \ldots, \theta_7)$$

subject to

$$\mathbf{g}(\theta_1, \ldots, \theta_7) - \mathbf{g}_\mathrm{d} = \mathbf{0}$$
$$\theta_{i\,\min} < \theta_i < \theta_{i\,\max}; \quad i = 1 \ldots 7.$$

A minimum of the problem above is guaranteed to solve the inverse kinematics problem, but now the constraints are nonlinear, which makes the optimization problem very difficult to solve.

In contrast, our method is a hybrid of analytical and numerical techniques. The analytical phase is used to simplify the dimension of the problem to a single variable ϕ and to establish the feasible set of solutions by finding linear inequality limits on ϕ. A numerical method can then be used to find the solution to the one-dimensional optimization problem of minimizing $f(\phi)$ instead of a seven-dimensional function $f(\theta_1, \ldots, \theta_7)$. Unlike the case of multivariate optimization, fast and reliable techniques exist for finding the minimum of a function of one variable.

Unsolvable problems. Sometimes it is not possible to find a value of ϕ that satisfies the limits of all the joint variables. In these cases, it is often useful to give the user an

approximate solution that satisfies the joint limits. The optimization method used by Zhao and Badler [27] incorporates this case implicitly. We use an optimization-based method to handle this case as well, but we take advantage of properties of the problem to simplify the optimization task.

We first notice that we can usually solve for the desired position of the wrist provided that the goal position is close enough to the shoulder. This allows us to compute θ_4 and the valid ranges of ϕ for variables θ_1, θ_2, and θ_3. Let $V = A_1 \cup A_2$ denote the set of ϕ that satisfy the joint limits for the two families of solutions for θ_1, θ_2, and θ_3. Since the location of the wrist does not depend on the other joints, any $\phi \in V$ satisfies the joint limits for the shoulder angles and is a solution for the goal position. Thus, the problem has been reduced to finding a suitable value of $\phi \in V$ that minimizes the orientation error between the wrist frame and the goal frame.

There are many possible ways of measuring orientation error between two rotation matrices $\mathbf{R}_\mathrm{d} = [\hat{\mathbf{n}}_\mathrm{d}, \hat{\mathbf{s}}_\mathrm{d}, \hat{\mathbf{a}}_\mathrm{d}]$ and $\mathbf{R} = [\hat{\mathbf{n}}, \hat{\mathbf{s}}, \hat{\mathbf{a}}]$. The approach used by Zhao and Badler [27] uses the term $\|(\hat{\mathbf{n}}_\mathrm{d} - \hat{\mathbf{n}})\| + \|(\hat{\mathbf{s}}_\mathrm{d} - \hat{\mathbf{s}})\|$. Another approach is to convert the rotations into their unit quaternion representations and take the absolute value of their dot product. A value of 1 means that the two rotations are identical, and a value of 0 indicates that the two rotations are far apart. Another approach is to find the axis $\hat{\mathbf{u}}$ and angle ν of equivalent rotation of the matrix

$$\mathbf{R}_{\hat{\mathbf{u}}}(\nu) = \mathbf{R}_\mathrm{d}\mathbf{R}^T, \quad -\frac{\pi}{2} \le \nu \le \frac{\pi}{2}$$

and to take the orientation error as $\sin(\nu)^2$. It is a simple matter to show that

$$4\sin(\nu)^2 = (r_{32} - r_{23})^2 + (r_{13} - r_{31})^2 + (r_{21} - r_{12})^2,$$

where r_{ij} are the components of $\mathbf{R}_{\hat{u}}(\nu)$.

The method used to calculate orientation error can be written as an objective function for a nonlinear optimization problem with linear inequality constraints in terms of the variables ϕ, θ_5, θ_6, and θ_7. This approach has several advantages compared with the approach used in [27]. The objective function is simpler and the problem is only four-dimensional rather than seven-dimensional. Additionally, the objective function used in [27] constrains both position and orientation terms and requires scale factors to weight the relative importance of these terms. In contrast, the objective function used in our algorithm contains only orientation constraints, the position constraint is always satisfied.

3.3. Partial Orientation Constraints

It is often the case that a user wants to pose an inverse kinematics solution for only one of the columns of the orientation component of the goal matrix. Consider the case of a person hammering a nail. In this example, it is not necessary to solve completely for the orientation of the end effector to obtain a plausible-looking solution. The only requirements are that the position of the tip of the hammer coincide with the nail and the axis pointing out of the head of the hammer align with the shaft of the nail. In this case, the system has an extra degree of freedom parameterized by the angle of rotation about the axis of the hammer. For these types of problems, the inverse kinematics problem has the form

$$\mathbf{T}_1\mathbf{A}\mathbf{T}_y\mathbf{B}\mathbf{T}_2\mathbf{E} = \mathbf{G}(\psi),$$

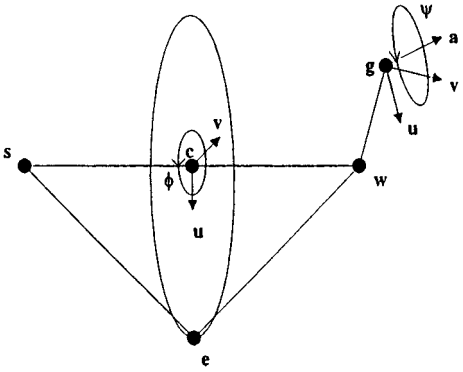

FIG. 8. A partial orientation problem is parameterized by ϕ and ψ.

where $\mathbf{E} = \left[\begin{smallmatrix} \mathbf{R}_e & \mathbf{t}_e \\ 0\,0\,0 & 1 \end{smallmatrix}\right]$ specifies the constant transformation from the proximal frame of the wrist to the end effector. As shown in Fig. 8, the orientation component of the goal matrix $\mathbf{G}$ depends on three vectors $\hat{\mathbf{a}}$, $\hat{\mathbf{u}}_g$, $\hat{\mathbf{v}}_g$, where $\hat{\mathbf{a}}$ is the axis that the user wants to align with the end effector and the $\hat{\mathbf{u}}_g$ and $\hat{\mathbf{v}}_g$ vectors are defined by constructing a local coordinate system about the plane whose normal is given by $\hat{\mathbf{a}}$,

$$\hat{\mathbf{u}}_g = \frac{\hat{\mathbf{p}} - (\hat{\mathbf{p}} \cdot \hat{\mathbf{a}})\hat{\mathbf{a}}}{\|\hat{\mathbf{p}} - (\hat{\mathbf{p}} \cdot \hat{\mathbf{a}})\hat{\mathbf{a}}\|}$$
$$\hat{\mathbf{v}}_g = \hat{\mathbf{a}} \times \hat{\mathbf{u}}_g,$$

where $\hat{\mathbf{p}}$ is the projection axis defined in the previous section. The structure of $\mathbf{G}$ depends on which column of the goal matrix $\hat{\mathbf{a}}$ corresponds to:

$$\hat{\mathbf{a}} \equiv \hat{\mathbf{x}} \quad \mathbf{G}(\psi) = \begin{bmatrix} \hat{\mathbf{a}} & \hat{\mathbf{u}}_g \cos(\psi) + \hat{\mathbf{v}}_g \sin(\psi) & \hat{\mathbf{v}}_g \cos(\psi) - \hat{\mathbf{u}}_g \sin(\psi) & \mathbf{t}_g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\mathbf{a}} \equiv \hat{\mathbf{y}} \quad \mathbf{G}(\psi) = \begin{bmatrix} \hat{\mathbf{v}}_g \cos(\psi) - \hat{\mathbf{u}}_g \sin(\psi) & \hat{\mathbf{a}} & \hat{\mathbf{u}}_g \cos(\psi) + \hat{\mathbf{v}}_g \sin(\psi) & \mathbf{t}_g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\mathbf{a}} \equiv \hat{\mathbf{z}} \quad \mathbf{G}(\psi) = \begin{bmatrix} \hat{\mathbf{u}}_g \cos(\psi) + \hat{\mathbf{v}}_g \sin(\psi) & \hat{\mathbf{v}}_g \cos(\psi) - \hat{\mathbf{u}}_g \sin(\psi) & \hat{\mathbf{a}} & \mathbf{t}_g \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The parameter ψ plays a role for the wrist position similar to the role ϕ plays for the elbow location. Changing the value of ψ moves the wrist relative to the end-effector position about a circle that lies on a plane whose normal is $\hat{\mathbf{a}}$.

We now compute the wrist and elbow positions as function of ϕ and ψ. The wrist position $\mathbf{w}$ is given by

$$\begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix} = \mathbf{T}_1 \mathbf{A} \mathbf{T}_y \mathbf{B} \mathbf{T}_2 [0, 0, 0, 1]^T = \mathbf{G}(\psi) \mathbf{E}^{-1} [0, 0, 0, 1]^T.$$

Expanding the right-hand side of the equation reveals that $\mathbf{w}$ is of the form

$$\mathbf{w}(\psi) = \begin{bmatrix} \alpha_1 \cos\psi + \beta_1 \sin\psi + \gamma_1 \\ \alpha_2 \cos\psi + \beta_2 \sin\psi + \gamma_2 \\ \alpha_3 \cos\psi + \beta_3 \sin\psi + \gamma_3 \end{bmatrix}, \tag{10}$$

where α_i, β_i, and γ_i are constants that depend on **G** and **E**. Equation (10) gives a convenient way of solving for a suitable value of ψ that optimizes for a criterion that involves the wrist position. For example, if the vertical axis is along the **z** direction, the values of ψ that minimize and maximize the height of the wrist can be computed by solving

$$\frac{d}{d\psi}(\alpha_3 \cos\psi + \beta_3 \sin\psi + \gamma_3) = 0, \tag{11}$$

which has a straightforward, trigonometric analytical solution. There are two solutions to Eq. (11) corresponding to the minimizing and maximizing values of ψ. They can be distinguished by checking whether the second derivative is positive or negative.

Similarly, the value of ψ that minimizes the distance from a desired wrist position $\mathbf{w}_\mathrm{d}$ is given by the solution to the equation

$$\frac{d}{d\psi}((\mathbf{w}(\psi) - \mathbf{w}_\mathrm{d}) \cdot (\mathbf{w}(\psi) - \mathbf{w}_\mathrm{d})) = 0. \tag{12}$$

Equation (12) has the form

$$a\cos^2\psi + b\sin^2\psi + c(\cos\psi \sin\psi) + d\cos\psi + e\sin\psi + f = 0.$$

We can convert the trigonometric terms into a polynomial by making the half angle substitution

$$u = \tan\frac{\psi}{2}$$

$$\cos\psi = \frac{1-u^2}{1+u^2}$$

$$\sin\psi = \frac{2u}{1+u^2}.$$

The equation with the substituted variable is a quartic polynomial in u which has a straightforward, trigonometric analytical solution. Finally, once the wrist position is known, the elbow equation can be computed using the method described in the previous section. However, the elbow position now depends upon the two variables ψ and ϕ.

Given a goal matrix **G** and the parameters ψ and ϕ, we can derive the wrist and elbow positions from Eqs. (10) and (6) with **w** substituted for $\mathbf{t}_\mathrm{g}$. θ_4 and $\mathbf{R}_0$ depend upon both **G** and ψ, and the $\mathbf{R}_\phi$ matrix depends upon ϕ and ψ. The inverse kinematics problem can be solved analytically by computing θ_4, $\mathbf{R}_1$, and $\mathbf{R}_2$ using the methods described earlier.

Joint limits. With a partial orientation constraint the inverse kinematics problem has 2 degrees of freedom instead of 1 and each joint variable can be thought of as a surface parameterized by ψ and ϕ. The joint limits are planes that partition the surface into regions that violate or satisfy the joint limits. Unfortunately, it is extremely difficult, perhaps impossible, to find a parameterization of these regions. Instead we will resort to the use of a numerical procedure to enforce the joint limits when required.

Inspecting the scalar components of the equation $\mathbf{R}_1(\theta_1, \theta_2, \theta_3) = \mathbf{R}_0\mathbf{R}_\phi$ we can always find a system of equation of the form

$$\sin(\theta_i) = f_1(\phi, \psi)$$

$$\cos(\theta_i)\cos(\theta_j) = f_2(\phi, \psi)$$

$$\begin{aligned}\cos(\theta_i)\sin(\theta_j) &= f_3(\phi,\psi)\\ \cos(\theta_i)\cos(\theta_k) &= f_4(\phi,\psi)\\ \cos(\theta_i)\sin(\theta_k) &= f_5(\phi,\psi)\end{aligned}$$

or

$$\begin{aligned}\cos(\theta_i) &= f_1(\phi,\psi)\\ \sin(\theta_i)\cos(\theta_j) &= f_2(\phi,\psi)\\ \sin(\theta_i)\sin(\theta_j) &= f_3(\phi,\psi)\\ \sin(\theta_i)\cos(\theta_k) &= f_4(\phi,\psi)\\ \sin(\theta_i)\sin(\theta_k) &= f_5(\phi,\psi),\end{aligned}$$

where i, j, and k are a permutation of the sequence $\{1, 2, 3\}$ and $f_l(\phi,\psi)$, $l = 1,\ldots,5$, are lengthy trigonometric expression in ϕ, ψ which have to be determined through the use of a symbolic mathematics package. Assume without loss of generality that we have a system of the first form and that $i = 1$, $j = 2$, and $k = 3$. We can calculate the partial derivatives of θ_1, θ_2, and θ_3,

$$\frac{\partial\theta_1}{\partial\phi} = \begin{cases}\left(\frac{\partial f_1}{\partial\phi}\right)\Big/\sqrt{1-f_1^2} & \text{family 1}\\ -\left(\frac{\partial f_1}{\partial\phi}\right)\Big/\sqrt{1-f_1^2} & \text{family 2}\end{cases}$$

$$\frac{\partial\theta_1}{\partial\psi} = \begin{cases}\left(\frac{\partial f_1}{\partial\psi}\right)\Big/\sqrt{1-f_1^2} & \text{family 1}\\ -\left(\frac{\partial f_1}{\partial\psi}\right)\Big/\sqrt{1-f_1^2} & \text{family 2}\end{cases}$$

and

$$\begin{aligned}\frac{\partial\theta_2}{\partial\phi} &= \left[\left(\frac{\partial f_3}{\partial\phi}\right)f_2 - f_3\left(\frac{\partial f_2}{\partial\phi}\right)\right]\Big/\sqrt{1-f_1^2}\\ \frac{\partial\theta_2}{\partial\psi} &= \left[\left(\frac{\partial f_3}{\partial\psi}\right)f_2 - f_3\left(\frac{\partial f_2}{\partial\psi}\right)\right]\Big/\sqrt{1-f_1^2}\\ \frac{\partial\theta_3}{\partial\phi} &= \left[\left(\frac{\partial f_5}{\partial\phi}\right)f_4 - f_5\left(\frac{\partial f_4}{\partial\phi}\right)\right]\Big/\sqrt{1-f_1^2}\\ \frac{\partial\theta_3}{\partial\psi} &= \left[\left(\frac{\partial f_5}{\partial\psi}\right)f_4 - f_5\left(\frac{\partial f_4}{\partial\psi}\right)\right]\Big/\sqrt{1-f_1^2}.\end{aligned}$$

By expanding the equation $\mathbf{R}_2 = (\mathbf{R}_0\mathbf{R}_\phi\mathbf{R}_a\mathbf{R}_y\mathbf{R}_b)^{-1}\mathbf{R}_g(\varphi)\mathbf{R}_e^{-1}$, we can obtain similar expressions for the partials of θ_5, θ_6, and θ_7.

We can now cast the problem of finding a solution to the partial orientation constraint with joint limits into an optimization problem in terms of ϕ and ψ. Define the objective function

$$F(\phi,\varphi) = \sum_{i=1, i\neq 4}^{7}(\theta_i(\varphi,\psi) - \theta_i^*)^2,$$

where $\theta_i^* = (\theta_i^{\max} + \theta_i^{\min})/2$ is the middle of the joint limit range of joint i. Minimizing F will tend to keep the joint variables as close as possible to the middle of their valid ranges. A local minimum of F can be found using a standard nonlinear optimization procedure to find where the gradient of F vanishes. The gradient of F is given by

$$\nabla F = \begin{bmatrix} \sum_{i=1, i\neq 4}^{7} 2(\theta_i(\varphi, \psi)_i - \theta_i^*)\frac{\partial \theta_i}{\partial \phi} \\ \sum_{i=1, i\neq 4}^{7} 2(\theta_i(\varphi, \psi)_i - \theta_i^*)\frac{\partial \theta_i}{\partial \psi} \end{bmatrix}.$$

The expressions for $\theta_i(\phi, \psi)$, $\frac{\partial \theta_i}{\partial \phi}$, and $\frac{\partial \theta_i}{\partial \psi}$ can be computed analytically as previously described.

There are several differences between this approach and the optimization-based method of Zhao and Badler [27]. In our approach the objective function involves the joint limits, whereas the formulation of the objective function in [27] is the inverse kinematics problem. Our approach guarantees that the inverse kinematics problem will be solved, at the expense of possibly violating the joint limits. Conversely, in [27] the joint limits are strictly observed but the inverse kinematics constraints might not be satisfied. Finally, our optimization problem is simpler in the sense that it only involves two variables with no constraints compared to seven variables with linear inequality constraints. On the other hand, our objective function is considerably more complicated and because there are two different families of solutions for θ_1 and θ_5, there are actually four optimization problems that must be solved, one for each combination of the two families.

3.4. Aiming Problems

Sometimes we want to aim an axis on the end effector toward a target point. This is not a traditional inverse kinematics problem encountered in robotics applications and it requires a slightly different formulation. For simplicity, we assume that wrist angles are held fixed and the only joints that contribute to the task are the shoulder and elbow joints. Because a pointing task defines two constraints, the system has 2 redundant degrees of freedom.

A physical interpretation of the redundant degrees of freedom is shown in Fig. 9. In the diagram L_1 and L_2 denote the length of the upper arm and the distance from the elbow to

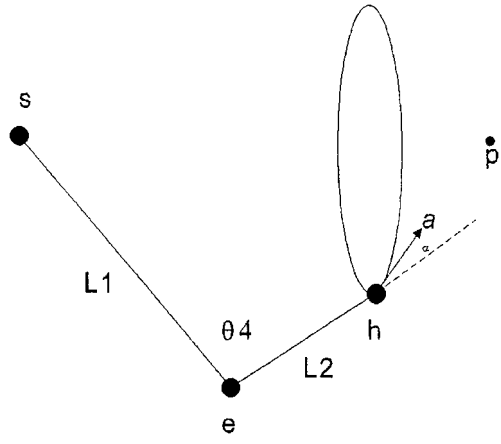

FIG. 9. Pointing an axis toward a target.

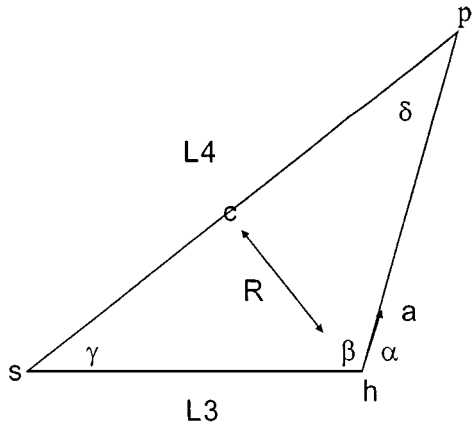

FIG. 10. The hand circle equation.

the hand position, **p** designates the target point, **h** denotes the hand position, **â** denotes the aiming axis, and α is the angle that **â** makes with $\overrightarrow{\mathbf{eh}}$. For a fixed value of θ_4, the hand position is free to swivel about a circle analogous to the elbow circle of the previous sections.

The system has 2 degrees of freedom: θ_4 controls the extent to which the hand is outstretched and ϕ moves the hand about a cone whose apex is the goal point. The hand position in terms of the shoulder frame is given by $\mathbf{h} = \mathbf{R}_1\mathbf{t}_a + \mathbf{R}_1\mathbf{R}_y(\theta_4)\mathbf{t}_b$. If $\hat{\mathbf{a}}_o$ is the aiming direction in the hand frame, the aiming direction in the shoulder frame **â** is given by $\hat{\mathbf{a}} = \mathbf{R}_1\mathbf{R}_y(\theta_4)\mathbf{t}_b\hat{\mathbf{a}}_o$.

The angle α between $\overrightarrow{\mathbf{sh}}$ and **â** depends only upon θ_4. From simple trigonometry (Fig. 10) we infer

$$\cos\alpha = \frac{\hat{\mathbf{a}}\mathbf{R}_y^T\mathbf{t}_a + \hat{\mathbf{a}}\mathbf{R}_y^T\mathbf{R}_y\mathbf{t}_b}{L_3}$$

$$L_3 = \left(\mathbf{t}_a^T\mathbf{t}_a + \mathbf{t}_a^T\mathbf{R}_y\mathbf{t}_b + \mathbf{t}_b^T\mathbf{R}_y^T\mathbf{t}_a + \mathbf{t}_b^T\mathbf{t}_b\right)^{1/2}.$$

To derive the equation of the circle governing the hand position in terms of ϕ we need to infer the angle γ of the triangle $\Delta\mathbf{shg}$. Assume that $\alpha \le \frac{\pi}{2}$ and that $\hat{\mathbf{a}} \cdot \overrightarrow{\mathbf{sh}} > 0$ implies that

$$\beta = a\cos(-\cos(\alpha))$$

$$\beta > \frac{\pi}{2}$$

$$\delta = a\sin\left(\frac{L_3}{L_4}\sin\beta\right)$$

$$\gamma = \pi - (\beta + \delta).$$

The center of the hand circle **c** and its radius R are given by

$$\mathbf{c} = L_3\cos\gamma\,\frac{\mathbf{p}}{\|\mathbf{p}\|}$$

$$R = L_3\sin\gamma.$$

Finally to obtain $\mathbf{h}(\phi)$ we define a local $\hat{\mathbf{u}}, \hat{\mathbf{v}}$ coordinate system by projecting an arbitrary

vector $\hat{\mathbf{p}}$ onto the plane of the circle as we did in the previous two sections.

$$\mathbf{h}(\phi) = \mathbf{c} + R(\cos\phi\hat{\mathbf{u}} + \sin\phi\hat{\mathbf{v}}). \tag{13}$$

In order to solve $\mathbf{R}_1$ we first consider the following subproblem. Given two points $\mathbf{x}$, $\mathbf{y}$ such that $\|\mathbf{x}\| = \|\mathbf{y}\|$ and a unit vector $\hat{\mathbf{z}}$ passing through the origin we want to find the angle of rotation θ about $\hat{\mathbf{z}}$ that rotates $\mathbf{x}$ onto $\mathbf{y}$. We also make the assumption that $\mathbf{x}$ and $\mathbf{y}$ are not collinear and that they do not lie along the $\hat{\mathbf{z}}$ axis. Let $\mathbf{x}^* = \mathbf{x} - \hat{\mathbf{z}}^{\mathrm{T}}\hat{\mathbf{z}}\mathbf{x}$ and $\mathbf{y}^* = \mathbf{y} - \hat{\mathbf{z}}^{\mathrm{T}}\hat{\mathbf{z}}\mathbf{y}$ be the projections of $\mathbf{x}$ and $\mathbf{y}$ onto the plane perpendicular to $\hat{\mathbf{z}}$. We can find θ by calculating the angle between $\mathbf{x}^*$ and $\mathbf{y}^*$. Since

$$\mathbf{x}^* \times \mathbf{y}^* = \hat{\mathbf{z}}\sin(\theta)\|\mathbf{x}^*\|\|\mathbf{y}^*\| \tag{14}$$

and

$$\mathbf{x}^* \cdot \mathbf{y}^* = \cos(\theta)\|\mathbf{x}^*\|\|\mathbf{y}^*\| \tag{15}$$

we have $\theta = a\tan 2(\hat{\mathbf{z}}^{\mathrm{T}}(\mathbf{x}^* \times \mathbf{y}^*), \mathbf{x} \cdot \mathbf{y})$.

For a specified θ_4 and $\mathbf{h}(\phi)$ we find $\mathbf{R}_1$ by expressing it as the product of two rotation matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ (Fig. 11). Let $\mathbf{h}_o$ denote the position of the hand after the elbow joint rotates by θ_4. $\mathbf{S}_1$ is the rotation about the vector $\mathbf{h}_o \times \mathbf{h}(\phi)/\|\mathbf{h}_o \times \mathbf{h}(\phi)\|$ that moves the hand from $\mathbf{h}_o$ to $\mathbf{h}(\phi)$. Now let $\hat{\mathbf{a}}_1$ denote the orientation of the aim vector after we apply $\mathbf{S}_1$. $\mathbf{S}_2$ is the rotation about the $\overrightarrow{\mathbf{sh}}/\|\overrightarrow{\mathbf{sh}}\|$ axis that aligns $\hat{\mathbf{a}}_1$ to $\hat{\mathbf{a}}$.

If θ_4 is specified, the valid ranges of ϕ that satisfy the joint limits can be computed as described in Section 4.3. If θ_4 is allowed to vary, the optimal values of θ_4 and ϕ satisfying the joint limits can be found by solving an optimization problem as described in Section 4.4. If the aiming constraint forces a joint limit violation, then an optimization algorithm with nonlinear constraints on the joint limits for θ_1, θ_2, and θ_3 can be used; however, since optimization with nonlinear constraints can have poor convergence properties a traditional optimization formulation might work better in this case.

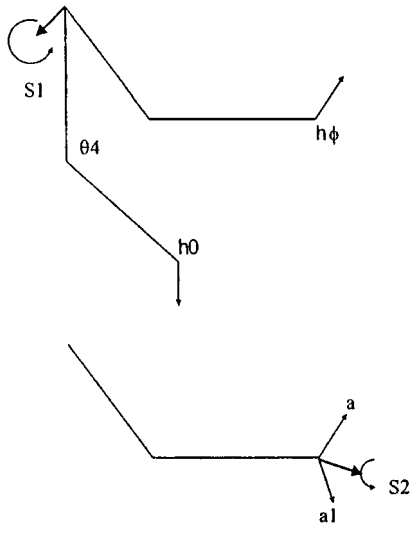

FIG. 11. Decomposing R_1 into two rotations.

TABLE 2
Summary of Methods Used When the Goal Is Reachable

	Goal reachable (joint limits off)	Goal reachable (joint limits on)
Position	Analytic	Analytic
Position and orientation	Analytic	Analytic
Position and partial orientation	Analytic	Analytic + 2DOF unconstrained optimization
Aiming	Analytic	Analytic if θ_4 given 2DOF unconstrained optimization otherwise

3.5. Benchmarks

In this section, we summarize the results of time and accuracy benchmarks comparing our algorithms against an optimization approach and an inverse Jacobian routine (Tables 2–7). The optimization implementation used in the study is based on the CFSQP algorithm developed by Lawrence *et al.* [13]. We believe their algorithm is among the best public domain optimization routines available. For the pseudo-inverse Jacobian implementation, we used the singular value decomposition and the Bulirsch and Stoer ODE routines from *Numerical Recipes in C* [18]. To generate a suitable end-effector trajectory we used Hermite interpolation to construct a curve connecting the start and goal positions. A suitable angular velocity profile was generated by using a screw motion between the start and goal orientations.

It should be noted that most numerical routines require a floating point number specifying the maximum error tolerance and an upper bound on the number of iterations permitted. Obviously both these parameters have a significant effect on the time and accuracy of the routine. In our benchmarks we used the default values furnished by the authors.

We measure position error by the norm of the vector from the final end-effector position to the goal position. The measure of orientation error depends upon the type of inverse kinematics problem. If the problem requires aligning two rotation matrices, orientation error is measured by subtracting one from the dot product of the quaternions corresponding to the goal and final rotation matrices. For partial orientation and aiming problems the orientation error is taken as the square of one minus the dot product of the end effector and goal axis.

TABLE 3
Summary of Methods Used When the Goal Is Unsolvable

	Goal unreachable
Position and orientation	4DOF constrained optimization (4 linear constraints)
Position and partial orientation	2DOF constrained optimization (6 nonlinear constraints)
Aiming	2DOF constrained optimization (3 nonlinear constraints)

TABLE 4
Position Only

	Our approach	Optimization	Inverse Jacobian
Joint limits off			
Absolute time	0.000024	0.007	0.5
Relative time	1	292	20,833
Joint limits on			
Absolute time	0.00023	0.009	0.77
Relative time	1	39	3347
Average position error	2.5×10^{-8}	3.3×10^{-7}	5.6×10^{-7}
Percentage failures	0	0.5%	1.02%

TABLE 5
Position and Orientation

	Our approach	Optimization	Inverse Jacobian
Joint limits off			
Absolute time	0.0001	0.02	1.5
Relative time	1	200	15,000
Joint limits on			
Absolute time	0.0015	0.03	2.23
Relative time	1	20	1487
Average position error	2.6×10^{-8}	5.4×10^{-7}	3.2×10^{-7}
Average orientation error	1.0×10^{-8}	1.2×10^{-7}	1.4×10^{-7}
Percentage failures	0	0.6%	1.75%
Goal unreachable			
Absolute time	0.005	0.037	3.65
Relative time	1	12.3	730

TABLE 6
Partial Orientation

	Our approach	Optimization
Joint limits off		
Absolute time	0.00012	0.015
Relative time	1	125
Joint limits on		
Absolute time	0.008	0.024
Relative time	1	1
Relative time (joint limits enforced)	1	3
Average position error	2.7×10^{-8}	6.0×10^{-7}
Average orientation error	0.9×10^{-8}	1.2×10^{-7}
Percentage failures	1.1%	0.74%
Goal unreachable		
Absolute time	0.054	0.028
Relative time	1.93	1

TABLE 7
Aiming

	Our approach	Optimization
Joint limits off		
Absolute time	0.000025	0.0065
Relative time	1	260
Joint limits on		
Absolute time	0.0067	0.0092
Relative time	1	1.373
Average aiming error	1.25×10^{-8}	1.43×10^{-7}
Percentage failures	0	0.45%
Goal unreachable		
Absolute time	0.014	0.011
Relative time	1.27	1

All the algorithms tested have accuracy that is more than adequate for most applications. However, none of the numerical algorithms are reliable all of the time. As long as the goal is reachable our methods will always find a solution. However, the optimization and inverse Jacobian methods sometimes fail even when the goal is within the workspace of the arm. Optimization can fail because of local minima. For example, the optimization routine may move the arm in a direction that locally moves it toward a goal only to have the arm lock at a joint limit. On the other hand, if the routine had moved the arm in another direction the goal might be satisfiable. The inverse Jacobian method typically fails when the arm is moved in a configuration where the Jacobian is close to singular. When this occurs, the joint velocities become very large and change rapidly. This instability in the joint velocities is detected by the numerical integrator which shrinks the step size to compensate. The result is very slow convergence near the vicinity of the singularity.

When we can utilize an analytical method or a hybrid of analytical and numerical techniques, our methods are significantly faster than their purely numerical counterparts. The inverse Jacobian method is by far the slowest. This is hardly surprising since the inverse Jacobian method actually solves for an entire trajectory rather than just the final posture of the arm. Thus the inverse Jacobian method must take relatively small step sizes at each iteration compared to an optimization-based routine.

The only case where our methods underperform is in partial orientation and aiming problems where the goal is actually not reachable. In these cases our method is slightly slower than a traditional optimization-based formulation. Apparently the presence of nonlinear constraints more than offsets the efficiency gained by reducing the optimization problem to a lower dimensional space. Fortunately, in problems involving partial orientation and aiming constraints the goal is seldom unreachable.

3.6. Additional Performance Considerations

One of the advantages of our analytical approach versus the numerical algorithms is in handling singularities. Geometrically, this occurs when the arm is fully stretched out. In this case, the Jacobian is no longer full rank and it is impossible to generate velocity along

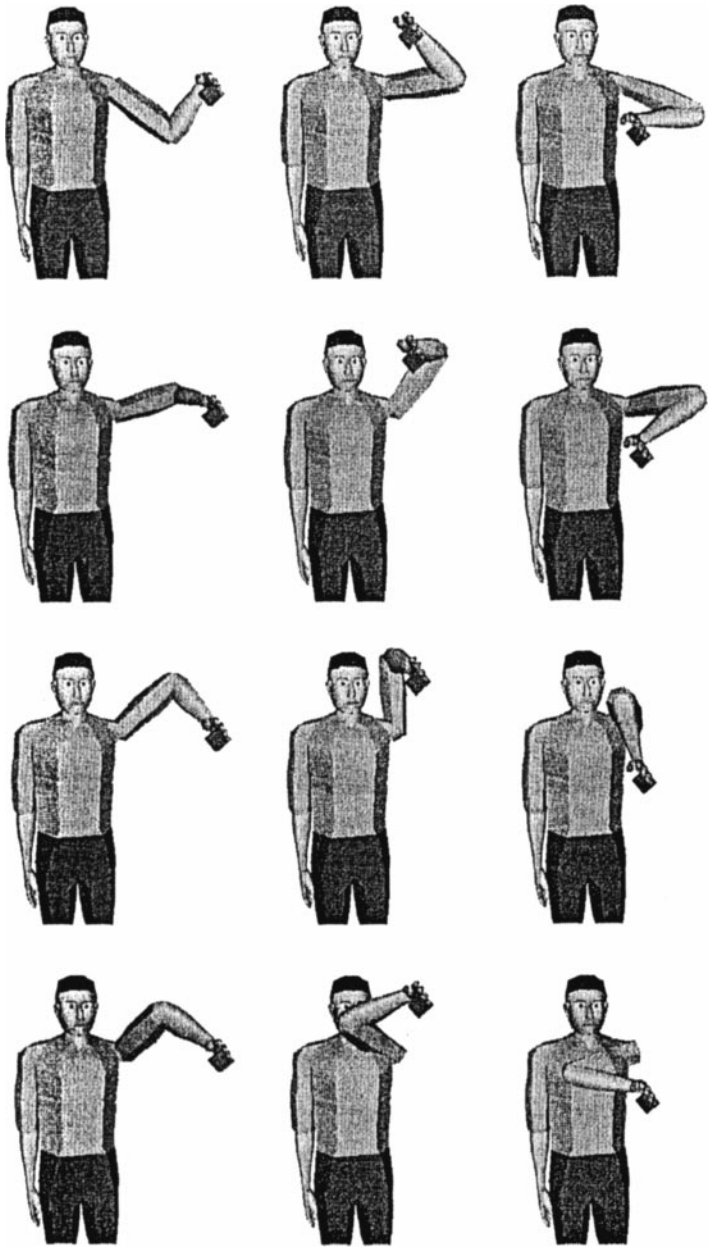

FIG. 12. A pseudoinverse solution does not yield a cyclical solution in joint space for a cyclical motion in end effector space.

one Cartesian direction irrespective of how large the joint velocities are. Singularities cause problems for both the pseudo-inverse and optimization approaches. In the pseudo-inverse algorithm, a singularity can cause abrupt discontinuities in the joint velocities. The discontinuities are detected by the numerical integrator which reduces the step size taken by the algorithm. This can significantly degrade the performance of the algorithm. Moreover, unless special care is taken in the implementation, the singularity can cause the integrator to reduce the step size to the machine precision, possibly producing a run-time error

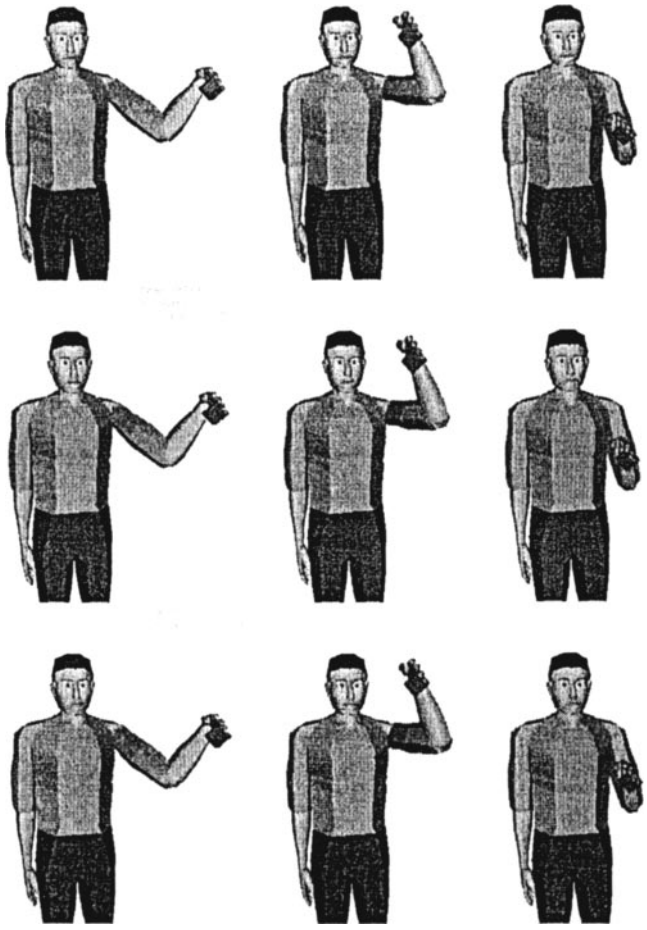

FIG. 13. Our algorithm produces a repeatable motion in joint space for a closed cycle in end effector space.

or preventing the algorithm from converging to a solution. In an optimization algorithm, the Jacobian is used to compute the gradient of the objective function. When the Jacobian loses rank, the gradient might become zero causing the algorithm to terminate in a local minimum. By contrast, in our approach a singularity merely causes the radius of the elbow circle to vanish. This means that the swivel angle is no longer relevant; regardless of its value the elbow position is constant. However, a solution for the problem can still be obtained.

Another problem with numerical approaches is that the solution returned depends upon the initial guess. In an interactive system this lack of repeatability can be frustrating to the user. To prevent this problem some interactive systems always begin the figure in a neutral posture before invoking the inverse kinematics routine. However, this approach is not satisfactory either since the neutral posture may be a poor initial guess. Additionally, if a pseudo-inverse approach is used to generate solutions for closed end-effector paths the corresponding joint trajectories will in general not be closed. This is a significant problem for computer animation where many tasks tend to be cyclical in nature. For example, if a pseudo-inverse solution is used to generate a walking sequence the gait will vary from step to step. This problem is illustrated in Fig. 12, where a pseudo-inverse method is used to trace

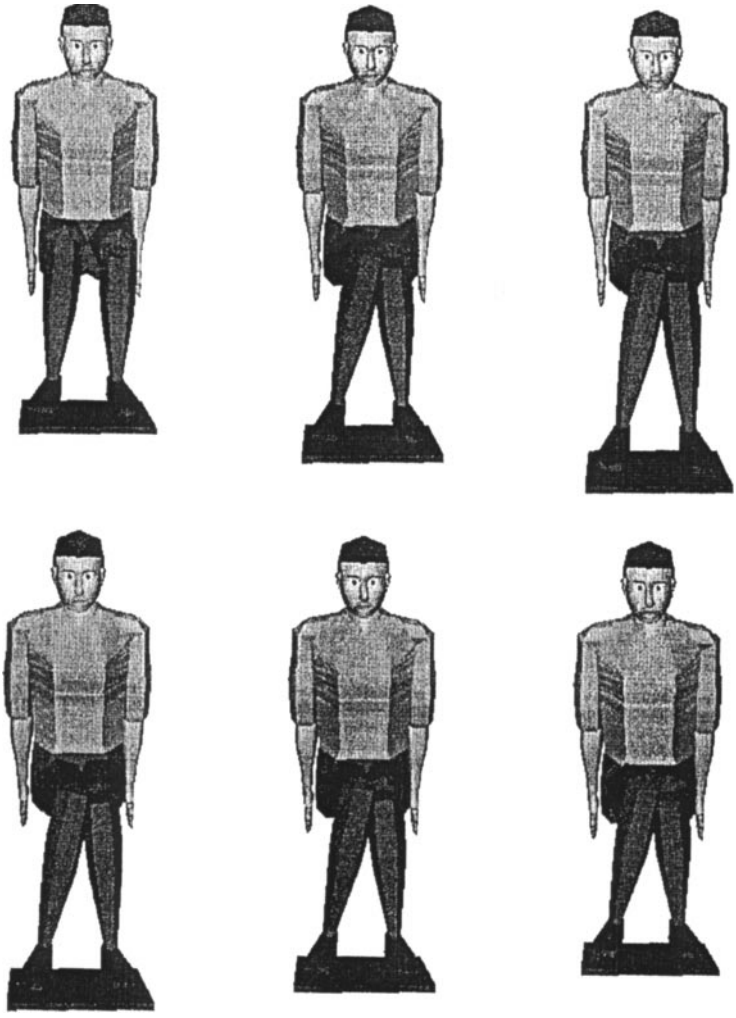

FIG. 14. An optimization approach does not reproduce the same answer when the goal is restored to its original location.

a circular path. After three cycles, the posture of the arm has drifted significantly from the initial posture. In contrast, our algorithm produces a repeatable and more natural-looking solution (Fig. 13).

The dependence of a numerical solution upon the initial guess can also be frustrating for a user in interactive applications because it may be difficult to reproduce a solution consistently from session to session. In our approach, the solution returned is independent of the starting posture. Thus our solutions are always repeatable. This problem is illustrated in Fig. 14, where an optimization procedure is used to solve an inverse kinematics problem for the legs. If the user adjusts the position of the goal it becomes difficult to obtain the original solution when the goal is restored to its original state. In contrast, our algorithm always gives a consistent solution for a desired goal irrespective of the starting posture (Fig. 15).

For position and orientation problems, our methods can identify the relationship between the extra degree-of-freedom and the joint limits. This allows the algorithm to automatically

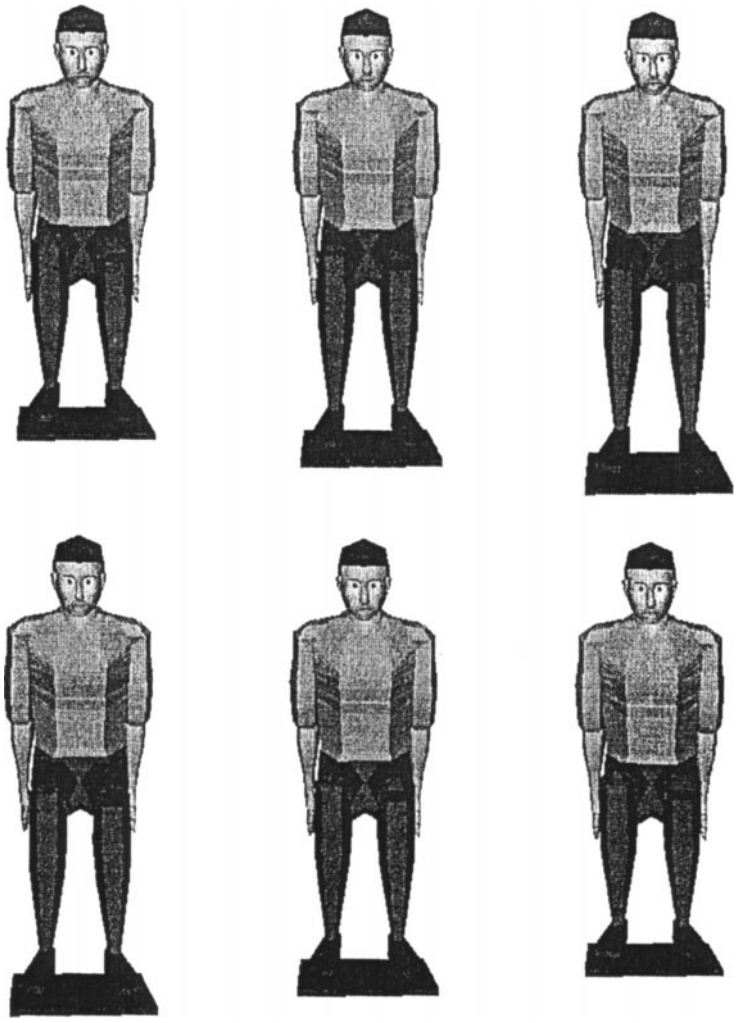

FIG. 15. Our algorithm always yields a reproducible solution.

detect when a solution is possible and which joint limits are sensitive to changes in the elbow posture.

4. CONCLUSIONS AND FUTURE WORK

4.1. Contributions

In this paper we have developed a set of kinematic tools for modeling the forward and inverse kinematics of a model of the human arm or leg. The advantages and novel aspects of our work include the following:

1. We have devised analytical and hybrid analytical numerical algorithms for solving inverse kinematics problems and other constraints such as partial orientation and aiming problems. Our algorithms are faster to compute and more reliable than purely numerical methods. For example, Jacobian-based methods become unstable near a singular configuration of the arm and optimization-based methods are subject to local minima problems.

Neither of these problems occurs when we use an analytical scheme. For each problem, we identify a minimal set of variables that identify the degrees of freedom in the space of solutions. These variables also provide the user with an intuitive set of control variables in interactive applications.

2. Occasionally, we are forced to rely on optimization techniques. However, in many cases, we are able to reduce the complexity of the optimization problem by decomposing the problem into an analytical component and a numerical one, and also by expressing the problem in terms of a lower number of variables than the number of joint variables. Further efficiency is obtained by converting some constrained optimization problems into unconstrained ones.

3. Most of our algorithms are repeatable in that the solution does not depend upon the starting posture of the arm. Repeatability is a significant issue in interactive applications where a user spends a great deal of time manipulating the solver into finding a satisfactory final posture. In these cases, it is very important to the user that the solver behave consistently and not be sensitive to minor perturbations of the starting state.

4. With a numerical approach it is difficult to tell whether algorithm failure indicates the absence of a feasible solution or a limitation of the technique such as a singular Jacobian or local minimum. By contrast, for position and orientation problems our approach can reliably determine whether a solution exists. This feature is particularly useful in task analysis where it is important to determine which regions of the workspace are reachable.

5. In the case of redundant systems, our algorithms permit the user to explore multiple solutions using a set of intuitive parameters. By contrast, numerical algorithms only converge to a single solution, which restricts their usefulness in interactive applications.

4.2. Future Work

We conclude with a list of potential enhancements and topics for future research. To develop a commercial quality and general purpose inverse kinematics subroutine library for human body animation, additional issues should be addressed.

1. Since our techniques already work for the arms and legs, the addition of a spine model would permit us to solve inverse kinematics problems involving the entire body. Although it is probably not practical to treat the entire body as a gigantic kinematic chain it may be possible to use an approach in which an inverse kinematics problem is broken into a smaller set of problems each of which can be solved efficiently using our methods. Zhao [28] described a “task information” system which converts a set of high-level goals into a sequence of low-level constraints on key positions of the body. The system first determines how to position the figure’s center of mass and global orientation so that the task can be performed. Once all the goals lie within the workspace of the corresponding end effector, the inverse kinematics routines for individual kinematic units such as the spine, arms, and legs are invoked. This system decouples a complex inverse kinematics problem into simpler ones that can be solved efficiently with our methods.

2. Animation and task simulation systems involve more than just determining if a goal is reachable. Other useful features are constraints such as balance control and collision avoidance [3, 29]. It may be possible to formulate these constraints in such a way that analytical or a hybrid of analytical and numerical solutions to them can be found. Efficient solutions for these additional constraints would be valuable to an animation system for obvious reasons.

ACKNOWLEDGMENTS

This research is partially supported by Office of Naval Research K-5-55043/3916-1552793 and AASERT N0014-97-1-0605, NASA NRA NAG 5-3990, NSF IRI95-04372, and the IRCS NSF STC at the University of Pennsylvania.

REFERENCES

1. K. Asano, Human arm kinematics, in *Proceedings of the 5th International Symposium on Robotics Research, 1990*.
2. N. Badler, K. Manoochehri, and G. Walters, Articulated figure positioning by multiple constraints, *IEEE Comput. Graph. Appl.* **7**(6), 1987, 28–38.
3. N. Badler, C. Phillips, and B. Webber, *Simulating Humans: Computer Graphics Animation and Control*, Oxford Univ. Press, New York, 1993.
4. H. Cheng and K. Gupta, A study of robot inverse kinematics based upon the solution of differential equations, *J. Robot. Systems* **8**(2), 1991, 115–175.
5. A. Hemami, A more general closed form solution to the inverse kinematics of mechanical arms, *Adv. Robot.* **2**, 1988.
6. J. M. Hollerbach, Optimal kinematic design for a seven degree of freedom manipulator, in *Proceedings of the 5th International Symposium on Robotics Research, 1990.*
7. J. M. Hollerbach and G. Sahar, Wrist-partitioned inverse kinematic accelerations and manipulator dynamics, *Internat. J. Robot. Res.* **2**, 1983, 61–76.
8. J. M. Hollerbach and K. C. Suh, Redundancy resolution of manipulators through torque optimization, in *Proceeding of the International Conference on Robotics and Automation, 1985*, pp. 1016–1021.
9. C. Klein and C. Huang, Review of pseudoinverse control for use with kinematically redundant manipulators, *IEEE Trans. Systems Man Cybernet.* **7**, 1997, 868–871.
10. Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe, Planning motions with intentions, in *ACM Computer Graphics, Annual Conf. Series, 1994*, pp. 395–408.
11. K. Kondo, Inverse kinematics of a human arm, *J. Robot. Systems* **8**(2), 1991, 115–175.
12. J. Korein, *A Geometric Investigation of Reach*, Ph.D. thesis, University of Pennsylvania, 1985.
13. C. Lawrence, J. Zhou, and A. Tits, *User's Guide for CFSQP Version 2.5*, Technical report.

13a. J. Lee and S. Y. Shin, A hierarchical approach to interactive motion editing for human-like figures, in *Computer Graphics Proceedings (SIGGRAPH), 1999*, pp. 39–48.

14. D. Manocha and J. Canny, *Real Time Inverse Kinematics for General 6R Manipulators*, Technical report, University of California, Berkeley, 1992.
15. W. Maurel, D. Thalamann, P. Hoffmeyer, P. Beylot, P. Gingins, P. Kalra, and N. Thalmann, A biomechanical musculoskeletal model of human upper limb for dynamic simulation, in *Proceedings of the Eurographics Computer Animation and Simulation Workshop in Poitiers, France, 1996*, pp. 121–136.
16. Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*, Addison–Wesley, Reading, MA, 1991.
17. D. Pieper, *The Kinematics of Manipulators under Computer Control*, Ph.D. thesis, Stanford University, 1968.
18. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, Cambridge Univ. Press, Cambridge, UK, 1992.
19. M. Raghavan and B. Roth, Kinematic analysis of the 6R manipulator of general geometry, in *Proceeding of the 5th International Symposium on Robotics Research, 1990.*
20. H. Rieseler, H. Schrake, and F. M. Wahl, Symbolic computation of closed form solutions with prototype equations, in *Proceedings, Advances in Robot Kinematics, 1991*.
21. L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*, McGraw–Hill, New York, 1996.
22. J. F. Soechting and M. Flanders, Errors in pointing are due to approximations in sensorimotor transformations, *J. Neurophys*. **62**(2), 1989, 595–608.

22a. D. Tolani, *Analytical Inverse Kinematics Techniques for Anthropometric Limbs*, Ph.D. thesis, University of Pennsylvania, 1998.

23. X. Wang, A behavior-based inverse kinematics algorithm to predict arm prehension for computer-aided ergonomic evaluation, *J. Biomechan*. **32**, 1999, 453–460.

24. X. Wang and J.-P. Verriest, A geometric algorithm to predict the arm reach posture of computer-aided ergonomic evaluation, *J. Visualization Comput. Animat.* **9**, 1998, 33–47.

25. Z. Wang and K. Kazerounian, Application of symbolic computation in analytic determination of the null space of human arm kinematics, in *Proceedings, Advances in Robot Kinematics, 1991*.

26. D. E. Whitney, Resolved motion rate control of manipulators and human prostheses, *IEEE Trans. Man–Machine Systems* **10**, 1969, 47–63.

27. J. Zhao and N. Badler, Inverse Kinematics positioning using nonlinear programming for highly articulated figures, *Trans. Comput. Graph.* **13**(4), 1994, 313–336.

28. X. Zhao, *Kinematic Control of Human Postures for Task Simulation*, Ph.D. thesis, University of Pennsylvania, 1996.

29. X. Zhao and N. Badler, Near real-time body awareness, *Computer-Aided Design* **26**.