EFFICIENT DISTANCE CALCULATION USING THE SPHERICALLY-EXTENDED POLYTOPE (S-TOPE) MODEL

NAGW-1333

by

Gregory J. Hamlin, Robert B. Kelley, and Josep Tornero*

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering Department
Troy, New York 12180-3590

*Departamento de Ingenierias de Sistemas
Computadores y Automatica (DISCA)
Universidad Politecnica de Valencia
Valencia, SPAIN

September 1991

CIRSSE REPORT #104

Efficient Distance Calculation using the Spherically-Extended Polytope (S-tope) Model

Gregory J. Hamlin† Robert B. Kelley† Josep Tornero‡

†Center for Intelligent Robotic Systems for Space Exploration
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

‡Departamento de Ingenierias de Sistemas, Computadores y Automatica (DISCA)
Universidad Politecnica de Valencia
Valencia, SPAIN

Abstract

An object representation scheme which allows fast Euclidean distance calculation is presented. The object model extends the polytope model by representing objects as the convex hull of a finite set of spheres. An algorithm for calculating distances between objects is developed which is linear in the total number of spheres specifying the two objects.

1 Introduction

The efficient calculation of distances between objects is an important problem in the field of robotics. Many task planning and control algorithms require a knowledge of distances between objects. Collision detection and avoidance depends upon distance calculation between a robot and other obstacles in the environment. Many path planning algorithms use the shortest distance between objects, particularly those involving potential fields or iterative methods which test a path and then adjust it if there is a collision. All of these applications require an efficient distance computation method if they are to be used in a real time environment.

In addition to applications in robotics, there are many computer graphics techniques which can benefit from distance or collision detection calculations. For example, collision detection can be used in ray-tracing to reduce the number of calculations required to render a scene by only considering those objects which lie in the path of the rays. In a virtual reality simulation, where a person interacts with objects in the environment, collision detection can be used to tell when the person is touching something.

A distance calculation method that is often used involves checking for intersections between all of the faces, edges, and vertices of polygonal objects. This brute force method, although straightforward, is extremely slow. Many different techniques have been proposed to speed up distance computation. Quadtree representations can be used to quickly detect collisions between objects and a stationary set of obstacles, but become awkward and slow when the obstacles are moving. Some methods use extremely simple approximations to all the objects such as calculating distances between bounding spheres. This is very fast, but quite inaccurate in most cases. Still others use norm one or norm infinity for the distance calculation, to avoid the computationally expensive square root operation. However, this approximation can give inaccurate or misleading results.

A common method used to represent convex objects is the polytope model. In the polytope model, objects are represented as the convex hull of a set of points. Gilbert, Johnson, and Keerthi [1] present a powerful iterative technique for calculating the distance between polytopes. The execution time of their algorithm is approximately linear with respect to the total number of vertices in the two objects. The algorithm is roughly $O(n)$ where n is the total number of vertices in the two objects. Later, Gilbert and Foo [2] generalize their result to include any convex object. However, in doing so, the advantages of having a single model for all objects is lost.

This paper presents the concept of a spherically extended polytope, or s-tope. Instead of representing objects as the convex hull of a set of points, an s-tope is the convex hull of a set of spheres. This allows a much wider range of objects to be easily described. Objects with rounded surfaces can be represented with far fewer vertices than would be required with the polytope model. Additionally, an efficient distance computation method based on that of Gilbert et al., is developed and experimental results presented. The algorithm still exhibits the $O(n)$ execution time as does the polytope method, but the spherical extension provides a more general object model. The s-tope concept is a generalization of a spherical object geometry presented by Tornero, Hamlin, and Kelley [3], [4].

2 Notational Conventions

The notation used in this paper is similar to that used in [2], especially when dealing with concepts originating in that or previous ([1],[5],[6]) works.

Vectors and points are represented by lowercase

boldfaced letters ($\mathbf{x}$,$\mathbf{y}$), and scalars by lowercase italics, (a, b). Lengths of vectors and distances are defined as the Euclidean norm

$$\text{length of } \mathbf{v} = \|\mathbf{v}\| = \sqrt{\mathbf{v} \cdot \mathbf{v}}, \tag{1}$$

where the $(\cdot)$ is the inner, or dot, product. Unit vectors are written with a hat $(\hat{\mathbf{v}})$.

Sets are denoted by uppercase boldfaced letters ($\mathbf{P}$,$\mathbf{S}$), and $\mathbf{Co\ P}$ is the convex hull of the set $\mathbf{P}$. Other notation is introduced later in the presentation.

To define the spherically extended polytope model, it is first necessary to discuss the notion and notation of a polyope. A polytope is the convex hull of a finite set of points. When dealing with polytopes, the vertices which define the polytope are given the symbol $\mathbf{p}$. Therefore, if $\mathbf{P}$ is a set of points $\{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_n\}$, then the convex hull of $\mathbf{P}$ is an infinite set of points $\mathbf{X}$ expressed as

$$\mathbf{X} = \left\{ \mathbf{x} : \mathbf{x} = \sum_{i=0}^{n} \lambda_i \mathbf{p}_i, \mathbf{p}_i \in \mathbf{P}, \lambda_0 + \cdots + \lambda_n = 1 \right\}. \tag{2}$$

3 S-topes

An s-tope is the convex hull of a finite set of spheres. A sphere is denoted $s = (\mathbf{c}, r)$, where $\mathbf{c}$ is the center and r is the radius. A set of spheres $\mathbf{S} = \{s_0, s_1, \cdots, s_n\}$ consists of the set of centers $\mathbf{C} = \mathbf{c}_0, \mathbf{c}_1, \cdots, \mathbf{c}_n$ and the set of radii $\mathbf{R} = \{r_0, r_1, \cdots, r_n\}$. The convex hull of $\mathbf{S}$ contains an infinite set of spheres called the set of *swept spheres* which is expressed as

$$\mathbf{Co\ S} = \left\{ s = (\mathbf{c}, r) : \mathbf{c} = \sum_{i=0}^{n} \lambda_i \mathbf{c}_i, r = \sum_{i=0}^{n} \lambda_i r_i \right\} \tag{3}$$

$$\text{where } \mathbf{c}_i \in \mathbf{C}, r_i \in \mathbf{R}, \tag{4}$$

$$\text{and } \lambda_0 + \cdots + \lambda_n = 1. \tag{5}$$

Note that the set of swept spheres does not include all possible spheres which can fit inside the s-tope, only those which are generated by Equation 3. An s-tope is represented in this paper by the symbol $\mathcal{S}$, where an s-tope formed from a set of spheres $\{s_0, s_1, \cdots, s_n\}$ is denoted as $\mathcal{S}_{0-n}$. The spheres which define the s-tope are called the spherical-vertices of the s-tope. For simple s-topes (those made from four or fewer spheres) the spheres are listed explicitly. For example, $\mathcal{S}_{ijk}$ is an s-tope with spherical-vertices s_i, s_j and s_k.

An s-tope is said to be *overspecified* if one or more of its spherical-vertices may be removed without changing the convex hull. An s-tope which is not overspecified is said to be a *valid* s-tope.

4 The Simple S-topes

As preparation for the discussion on distance calculation, it is useful to have a detailed description of the s-topes formed from sets of one to four spheres.

4.1 Sphere

The simplest s-tope is a single sphere. An s-tope formed from a sphere, s_i, is denoted $\mathcal{S}_i$ and is identical to that sphere.

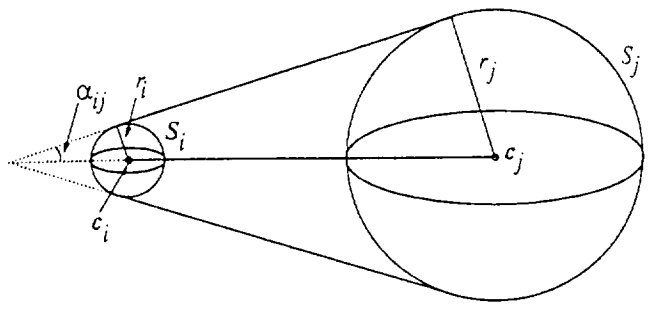


Figure 1: An example of a bi-sphere

4.2 Bi-Sphere

An s-tope formed from two spheres, s_i and s_j, is denoted $\mathcal{S}_{ij}$. A set of expressions describing the set of swept spheres which defines the bi-sphere is obtained by rewriting Equation 3 in the following manner,

$$\mathbf{c}_{ij}(\lambda_{ij}) = \mathbf{c}_i + \lambda_{ij}(\mathbf{c}_j - \mathbf{c}_i), \tag{6}$$

$$r_{ij}(\lambda_{ij}) = r_i + \lambda_{ij}(r_j - r_i), \tag{7}$$

$$\text{where } \lambda_{ij} \in [0, 1]. \tag{8}$$

An example of a bi-sphere is shown in Figure 1. Hereafter, the conical surface generated between the two spheres is referred to as the *side* of the bi-sphere. There are several parameters which may be used to describe a bi-sphere. The *axis*, $\mathbf{v}_{ij}$, is the vector from $\mathbf{c}_i$ to $\mathbf{c}_j$. The *length* of the bi-sphere is the magnitude of the axis vector. The *convergence angle*, α_{ij} of the bi-sphere is the angle between the axis and the side. The sine of α_{ij} appears frequently and is given the symbol η_{ij}. Another useful parameter is the direction along the axis, $\hat{\mathbf{v}}_{ij}$. These parameters have the following relations,

$$\mathbf{v}_{ij} = \mathbf{c}_j - \mathbf{c}_i, \tag{9}$$

$$\hat{\mathbf{v}}_{ij} = \frac{\mathbf{v}_{ij}}{\|\mathbf{v}_{ij}\|}, \tag{10}$$

$$\eta_{ij} = \frac{r_j - r_i}{\|\mathbf{v}_{ij}\|}, \tag{11}$$

$$\alpha_{ij} = \sin^{-1} \eta_{ij}. \tag{12}$$

The vector $\hat{\mathbf{n}}_s$ is normal to the side of a bi-sphere, projecting away from the surface, and satisfies the following constraint,

$$\hat{\mathbf{n}}_s \cdot \hat{\mathbf{v}}_{ij} = -\eta_{ij}. \tag{13}$$

A bi-sphere is valid if and only if neither of the spheres s_i or s_j is completely contained within the other. Equivalently, a bi-sphere is not valid if $|\eta_{ij}| \geq 1$.

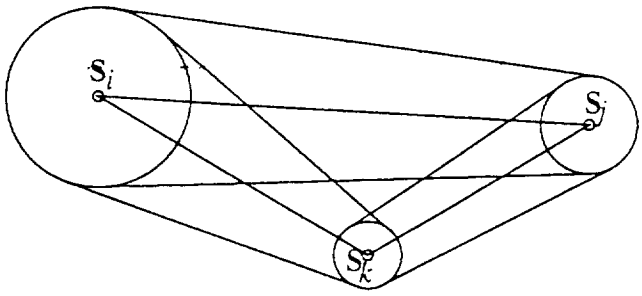


Figure 2: An example of a tri-sphere

4.3 Tri-Sphere

A Tri-sphere is an s-tope formed from three spheres, s_i, s_j, and s_k, and is denoted $\mathcal{S}_{ijk}$. Figure 2 shows an example of a tri-sphere. The set of equations describing all the swept-spheres in a tri-sphere is

$$\mathrm{c}_{ijk}(\lambda_{ijk}) = \mathrm{c}_i + \lambda_{ij}(\mathrm{c}_j - \mathrm{c}_i) + \lambda_{ik}(\mathrm{c}_k - \mathrm{c}_i) \quad (14)$$

$$r_{ijk}(\lambda_{ijk}) = r_i + \lambda_{ij}(r_j - r_i) + \lambda_{ik}(r_k - r_i), \quad (15)$$

$$\text{where } \lambda_{ij} \in [0,1], \lambda_{ik} \in [0,1]. \quad (16)$$

A tri-sphere $\mathcal{S}_{ijk}$ has three bi-sphere edges, and two planar faces. The three edges are the bi-spheres $\mathcal{S}_{ij}$, $\mathcal{S}_{ik}$, and $\mathcal{S}_{jk}$.

The two planar faces are tangent to all three spheres s_i, s_j, and s_k, as well as to all three edges $\mathcal{S}_{ij}$, $\mathcal{S}_{ik}$, and $\mathcal{S}_{jk}$. Therefore, the normals of the two faces must satisfy Equation 13 for each of the edges. That is,

$$\hat{\mathbf{n}}_s \cdot \hat{\mathbf{v}}_{ij} = -\eta_{ij}, \quad (17)$$

$$\hat{\mathbf{n}}_s \cdot \hat{\mathbf{v}}_{ik} = -\eta_{ik}, \quad (18)$$

$$\hat{\mathbf{n}}_s \cdot \hat{\mathbf{v}}_{jk} = -\eta_{jk}. \quad (19)$$

These equations imply an angle of $90 - \alpha_{ij}$ between $\hat{\mathbf{n}}_s$ and $\hat{\mathbf{v}}_{ij}$, and an angle of $90 - \alpha_{ik}$ between $\hat{\mathbf{n}}_s$ and $\hat{\mathbf{v}}_{ik}$. Since the three equations are not independent, it is necessary to add the following constraint on $\hat{\mathbf{n}}_s$ in order to calculate the two normals $\hat{\mathbf{n}}_s$ and $\hat{\mathbf{n}}_s'$,

$$\|\hat{\mathbf{n}}_s\| = 1. \quad (20)$$

Figure 3 shows the relationships expressed in the preceeding equations.

A tri-sphere is not valid if one of the spheres is completely contained within the bi-sphere formed by the other two spheres.

4.4 Quad-Sphere

A quad-sphere is an s-tope formed from four spheres. The notation for a quad-sphere is the same as for the other objects. Namely, for a quad-sphere with spherical-vertices s_i, s_j, s_k, and s_l, the symbol is $\mathcal{S}_{ijkl}$. The equations for the quad-sphere are similar to those for the tri-sphere.

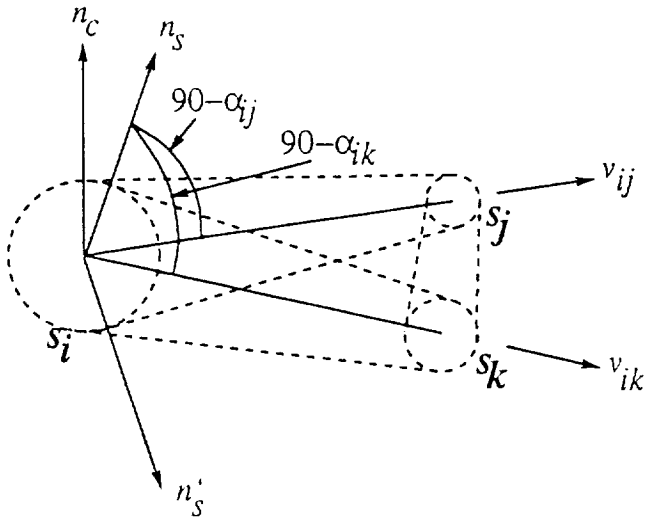


Figure 3: Normals to the faces of a tri-sphere

$$\mathrm{c}_{ijkl}(\lambda_{ijkl}) = \mathrm{c}_i + \lambda_{ij}(\mathrm{c}_j - \mathrm{c}_i) + \lambda_{ik}(\mathrm{c}_k - \mathrm{c}_i) + \lambda_{il}(\mathrm{c}_l - \mathrm{c}_i),$$

$$r_{ijk}(\lambda_{ijk}) = r_i + \lambda_{ij}(r_j - r_i) + \lambda_{ik}(r_k - r_i) + \lambda_{il}(r_l - r_i),$$

$$\text{where } \lambda_{ij} \in [0,1], \lambda_{ik} \in [0,1], \lambda_{il} \in [0,1]. \quad (21)$$

Just as a polytope with four or more points is generally a polyhedral object with triangular facets, a quad-sphere (or an s-tope with more than four spherical-vertices) is an object with tri-sphere *facets*. Therefore, the normal to the surface of any of the facets may be calculated in the same manner as for a tri-sphere.

A quad-sphere is not valid if any of its spherical-vertices are completely contained within the tri-sphere formed from the other three spheres.

5 Polytope Distance Calculation

The basic algorithm for calculating the distance between s-topes is similar to an algorithm for polytopes developed by Gilbert, Johnson, and Keerthi [1], which in turn is based on work by Barr and Gilbert [5], [6]. Since the s-tope distance algorithm is an extension of the polytope distance algorithm, an overview of the polytope distance algorithm is presented here. For more details, refer to the works cited above.

5.1 Polytope algorithm

The polytope distance algorithm takes advantage of the fact that the shortest distance between two objects is the same as the shortest distance from the origin to the object formed by taking the Minkowski difference of the two objects. Therefore, given two sets of points $\mathbf{P}_1$ and $\mathbf{P}_2$ which define polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$, a new polytope $\mathcal{P}_{21}$ can be formed from the set $\mathbf{P}_{21}$ as

$$\mathbf{P}_{21} = \mathbf{P}_2 - \mathbf{P}_1 = \{\mathrm{p}_2 - \mathrm{p}_1 : \mathrm{p}_1 \in \mathbf{P}_1, \mathrm{p}_2 \in \mathbf{P}_2\} \quad (22)$$

The algorithm requires the ability to determine the point in a set which is farthest in a particular direction. The support function, $h_{\mathbf{P}}$, for a set of points $\mathbf{P}$, is defined in [1] as

$$h_{\mathbf{P}}(\mathbf{v}) = \max\{\mathbf{p} \cdot \mathbf{v} : \mathbf{p} \in \mathbf{P}\}. \quad (23)$$

The point $\mathbf{p}$ which satisfies this equation is furthest in the direction $\mathbf{v}$.

The basic algorithm used is as follows. Given two polytopes defined by their sets of vertices $\mathbf{P}_1$ and $\mathbf{P}_2$, create a new set of points $\mathbf{P}_{12}$ using Equation 22.

1. Start with an empty *working set*, $\mathbf{P}$
2. Guess at the direction of shortest distance, $\mathbf{v}$. Use the direction between the centroids if you don't have any better information.
3. Find the point $\mathbf{p}(\mathbf{v})$ in $\mathbf{P}_{12}$, furthest in the direction $\mathbf{v}$, and add it to the set $\mathbf{P}$.
4. Find the closest point to the origin, $\mathbf{d}$, on the polytope formed from the set $\mathbf{P}$ using the polytope distance sub-algorithm.
5. Discard any points in $\mathbf{P}$ which are not necessary to define the point $\mathbf{d}$.
6. Repeat steps 2-4 until $|(h_{\mathbf{P}_{12}}(\mathbf{v}))^2 - (\|\mathbf{d}\|)^2| \leq$ *Tolerance*

Figure 4 show four iterations of the algorithm on a planar polytope, $\mathcal{P}_{1-6}$.

For two polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$, which have n and m vertices respectively, the polytope $\mathcal{P}_{12}$ formed from the Minkowski difference of the two polytopes has $n \times m$ vertices. This means that for each iteration of the algorithm, $n \times m$ inner products must be evaluated to calculate the support function $h_{\mathcal{P}_{12}}$. However, the support function may also be calculated as

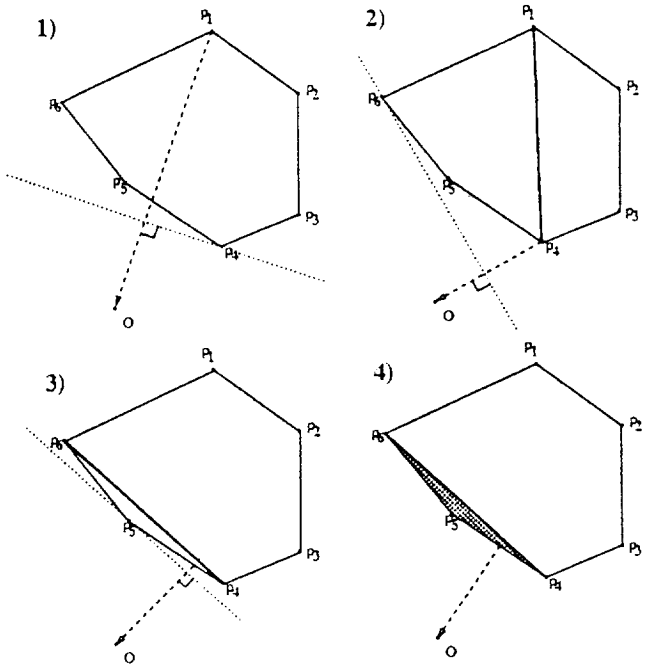
$$h_{\mathcal{P}_{12}}(\mathbf{v}) = h_{\mathcal{P}_2}(\mathbf{v}) - h_{\mathcal{P}_1}(\mathbf{v}). \quad (24)$$

This formulation only requires $n+m$ evaluations of the inner product.

Amazingly, the polytope distance algorithm almost always converges within four or five iterations, even when there are hundreds of vertices. This makes the polytope algorithm approximately $O(n+m)$ in computational complexity [1].

5.2 Polytope distance sub-algorithm

The *distance sub-algorithm* [1] calculates the closest point to the origin on a polytope with one, two, three, or four vertices. Additionally, it returns the minimal number of vertices necessary to define this point. For example, if the closest point is one of the vertices, then that vertex is sufficient to define the closest point. If the closest point is on an edge, then two vertices (the end points of the edge) are necessary to define the closest point. Similarly, if the closest point is on a face of the polytope, three vertices are necessary to define the closest point. Note that the only time four vertices are required to define the closest point is when the point is actually *inside* the object. Therefore, after each iteration of the distance algorithm, there are never more than three points in the working set.



1. calculate shortest distance to initial working set ($\mathbf{p}_1$)
2. add $\mathbf{p}_4$ to working set and calculate shortest distance to working set ($\mathbf{p}_1$,$\mathbf{p}_4$)
3. discard $\mathbf{p}_1$ and add $\mathbf{p}_6$ and calculate shortest distance to working set ($\mathbf{p}_4$,$\mathbf{p}_6$)
4. add $\mathbf{p}_5$ and calculate shortest distance to working set ($\mathbf{p}_4$,$\mathbf{p}_5$,$\mathbf{p}_6$)

Figure 4: Example of the polytope distance algorithm

6 S-tope Distance Calculation

Many of the advantages of the polytope representation are also present with the s-tope representation. For example, the Minkowski difference and the support function are both simply calculated. The Minkowski difference of the s-topes, $\mathcal{S}_1$ and $\mathcal{S}_2$ defined by two sets of spheres $\mathbf{S}_1$ and $\mathbf{S}_2$ is the s-tope $\mathcal{S}_{12}$ defined by the set of spheres $\mathbf{S}_{12}$, where

$$\begin{aligned} \mathbf{S}_{12} = \{&s_{12} = (\mathbf{c}_{12}, r_{12}) : \mathbf{c}_{12} = \mathbf{c}_2 - \mathbf{c}_1, && (25)\\ &\mathbf{c}_1 \in \mathbf{C}_1, \mathbf{c}_2 \in \mathbf{C}_2, && (26)\\ &r_{12} = r_1 + r_2, && (27)\\ &r_1 \in \mathbf{R}_1, r_2 \in \mathbf{R}_2\}. && (28) \end{aligned}$$

The support function for an s-tope is also very similar to that of a polytope. For an s-tope $\mathcal{S}$ defined by spherical vertices in the set $\mathbf{S}$, the support function is computed as

$$h_{\mathbf{S}}(\mathbf{v}) = \max\{(\mathbf{c}_i \cdot \mathbf{v}) + r_i : (\mathbf{c}_i, r_i) = s_i, s_i \in \mathbf{S}\}. \quad (29)$$

6.1 S-tope algorithm

The distance calculation algorithm is very similar to the one presented above for polytopes. Given two s-topes defined by their sets of spherical-vertices S_1 and S_2, create a new set of points S_{12} using Equation 26.

1. Start with an empty *working set*, S
2. Guess at the direction of shortest distance, v. Use the direction between the centroids if you don't have any better information.
3. Find the sphere $s(v)$ in S_{12}, furthest in the direction v, and add it to the set S.
4. Find the closest point to the origin, d, on the s-tope formed from the set S using the s-tope distance sub-algorithm.
5. Discard any spherical-vertices in S which are not necessary to define the point d.
6. Repeat steps 2-4 until $|((h_{S_{12}}(v))^2 - (\|d\|)^2| \leq$ *Tolerance*

6.2 S-tope distance sub-algorithm

The s-tope distance sub-algorithm is actually a collection of four separate but interdependent algorithms. There are algorithms for finding the shortest distance from the origin to a sphere, from the origin to a bi-sphere, from the origin to a tri-sphere, and from the origin to a quad-sphere. The algorithms for calculating the distance to a sphere, bi-sphere, and tri-sphere, are based on work by Tornero, Hamlin, and Kelley [3], [4].

The sub-algorithm takes advantage of the fact that each of the collection is called under specific circumstances by the s-tope distance algorithm. For example, the tri-sphere sub-algorithm will only be called after the bi-sphere sub-algorithm has been called. In general, if the distance is to be calculated from the origin to an s-tope, $\mathcal{S}_n$, with spherical-vertices s_1-s_n, it can be assumed that the distance has already been calculated from the origin to the s-tope $\mathcal{S}_{(n-1)}$ formed by the spherical-vertices s_1-$s_{(n-1)}$, and that all the spheres s_1-$s_{(n-1)}$ were necessary to define the closest sphere. Thus, it is useful to think of the sphere s_n as being *added* to the s-tope $\mathcal{S}_{(n-1)}$, and the distance computed to the resulting object.

6.2.1 Distance to a Sphere

In this paper, the shortest distance between the origin and an object is written as

$$\mathcal{DIST}(object).$$

The distance between the origin and a sphere is calculated by the following formula,

$$\mathcal{DIST}(s) = \|c\| - r. \tag{30}$$

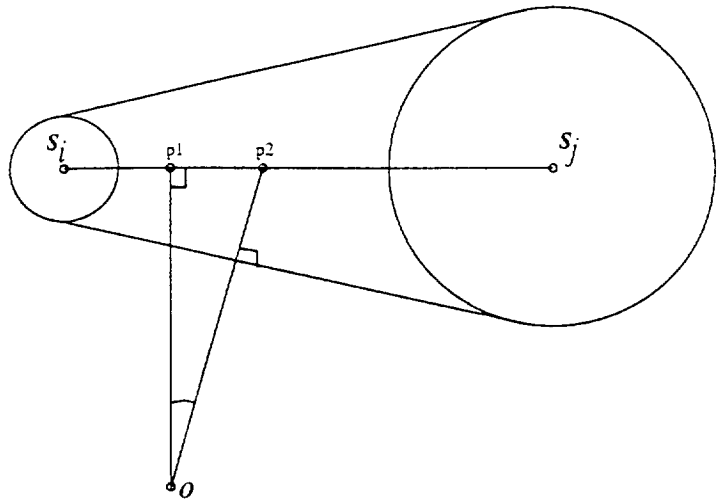


Figure 5: Distance from the origin to a bi-sphere

6.2.2 Distance to a Bi-Sphere

Calculation of the shortest distance between the origin and a bi-sphere $\mathcal{S}_{ij}$ is performed in several steps. First, the point on the axis of the bi-sphere closest to the origin is found by calculating $\lambda_{ij}{}^*$,

$$\lambda_{ij}{}^* = -\frac{c_i \cdot v_{ij}}{\|v_{ij}\|^2}. \tag{31}$$

Second, to satisfy the angular constraint, the value of λ_{ij} is updated,

$$\lambda_{ij} = \lambda_{ij}{}^* + \frac{\|c_{ij}(\lambda_{ij}{}^*)\|}{\|v_{ij}\|} \tan \alpha_{ij}. \tag{32}$$

Figure 5 illustrates these ideas.

Simple case ($\lambda_{ij} \in [0,1]$). For the simple case of $\lambda_{ij} \in [0,1]$ the distance is calculated between the origin and $s_{ij}(\lambda_{ij})$ using Equation 30. That is,

$$\mathcal{DIST}(\mathcal{S}_{ij}) = \mathcal{DIST}(s_{ij}(\lambda_{ij})) = \|c_{ij}(\lambda_{ij})\| - r_{ij}(\lambda_{ij}). \tag{33}$$

Handling λ out of range [0,1]. When the λ_{ij} is not in the range [0,1], the shortest distance is calculated from the origin to one of the ends of $\mathcal{S}_{ij}$. In particular,

$$\begin{aligned} \lambda_{ij} >= 1 \quad &\Rightarrow \quad \mathcal{DIST}(\mathcal{S}_{ij}) = \mathcal{DIST}(s_j) \\ \lambda_{ij} <= 0 \quad &\Rightarrow \quad \mathcal{DIST}(\mathcal{S}_{ij}) = \mathcal{DIST}(s_i) \end{aligned} \tag{34}$$

However, since s_j is chosen by the distance algorithm and is closer (along the vector c_i) to the origin, λ_{ij} will never be less than zero, and that case does not need to be tested.

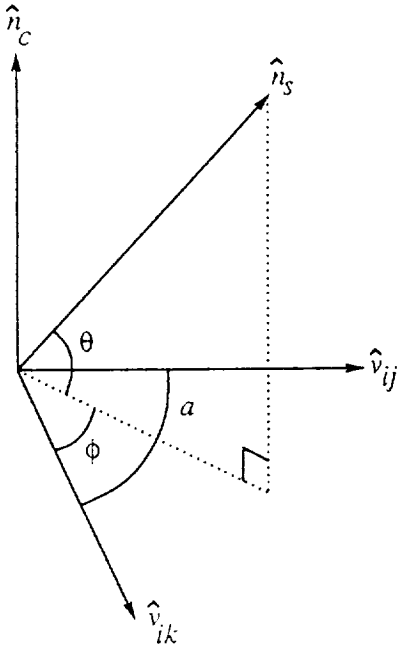


Figure 6: Pseudo-Polar Coordinate System

6.2.3 Distance to a Tri-Sphere

The process of calculating the shortest distance from the origin to a tri-sphere $\mathcal{S}_{ijk}$ has three steps. First, the direction of shortest distance to the origin is calculated. Then, the λ's (see Equation 14) are calculated which correspond to the closest sphere on the tri-sphere $\mathcal{S}_{ijk}$ to the sphere s_i. The distance is then calculated from this sphere to the origin, unless the λ's are out of range (see Equation 16), in which case further calculation is necessary.

Calculation of surface normals. The direction of shortest distance, $\hat{\mathbf{n}}_s$, when the λ's are in range, is simply the normal to one of the planar surfaces bounding the tri-sphere. These normals must satisfy Equations 17 to 20. Two of the three independent constraint equations are repeated here.

$$\hat{\mathbf{n}}_s \cdot \hat{\mathbf{v}}_{jk} = -\eta_{jk}, \tag{35}$$

$$\hat{\mathbf{n}}_s \cdot \hat{\mathbf{v}}_{jl} = -\eta_{jl}. \tag{36}$$

A pseudo-polar coordinate system is formed using $\mathbf{v}_{ij}$, $\mathbf{v}_{kl}$, and $\hat{\mathbf{n}}_c = \mathbf{v}_{ij} \times \mathbf{v}_{kl}$. The vector $\hat{\mathbf{n}}_s$ is located in this system using two variables, θ and ϕ. θ is the angle between $\hat{\mathbf{n}}_s$ and the plane formed by $\mathbf{v}_{ij}$ and $\mathbf{v}_{kl}$. ϕ is the angle from $\mathbf{v}_{ij}$ to the projection of $\hat{\mathbf{n}}_s$ in this plane. a represents the angle between $\mathbf{v}_{ij}$ and $\mathbf{v}_{kl}$. This coordinate system is shown in Figure 6.

Using these three angles, the constraint equations may be rewritten as

$$a = \arccos(\hat{\mathbf{v}}_{ij} \cdot \hat{\mathbf{v}}_{ik}). \tag{37}$$

$$\cos\theta\cos\phi = \eta_{ij}. \tag{38}$$

$$\cos\theta\cos(a+\phi) = \eta_{ik}. \tag{39}$$

Solving for ϕ and θ gives

$$\phi = \arctan\left(\frac{\eta_{ik} - \eta_{ij}\cos a}{\eta_{ij}\sin a}\right), \tag{40}$$

$$\theta = \arccos\left(\frac{\eta_{ij}}{\cos\phi}\right). \tag{41}$$

The two values of θ specify the two faces of the tri-sphere. The vectors $\hat{\mathbf{n}}_s$ are then constructed using the equations

$$g_{ij} = \frac{\cos\theta\sin(a-\phi)}{\sin a}, \tag{42}$$

$$g_{ik} = \frac{\cos\theta\sin\phi}{\sin a}, \tag{43}$$

$$g_c = \sin\theta, \tag{44}$$

$$\hat{\mathbf{n}}_s = g_{ij}\hat{\mathbf{v}}_{ij} + g_{ik}\hat{\mathbf{v}}_{ik} + g_c\hat{\mathbf{n}}_c. \tag{45}$$

Calculation of λ's. Let d be the length of the line segment joining the origin with the closest sphere in the tri-sphere $\mathcal{S}_{ijk}$. The following equation may be obtained from Equation 14,

$$d\,\hat{\mathbf{n}}_s = -\mathbf{c}_j - \lambda_{ij}(\mathbf{c}_j - \mathbf{c}_i) - \lambda_{ik}(\mathbf{c}_k - \mathbf{c}_i). \tag{46}$$

Multiplying Equation 46 by $\hat{\mathbf{n}}_{ij} = \hat{\mathbf{n}}_s \times \hat{\mathbf{v}}_{ij}$ and $\hat{\mathbf{n}}_{ik} = \hat{\mathbf{n}}_s \times \hat{\mathbf{v}}_{ik}$ gives the following two equations for λ_{ij} and λ_{ik},

$$\lambda_{ij} = -\frac{\mathbf{c}_i \cdot \hat{\mathbf{n}}_{ik}}{\mathbf{v}_{ij} \cdot \hat{\mathbf{n}}_{ik}}. \tag{47}$$

$$\lambda_{ik} = -\frac{\mathbf{c}_i \cdot \hat{\mathbf{n}}_{ij}}{\mathbf{v}_{ik} \cdot \hat{\mathbf{n}}_{ij}}. \tag{48}$$

Shortest distance (simple case). Assuming both λ's are in the ranges specified by equation 16 these λ's may be used in Equations 14 and 15 to calculate the closest sphere $s_{ijk}(\lambda_{ij}, \lambda_{ik})$ in the tri-sphere. The shortest distance between the origin and the tri-sphere $\mathcal{S}_{ijk}$ is then calculated as,

$$\mathcal{DIST}(\mathcal{S}_{ijk}) = \mathcal{DIST}(s_{ijk}(\lambda_{ij}, \lambda_{ik}).) \tag{49}$$

Table 1: Handling λ's out of range

λ_{ij}	λ_{ik}	$\lambda_{ij} + \lambda_{ik}$	$DIST(\mathcal{S}_{ijk}) =$
>= 0	>= 0	<= 1	Simple case
< 0	>= 0	<= 1	$DIST(\mathcal{S}_{ik})$
>= 0	< 0	<= 1	$DIST(\mathcal{S}_{ij})$
>= 0	>= 0	> 1	$DIST(\mathcal{S}_{jk})$
< 0	< 0	<= 1	MIN($DIST(\mathcal{S}_{ij})$, $DIST(\mathcal{S}_{ik})$)
< 0	>= 0	> 1	MIN($DIST(\mathcal{S}_{ik})$, $DIST(\mathcal{S}_{jk})$)
>= 0	< 0	> 1	MIN($DIST(\mathcal{S}_{ij})$, $DIST(\mathcal{S}_{jk})$)

Handling λ's out of range. If the λ's obtained above fail to satisfy one or more of the inequality constraints, then the shortest distance is between the origin and one of the three edges of the tri-sphere $\mathcal{S}_{ijk}$. Table 1 shows all the possible situations. In the table $DIST(\mathcal{S}_{ij})$ refers to the calculation of the distance between the origin and $\mathcal{S}_{ij}$ using the method of Section 6.2.2.

As was the case when calculating the distance to a bi-sphere, it is possible to eliminate some of the cases in Table 1. Sphere s_k was added to $\mathcal{S}_{ij}$ to form $\mathcal{S}_{ijk}$, and that sphere is closer to the origin along the direction of shortest distance calculated for $\mathcal{S}_{ij}$. Therefore, the closest sphere in $\mathcal{S}_{ijk}$ will never be in $\mathcal{S}_{ij}$, and λ_{ik} will never be less than zero.

6.2.4 Distance to a Quad-Sphere

The method for calculating the distance from the origin to a quad-sphere $\mathcal{S}_{ijkl}$ relies heavily on the fact that the distance has already been calculated to $\mathcal{S}_{ijk}$. This insures that when the origin is projected along $\hat{\mathbf{n}}_s$, the direction of shortest distance from the origin to $\mathcal{S}_{ijk}$, it will lie within the triangle formed by $\mathbf{c}_i$, $\mathbf{c}_j$, and $\mathbf{c}_k$. Adding the fourth spherical-vertex to the s-tope $\mathcal{S}_{ijk}$ creates three tri-sphere faces $\mathcal{S}_{ijl}$, $\mathcal{S}_{ikl}$, and $\mathcal{S}_{jkl}$. The shortest distance to the s-tope $\mathcal{S}_{ijkl}$ is identical to the shortest distance to one of these three tri-spheres.

To determine which tri-sphere face of the quad-sphere contains the closest sphere to the origin, the centers of all the spherical vertices of $\mathcal{S}_{ijkl}$, and the origin, are projected into a plane perpendicular to $\hat{\mathbf{n}}_s$, the result will appear something like Figure 7. The triangle in which the projected origin lies determines which tri-sphere face contains the closest sphere to the origin.

A simple procedure to determine which of the projected triangles $\mathbf{c}_i\mathbf{c}_j\mathbf{c}_l$, $\mathbf{c}_i\mathbf{c}_k\mathbf{c}_l$, or $\mathbf{c}_j\mathbf{c}_k\mathbf{c}_l$, contains the origin has been developed which avoids actually having to project the vertices into a plane.

First, three vectors are calculated as follows:

if $\hat{\mathbf{n}}_s \cdot \mathbf{c}_i > 0$ then

$$\mathbf{n}_{il} = \hat{\mathbf{n}}_s \times \mathbf{v}_{il}, \quad (50)$$

$$\mathbf{n}_{jl} = \hat{\mathbf{n}}_s \times \mathbf{v}_{jl}, \quad (51)$$

$$\mathbf{n}_{kl} = \hat{\mathbf{n}}_s \times \mathbf{v}_{kl}; \quad (52)$$

otherwise,

$$\mathbf{n}_{il} = \mathbf{v}_{il} \times \hat{\mathbf{n}}_s. \quad (53)$$

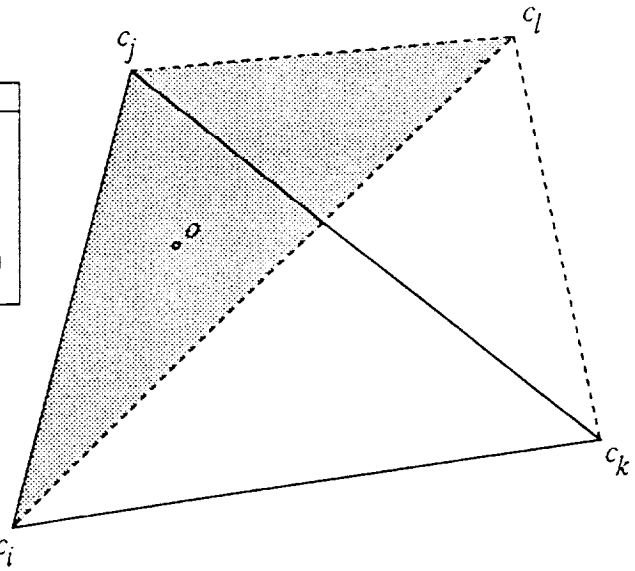


Figure 7: Projection of the centers of a Quad-Sphere into a plane

Table 2: Determining the face which contains the closest sphere

x_{il}	x_{jl}	x_{kl}	Candidate tri-sphere
> 0	< 0	?	$\mathcal{S}_{ijl}$
< 0	?	> 0	$\mathcal{S}_{ikl}$
?	> 0	< 0	$\mathcal{S}_{jkl}$

$$\mathbf{n}_{jl} = \mathbf{v}_{jl} \times \hat{\mathbf{n}}_s, \quad (54)$$

$$\mathbf{n}_{kl} = \mathbf{v}_{kl} \times \mathbf{n}_s. \quad (55)$$

These vectors, define three planes which contain the point $\mathbf{c}_l$ and are parallel to $\hat{\mathbf{n}}_s$ and to $\mathbf{v}_{il}$, $\mathbf{v}_{jl}$, and $\mathbf{v}_{kl}$ respectively. The dot product of the vector $\mathbf{c}_l$ with each of these vectors is taken, to give,

$$x_{il} = \bar{n}_{il} \cdot \mathbf{c}_l, \quad (56)$$

$$x_{jl} = \bar{n}_{jl} \cdot \mathbf{c}_l, \quad (57)$$

$$x_{kl} = \bar{n}_{kl} \cdot \mathbf{c}_l. \quad (58)$$

The tri-sphere containing the closest sphere is then determined from Table 2.

The question marks in the table indicate that any value is allowed. The tri-sphere face chosen by this method contains the closest point to the origin unless the origin lies within the interior of the quad-sphere (or more specifically, within the interior of the tetrahedron formed by the centers of the spherical-vertices of the quad-sphere). The origin is tested to determine whether it lies within the quad-sphere by calculating the normal to the plane of the centers of the *candidate* tri-sphere, and then taking the dot product of $\mathbf{c}_l$

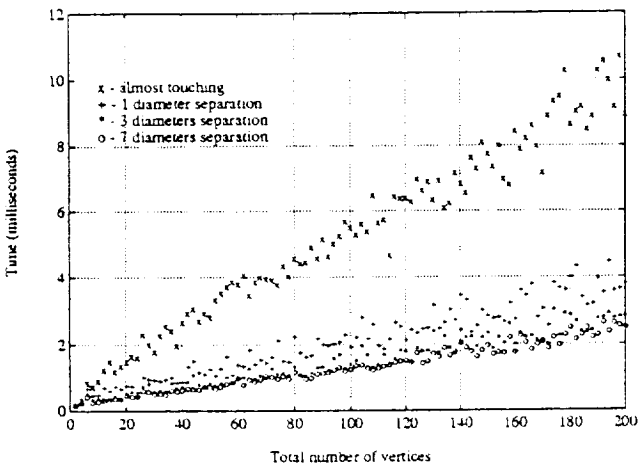


Figure 8: Calculation times for Polytopes on a Sparc I

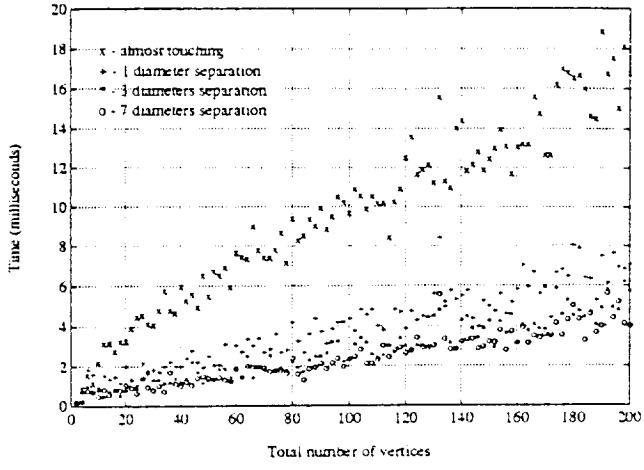


Figure 9: Calculation times for Sperically Extended Polytopes on a Sparc I

with this normal. If the origin lies within the quad-sphere, the distance algorithm halts and a collision is reported. Otherwise, the distance to the quad-sphere is found by calculating the distance to the candidate tri-sphere using the method of Section 6.2.3.

7 Experimental Results

To determine the efficiency of the distance computation algorithms, a program was written and tested using randomly generated objects with varying numbers of vertices.

Many of the equations used for calculating distances can be greatly simplified if all the radii of the spheres in a given object are assumed to be constant. In this case, the problem is reduced to calculating the distance between polytopes and then subtracting the sum of the radii of the two objects. The program was written to take advantage of this degenerate case, and runs faster when given sets of polytopes instead of s-topes. Consequently, two sets of experiments were performed. The first using all polytopes, and the second using all s-topes.

In both cases, objects with from one to one hundred vertices were generated by randomly placing the vertices on the surface of a sphere. To generate polytopes, the vertices were all spheres of zero radius. To generate s-topes, the vertices were spheres of random radii. The times for calculating distances between pairs of these objects at varying separations was measured on a Sun Sparcstation I. The results of these measurements are presented in Figures 8 and 9. In each figure calculation times are presented for the following cases: just touching, separated by a distance equal to the approximate diameter of the objects, separated by three diameters, and separated by seven diameters. The two figures show the linear relationship between the total number of vertices and the calculation time. Additionally, it can be seen that the calculation time is greatly reduced when the objects are not in close proximity.

8 Conclusions

The s-tope model is a simple extension to the polytope model which allows many types of objects to be represented by fewer vertices. An efficient distance calculation method has been presented which has an execution time linearly related to the number of vertices in the two objects. The distance calculation time for s-topes is slightly greater than for polytopes with an equal number of vertices. However, since many objects may be represented with far fewer vertices using the s-tope model, there may be a net reduction in computation time.

References

[1] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal of Robotics and Automation*, vol. 4, pp. 193–203, April 1988.

[2] E. G. Gilbert and C. P. Foo, "Computing the distance between general convex objects in three-dimensional space," *IEEE Transactions on Robotics and Automation*, vol. 6, pp. 53–61, February 1990.

[3] J. Tornero, G. J. Hamlin, and R. B. Kelley, "Spherical-object representation and fast distance computation for robotic applications," in *IEEE International Conference on Robotics and Automation*, (Sacramento, CA), pp. 1602–1608, May 1991.

[4] J. Tornero, G. J. Hamlin, and R. B. Kelley, "Collision-detection based on a fast distance computation technique," in *Proceedings of the European Robotics and Intelligent Systems Conference*, (Corfu, Greece), June 1991. (in press).

[5] R. O. Barr, "An efficient computational procedure for a generalized quadratic programming problem," *SIAM J. Contr.*, vol. 7, pp. 415–419, 1969.

[6] R. O. Barr and E. G. Gilbert, "Some efficient algorithms for a class of abstract optimization problems arising in optimal control," *IEEE Transactions on Automatic Control*, vol. AC-14, pp. 640–652, 1969.