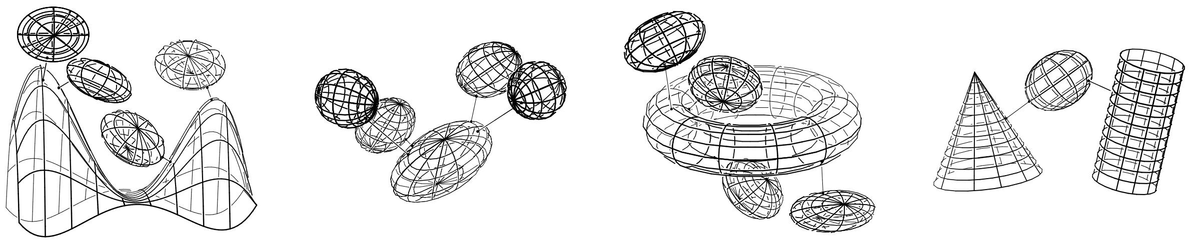
Computing the Distance Between Two Surfaces via Line Geometry

Kyung-Ah Sohn[a], Bert Jüttler[b,1], Myung-Soo Kim[a], and Wenping Wang[c]
[a] School of Computer Science and Engineering,
Seoul National University, South Korea.
[b] Institute of Analysis, Dept. of Applied Geometry,
Johannes Kepler University of Linz, Austria.
[c] Dept. of Computer Science and Information Systems,
The University of Hong Kong, China.



Abstract

This paper presents an algorithm for computing the distance between two free-form surfaces. Using line geometry, the distance computation is reformulated as a simple instance of a surface-surface intersection problem, which leads to a low-dimensional root finding in a system of equations. This approach produces an efficient algorithm for computing the distance between two ellipsoids, where the problem is reduced to finding a specific solution in a system of two equations in two variables. Similar algorithms can be designed for computing the distance between an ellipsoid and a simple surface (such as cylinder, cone, and torus). In an experimental implementation (on a 500 MHz Windows PC), the distance between two ellipsoids was computed in less than 0.3 msec on average; and the distance between an ellipsoid and a simple convex surface was computed in less than 0.15 msec on average.

Keywords: Distance computation, collision detection, line geometry, normal congruence, ellipsoid, cylinder, cone, torus.

1 Introduction

Computing the minimum distance between two objects of arbitrary shape has important applications in diverse areas such as computational biology, robotics, CAD/CAM, computer graphics, computer animation, computer games, and virtual reality [7, 15, 23]. For example, distance computation can be used for collision detection in robotics and computer games, and for generating force feedback in haptic rendering, to mention only a few.

There are many efficient algorithms for computing the distance between two polyhedral objects [4, 5, 12, 14, 19, 21, 22, 24, 32]. Distance computation for general free-form objects is, however, more difficult [1, 13, 30]. Conventional algorithms solve a set of polynomial equations, which is time-consuming. In a recent work, Johnson and Cohen [16] present an interesting approach that combines polyhedral approximation and subdivision to NURBS primitives. Using the convex hulls of control polyhedra, two closest NURBS patches are detected and their minimum

[1] The second author has partially been supported by the Austrian Science Fund (FWF) through the Special research programme (SFB) F013, project 15.

distance is then computed by recursive subdivision to the NURBS patches.

In the core of efficient algorithms for collision detection and distance computation are fast geometric tests for two primitives such as spheres, OBB (Oriented Bounding Boxes), LSS (Line Swept Sphere), RSS (Rectangle Swept Sphere), and k-DOP (k-Discrete Oriented Polytope) [14, 21, 22, 32]. We may also consider some other simple geometric primitives for this purpose. Cylinder, cone, torus, surface of revolution, surface of linear extrusion, and canal surface are good candidates for these primitives.

A canal surface is the envelope of a sphere with varying radius, which moves along a space curve [28]. Cylinder, cone, torus, and cyclide are special types of canal surfaces. Kim [20] computes the distance between a canal surface and a simple surface (cylinder, cone, and torus) by reducing the problem to solving a polynomial equation in one variable, which can be computed very quickly. Seong et al. [29] compute the distance between two surfaces of revolution using a simple structure of their Gauss maps, where the normal matching between two surfaces is explicitly given in a closed-form equation. (They consider the surfaces of revolution generated by slope-monotone closed curves; however, the basic approach can be applied to more general surfaces of revolution.)

Wang et al. [33] present an efficient algorithm for testing the separation of two ellipsoids. Separation test is easier than distance computation since a positive distance implies the separation of two objects.

In this paper, we consider the distance computation for two ellipsoids. Typically, the existing algorithms formulate this problem as a system of four equations in four variables. Using line geometry, we obtain a particularly compact formulation of the problem. We transform the distance computation to a system of two equations in two variables.

Rimon and Boyd [27] approximate the distance between two ellipsoids by iteratively computing a sequence of point-ellipsoid distances, where each point-ellipsoid distance is computed by solving a polynomial equation in one variable. In the present paper, we use point-ellipsoid distance computations only a couple of times and generate an initial solution for the distance of two ellipsoids. Then we find a solution in a system of two equations in two variables. The solution is tested whether the corresponding foot points on the ellipsoids realize the minimum distance; otherwise, other solution is searched for a shorter distance. In an experimental implementation, the distance between two ellipsoids was computed in less than 0.3 msec on average (on a 500 MHz Windows PC).

A similar approach can be applied to the distance computation between an ellipsoid and a simple surface (cylinder, cone, and torus). In an experimental implementation, the distance between an ellipsoid and a cylinder was computed in less than 0.1 msec on average; and the distance between an ellipsoid and a cone or the convex part of a torus was computed in less than 0.15 msec on average. Clearly, the problem becomes more difficult when we consider general free-form surfaces, and we are currently exploring the potential of our method in this case. In a recent paper, Bischoff and Kobbelt et al. [2] develop multiresolution techniques for approximating general objects by a collection of ellipsoids, which may be used in order to reduce the general situation to the ellipsoidal case.

The rest of this paper is organized as follows. In Section 2, we briefly review line coordinates. Section 3 presents the normal congruence of a surface, both in parametric and implicit representations. In Section 4, we outline the procedure of computing the distance between two surfaces based on line geometry. Section 5 considers the distance of two ellipsoids, and Section 6 considers the distance between an ellipsoid and a simple surface (such as cylinder, cone, and torus). Experimental results are given in Section 7, where some implementation details are briefly explained. Finally, Section 8 concludes this paper.

2 Line Coordinates

The geometry of lines in three-dimensional space is a classical subject of advanced geometry. Its origins can be traced back to Monge (1771), Plücker (1846) and Klein (1868). Standard textbooks are due to Hoschek [9], Hlavatý [11], and, more recently, to Pottmann and Wallner [28].

Throughout this paper, we shall use *homogeneous coordinates*

$$\mathbf{p} = (p_0, p_1, p_2, p_3) \neq (0, 0, 0, 0) \tag{1}$$

to describe points in 3-space. These coordinates are homogeneous; for any $\lambda \neq 0$, the vectors $\mathbf{p}$ and $\lambda\mathbf{p}$ describe the same point. Consequently, the points in three-dimensional space are identified with one-dimensional subspaces of $\mathbb{R}^4$.

If $p_0 \neq 0$, then the associated Cartesian coordinates of $\mathbf{p}$ are

$$\underline{\mathbf{p}} = (\underline{p}_1, \underline{p}_2, \underline{p}_3) = \left(\frac{p_1}{p_0}, \frac{p_2}{p_0}, \frac{p_3}{p_0}\right). \tag{2}$$

The Cartesian coordinates can be obtained by intersecting the one-dimensional subspace spanned by $\mathbf{p}$ with the plane $p_0 = 1$, and omitting the 0-th coordinate. In the sequel, underlined letters always refer to Cartesian coordinates.

Otherwise, if $p_0 = 0$, the coordinates $\mathbf{p}$ correspond to a so-called ideal point; it can be used to represent the intersection point of all lines with a direction parallel to (p_1, p_2, p_3). If two vectors of homogeneous coordinates are linearly dependent, then they correspond to the same point in 3-space.

The set of lines in 3-space is a four-dimensional manifold. Lines in 3-space can be equipped with homogeneous *line coordinates* (sometimes called the "Plücker coordinates")

$$\begin{aligned}\mathbf{L} &= (L_1, \ldots, L_6) \\ &= (\hat{L}_{01}, \hat{L}_{02}, \hat{L}_{03}, \hat{L}_{23}, \hat{L}_{31}, \hat{L}_{12}). \end{aligned} \qquad (3)$$

If a line is spanned by two points $\mathbf{p}$ and $\mathbf{q}$, then the line coordinates are given by

$$\hat{L}_{ij} = p_i q_j - p_j q_i. \qquad (4)$$

These line coordinates are homogeneous; the vectors $\mathbf{L}$ and $\lambda\mathbf{L}$ correspond to the same line. If we replace $\mathbf{p}$ and $\mathbf{q}$ with two other distinct points on the line, then we get again the original line coordinates, multiplied with a non-zero factor.

In particular, one may generate the line coordinates from a point $(1, \underline{\mathbf{p}})$ with normalized homogeneous coordinates, and from a point $(0, \vec{\mathbf{q}})$ at infinity with $||\vec{\mathbf{q}}|| = 1$ (the latter corresponding to one of the two unit direction vectors of the given line). In this special case, the line coordinates are given by the six-dimensional vector

$$\mathbf{L}^{(0)} = (\vec{\mathbf{q}}, \underline{\mathbf{p}} \times \vec{\mathbf{q}}) \qquad (5)$$

which consists of the *direction vector* $\vec{\mathbf{q}}$ and the so-called *momentum vector* $\underline{\mathbf{p}} \times \vec{\mathbf{q}}$. Note that the latter vector does not depend on the specific choice of the point $\underline{\mathbf{p}}$. The momentum vector is perpendicular to the plane that contains the origin and the line. The general homogeneous line coordinates are obtained by multiplying $\mathbf{L}^{(0)}$ with arbitrary non-zero factors.

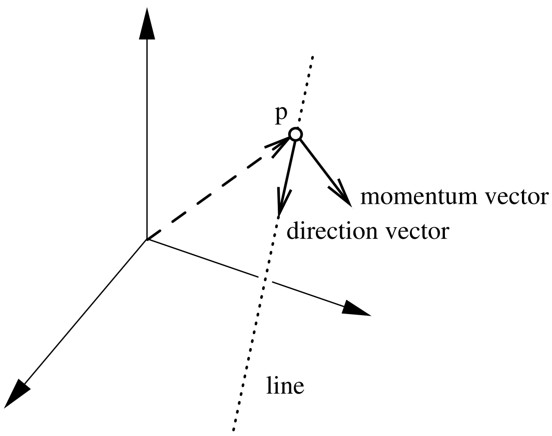


Figure 1: Direction and momentum vector of a line.

The coordinates of any given line satisfy the quadratic equation (Plücker's condition)

$$L_1\, L_4 + L_2\, L_5 + L_3\, L_6 = 0. \qquad (6)$$

This equation is obviously satisfied by the normalized coordinates (5). Due to its homogeneous nature, it is also satisfied for general coordinates (3).

The incidence of two lines can be expressed by line coordinates: if two lines $\mathbf{L}$ and $\mathbf{M}$ intersect, then

$$\begin{aligned}&L_1\, M_4 + L_4\, M_1 + L_2\, M_5 + L_5\, M_2 \\ &+ L_3\, M_6 + L_6\, M_3 = 0\end{aligned} \qquad (7)$$

holds. Consequently, Plücker's condition means that every line intersects with itself.

Using these homogeneous line coordinates, one may identify a line in three-dimensional space with a point $\mathbf{L}$ on the hyperquadric (6). This point will be called the *Plücker image* of the line.

The quadric surface (6), which is embedded in a 5-dimensional real projective space, is called the "Klein quadric" $\mathcal{K}$. Any point $\mathbf{L}$ on it corresponds to a line. Note that points with $L_1 = L_2 = L_3 = 0$ correspond to lines at infinity.

3 The Normal Congruence of a Surface

Consider a surface in three-dimensional space. We may associate with each point the normal, i.e., line through this point which is spanned by the normal vector. The normals form a two-dimensional family (called a "congruence") of lines. It is called the *normal congruence* of the given surface.

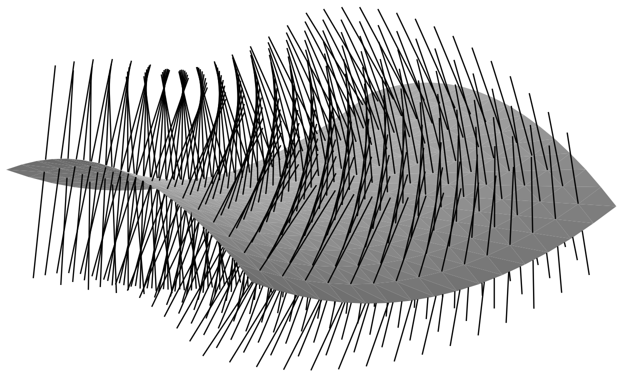


Figure 2: The normal congruence of a surface.

Using line coordinates, the normal congruence can be identified with a two-dimensional surface which is contained in the Klein quadric $\mathcal{K}$. We will refer to this surface as the *Plücker image* of the normal congruence.

3.1 Parametric Representation

If the surface is given by a parametric representation,

$$\mathbf{x}(u, v) = (x_0(u, v), x_1(u, v), x_2(u, v), x_3(u, v)), \quad (8)$$

where $(u, v) \in \Omega \subset \mathbb{R}^2$, then a normal vector can be obtained from

$$\vec{\mathbf{N}}(u, v) = \frac{\partial}{\partial u}\underline{\mathbf{x}}(u, v) \times \frac{\partial}{\partial v}\underline{\mathbf{x}}(u, v). \tag{9}$$

Using the definition of the line coordinates we arrive at a parametric representation of the normal congruence

$$\mathbf{L}(u, v) = (x_0\vec{\mathbf{N}},\ (x_1, x_2, x_3)^T \times \vec{\mathbf{N}}). \tag{10}$$

Clearly, as this is a homogeneous representation, it can be multiplied by arbitrary non-zero factors. If the given surface is rational (i.e., the coordinate functions $x_i(u, v)$ are polynomials), then the normal congruence is again given by a rational parametric representation.

Example 1: We consider the normal congruence of an ellipsoid[1]. Its equation in Cartesian coordinates is

$$\frac{\underline{x}_1^2}{a^2} + \frac{\underline{x}_2^2}{b^2} + \frac{\underline{x}_3^2}{c^2} = 1, \tag{11}$$

and the equation in homogeneous coordinates is

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} + \frac{x_3^2}{c^2} = x_0^2, \tag{12}$$

where the constants $a, b, c > 0$ specify the principal diameters.

A rational quadratic parametric representation (which can be obtained by stereographic projection with the center at the "south pole") is

$$\begin{aligned} x_0 &= 1 + u^2 + v^2, \qquad (13)\\ x_1 &= 2au,\\ x_2 &= 2bv,\\ x_3 &= (1 - u^2 - v^2)c, \end{aligned}$$

with parameters $(u, v) \in \mathbb{R}^2$. (See [6] for a detailed discussion of rational parameterizations of quadric surfaces.) From (9) we get the normal vector

$$\vec{\mathbf{N}} = \frac{4\left(\ 2bcu,\ 2acv,\ ab(1 - u^2 - v^2)\right)^T}{(1 + u^2 + v^2)^3}. \tag{14}$$

Finally, after taking out common factors, we arrive at a parametric representation of the normal congruence,

$$\begin{aligned} \mathbf{L}(u, v) = (\ & 2bcu(1 + u^2 + v^2),\\ & 2acv(1 + u^2 + v^2),\\ & ab(1 + u^2 + v^2)(1 - u^2 - v^2), \qquad (15)\\ & 2(b^2 - c^2)av(1 - u^2 - v^2),\\ & 2(c^2 - a^2)bu(1 - u^2 - v^2),\\ & 4(a^2 - b^2)cuv\ ) \end{aligned}$$

In order to cover the whole ellipsoid with regular parameterizations, we have to use another stereographic projection[2]. For instance, choosing the center at the north pole of the unit sphere, we obtain the same as (13), but with the opposite sign of x_3. The normal congruence is then again given by (15), but with opposite signs of L_1, L_2, and L_6.

By restricting both stereographic projections to parameters (u, v) from the unit disc, $u^2 + v^2 \leq 1$, we will cover both the upper and the lower hemisphere of the ellipsoid. In order to obtain two triangular Bézier patches covering the whole ellipsoid, we apply the mappings to a circumscribed triangle of the unit circle, see Figure 3. The associated normal congruences are then given by quartic triangular Bézier patches.

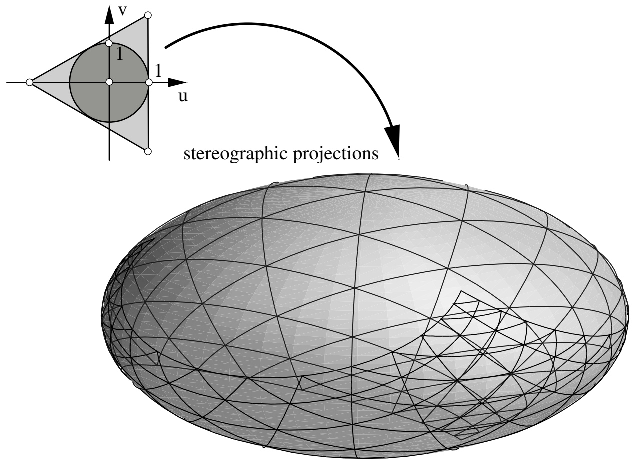


Figure 3: Covering the ellipsoid with triangular patches.

3.2 Implicit Representation

Alternatively, one may describe the normal congruence in implicit form. The implicit representation can be computed either by implicitizing a parametric representation, such as (10), or directly from an implicit representation of the surface.

Suppose that the surface is algebraic, i.e., the zero contour of a trivariate polynomial,

$$f(\underline{x}_1, \underline{x}_2, \underline{x}_3) = 0. \tag{16}$$

Clearly, the normal vector at a surface point is the gradient

$$(N_1, N_2, N_3) = \nabla f(\underline{x}_1, \underline{x}_2, \underline{x}_3). \tag{17}$$

The normal at a point of the surface is spanned by the two points (in homogeneous coordinates) $(1, \underline{x}_1, \underline{x}_2, \underline{x}_3)$ and $(0, N_1, N_2, N_3)$. Consequently, the Plücker coordinates of the normals satisfy the equations

$$\begin{aligned} L_1 &= \lambda N_1,\\ L_2 &= \lambda N_2, \end{aligned}$$

[1] A classical source, including many related references, is [31, No.44], entitled "Die Normalenkongruenz der Flächen zweiter Ordnung".

[2] Otherwise, the parameterization misses the center of projection, and the distance computation may run into numerical problems in its vicinity.

$$
\begin{aligned}
L_3 &= \lambda N_3, \\
L_4 &= \lambda(\underline{x}_2 N_3 - \underline{x}_3 N_2), \\
L_5 &= \lambda(\underline{x}_3 N_1 - \underline{x}_1 N_3), \\
L_6 &= \lambda(\underline{x}_1 N_2 - \underline{x}_2 N_1),
\end{aligned} \tag{18}
$$

for some homogenizing factor $\lambda \neq 0$. Thus, we got 1+3+6=10 equations (16), (17), (18) for the 13 unknowns $\underline{x}_i, N_i, L_i, \lambda$. Using suitable algebraic techniques (direct elimination and/or resultants) we may eliminate the unknowns $\underline{x}_i$, N_i, and λ, giving 3 algebraic equations for the line coordinates $L_1, \dots, L_6$. One of these equations can be chosen as Plücker's condition (6).

In principle, this elimination works for any surface. The resulting equations, however, will be useful for low degree surfaces only.

Example 2: We consider an ellipsoid (11) as an example,

$$
\begin{aligned}
f &= \frac{\underline{x}_1^2}{A^2} + \frac{\underline{x}_2^2}{B^2} + \frac{\underline{x}_3^2}{C^2} - 1 = 0, \\
\nabla f &= \left(\frac{2\underline{x}_1}{A^2}, \frac{2\underline{x}_2}{B^2}, \frac{2\underline{x}_3}{C^2} \right)
\end{aligned} \tag{19}
$$

We may easily eliminate $\underline{x}_1, \underline{x}_2, \underline{x}_3$ and N_1, N_2, N_3 from the equations (16), (17), (18), as it is possible to solve for them and substitute the results into the remaining equations. This leads to 4 equations for the remaining 7 unknowns $L_1, \dots, L_6$ and λ.

$$
\begin{aligned}
2\lambda L_4 &= L_2 L_3 (B^2 - C^2), \\
2\lambda L_5 &= L_1 L_3 (C^2 - A^2), \\
2\lambda L_6 &= L_1 L_2 (A^2 - B^2), \\
4\lambda^2 &= L_1^2 A^2 + L_2^2 B^2 + L_3^2 C^2
\end{aligned} \tag{20}
$$

Note that Plücker's condition (6) can be derived from these equations by computing the sum of the first three equations, multiplied by L_1, L_2, and L_3, respectively.

Still, the homogenizing factor λ has to be eliminated.

Case 1: All three principal diameters are identical, $A = B = C$ (sphere). In this case, the normal congruence, which consists of all lines which pass through the origin of the Cartesian coordinate system, is described by the three equations

$$
L_4 = L_5 = L_6 = 0. \tag{21}
$$

Case 2: Two of the principal diameters are identical, e.g., $A = B$ (ellipsoid of revolution). This entails

$$
L_6 = 0. \tag{22}
$$

Another equation is obtained by solving any of the first two equations for λ and substituting it into the fourth equation. With the help of the first equation we get

$$
L_4^2\,(A^2 L_1^2 + B^2 L_2^2 + C^2 L_3^2) = L_2^2 L_3^2\,(B^2 - C^2)^2. \tag{23}
$$

Case 3: All principal diameters are different. First we obtain certain quadratic equations by solving any two of the first three equations for λ and equating the results. This results in three equations,

$$
L_2 L_5 (A^2 - B^2) + L_3 L_6 (A^2 - C^2) = 0, \tag{24}
$$

and

$$
L_1 L_4 (A^2 - C^2) + L_2 L_5 (B^2 - C^2) = 0, \tag{25}
$$

$$
L_1 L_4 (A^2 - B^2) + L_3 L_6 (C^2 - B^2) = 0. \tag{26}
$$

Note that Plücker's condition (6), combined with any of these three equations, is equivalent to the remaining two equations.

Second, we obtain a quartic equation by eliminating λ from the fourth equation in (20) with the help of any of the first three equations. For example, using the third one, we arrive at

$$
L_6^2\,(A^2 L_1^2 + B^2 L_2^2 + C^2 L_3^2) = L_1^2 L_2^2\,(A^2 - B^2)^2. \tag{27}
$$

To sum up, the homogeneous line coordinates of the normal congruence of a general ellipsoid without rotational symmetries are characterized by equations (24) and (27).

4 Distance Computation

Consider solids in three-dimensional space. We will assume that the two solids are disjoint. Their boundaries are assumed to be C^1 surfaces (orientable manifolds). We denote the boundary surfaces with $\mathcal{L}$ and $\mathcal{M}$. Their minimum distance d is defined as

$$
d = \min_{\mathbf{p} \in \mathcal{L},\, \mathbf{q} \in \mathcal{M}} \|\mathbf{p} - \mathbf{q}\|. \tag{28}
$$

If the minimum distance is reached at the two points $\mathbf{p}$ and $\mathbf{q}$, then the line spanned by $\mathbf{p}$ and $\mathbf{q}$ is a normal of *both* surfaces. Thus, the minimum distance can be computed using the following strategy.

1. Generate the line coordinates of the normal congruences of both surfaces. The Plücker images of the normal congruences are two two-dimensional surfaces which are embedded in the Klein quadric.

2. Intersect the two normal congruences in order to find the joint normals of both surfaces. That is, we have to intersect two two-dimensional surfaces embedded in a four-dimensional manifold. This will result in a finite number of joint normals.

3. Find the two foot points of all joint normals and check their distances.

Clearly, surface-surface intersection (SSI) is a standard problem in geometric computing, and it can be attacked with various techniques, see [10, 25, 26]. In particular, the computations get particularly simple, if one of the surfaces is given in implicit form, and the other surface by a parametric representation. Other techniques are based on the comparison of bounding volumes, such as bounding boxes, which can be derived with the help of the control structure. Many SSI algorithms can easily be generalized to the higher-dimensional situations [26].

5 Distance of Two Ellipsoids

In order to compute the distance of two ellipsoids, we have to intersect the Plücker images of their normal congruences. We formulate this problem in implicit/parametric form.

5.1 First Ellipsoid

The first ellipsoid is given in standard form,

$$\frac{x_1^2}{A^2} + \frac{x_2^2}{B^2} + \frac{x_3^2}{C^2} = x_0^2. \tag{29}$$

Its principal radii A, B, C are assumed to be mutually different. Consequently, the normal congruence satisfies – in addition to Plücker's condition – the equations

$$F(\mathbf{L}) = L_2 L_5 (A^2 - B^2) + L_3 L_6 (A^2 - C^2) = 0, \tag{30}$$

$$\begin{aligned} G(\mathbf{L}) = {} & (L_4^2 + L_5^2 + L_6^2)(A^2 L_1^2 + B^2 L_2^2 + C^2 L_3^2) \\ & - L_1^2 L_2^2 (A^2 - B^2)^2 \\ & - L_2^2 L_3^2 (B^2 - C^2)^2 \\ & - L_1^2 L_3^2 (C^2 - A^2)^2 = 0, \end{aligned} \tag{31}$$

see Case 3 of Section 3.2. (If some of the principal diameters are identical, however, then the simpler equations of Cases 1 and 2 should be used.) Note that (31) is obtained by summing up three alternative derivations of (27), each from the first three equations in (20). We use this slightly more complicated equation since we can deal with degenerate cases easily using this formulation.

5.2 Second Ellipsoid

Again, we start from the standard form of this ellipsoid,

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} + \frac{x_3^2}{c^2} = x_0^2, \tag{32}$$

where the constants a, b, c specify the principal diameters. First we parameterize the normal congruence, see Section 3.1, which leads to a quartic rational surface (15) in line space. Then, we use dual quaternion calculus to move the ellipsoid to a general position in space. More precisely, we apply a translation and a rotation to the ellipsoid. The most compact representation of the transformation of the line coordinates can be given with the help of *dual quaternions*, see [3, 17]. We identify ("∼") the homogeneous line coordinates with vectors consisting of three real and three dual components,

$$\begin{aligned} & \vec{\mathbf{L}} = (L_1, \ldots, L_6) \\ \sim \quad & \mathcal{L} = (L_1, L_2, L_3) + \epsilon (L_4, L_5, L_6), \end{aligned} \tag{33}$$

where ϵ is the dual unit, $\epsilon^2 = 0$. These vectors are then embedded into the ring of dual quaternions, $\mathbb{H}_\epsilon = \mathbb{H} + \epsilon\mathbb{H}$, where the quaternions $\mathcal{Q} \in \mathbb{H}$ are (formally) written as a sum of a scalar part and a three-dimensional vector,

$$\mathcal{Q} = q_0 + (q_1, q_2, q_3) = q_0 + \vec{\mathbf{q}} \tag{34}$$

The conjugate quaternion equals $\tilde{\mathcal{Q}} = q_0 - \vec{\mathbf{q}}$. The quaternion multiplication $\star$ is defined as

$$\begin{aligned} & (q_0 + \vec{\mathbf{q}}) \star (r_0 + \vec{\mathbf{r}}) \\ = \quad & (q_0 r_0 - \vec{\mathbf{q}} \cdot \vec{\mathbf{r}}) + (q_0 \vec{\mathbf{r}} + r_0 \vec{\mathbf{q}} + \vec{\mathbf{q}} \times \vec{\mathbf{r}}). \end{aligned} \tag{35}$$

Analogous rules (always taking the identity $\epsilon^2 = 0$ into account) are valid for dual quaternions.

Now we are ready to represent spatial displacements by dual quaternions. The rotation around the axis spanned by $\vec{\mathbf{a}}$, $||\vec{\mathbf{a}}|| = 1$, through the angle ϕ corresponds to $\mathcal{R}$, and a translation by a vector $\vec{\mathbf{v}}$ corresponds to $\mathcal{T}$, where

$$\mathcal{R} = \cos\frac{\phi}{2} + \sin\frac{\phi}{2}\vec{\mathbf{a}}, \quad \text{and} \quad \mathcal{T} = 1 + \frac{1}{2}\epsilon\vec{\mathbf{v}}. \tag{36}$$

If a spatial displacement is applied to a line $\mathcal{L}$, then the new homogeneous coordinates can be computed from

$$\hat{\mathcal{L}} = \mathcal{T} \star \mathcal{R} \star \mathcal{L} \star \tilde{\mathcal{R}} \star \tilde{\mathcal{T}}. \tag{37}$$

Note that this transformation transforms the pure[3] dual quaternion $\mathcal{L}$ into another pure dual quaternion, and also Plücker's identity is still satisfied. Thus, one may again identify $\hat{\mathcal{L}}$ with homogeneous line coordinates $\hat{\mathbf{L}}$.

Summing up, if the second ellipsoid is subject to a spatial displacement, then the Plücker image transforms according to (37). The resulting normal congruences are denoted with $\hat{\mathbf{L}}(u, v)$.

5.3 Distance Computation

In order to find common normals of the two normal congruences, we substitute the parametric representations $\hat{\mathbf{L}}(u, v)$ into equations (30) and (31). Then we obtain two polynomials of degrees 8 and 16,

$$f(u, v) = F(\hat{\mathbf{L}}(u, v)) = 0, \tag{38}$$

$$g(u, v) = G(\hat{\mathbf{L}}(u, v)) = 0. \tag{39}$$

[3] A quaternion without scalar part is called 'pure'.

The common roots of these polynomials correspond to the joint normals of the ellipsoids.

The minimum distance between two ellipsoids is realized at a specific root of the above system of equations. To find this root efficiently, it is important to generate a good initial solution. Then we apply a standard Newton method in two variables.

To generate an initial solution, we use point-ellipsoid distance computation for a couple of times. Starting from the center of one ellipsoid, we generate a sequence of points by generating alternating foot points on both surfaces. We consider the distance between a point $\underline{\mathbf{p}} = (\underline{p}_1, \underline{p}_2, \underline{p}_3)$ and an ellipsoid

$$\begin{aligned} f &= \frac{\underline{x}_1^2}{A^2} + \frac{\underline{x}_2^2}{B^2} + \frac{\underline{x}_3^2}{C^2} - 1 = 0, \qquad (40) \\ \nabla f &= \left(\frac{2\underline{x}_1}{A^2}, \frac{2\underline{x}_2}{B^2}, \frac{2\underline{x}_3}{C^2} \right). \end{aligned}$$

A foot point $(\underline{x}_1, \underline{x}_2, \underline{x}_3)$ satisfies the following condition, for some t,

$$\begin{aligned} & (\underline{p}_1, \underline{p}_2, \underline{p}_3) - (\underline{x}_1, \underline{x}_2, \underline{x}_3) \\ = \quad & t\nabla f(\underline{x}_1, \underline{x}_2, \underline{x}_3) = t\left(\frac{2\underline{x}_1}{A^2}, \frac{2\underline{x}_2}{B^2}, \frac{2\underline{x}_3}{C^2} \right). \end{aligned}$$

Solving for $\underline{x}_1, \underline{x}_2, \underline{x}_3$, we get

$$\underline{x}_1 = \frac{A^2\underline{p}_1}{A^2 + 2t}, \quad \underline{x}_2 = \frac{B^2\underline{p}_2}{B^2 + 2t}, \quad \underline{x}_3 = \frac{C^2\underline{p}_3}{C^2 + 2t}.$$

Using (40), we get the following equation in t

$$\left(\frac{A^2\underline{p}_1}{A^2 + 2t} \right)^2 + \left(\frac{B^2\underline{p}_2}{B^2 + 2t} \right)^2 + \left(\frac{C^2\underline{p}_3}{C^2 + 2t} \right)^2 = 1.$$

Multiplying both sides by

$$\left(A^2 + 2t\right)^2 \left(B^2 + 2t\right)^2 \left(C^2 + 2t\right)^2,$$

we obtain a polynomial equation of degree 6 in t. The largest positive root corresponds to the footpoint realizing the minimum distance between the point $\underline{\mathbf{p}}$ and the ellipsoid.

6 Distance with Other Simple Surfaces

A similar approach can be used in computing the distance between an ellipsoid and a simple surface, such as cylinder, cone, and torus.

6.1 Cylinder

We consider a cylinder of radius R

$$\begin{aligned} f &= \underline{x}_1^2 + \underline{x}_2^2 - R^2 = 0, \qquad (41) \\ \nabla f &= (2\underline{x}_1, 2\underline{x}_2, 0). \end{aligned}$$

The normal at a point $\underline{\mathbf{x}}$ has the Plücker coordinates

$$\mathbf{L} = (\underline{x}_1, \underline{x}_2, 0, -\underline{x}_2\underline{x}_3, \underline{x}_1\underline{x}_3, 0).$$

Consequently, the normal congruence is characterized by

$$L_3 = 0, \quad L_6 = 0.$$

Thus, the distance computation reduces to solving the following system of polynomial equations of degree four

$$\begin{aligned} f(u,v) &= \hat{L}_3(u,v) = 0, \\ g(u,v) &= \hat{L}_6(u,v) = 0. \end{aligned}$$

6.2 Cone

We consider a cone

$$f = \underline{x}_1^2 + \underline{x}_2^2 - c\underline{x}_3^2 = 0 \qquad (42)$$

Similar to the case of a cylinder, we obtain two simple equations which characterize the normal congruence,

$$(c+1)L_2L_3 - cL_4 = 0, \quad L_6 = 0.$$

Consequently, the distance computation essentially reduces to solving the following system of polynomial equations

$$\begin{aligned} f(u,v) &= (c+1)\hat{L}_2(u,v)\hat{L}_3(u,v) - c\hat{L}_4(u,v) = 0, \\ g(u,v) &= \hat{L}_6(u,v) = 0 \end{aligned}$$

of degree 8 and 4, respectively.

6.3 Torus

We consider a torus with major radius R and minor radius r,

$$(\underline{x}_1^2 + \underline{x}_2^2 + \underline{x}_3^2 + R^2 - r^2)^2 - 4R^2(\underline{x}_1^2 + \underline{x}_2^2) = 0$$

The normal congruence is then characterized by

$$L_4^2 + L_5^2 - R^2L_3^2 = 0, \quad L_6 = 0.$$

Thus, the distance computation leads to the system

$$\begin{aligned} f(u,v) &= \hat{L}_4^2(u,v) + \hat{L}_5^2(u,v) - R^2\hat{L}_3^2(u,v) = 0, \\ g(u,v) &= \hat{L}_6(u,v) = 0. \end{aligned}$$

of polynomial equations of degree 8 and 4, respectively.

7 Experimental Results

We have implemented the proposed algorithm of this paper on a 500 MHz Windows PC using C++. Figure 4 shows an example where an ellipsoid moves around another ellipsoid while smoothly changing its position and orientation. The figure shows four 'snapshots' of the motion, at each of which the distance between two ellipsoids was computed.

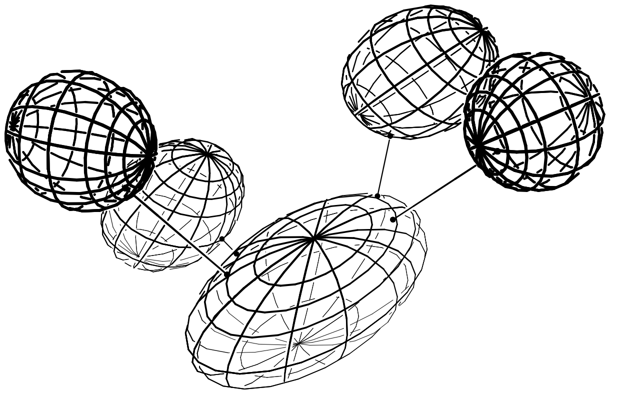


Figure 4: Distance between two ellipsoids.

Without using any witness information from the previous computations and thus starting from the scratch, each distance computation took less than 0.3 msec on average. The performance demonstrates the potential of our approach in real-time collision detection and avoidance.

In the case of computing the distance between an ellipsoid and a cylinder or a cone (see Figure 5), the two equations $f(u,v) = 0$ and $g(u,v) = 0$ have lower degrees (cf. Sections 6.1–6.2). Consequently, the computation can be carried out more efficiently. Using our experimental implementation, the distance between an ellipsoid and a cylinder was computed in less than 0.1 msec on average; and the distance between an ellipsoid and a cone was computed in less than 0.15 msec on average.

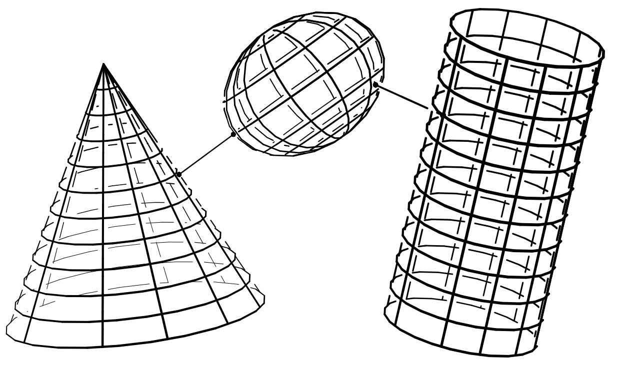


Figure 5: Distance between an ellipsoid and a cylinder/cone.

According to the analysis for a torus presented in Section 6.3, the degrees of $f(u,v) = 0$ and $g(u,v) = 0$ are 8 and 4, respectively, which are the same as in the case of a cone. Consequently, each local minimum distance between an ellipsoid and a torus was computed in less than 0.15 msec on average. When the foot point appears on the convex part of the torus, the distance thus computed is the global minimum distance. However, the foot point may appear on the non–convex part of the torus as shown in the second and third 'snapshots' of the moving ellipsoid of Figure 6. In this more difficult case, the iterative search may fall into a local minimum, in particular, when the ellipsoid passes through the center of the torus. In the worst case, one may have to compute all discrete solutions of $f(u,v) = g(u,v) = 0$, and search for the solution that corresponds to the global minimum distance[4]. We are currently investigating an efficient method for pruning redundant solutions.

Computing the distance between an ellipsoid and a free-form surface is the most difficult case. In this case, we use a parametric representation for the line coordinates of the free-form surface. The line coordinates of an ellipsoid are represented implicitly; and they are transformed using dual quaternions for a translation and a rotation. Again the main difficulty arises when a local foot point appears in a non–convex part of the free-form surface. A technique similar to Johnson and Cohen [16] would generate candidate pairs of foot points efficiently. We are currently investigating a more efficient method that is based on the special structure of an ellipsoid. For example, an ellipsoid can be bounded by a discrete union of balls very compactly. The distances from the centers of these balls may provide a reasonably good approximation to the global minimum distance. An ellipsoid has a relatively simple medial axis transformation, which may provide a compact way of approximating an ellipsoid by a union of balls.

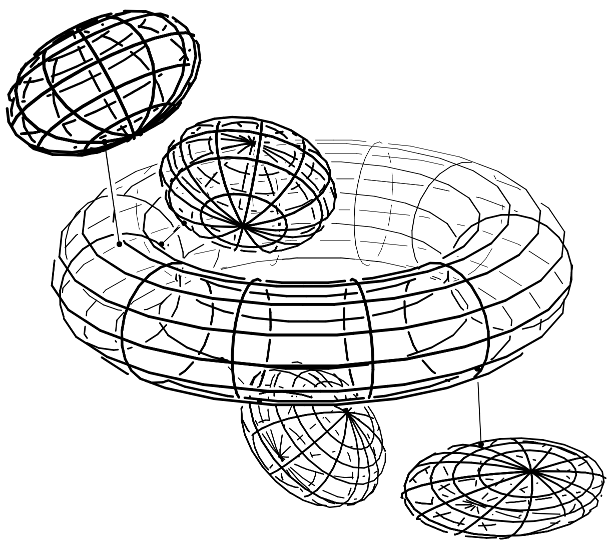


Figure 6: Distance between an ellipsoid and a torus.

[4]Note that, by composing the parametric and implicit equations of the normal congruences, we are able to detect *all* potential joint normals. Clearly, this requires some additional computational effort, which is not justified in the general case.

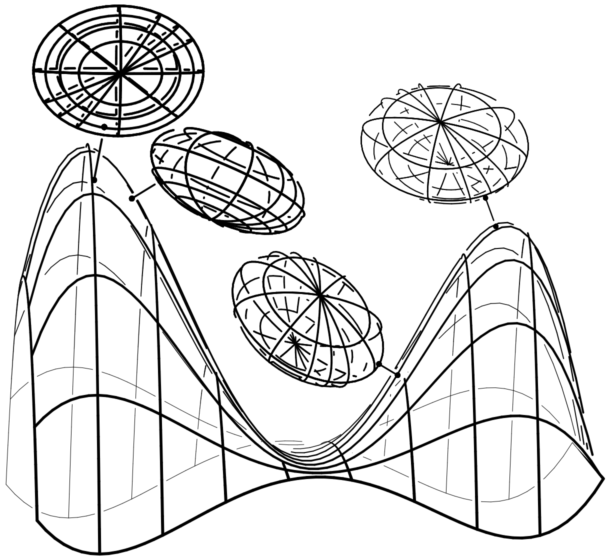


Figure 7: Distance between an ellipsoid and a free-form surface.

8 Conclusions and Future Work

Using line geometry, it is possible to formulate the task of distance computation as a surface-surface-intersection (SSI) problem, involving two two-dimensional surfaces which are embedded in a four-dimensional manifold. This has been demonstrated in the case of two ellipsoids. In this situation, we are able to formulate the SSI problem in implicit/parametric form, leading to a system of only two polynomial equations in two variables. Generally, these equations are of degree 8 and 16.

This technique works is not limited to ellipsoids; it can be applied to any pair of quadric surfaces. If one of them is a simple surface, such as circular cone or cylinder, then the degree of these equations reduces substantially.

Conventional methods attack the same problem by solving a system of four equations in four variable. In principle, by eliminating two variables, the system can be reduced to two equations in two variables. However, multivariate elimination is a non-trivial task. In this paper, we have demonstrated some special yet important cases, where the elimination can be done in a straightforward manner.

According to our experiments, our method appears to be particularly useful in complicated cases, where the joint normal of the two surfaces is attached to non–convex, or nearly parallel, regions of both surfaces. In this situation, our technique will help to avoid potential problems with local minima, as we are able to detect all potential intersections of the two normal congruences. Clearly, this issue will become even more important for non–convex objects, such as general free–form surfaces.

Future research will focus on more general classes of surfaces, which are characterized by particularly simple normal congruences. Here, among other candidates, we plan to investigate the potential of the class of LN (Linear Normal) surfaces [18], which are characterized by a *linear* distribution of normal vectors. (Note that Phong shading model is based on the assumption that the surface normal changes linearly along each scanline [8].) These surfaces can be shown to have rational offset surfaces and normal congruences of relatively low degree.

References

[1] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics (Proc. of ACM SIGGRAPH'90)*, Vol. 24, No. 4, pp. 19–28, 1990.

[2] S. Bischoff and L. Kobbelt. Ellipsoid decomposition of 3D-models. Manuscript.

[3] O. Bottema and B. Roth. *Theoretical Kinematics*. Dover, New York 1990.

[4] S. Cameron. A comparison of two fast algorithm for computing the distance between convex polyhedra. *IEEE Trans. on Robotics and Automation*, 13(6):915–920, 1997.

[5] K. Chung and W. Wang. Quick collision detection of polytopes in virtual environments. *Prof. of VRST'96*, pp. 125-131, 1996.

[6] R. Dietz, J. Hoschek, and B. Jüttler. An algebraic approach to curves and surfaces on the sphere and on other quadric. *Comp. Aided Geom. Design*, 10:211–229, 1993.

[7] D.H. Eberly. *3D Game Engine Design*. Morgan Kaufmann, San Francisco, 2001.

[8] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice*, 2nd Ed. Addision-Wesley, Reading, 1990.

[9] J. Hoschek. Liniengeometrie, BI Hochschulskripten 733a/b, Bibliographisches Institut Zürich, 1971.

[10] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. AK Peters, Wellesley, 1993.

[11] V. Hlavatý. *Differential Line Geometry*. Noordhoff, Groningen 1953.

[12] E. Gilbert, D. Johnson, and S. Keerthi. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE J. of Robotics and Automation*, 6:193–203, 1988.

[13] E. Gilbert and C. Foo. Computing the distance between general convex objects in three-dimensional space. *IEEE Trans. on Robotics and Automation*, 6(1):53–61, 1990.

[14] S. Gottschalk, M. Lin and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection, *Proc. of ACM SIGGRAPH'96*, pp. 171–180, 1996.

[15] P. Jiménez, F. Thomas, and C. Torras. 3D collision detection: a survey. *Computers & Graphics*, 25:269–285, 2001.

[16] D. Johnson and E. Cohen. A framework for efficient minimum distance computations. *Proc. of IEEE Conf. on Robotics and Automation*, pp. 3678–368, 1998.

[17] B. Jüttler. Visualization of moving objects using dual quaternion curves. *Computers & Graphics*, 18(3):315–326, 1994.

[18] B. Jüttler and M.L. Sampoli. Hermite interpolation by piecewise polynomial surfaces with rational offsets, *Comp. Aided Geom. Design* 17:361–385, 2000.

[19] K. Kawachi and H. Suzuki. Distance computation between non-convex polyhedra at short range based on discrete Voronoi regions. *Proc. of Geometric Modeling and Processing*, Hong Kong, 123–128, 2000.

[20] K.-J. Kim. Minimum distance between a canal surface and a simple surface. Submitted to *Computer-Aided Design*, 2001.

[21] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[22] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. *Proc. of IEEE Conf. on Robotics and Automation*, 1999.

[23] M.C. Lin and S. Gottschalk. Collision detection between geometric models: a survey. *Mathematics of Surfaces VIII*, R. Cripps, editor, Information Geometers, 37–56, 1998.

[24] M.C. Lin and J.F. Canny. A fast algorithm for incremental distance calculation. *Proc. of IEEE Int'l Conference on Robotics and Automation*, Sacramento, Califonia, 1008–1014, 1991.

[25] N. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, 1993.

[26] N. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*, Springer-Verlag, Heidelberg, 2002.

[27] E. Rimon and S. Boyd. Obstacle collision detection using best ellipsoid fit. *Journal of Intelligent and Robotics Systems*, 18:105–126, 1997.

[28] H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer, Heidelberg, 2001.

[29] J.-K. Seong, M.-S. Kim, and K. Sugihara. The Minkowski sum of two simple surfaces generated by slope-monotone closed curves. To appear in *Proc. of Geometric Modeling and Processing*, Tokyo, July 10–12, 2002.

[30] J. Snyder, A. Woodbury, K. Fleischer, B. Currin, and A. Barr. Interval methods for multi-point collisions between time-dependent curved surfaces. *Proc. of ACM SIGGRAPH'93*, pp. 321–334, 1996.

[31] O. Staude, Flächen zweiter Ordnung und ihre Systeme und Durchdringungskurven, Encykl. d. math. Wiss. 3_2 (1904), 161–256.

[32] S. Quinlan. Efficient distance computation between non-convex objects. *Proc. of IEEE Conf. on Robotics and Automation*, pp. 3324–3329, 1994.

[33] W. Wang, J. Wang, and M.-S. Kim. An algebraic condition for the separation of two ellipsoids. *Comp. Aided Geom. Design*, 18(6):531–539, 2001.