# Fast and Easy Reach-Cone Joint Limits

Jane Wilhelms and Allen Van Gelder
Computer Science Dept., University of California, Santa Cruz, CA 95064 *

August 2, 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 and Background

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.

A body segment attached to an unrestricted (*ball-and-socket*) joint is capable of three degrees-of-freedom (DOF) of movement. Rotations of two of these DOF sweep out a sphere; the third DOF is for rotation around the longitudinal axis of the segment. Removing degrees of freedom produces two-DOF *universal* joints lacking longitudinal rotation and one-DOF *hinge* joints. Often, articulated body models used in computer graphics have modeled joints as a sequence of one-DOF joints specified as Euler angles.

For realism, further *joint limit* restrictions within degrees of freedom must be specified. Joint limits on one-DOF joints are given as a maximum-minimum range pair. Joint limits on two- and three-DOF joints represented by Euler angles then become independent ranges around each single axis. However, independent range specification is inadequate for joints with more than one DOF [1]. A more natural specification, sometimes called a *joint sinus cone*, has been used in biomechanics, simulation, and in computer graphics. Here the range of motion for universal joints is described as an irregular cone, defined by a closed curve (and its interior) on the sphere that represents free universal movement.

James Korein developed the idea of reach cones as part of his Ph.D. thesis [3] and implemented it in the system named TEMPUS. While there are general similarities between his work and ours, there are also significant differences. In many cases, Korein seems to have opted for simplicity at the expense of efficiency. No timing information is given in the thesis and his code is not available to us. For example, the thesis solves the problem of testing whether a point is inside a spherical polygon and related problems by using 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. For further details, please see the supplemental technical report [8].

*wilhelms@cse.ucsc.edu, avg@cse.ucsc.edu

Engin and Chen developed a biomechanical database in which they treated the whole shoulder complex, for their measurements, as one joint and defined a joint sinus cone covering more than one hemisphere [1]. Wang *et al.* extended the conical limit to include limits on axial rotation (i.e., rotation around the longitudinal axis, or $z$-axis) of the upper arm, because these limits were found to change depending on arm orientation [6]. The model discussed in Section 3 uses data from both these sources.

Recently, Maurel and Thalmann [4] described a graphical model of the human shoulder complex which closely follows Engin and Tumer; their work is significant for being the first use of biomechanically based joint limits in graphical human modeling. Their reach cone is created by having the user draw a 2D polygon representing a planar slice through it; such reach cones are inherently limited to one hemisphere. They use a planar point-in-polygon test to detect whether the segment is in the cone, and a planar line-intersect-polygon test to detect where the trajectory of a segment intersects the boundary. They state that use of this model with joint limits significantly slows interaction and animation. Our approach offers greater generality with less overhead.

In this paper, we describe a new method for specifying joint sinus cones and give procedures for recognizing inclusion of a point in the cone and for intersecting a great circle trajectory with the cone boundary. We refer to such joint 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 axial rotation (i.e., rotation around the longitudinal axis of the segment), because these limits generally vary with the orientation of the longitudinal axis [6].

### Web Information

At `http://www.acm.org/jgt/papers/WilhelmsVanGelder01` there are supplements for this paper consisting of C code for the main routines, C procedures to construct a reach cone directly in 3D, and a longer technical report with more extensive bibliography and background [8].

## 2 Method

After defining some key terms (Section 2.1) and describing the star-polygon restriction on our method (Section 2.2), we explain how to detect inclusion in a reach cone (Section 2.3), how to calculate the boundary position where a trajectory exits the reach cone (Section 2.4), how to implement varying limits on axial rotation within the reach cone (Section 2.5), and how to smooth a coarse reach cone (Section 2.6).

### 2.1 Definitions

To clarify our use of terms, a *joint* is an articulation between a *parent segment* and *child segment*. 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. 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. Both coordinate frames have their origin at the joint. The *longitudinal segment axis* is the $z$-axis in the state coordinate frame, and is considered to be a (bounded) straight line segment. In equations, **boldface** indicates a vector and *italics* indicates a scalar variable.

Formally, a *cone* is a set of rays (or half-lines) emanating from the origin. That is, if a point $\mathbf{p}$ is in a given cone then all points of the form $\lambda\mathbf{p}$, 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 sometimes by a spherical polygon (for conceptual purposes) and sometimes by a set of tetrahedra (for computational purposes).

Without any joint limit, the longitudinal axis of a segment might be in any direction from the origin, where the joint is located. Conceptually, the *sinus cone* or *reach cone* defines the set of directions that can *actually* be taken on by the longitudinal axis.

Our spherical joint limits are specified as a reach cone intersected with a sphere of radius 1 centered at the joint, together with limits on *axial* rotation that may vary throughout the reach cone. The reach cone is

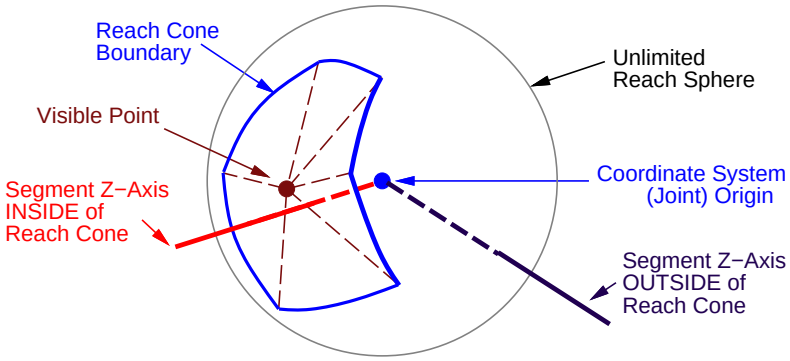


Figure 1: A reach-cone polygon with 5 boundary points and a visible point is shown.

specified by a spherical polygon, called the *reach-cone polygon*, whose vertices are a series of *boundary points* on this unit sphere, and whose edges are great-circle arcs on the sphere. 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 that is distal to the joint; when the segment changes direction the default frame remains fixed. 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. This only depends on the $x$ and $y$ rotations applied to the segment.

At each vertex, limits on axial rotation (i.e., rotation about the longitudinal segment axis or $z$ axis, called *twist* by Korein) may be specified by maximum and minimum angles. Limits on axial rotation at any position in the reach cone are defined by an interpolant of these values. Being able to vary the axial limits depending on where the segment is within the reach cone is important because biomechanical studies show that the axial limits vary considerably throughout the reach cone [6]. See Section 2.5 for further discussion.

## 2.2 Restriction to Star Polygons

To rapidly discover whether any new location of the longitudinal segment axis is within the reach cone, we restrict reach cones to *spherical star polygons*. I.e., the reach cone spherical polygon has a *visible 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). 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 Figure 6. For clarity, our explanatory figures show small reach cones. 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.

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.

## 2.3 Detecting Reach Cone Inclusion

In this section, we describe how to detect if the segment axis is within the reach cone. 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.

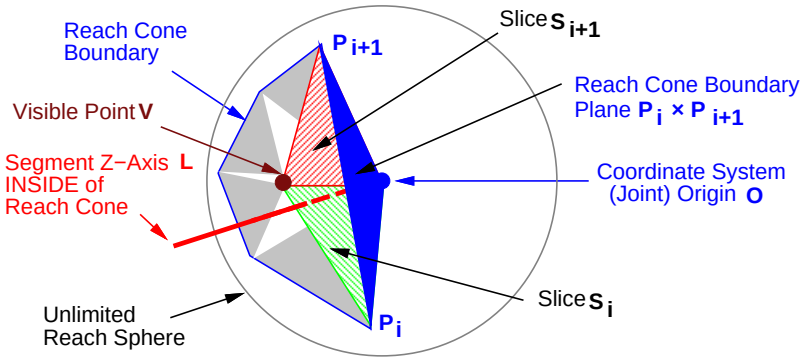


Figure 2: A reach cone with five boundary points showing slice planes $\mathbf{S}_i$ and $\mathbf{S}_{i+1}$ and the boundary plane defined by $\mathbf{O}, \mathbf{P}_i, \mathbf{P}_{i+1}$ that enclose the longitudinal segment axis vector $\mathbf{L}$.

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 [5]:

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

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,

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

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.

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 $\mathbf{V} \times \mathbf{P}_i \cdot \mathbf{L} \ge 0$. 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 $\mathbf{V} \times \mathbf{P}_{i+1} \cdot \mathbf{L} \le 0$ and $\mathbf{P}_i \times \mathbf{P}_{i+1} \cdot \mathbf{L} \ge 0$. 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 $i$ $(0 \le i \le n-1$, all subscripts modulo $n)$ such that $p_i = \mathbf{S}_i \cdot \mathbf{L} \ge 0$ 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.[1]

[1] 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 $\mathbf{V} \times \mathbf{P}_{j+1} \cdot \mathbf{L} \ge 0$, but this is ignored.

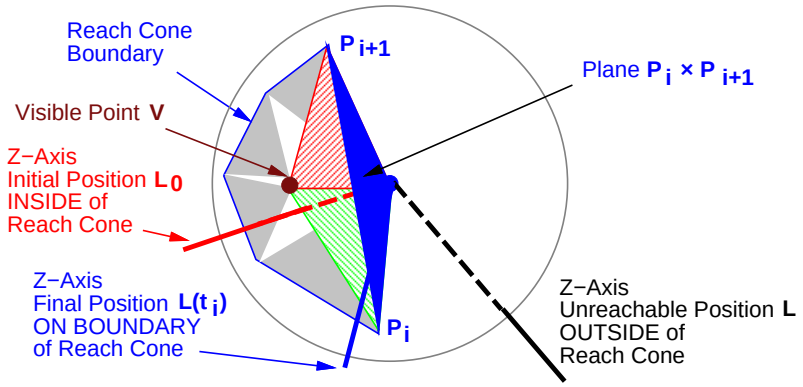


Figure 3: The reach cone from Figure 2 showing an initial $z$-axis position $\mathbf{L}_0$ inside the reach cone, a tested $z$-axis position $\mathbf{L}$ found to be outside the reach cone, and the intersection $\mathbf{L}(t_i)$ of the line between the two with the reach cone boundary. This intersection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

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 2.5.

A different method for testing point inclusion in a spherical polygon, which is not limited to star polygons, is sketched in Korein's thesis without proof. We summarize it in the supplemental technical report [8].

## 2.4 Exit of Trajectory from Reach Cone

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 a segment *trajectory* (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.

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 $\mathbf{L}$. (This follows by observing that the great circle arc and the line segment are in the same plane and have the same endpoints, hence they generate the same cone. Intersect this cone with the reach cone.) 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 2.3. Homogeneous coordinates are all 0 because the planes pass through the origin.

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

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 2.3. 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 (wrapping around from 0 to $n-1$ if necessary), 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 (wrapping around from $n-1$ to 0 if necessary), 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.

## 2.5 Reach Cone Axial Limits

Limits on axial rotation (i.e., around the $z$-axis, which is the longitudinal axis of the segment) can be specified as a minimum-maximum range. However, the axial 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 [6]. 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.

Fortunately, the calculations to locate the segment $z$-axis point $\mathbf{L}$ within two radial slices and one boundary slice (in Section 2.3) provide a form of barycentric coordinates [2] 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:

$$\begin{aligned} s &= p_i + p_{i+1} + v_i \\ w_i &= \frac{p_i}{s} \\ w_{i+1} &= \frac{p_{i+1}}{s} \\ w_V &= \frac{v_i}{s} \end{aligned} \tag{5}$$

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

$$\begin{aligned} \theta_{min}(\mathbf{L}) &= w_i\,\theta_{min}(\mathbf{P}_i) + w_{i+1}\,\theta_{min}(\mathbf{P}_{i+1}) + w_V\,\theta_{min}(\mathbf{V}) \\ \theta_{max}(\mathbf{L}) &= w_i\,\theta_{max}(\mathbf{P}_i) + w_{i+1}\,\theta_{max}(\mathbf{P}_{i+1}) + w_V\,\theta_{max}(\mathbf{V}) \end{aligned} \tag{6}$$

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

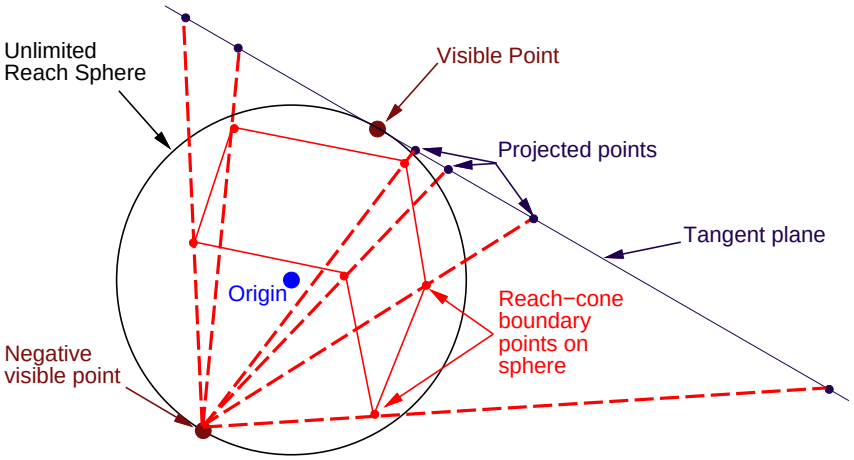


Figure 4: The stereographic projection provides an invertible mapping between the entire sphere (except for one point) and the tangent plane.

kinematics or motion capture, such as penalties for being close to $\theta_{min}$ or $\theta_{max}$. Also, $v_i$ measures closeness to the reach-cone boundary and penalties might be assigned for small values of $v_i$.

## 2.6 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. 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. (However, edges of the spherical star polygon do *not* map into the planar star polygon.) 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 origin, and let $Q(\mathbf{p})$ denote the mapping into the tangent plane. We compute $Q(\mathbf{p})$ by

$$\begin{aligned} \lambda &= \mathbf{V} \cdot \mathbf{p} \\ \mu &= 2r^2/(r^2 + \lambda) \\ Q(\mathbf{p}) &= \mu\mathbf{p} + (1-\mu)\widetilde{\mathbf{V}} \end{aligned} \tag{7}$$

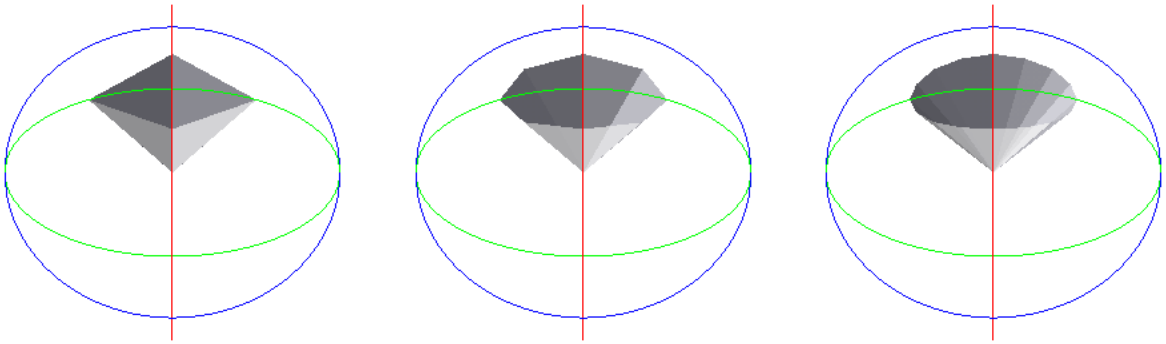

Figure 5: Left: initial reach cone with four boundary points at 30° East, 15° North, 30° West, 15° South. Center: after one smoothing step. Right: after two smoothing steps.

Note that this is actually an extrapolation formula because $\mu \geq 1$. Of course, $Q(\widetilde{\mathbf{V}})$ is not defined and numerical problems will occur for points very near $\widetilde{\mathbf{V}}$.

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

$$\begin{aligned} \lambda &= (\mathbf{q} - \mathbf{V}) \cdot (\mathbf{q} - \mathbf{V}) \\ \mu &= 4r^2/(4r^2 + \lambda) \\ Q^{-1}(\mathbf{q}) &= \mu\mathbf{q} + (1 - \mu)\widetilde{\mathbf{V}} \end{aligned} \tag{8}$$

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

Correctness of Equations 7 and 8 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 $\mathbf{q}_0, \ldots, \mathbf{q}_{k-1}$, 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 $\alpha_k$ is a scalar:

$$\mathbf{q}_{i+1/2} = \tfrac{1}{2}(\mathbf{q}_i + \mathbf{q}_{i+1}) + \alpha_k (\mathbf{q}_{i+1} - \mathbf{q}_{i-1}) + \alpha_k (\mathbf{q}_i - \mathbf{q}_{i+2}) \tag{9}$$

| $k$ | $\alpha_k$ | $k$ | $\alpha_k$ |
|---|---|---|---|
| 3 | .1667 | 6 | .0773 |
| 4 | .1036 | 7 | .0700 |
| 5 | .0850 | 8+ | .0625 |

(10)

The value of $\alpha_8$ is based on a cubic spline fit and the values for $\alpha_3$, $\alpha_4$, and $\alpha_6$ are calculated (by analytical geometry) so that an initial isosceles triangle, rhombus, or regular hexagon generates new points that lie on an ellipse in common with the original points. The values of $\alpha_5$ and $\alpha_7$ were estimated by interpolation. 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.

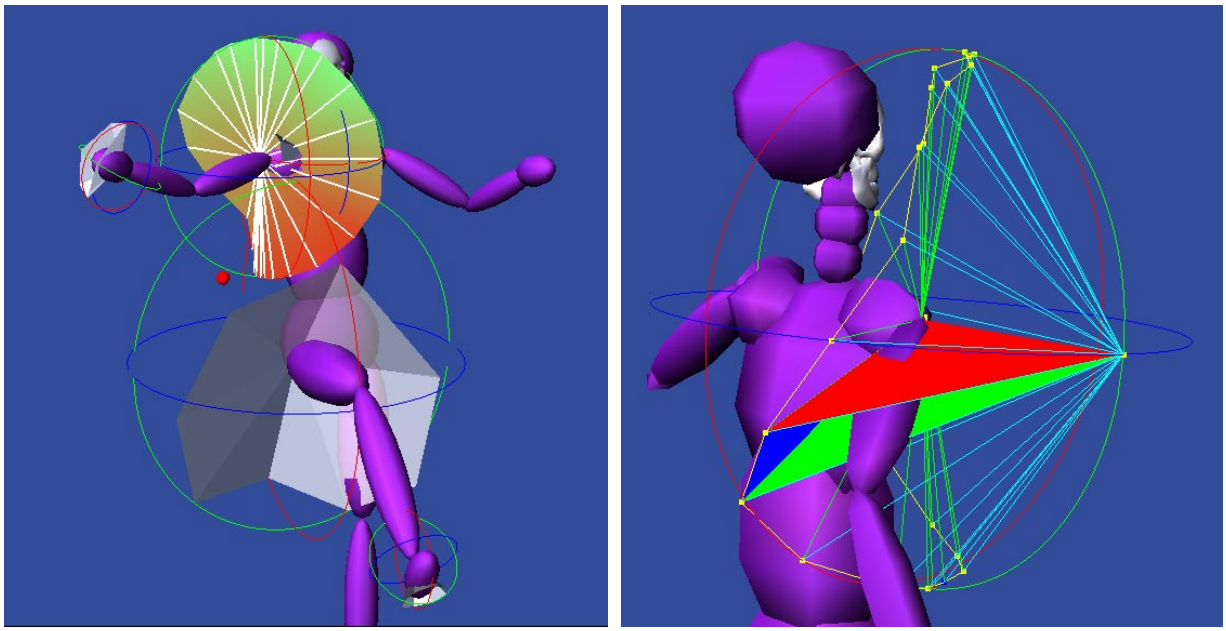

Figure 6: Left: The reach cones of the right side of the body. On the upper arm reach cone, the range of allowable $z$-axis rotation is color-coded (green denotes a lesser range and red a larger). Right: The upper arm reach cone, with the slice containing the arm defined by three triangles. Red and green are consecutive radial slices in counterclockwise order; blue is the reach-cone boundary.

| | Number of Executions | Total Time (secs.) | Average Time ($\mu$-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).

# 3 Example and Discussion

We tested our joint limits using a human model based on the *fauna* software [7] in its upgraded version *zoo*. Figure 6 (left) shows the reach cones on the right limbs. For most joints, limits were created interactively based on observations. The shoulder-arm complex, however, is based on published data [1, 6]. The upper arm reach cone, in particular, is defined by 25 boundary points, and the allowable range of $z$-axis rotation is defined at each boundary point and at the visible point. On the boundary polygons of the upper arm reach cone, the range of $z$-axis rotation values (from 94 to 157 degrees) is shown with color coding, where green is a smaller range and red is a larger range. Figure 6 (right) 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.

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 sphere. However, to obtain a more quantitative evaluation, we did some timing tests for joint limits using 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.

## Acknowledgements

This work was supported by NSF grants CCR 9972464 and MRI 9724237. Jennifer Bevan, Alison Luo, and Mark Slater contributed to the software development.

## References

[1] 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.

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

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

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

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

[6] 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.

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

[8] Jane Wilhelms and Allen Van Gelder. Efficient spherical joint limits with reach cones. Technical report, UCSC, April 2001. Available at `http://www.acm.org/jgt/papers/WilhelmsVanGelder01; includes C code.`