Efficient Spherical Joint Limits with Reach Cones*

Technical Report

Jane Wilhelms and Allen Van Gelder
wilhelms@cse.ucsc.edu, avg@cse.ucsc.edu
Computer Science Dept., University of California, Santa Cruz, CA 95064

April 17, 2001

Abstract

We describe a simple and fast method of creating and using joint sinus reach cones to limit the range of motion on universal and ball-and-socket joints. Reach cones can be created interactively or based on results from biomechanics. A reach cone is specified as a spherical polygon on the surface of a reach sphere that defines all positions the longitudinal segment axis beyond the joint can take. Reach polygons can be concave or convex, and cover more than one hemisphere. Longitudinal axis rotation limits can also be defined for locations in the reach cone. Calculations to determine whether a segment is within the reach cone are based on the relation of the segment longitudinal axis to the planes defining the reach cone, and are done in the local coordinate space of the segment. The calculations have no discernible effect on the speed of interactive display or animation.

Keywords: Computer Graphics: Modeling. Animation; Human Motion; Joint Limits.

1 Introduction

The task of animating humans and animals is greatly aided by automatic constraints, such as joint limits and collision detection, which provide natural restrictions on realistic motion. Such constraints are important in interactive placement, taking from the user some of the strain of recognizing unrealistic positions visually. They are essential in more automated methods, such as physical simulation, inverse kinematics, and motion capture from monocular video.

Generally, articulated body models used in computer graphics model joints with more than one degree of freedom (DOF) as a sequence of one DOF joints specified as Euler angles. The obvious way to specify a range of movement for one-DOF *hinge joints* is to give a minimum and maximum value in degrees. Joint limits on two- or three-DOF joints then become ranges around each single axis. If the longitudinal axis of a segment distal to the joint is the z-axis, a 2-DOF *universal joint* allows rotation about the x- and y-axes. If the joint also allows longitudinal z-axis rotation, it is a 3-DOF *ball-and-socket joint*. Single-axis range specification is inadequate for such joints.

A more natural specification, sometimes called a *joint sinus cone*, has been used in biomechanics [2, 3, 4, 14], simulation [10], and in computer graphics [7, 9]. Here the range of motion for universal joints is described as an irregular cone, defined by a cyclical sequence of points on the sphere that represents free universal movement.

In this paper, we describe a new method for specifying, recognizing inclusion in, and intersecting joint sinus cones. We refer to such limits as *reach cones*. The method works for reach cones covering more than one hemisphere. Calculations are so simple as to have no discernible impact on interaction or animation speed. We extend the reach-cone definition to include range limits on longitudinal rotation, because these limits generally depend on the orientation of the longitudinal axis. Source code for the main routines is available at `ftp://ftp.cse.ucsc.edu/pub/avg/Jtl`.

---

*An updated and condensed version, entitled “Fast and Easy Reach-Cone Joint Limits”, appears in *Journal of Graphics Tools* (6)2, 2002.

2 Background

An unrestricted articulated body would consist of joints all capable of their full 3 degree-of-freedom (DOF) movement, so that rotation of a joint would sweep out a sphere, and longitudinal axes could spin freely. Such a model is only possible in a virtual world, and realism requires that constraints be imposed. Joints are often simplified by removing some degrees of freedom. For example, the knee is generally treated as a 1-DOF *hinge* joint and the ankle as a 2-DOF *universal* joint. The shoulder and hip generally remain 3-DOF *ball-and-socket* joints. The range of motion within each degree of freedom must be further constrained. This is generally done by providing a maximum and minimum range of motion at each degree of freedom, though this is inadequate for 2- and 3-DOF joints.

In an interesting early paper, Korein and Badler [8] discussed inverse kinematics on universal and ball-and-socket joints and proposed to limit the reach space to be the interior of a *joint limit boundary curve*, which is shown as a spherical polygon in a figure. This figure inspired our present method. Korein further developed the idea as part of his Ph.D. thesis [7] and implemented it in a system named TEMPUS [1]. While there are general similarities between his work and ours, there are also significant differences, which we point out as our method is described. In many cases, Korein seems to have opted for simplicity at the expense of efficiency. No timing information is given in the thesis. For example, it seems[1] that Korein's procedures for several problems, such as testing whether a point is inside a spherical polygon, use trigonometric and inverse trigonometric functions. Our procedures for these problems do not require such functions. Korein's system for joint limits is more general than ours in some ways and less general in others.

Engin *et.al.* introduced a similar joint limit restriction in their biomechanical study of the shoulder [2, 3, 4]. In the biomechanical literature, such a reach space is called a *joint sinus*, specifying "the total range of angular motion permitted by a moving link of a joint when the other link is fixed" [14]. The shoulder complex is a particularly complicated region and the subject of considerable research in biomechanics. Sometimes four articulations are defined: the *sternoclavicular joint* (the clavicle at the sternum in front); the *scapulothoracic joint* (the scapula moving on the rib cage, in back); the *claviscapular joint* (the clavicle and scapula meeting near the arm); and the *glenohumeral joint* (the arm at the scapula).

Engin and Chen [2] treated the whole shoulder complex, for their measurements, as one joint and defined a joint sinus cone covering more than one hemisphere. They describe the cone in terms of spherical coordinates relative to a fixed thorax. Wang *et.al.* [14] extended that conical limit to include longitudinal (axial) rotation limits on the upper arm [14], because these limits are known to change depending on arm orientation. Silva *et.al.* [10] also used joint sinus cone limits in their physical simulation of a 12-segment human body for crash studies. They describe the cone boundaries by a cubic spline in latitude and longitude coordinates. Contact forces push the segment away from joint boundaries. Their simulation is not graphical or interactive.

Engin and Tumer [3, 4] modeled the shoulder complex as a sequence of three spherical joints: the *sternoclavicular* (proximal shoulder at spine); the *claviscapular* (distal shoulder); and the *glenohumeral* (shoulder to upper arm). By using a sequence of three joints, each joint sinus cone is less than one hemisphere.

Recently, Maurel and Thalmann [9] described a graphical model of the human shoulder complex which closely follows Engin and Tumer. Their joint sinus cone is created by drawing a 2D polygon representing a planar slice through it. They use a 2D point-in-polygon test to detect whether the distal segment is in the cone, and a 2D line-intersect-polygon test to detect where the segment should be placed on the boundary. This method limits individual joint sinus cones to one hemisphere or less, which is fine for this shoulder model. They state that use of this model with joint limits significantly slows interaction and animation.

The Maurel-Thalmann paper is the first use of such biomechanically based joint limits in graphical human modeling. However, we believe our approach offers some significant improvements over this and the Silva model [10]:

- our reach space is three-dimensional and can extend beyond one hemisphere, making it usable for any universal or ball-and-socket joint model;
- our conical limits include longitudinal rotation limits;
- our calculations for inclusion and intersection are three-dimensional, very fast, and have no discernible impact on interaction and animation speed;
- the method of specifying the reach space uses actual limb positions in three dimensions.

[1] We are not certain, because only high-level descriptions are given and his code is not available to us.

3 Method

To clarify our use of terms, a *joint* is an articulation between a *parent segment* and *child segment* of a tree-structured articulated body. The end of the segment closest to the root of the tree is called *proximal* and the other end is called *distal*. The parent segment is said to be *proximal* to the joint and the child segment is said to be *distal*. When not otherwise qualified, the term *segment* should be understood to mean the segment distal to the joint under discussion, i.e., the child segment. Each segment longitudinal axis is considered to be a (bounded) straight line segment originating at the joint with its parent segment.

A *default coordinate frame* defines the default position of a segment on its parent segment, while a *state coordinate frame* defines a rotation relative to the default position. The *longitudinal segment axis* is the z-axis in the state coordinate frame. Both coordinate frames have their origin at the joint. When a state (or default) coordinate frame is represented by Euler angles, they are (x, y, z) rotations where the z rotation is always a longitudinal axis rotation. In transforming a point defined in a local state coordinate frame back to world-space, rotations occur in the order z, then y, then x. In equations, **boldface** indicates a vector and *italics* a scalar variable.

Formally, a *cone* is a set of rays (or half-lines) emanating from the origin. That is, if a point p is in a given cone then all points of the form λp, where $\lambda \geq 0$, are in the cone. Thus a cone can be *represented* (or *defined*) by any set of points that generate it. We take advantage of this flexibility to represent the same cone in various ways, as convenient. One natural representation is a set of points on a sphere centered at the origin. Without any joint limit, the distal end of the segment might be anywhere on this sphere. With joint limits, the set of points that can actually be occupied by the distal end represents, or defines, the *sinus* or *reach cone*. Alternatively, the intersection of the sinus cone with this sphere defines the set of points that can be occupied by the distal end of the segment.

Our spherical joint limits are specified as a reach cone inscribed on a sphere of radius 1 centered at the joint, together with limits on z rotation that may vary throughout the reach cone. The reach cone is specified by a spherical polygon, called the *reach-cone polygon*. The vertices of this spherical polygon are a series of *boundary points* on this unit sphere and great-circle arcs on the sphere form its edges. Points inside or on the reach-cone polygon are in the reach cone. Reach cones are defined in the default coordinate frame of the segment distal to the joint. They do not move about with state rotations of this segment. Detection of allowable positions in the reach cone is done by testing whether the longitudinal segment axis intersects the unit sphere inside or outside the reach cone polygon. Note that this only depends on the x and y rotations applied to the segment.

At each vertex, limits on z rotation (i.e., rotation about the longitudinal segment axis, called *twist* by Korein) may be specified by maximum and minimum angles. Limits on z rotation at any position in the reach cone are defined by an interpolant of these values. This capability is important because it has been found that the range of motion for longitudinal rotations is a function of the direction of the longitudinal axis [14]. See Section 3.6 for further discussion.

3.1 The Visible-Point Restriction

We want to be able to discover quickly whether any new location of the longitudinal segment axis is within the reach cone. To facilitate this operation, we require the reach-cone polygon to have a *visible point*, that is, a point that can be "seen" by all of the boundary points in the sense that a great-circle arc (or line segment) joining the boundary point with the visible point lies entirely inside the reach cone (see Figure 1). Reach cones must be defined such that boundary points are in counter-clockwise order viewed from the outside, "above" the visible point. Also, the point on the sphere exactly opposite the visible point is not allowed to be in the reach cone.

Polygons for which such a visible point exists are called *star polygons*. Note that a star polygon need not be convex and can easily span more than one hemisphere, as is the case for our human models, as seen in Figures 6–7. For clarity, our explanatory figures show small reach cones.

In practice, we have found star polygons to be sufficiently general for realistic sinus cones. The method could be extended to define a very convoluted reach cone with several star polygons, but we have not done so. The system of spherical polygons in Korein's thesis is not limited to star polygons and consequently he has to solve problems differently from the solutions given here.

3.2 Creating a Reach Cone Interactively

Boundary points can be interactively created by the user or they can be read from a file. A boundary point is created interactively in our GUI by rotating the longitudinal z-axis of the body segment distal to the joint and making a

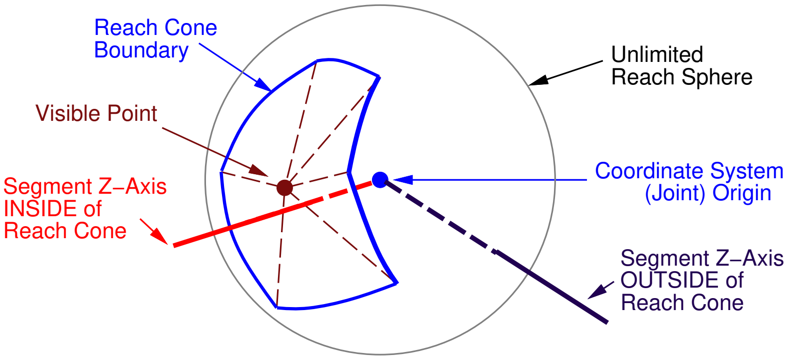

Figure 1: A reach-cone polygon with 5 boundary points and a visible point is shown.

selection with the mouse. The point $(0, 0, 1)$ (on the longitudinal segment axis) in the *rotated* segment state coordinate frame becomes a boundary point. The user can add and delete points at any location on the sphere, and designate the order of the points to define the reach-cone polygon.

As boundary points are being created interactively, a provisional visible point is calculated by adding the boundary points as 3D vectors and normalizing the result to unit length. If this provisional visible point does not fulfill the criteria for a visible point, or the user is not satisfied with it, then the user can position the longitudinal segment axis at a suitable point, and set that as the visible point.

To save the user from having to enter many positions to create a smooth reach cone, we provide the ability to subdivide the already-entered points to create a smoothly curving boundary, as described in Section 3.7. For many human joints, such as the wrist, the reach cone can be approximated well by an ellipse, and the cone angles are sometimes provided in the biomechanical literature. For these cases, the user can create four positions and subdivide once or twice to create an approximately elliptical reach space.

3.3 Detecting Reach Cone Inclusion

For efficient algorithms concerning reach cones we use the property that any set of points that generates the entire cone (by extending rays from the origin through these points) can be used to represent the cone. Suppose a reach-cone polygon has been defined with origin $\mathbf{O}$, visible point $\mathbf{V}$, and boundary points $\mathbf{P}_i$ for $i = 0, \ldots, n-1$. This notation is used throughout; the points are treated as 3D vectors from the origin and arithmetic on indexes is understood to be modulo n. The vector cross product and dot product, denoted by "$\times$" and "$\cdot$," respectively, permit many operations to be expressed in a coordinate-free manner.

Observe that for each i, the four points $(\mathbf{O}, \mathbf{V}, \mathbf{P}_i, \mathbf{P}_{i+1})$ define a tetrahedron. The order of the points imparts an orientation to the tetrahedron. These n tetrahedra also generate the reach cone, so can be used as an alternative representation. The volume of an oriented tetrahedron with one point at the origin is given by the *triple scalar product* expression [12]:

$$\text{vol}(\mathbf{O}, \mathbf{a}, \mathbf{b}, \mathbf{c}) \quad = \quad \tfrac{1}{6}\mathbf{a} \times \mathbf{b} \cdot \mathbf{c}. \tag{1}$$

The visible point is properly positioned with respect to the boundary points if and only if each of the n oriented tetrahedra has positive volume; that is,

$$\mathbf{V} \times \mathbf{P}_i \cdot \mathbf{P}_{i+1} > 0 \qquad \text{for } 0 \leq i \leq n-1.$$

We consider the problem of deciding whether a specified vector $\mathbf{L}$ is in the reach cone. A segment is considered to be in the reach cone if the z-axis of the segment's state coordinate frame is in the cone. To detect this, we define the unit vector $\mathbf{L}$ in the direction of this z-axis, that is $\mathbf{L} = (0, 0, 1)$ in the segment's state coordinate frame.

Clearly, $\mathbf{L}$ is in the reach cone if and only if $\mathbf{L}$ passes through one of the n tetrahedra that define the reach cone. Also, $\mathbf{L}$ passes through the tetrahedron $(\mathbf{O}, \mathbf{V}, \mathbf{P}_i, \mathbf{P}_{i+1})$ if and only if each of the three oriented tetrahedra, $(\mathbf{O}, \mathbf{V}, \mathbf{P}_i, \mathbf{L})$, $(\mathbf{O}, \mathbf{P}_i, \mathbf{P}_{i+1}, \mathbf{L})$, and $(\mathbf{O}, \mathbf{P}_{i+1}, \mathbf{V}, \mathbf{L})$ has nonnegative volume, using Equation 1.

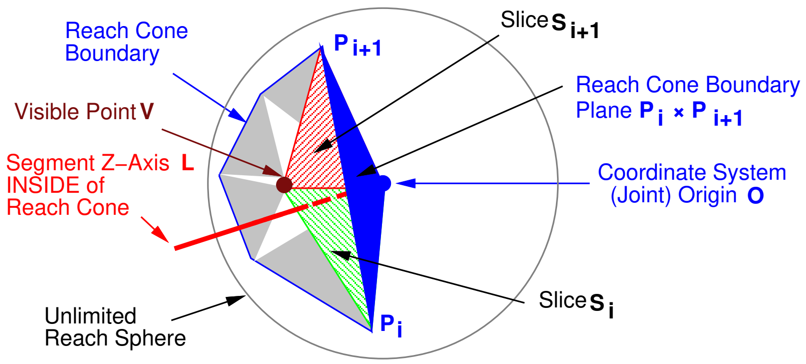

Figure 2: A reach cone with five boundary points showing slice planes $\mathbf{S}_i$ and $\mathbf{S}_{i+1}$ and the boundary plane defined by $\mathbf{O}, \mathbf{P}_i, \mathbf{P}_{i+1}$ that enclose the longitudinal segment axis vector $\mathbf{L}$.

Another way to view this is to consider the plane defined by $\mathbf{O}, \mathbf{V}, \mathbf{P}_i$, which has a normal vector (not necessarily unit length) $\mathbf{V} \times \mathbf{P}_i$. If $\mathbf{L}$ is on the same side of the plane as this normal vector points, then $\mathbf{V} \times \mathbf{P}_i \cdot \mathbf{L} \geq 0$. The plane defined by $\mathbf{O}, \mathbf{V}, \mathbf{P}_i$ is called a *radial slice plane* and is denoted by $\mathbf{S}_i$ (see Figure 2). Similarly, for $\mathbf{L}$ to pass through the tetrahedron $(\mathbf{O}, \mathbf{V}, \mathbf{P}_i, \mathbf{P}_{i+1})$ it is necessary that $\mathbf{V} \times \mathbf{P}_{i+1} \cdot \mathbf{L} \leq 0$ and $\mathbf{P}_i \times \mathbf{P}_{i+1} \cdot \mathbf{L} \geq 0$. Note that the cross products depend on the boundary points and visible point, but not on $\mathbf{L}$, so they can be computed just once, when the reach cone is specified:

$$\begin{aligned} \mathbf{S}_i &= \mathbf{V} \times \mathbf{P}_i \\ \mathbf{B}_i &= \mathbf{P}_i \times \mathbf{P}_{i+1} \end{aligned} \tag{2}$$

To summarize, the procedure to decide whether $\mathbf{L}$ is in the reach cone follows.

inReachCone(L)

1. Find the i such that $p_i = \mathbf{S}_i \cdot \mathbf{L} \geq 0$ and $p_{i+1} = \mathbf{S}_{i+1} \cdot \mathbf{L} < 0$. There is exactly one such i because the radial slices (as half planes) partition the sphere.[2]

   We start with the slice in which the axis was located previously, which is often where it is now. In the worse case, this step takes n dot-product operations, since the cross products are stored. We could speed this up by doing a binary search on the radial slice planes, but it didn't seem worth the coding time.

2. If $v_i = \mathbf{B}_i \cdot \mathbf{L} \geq 0$, then $\mathbf{L}$ is in the reach cone, otherwise it is not.

The values p_i, p_{i+1}, and v_i are retained for later possible interpolation; see Section 3.6.

A different method for testing point inclusion in a spherical polygon is sketched in Korein's thesis without proof. We summarize it here. To test $\mathbf{L}$ for inclusion, pass a great circle C through $\mathbf{P}_0$ and $\mathbf{L}$. Find the edges of the spherical polygon that intersect C. Among these, find the edge $(\mathbf{P}_i, \mathbf{P}_{i+1})$ whose intersection with C is closest to $\mathbf{L}$. There are several cases to consider, according to whether the closest intersection is exactly at $\mathbf{P}_0$, giving two candidate edges, $(\mathbf{P}_0, \mathbf{P}_1)$ and $(\mathbf{P}_{n-1}, \mathbf{P}_0)$. In the latter case consider the absolute spherical angles from $(\mathbf{P}_0, \mathbf{L})$ to $(\mathbf{P}_0, \mathbf{P}_1)$ and from $(\mathbf{P}_0, \mathbf{L})$ to $(\mathbf{P}_0, \mathbf{P}_{n-1})$. (Spherical angles are the angles between great-circle arcs, not chords.) Use $(\mathbf{P}_{n-1}, \mathbf{P}_0)$ if the second angle is smaller; otherwise use $(\mathbf{P}_0, \mathbf{P}_1)$. Having chosen $(\mathbf{P}_i, \mathbf{P}_{i+1})$ as the closest edge, look at the sphere from the outside and directly above $\mathbf{P}_i$. If $\mathbf{L}$ is counterclockwise from $\mathbf{P}_{i+1}$, then it is inside the spherical polygon. Korein defines a function named *RhAngle* to give the signed spherical angle and expresses the condition of being counterclockwise in terms of this angle being positive. This method is not limited to star polygons.

[2] Recall that a half plane consists of all points in a given plane on one side of a given line. In this case the given line is $\mathbf{V}$ and the side is that in which $\mathbf{P}_i$ lies. There is also a unique j such that $\mathbf{V} \times \mathbf{P}_j \cdot \mathbf{L} < 0$ and $\mathbf{V} \times \mathbf{P}_{j+1} \cdot \mathbf{L} \geq 0$, but this is ignored.

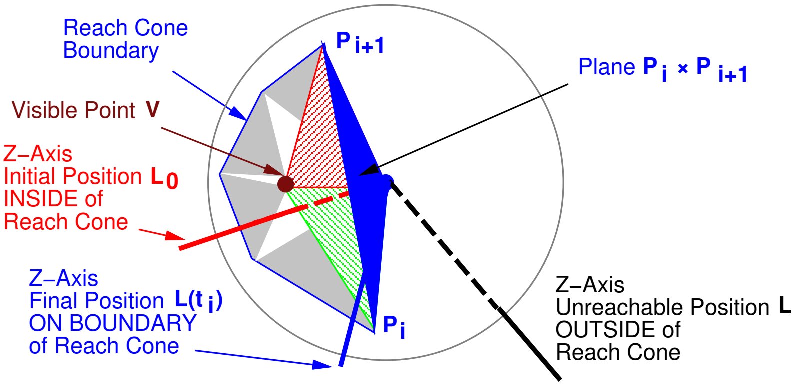

Figure 3: The reach cone from Figure 2 showing an initial z-axis position $\mathbf{L}_0$ inside the reach cone, a tested z-axis position $\mathbf{L}$ found to be outside the reach cone, and the intersection $\mathbf{L}(t_i)$ of the line between the two with the reach cone boundary. This intersection provides a new orientation for the z-axis.

3.4 Calculating a Boundary Position

In many applications, such as inverse kinematics, physical simulation, or motion capture from video, when the (proposed) new segment orientation is outside the reach cone, it might be useful to find where the path from the old segment position to the new position first exits the reach cone. Then a valid new position based on this exit point can be chosen. The situation is shown in Figure 3. This problem is not considered in Korein's thesis, but it might be implemented by a variant of his inclusion test, summarized at the end of Section 3.3,

In the case that the new position is not in the reach cone, we assume that the segment is going to rotate directly from its present position to the new one, in the plane defined by the origin and the old and new positions. We assume that the rotation is less than 180 degrees. The exit point is the first point reached on the shorter great-circle route from the old position $\mathbf{L}_0$ to the new position $\mathbf{L}$, which has been found to be outside the reach cone. However, it is sufficient, and much simpler, to find the exit point on a straight line from $\mathbf{L}_0$ to $\mathbf{L}$. The test is in the opposite order of that described for inclusion in the reach cone: first we test against the outer reach cone boundaries, and then against the slice planes.

We test a line $\mathbf{L}(t)$ from the present (old) position of the segment z-axis ($\mathbf{L}_0$) to the new position $\mathbf{L}$ for intersection with the reach cone boundary planes. The boundary planes B_i are defined by the precomputed cross products $\mathbf{B}_i$ from Equation 2 in Section 3.3. Homogeneous coordinates are all 0 because the planes pass through the origin. The algorithm is that described by Foley *et al.* [6].

$$\mathbf{L}(t) \quad = \quad \mathbf{L}_0 + t(\mathbf{L} - \mathbf{L}_0) \tag{3}$$

The value of t where the line intersects the plane is:

$$t_i \quad = \quad \frac{-\mathbf{L}_0 \cdot \mathbf{B}_i}{(\mathbf{L} - \mathbf{L}_0) \cdot \mathbf{B}_i} \tag{4}$$

For this discussion a slice is the region between two consecutive radial slice planes, treated as half planes. If $0 < t_i < 1$, an intersection occurred. (If $t_i = 0$ we would have been inside, and if $t_i = 1$ we know we are outside.) We use the line equation (Equation 3) to find the intersection point. We further test whether the intersection point is within the slice defined by points $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$, using the same test as in step 1 of the procedure **inReachCone** in Section 3.3. If it is, the intersection point is a *candidate exit point*.

For nonconvex reach cones, there may be several candidate exit points. We need to be sure that the candidate exit point chosen is the one with the smallest value of t. We start at the slice $\mathbf{S}_i$ that contains the old point $\mathbf{L}_0$, and find the value of t_i there. If an intersection point is found in this slice, it must have the minimum value of t_i, and we are done.

Otherwise, we compute $\mathbf{L}_0 \times \mathbf{L} \cdot \mathbf{V}$. If this triple scalar product, is negative, we continue around the reach cone in the direction of decreasing indexes, because this is the sequence in which $\mathbf{L}(t)$ passes through the slices as t increases; otherwise, we continue in the direction of increasing indexes, because $\mathbf{L}(t)$ passes through slices in this sequence as t

increases. In each case the first candidate exit point discovered will actually be the first exit point from the reach cone. The value of t is also used to interpolate the z-axis rotation.

For a quick upper bound, assume all n slices need to be processed completely. Computing $\mathbf{L} - \mathbf{L}_0$ is 3 floating-point operations (flops). Computing t_i requires 11 flops. Finding the intersection point $\mathbf{L}(t_i)$ is 6 flops. Testing for slice inclusion is 10 flops. The one-time triple scalar product is 14 flops. The total is $30n + 14$ flops. Finally, the exit point $\mathbf{L}(t)$ has to be normalized to unit length to find the corresponding point on the sphere. Finding the point where $\mathbf{L}(t)$ first exits the reach cone is one of our more expensive operations, but still not weighty. As it is only used in conjunction with automated methods such as inverse kinematics, it has a negligible impact.

3.5 Converting Between Positions and Euler Angles

In general, we need to find new Euler angles $(\theta_x, \theta_y, \theta_z)$ to move the segment to its new position, and store the state frame orientation. To do this, we find a rotation from the present (old) z-axis position ($\mathbf{L}_0$) to the new position ($\mathbf{L}$) (which may have been reset to the point where $\mathbf{L}$ exited the reach cone; see Section 3.4). The axis of rotation is the cross-product of these two vectors, and amount of rotation given by their dot product. Compositing the arbitrary rotation matrix for this rotation with the present state rotation matrix gives a matrix representing a direct move to the new position (These are standard techniques [6].)

To extract the new Euler angles from the composite matrix, we also use the standard method of extracting the known $\sin\theta_y$ from the matrix

$$\begin{bmatrix} \cos\theta_y\cos\theta_z & \cos\theta_x\sin\theta_z + \sin\theta_x\sin\theta_y\cos\theta_z & \sin\theta_x\sin\theta_z - \cos\theta_x\sin\theta_y\cos\theta_z \\ -\cos\theta_y\sin\theta_z & \cos\theta_x\cos\theta_z - \sin\theta_x\sin\theta_y\sin\theta_z & \sin\theta_x\cos\theta_z + \cos\theta_x\sin\theta_y\sin\theta_z \\ \sin\theta_y & -\sin\theta_x\cos\theta_y & \cos\theta_x\cos\theta_y \end{bmatrix} \tag{5}$$

and using that to find the rest of the Euler rotations. Because of the ambiguity in the sign of $\cos\theta_y$, we extract Euler angles using both negative and positive assumptions, and use the choice that is closest to the Euler angles that specify the old orientation. The comparison is done using the squared distance between old and new Euler angles.

Because this process is often necessary in moving to a new position without using joint limits, e.g., in using a virtual sphere or inverse kinematics, we don't consider this a part of the cost of our joint limit algorithm.

3.6 Reach Cone Z-Axis Limits

Limits on rotation of the z-axis (longitudinal axis of the segment) can be specified as a minimum-maximum range. However, the z-axis range can change considerably, depending on orientation of the axis in the reach cone. For the human shoulder, Wang *et al.* found that the range of axial rotation of the upper arm might be as low as 94 degrees or as high as 157 degrees, depending on the upper arm orientation [14]. Therefore, we also allow the user to specify longitudinal rotation limits for each reach cone boundary point and the visible point. We interpolate these values to find limits on rotation about the longitudinal axis for each orientation in the reach cone.

This is one area in which our system is more general than Korein's, which permits only one minimum-maximum range per joint for z-axis rotation: the range cannot depend on the orientation of the segment. The reason that Korein imposed this restriction might be that there is no clear-cut way to interpolate within his more general spherical polygons. Our additional visible point provides a natural and simple method.

Fortunately, the calculations to locate the segment z-axis point $\mathbf{L}$ within two radial slices and one boundary slice (in Section 3.3) provide a form of barycentric coordinates [5] for interpolation across the spherical triangle defined by boundary points $\mathbf{P}_i$, $\mathbf{P}_{i+1}$ and visible point $\mathbf{V}$, where these points surround $\mathbf{L}$. The values p_i, p_{i+1}, and v_i that were computed in the **inReachCone** procedure can be used to compute weights for interpolation, as follows:

$$\begin{aligned} s &= p_i + p_{i+1} + v_i \\ w_i &= \frac{p_i}{s} \\ w_{i+1} &= \frac{p_{i+1}}{s} \\ w_V &= \frac{v_i}{s} \end{aligned} \tag{6}$$

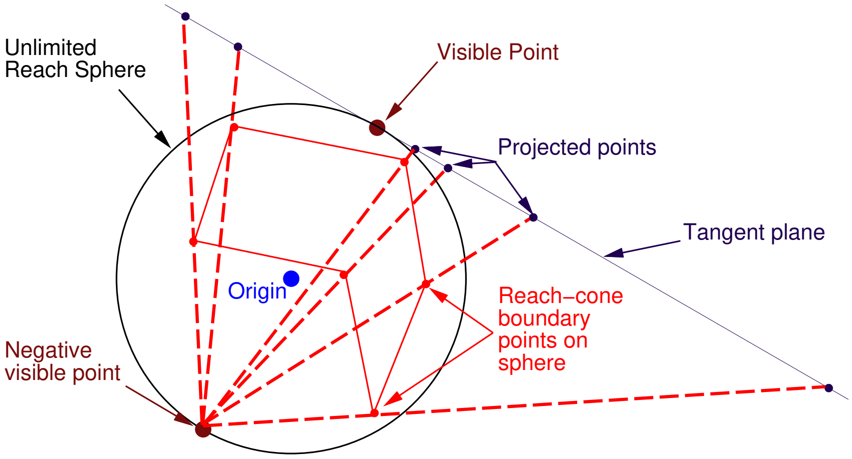

Figure 4: The stereographic projection provides an invertible mapping between the entire sphere (except for one point) and the tangent plane.

Each boundary point and the visible point have z-axis limits θ_{min} and θ_{max} associated with them. We interpolate them weighted by the barycentric coordinates to find the limits at $\mathbf{L}$.

$$\begin{aligned} \theta_{min}(\mathbf{L}) &= w_i\,\theta_{min}(\mathbf{P}_i) + w_{i+1}\,\theta_{min}(\mathbf{P}_{i+1}) + w_V\,\theta_{min}(\mathbf{V}) \\ \theta_{max}(\mathbf{L}) &= w_i\,\theta_{max}(\mathbf{P}_i) + w_{i+1}\,\theta_{max}(\mathbf{P}_{i+1}) + w_V\,\theta_{max}(\mathbf{V}) \end{aligned} \tag{7}$$

The cost is 15 floating-point operations.

In the current implementation, if the z-rotation at $\mathbf{L}$ lies outside these limits, it is replaced with the appropriate limit value. However, more sophisticated uses of these limits are possible, particularly in inverse kinematics or motion capture, such as penalties for being close to θ_{min} or θ_{max}. Also, v_i measures closeness to the reach-cone boundary and penalties might be assigned for small values of v_i.

3.7 Smoothing the Reach Cone

To save the user from having to enter many positions to create a smooth reach cone, we provide the ability to subdivide the already-entered points to create a smoothly curving boundary. We outline this feature briefly because it uses an invertible mapping between the entire sphere (except for one point) and the plane that appears not to be well known in the graphics literature. The most popular mappings cover only one hemisphere.

The *stereographic projection* covers the entire sphere, except for one point, and is well known in analysis of complex variables [11]. Our contribution is to give coordinate-free representations for both the forward and inverse mappings. Figure 4 illustrates its application to our situation. The *tangent plane* is defined to be the plane that is tangent to the unit sphere at the visible point. Observe that any ray cast from the *negative* visible point into the sphere intersects the sphere exactly once and intersects the tangent plane exactly once. Also, every point on the sphere, except the negative visible point, and every point on the tangent plane is intersected by some such ray.

Each boundary point of the existing reach cone is mapped into a *projected point* on the plane that is tangent to the unit sphere at the visible point. The projected points form a planar star polygon with the same visible point. Additional points can be defined for the planar polygon by any convenient interpolation method that gives a new planar star polygon. The vertices of the new planar star polygon can then be mapped back onto the unit sphere to define a new reach-cone polygon.

As suggested by Figure 4, the tangent plane is not necessarily in a convenient orientation for any of the coordinate frames in use. Therefore it is useful to have coordinate-free representations for both the forward and inverse mappings. Let us denote the negative visible point by $\widetilde{\mathbf{V}} = -\mathbf{V}$. As usual, points are also treated as vectors from the origin. Although we work with a unit sphere in our application, we give the formulas for a sphere of radius r for generality. Thus the lengths of $\mathbf{V}$ and $\widetilde{\mathbf{V}}$ are r also in these formulas. Let $\mathbf{p}$ be a point on the sphere of radius r centered at the

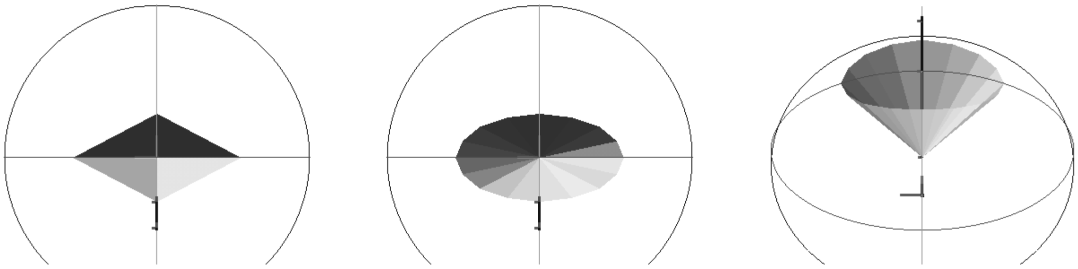

Figure 5: Left: initial reach cone with four boundary points at 30° East, 15° North, 30° West, 15° South. Center: after two smoothing steps. Right: a different view.

origin, and let $Q(\mathbf{p})$ denote the mapping into the tangent plane. We compute $Q(\mathbf{p})$ by

$$\begin{aligned} \lambda &= \mathbf{V} \cdot \mathbf{p} \\ \mu &= 2r^2/(r^2 + \lambda) \\ Q(\mathbf{p}) &= \mu\mathbf{p} + (1-\mu)\widetilde{\mathbf{V}} \end{aligned} \tag{8}$$

Note that this is actually an extrapolation formula because $\mu \geq 1$. Of course, $Q(\widetilde{\mathbf{V}})$ is not defined.

If $\mathbf{q} = Q(\mathbf{p})$ is any point in the tangent plane, the component of $\mathbf{q}$ (regarded as a vector) that is parallel to the tangent plane is simply $\mathbf{q} - \mathbf{V}$.

For the inverse mapping, let $\mathbf{q}$ be any point in the tangent plane. We compute $Q^{-1}(\mathbf{q})$ by

$$\begin{aligned} \lambda &= (\mathbf{q} - \mathbf{V}) \cdot (\mathbf{q} - \mathbf{V}) \\ \mu &= 4r^2/(4r^2 + \lambda) \\ Q^{-1}(\mathbf{q}) &= \mu\mathbf{q} + (1-\mu)\widetilde{\mathbf{V}} \end{aligned} \tag{9}$$

In this case $0 < \mu \leq 1$, so we have an interpolation formula.

Correctness of Equations 8 and 9 is easily established. Since they are extrapolation and interpolation formulas, it is only necessary to check the following:

1. If $\mathbf{p} \cdot \mathbf{p} = r^2$, then $Q(\mathbf{p}) \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{V}$, that is, $Q(\mathbf{p})$ is in the tangent plane.

2. If $\mathbf{q} \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{V}$, then $Q^{-1}(\mathbf{q}) \cdot Q^{-1}(\mathbf{q}) = r^2$, that is, $Q^{-1}(\mathbf{q})$ is on the sphere of radius r.

These conditions follow by straightforward application of vector-calculus methods.

Assume there are k boundary points and we want to interpolate k more. In the projected plane, say they are $\mathbf{q}_0$, …, $\mathbf{q}_{k-1}$, To put a new vertex between $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$, we use this formula, where indexes are interpreted mod k and α_k is a scalar:

$$\mathbf{q}_{i+1/2} \quad = \quad \tfrac{1}{2}\left(\mathbf{q}_i + \mathbf{q}_{i+1}\right) + \alpha_k\left(\mathbf{q}_{i+1} - \mathbf{q}_{i-1}\right) + \alpha_k\left(\mathbf{q}_i - \mathbf{q}_{i+2}\right) \tag{10}$$

(11)

k	α_k	k	α_k
3	.1667	6	.0773
4	.1036	7	.0700
5	.0850	8+	.0625

The value of α_8 is based on a cubic spline fit and the values for α_3 through α_7 are designed to cause an initial rhombus or isosceles triangle to smooth approximately to an ellipse. Figure 5 illustrates our subdivision procedure on an initial reach-cone polygon that is a spherical rhombus. We should add that certain pathological star polygons can smooth into self-intersecting nonstar polygons with this method. We do not believe that poses a practical problem.

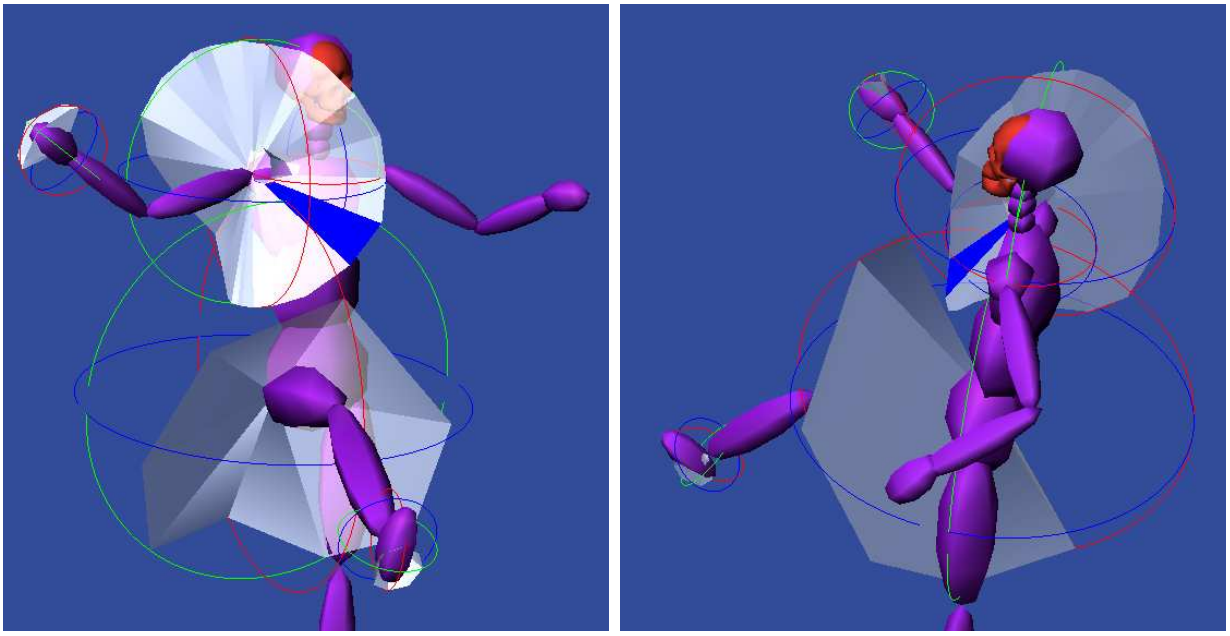

Figure 6: The reach cones of the right side of the body from two views. The blue triangle indicates which slice of the reach cone contains the upper arm.

4 Results

We tested our joint limits using a human model based on the *fauna* software [15] in its upgraded version *zoo*. In this model, all joints are inherently three degrees-of-freedom unless limits are imposed. The joints of the vertebral column, medial shoulder joints (where shoulders meet the vertebral axis), wrists, and ankles have reach cones that are roughly elliptical defined by eight boundary points. Joints between the shoulder and the upper arm and between the hip and the upper leg have more complex reach spaces. Figure 6 shows the reach cones on the right limbs. For most of the joints, limits were created interactively based on observations. The shoulder-arm complex, however, is based on published data [2, 4, 14, 13].

Our arm-shoulder complex has a 2-DOF universal joint for a medial shoulder joint, a 3-DOF ball-and-socket joint at the distal shoulder, a 1-DOF hinge joint at the elbow, and a three-DOF ball-and-socket joint at the wrist. We put the longitudinal rotation causing supination-pronation of the lower arm at the wrist rather than the elbow, because the effect of the rotation is largely there, and it is better at simulating soft tissue deformations. The lower arm is a strange segment, in that the longitudinal rotation is caused by the two lower arm bones (radius and ulna) crossing over.

Figure 7 shows the reach cone of the shoulder-upper-arm joint, which is defined by 25 boundary points. For this joint, the allowable range of z-axis rotation is defined at each boundary point and at the visible point, based on published values [14]. On the boundary polygons of the reach cone, the range of z-axis rotation values is shown with color coding, where green is a smaller range and red is a larger range. The z-axis range varies from 157 degrees when the arm is down, to 94 degrees when the arm is raised vertically. The elbow hinge joint (the x-axis in our case) has a range of 0 degrees extension to 142 degrees flexion. The elbow y- and z-axes cannot rotate (joint limits 0–0). The wrist joint has a reach cone associated with it. This cone, also shown in Figure 7, is small and roughly elliptical and approximated by 8 boundary points. The z-axis limits for the wrist are simple minimum-maximum limits between -60 degrees and 100 degrees, where 0 degrees is supinated.

Figure 8 shows the arm bounded by the three triangles used to locate it. Red and green are bounding slices and blue is the reach-cone boundary plane. If the z-axis of the segment being tested were on or outside the boundary, this plane would be shown in black.

As mentioned above, the use of joint limits has no discernible effect on interactive speeds, and the calculations are actually much fewer than those involved in displaying the figures, or interactively changing joint angles using a virtual

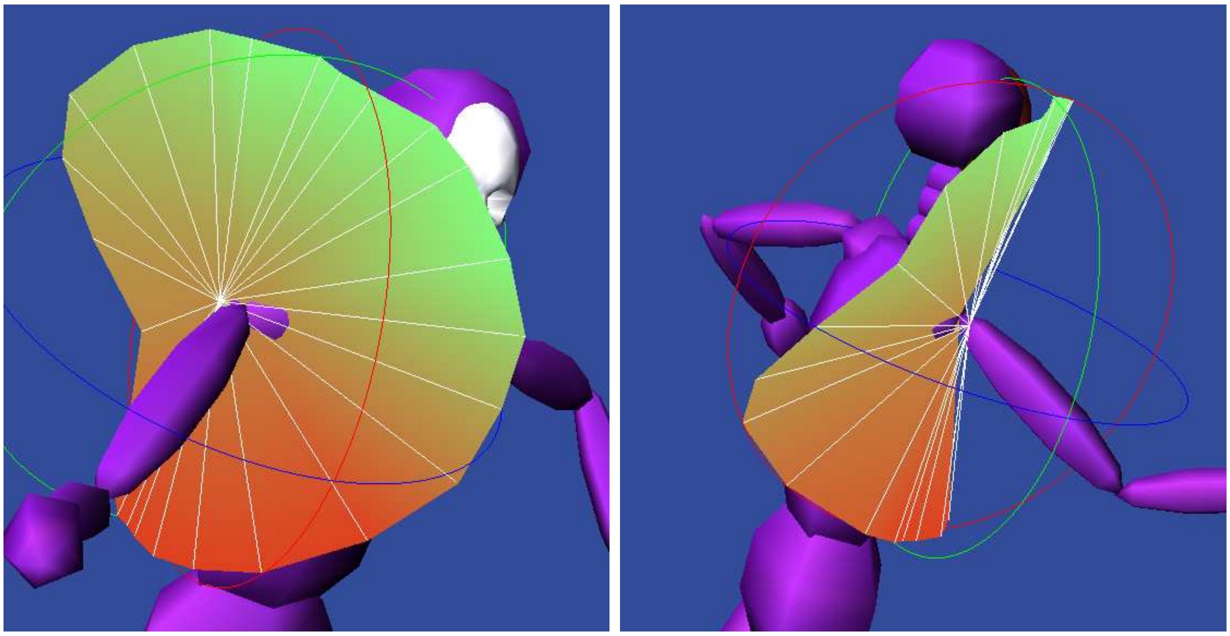

Figure 7: The reach cone of the upper arm from two views. The range of allowable z-axis rotation is color coded. Green denotes a lesser range and red denotes a larger range.

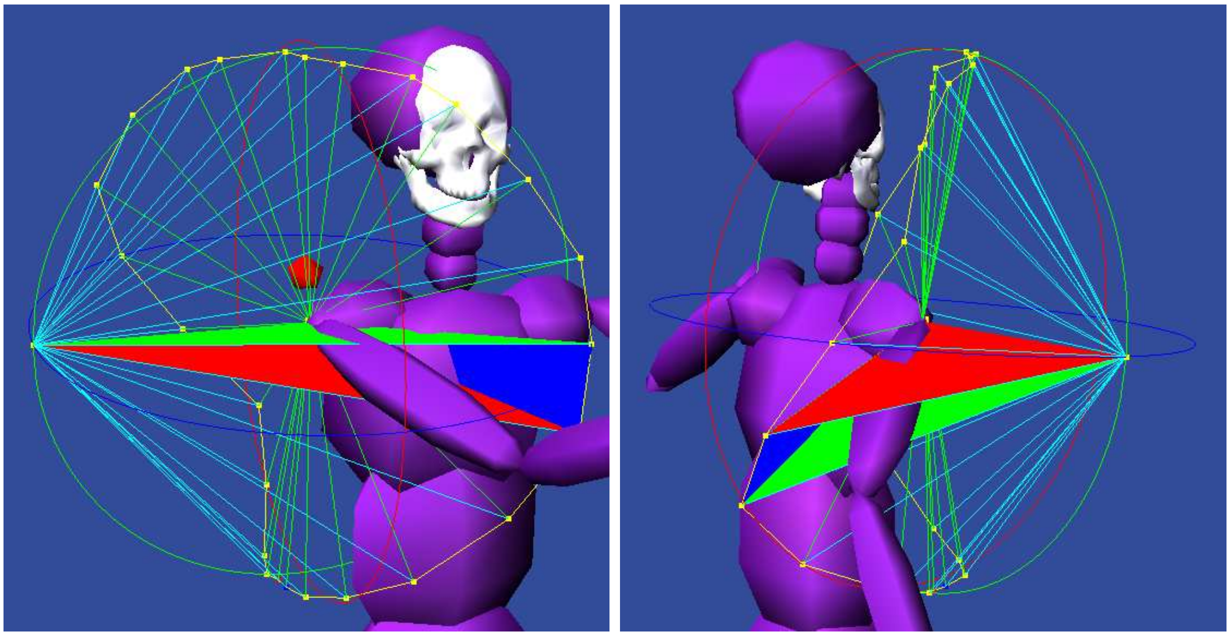

Figure 8: The arm in two positions, each bounded by three triangles used to locate it: red and green are consecutive radial slices in counterclockwise order; blue is the reach-cone boundary.

	Number of Executions	Total Time (secs.)	Average Time (μ-secs.)
With joint limits	166,027	28.70	173
Without joint limits	131,417	21.42	163
Ratio	1.26	1.34	1.06

Table 1: Times for inverse kinematics calculations with and without joint limits. Times are for an SGI Octane2 (360 MHz R12000).

sphere. However, to obtain a more quantitative evaluation, we did some timing tests for joint limits in our inverse kinematic procedure with all drawing disabled. Results are shown in Table 1.

The times are based on 10000 randomly generated goals, a chain of three segments, with reach-cone limits on two segments and Euler-angle limits on one. Each goal involves several iterative executions of the inverse-kinematics procedure for each segment in the chain. With joint limits active, about one fourth of the executions required finding a boundary intersection; that is, the attempted motion would have violated the joint limits, so a partial motion was computed. About half of the goals were too far away for the chain to succeed even without joint limits; in these cases the chain tried to get its distal end as close as possible to the goal. In this experiment the joint-limit overhead per procedure execution is about six percent. This value should remain stable across many applications of inverse kinematics, including multiple goals and chains. In addition, the presence of joints limits increased the number of executions about 26 percent, indicating slower convergence. We expect this value to be less stable across various applications.

5 Conclusions

We have described a method of limiting joint motion based on reach cones for joints of more than one degree of freedom. Reach cones are far more realistic than simple range limits on Euler angles. Calculations are very simple and have no discernible effect on interactions or animation. They provide an important tool for improving the realism of motion generated by automatic methods such as inverse kinematics and physical simulation.

The results presented here aim at explaining an improved approach to joint control, and only loosely approximate the actual range of motion of most joints. More realistic results can be created by using published joint data from the biomechanics literature.

References

[1] N. I. Badler, J. Korein, J.U. Korein, G. Radack, and L. Brotman. TEMPUS: A system for the design and simulation of human figures in a task-oriented environment. In *Proceedings of the First Annual Workshop on Robotics and Expert Systems*, June 1985.

[2] A. E. Engin and S.-M. Chen. Statistical data base for the biomechanical properties of the human shoulder complex. I. Kinematics of the shoulder complex. *Journal of Biomechanical Engineering*, 108:215–21, 1986.

[3] A. E. Engin and S. T. Tumer. Three-dimensional kinematic modelling of the human shoulder complex. I. Physical model and determination of joint sinus cones. *Journal of Biomechanical Engineering*, 111:107–12, 1989.

[4] A. E. Engin and S. T. Tumer. Three-dimensional kinematic modelling of the human shoulder complex. II. Mathematical modelling and solution via optimization. *Journal of Biomechanical Engineering*, 111:113–21, 1989.

[5] G. Farin. *Curves and Surfaced for Computer Aided Geometric Design*. Academic Press, San Diego, CA, 1990.

[6] James D. Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, Reading, Mass., 2nd edition, 1990.

[7] James U. Korein. *A Geometric Investigation of Reach*. ACM Distinguished Dissertations. MIT Press, Cambridge, MA, 1985.

[8] James U. Korein and Norman I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE Computer Graphics and Applications*, 2(9):71–81, November, 1982.

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

[10] M.P.T. Silva, J.A.C. Ambrosio, and M.S. Pereira. Biomechanical model with joint resistance for impact simulation. *Multibody System Dynamics*, 1:65–84, 1997.

[11] Richard A. Silverman. *Introductory Complex Analysis*. Dover, New York, 1972.

[12] Allen Van Gelder. Efficient computation of polygon area and polyhedron volume. In *Graphics Gems V*. Academic Press, Boston, Mass., 1998.

[13] X. Wang. A behavior-based inverse kinematics algorithm to predict arm prehension postures for computer-aided ergonomic evaluation. *J. of Biomechanics*, 32:453–460, 1999.

[14] X. Wang, M. Maurin, F. Mazet, N. De Castro Maia, K. Voinot, J.-P. Verriest, and M. Fayet. Three-dimensional modelling of the motion range of axial rotation of the upper arm. *J. of Biomechanics*, 31:899–908, 1998.

[15] Jane Wilhelms and Allen Van Gelder. Anatomically based modeling. In *Computer Graphics*, pages 173–180, Los Angeles, Ca., August 1997. ACM Siggraph Conference Proceedings.