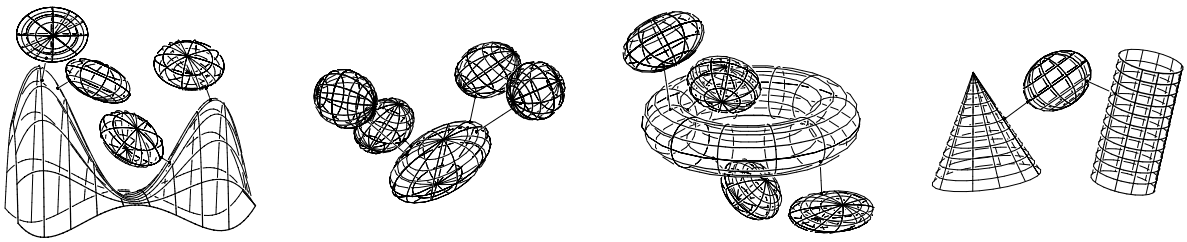
Computing Distances Between Surfaces Using Line Geometry

Kyung-Ah Sohn[a], Bert Jüttler[b], 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

We present 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 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, or 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, 24]. To give just a couple of examples, distance computation can be used for collision detection in robotics and computer games, or for generating force feedback in haptic rendering.

There are many efficient algorithms for computing the distance between two polyhedral objects [4, 5, 12, 14, 19, 21, 22, 25, 33]. Distance computations among general free-form objects are, however, more difficult [1, 13, 31]. 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 of NURBS primitives. Using the convex hulls of control polyhedra, the two closest NURBS patches are detected and their minimum distance is then computed by recursive subdivision.

At the core of efficient algorithms for collision detection and distance computation are fast geometric tests which operate on two enclosing volumes, which may be spheres,

Proceedings of the 10 th Pacific Conference on Computer Graphics and Applications (PG'02)
0-7695-1784-6/02 $17.00 © 2002 **IEEE**

oriented bounding boxes (OBB), line swept spheres (LSS), rectangle swept spheres (RSS), or k-discrete oriented polytopes (k-DOP) [14, 21, 22, 33]. We can also use other simple geometric primitives for this purpose: Cylinders, cones, tori, surfaces of revolution, surfaces of linear extrusion, and canal surfaces are good candidates.

A canal surface is the envelope of a sphere with varying radius, which moves along a space curve [29]. The 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, or torus) by reducing the problem to a polynomial equation in one variable, which can be solved very quickly. Seong et al. [30] compute the distance between two surfaces of revolution by exploiting the simple structure of their Gauss maps; the problem of matching normals between the surfaces can then be expressed as 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. [34] present an efficient algorithm for testing whether two ellipsoids are separate. A separation test is easier than a distance computation since a positive distance between the objects implies separation: its magnitude need not be computed.

In this paper, we consider how to compute the distance between two ellipsoids. Typically, existing algorithms formulate this problem as a system of four equations in four variables. By using line geometry, we are able to transform the distance computation to a system of two equations in two variables. In a recent work, Lennerz and Schömer [23] also reduce the distance computation between two general quadrics to a system of two equations in two variables. In their algebraic approach, the unknown variables are Lagrange multipliers, from which the corresponding foot-points can be computed. On the other hand, the equations of our method are explicitly represented by the surface parameters of an ellipsoid. Moreover, it is more efficient to manipulate line coordinates when the surfaces are moving under translation and rotation.

There are different motivations for studying the ellipsoid case. First, ellipsoids and degenerate quadrics, such as cylinders and cones, are important primitives for solid modeling systems, and the results of our paper can be applied directly to them. Second, ellipsoids can be used for efficiently bounding more general solids. Finally, the solution of the ellipsoid distance problem gives some insight into solving the same problem with more complicated surfaces.

Rimon and Boyd [28] approximate the distance between two ellipsoids by iteratively finding a sequence of point-ellipsoid distances, each of which is computed by solving a polynomial equation in one variable. In the present paper, we use point-ellipsoid distance computations only a few times, to generate an initial solution for the distance between two ellipsoids. Then we find a solution using a system of two equations in two variables. This solution is tested to determine whether the resulting foot-points on the ellipsoids correspond to the minimum distance; otherwise, a new solution is computed. 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, or 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 application of our method to this problem. Approximate solutions are always an alternative: in a recent paper, Bischoff and Kobbelt [2] develop multiresolution techniques for approximating general objects by a collection of ellipsoids, thus allowing wider use of an ellipsoidal solution.

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 our procedure for computing the distance between two surfaces based on line geometry. In Section 5 we consider the distance between two ellipsoids, and in Section 6 the distance between an ellipsoid and a simple surface (such as cylinder, cone, or torus). Experimental results are given in Section 7, where some implementation details are briefly explained. Section 8 concludes this paper.

2 Line Coordinates

Lines in three-dimensional space are an important topic in classical geometry, and can be traced back to Monge (1771), Plücker (1846) and Klein (1868). Standard textbooks include those by Hoschek [9], Hlavatý [11] and, more recently, Pottmann and Wallner [29].

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

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

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

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

Otherwise, if $p_0 = 0$, the coordinates $\mathbf{p}$ correspond to a so-called ideal point, which 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 in 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")

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

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

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

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 by a non-zero factor.

In particular, it is possible to generate the coordinates of a line 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 corresponds 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

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

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 momentum vector does not depend on the specific choice of the point $\underline{\mathbf{p}}$, and that it is perpendicular to the plane that contains the origin and the line. The general homogeneous line coordinates are obtained by multiplying $\mathbf{L}^{(0)}$ by arbitrary non-zero factors.

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

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

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

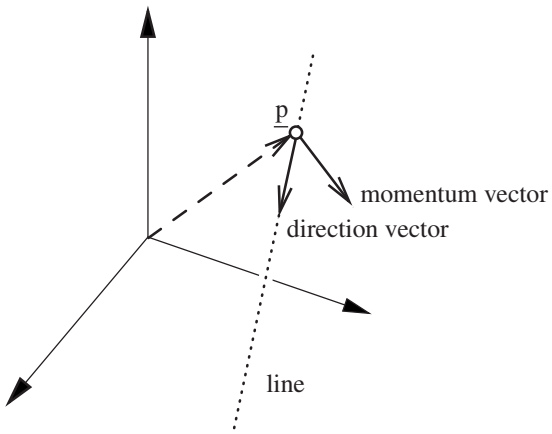


Figure 1: Direction and momentum vector of a line.

The incidence of two lines can be expressed in terms of the corresponding line coordinates: if two lines $\mathbf{L}$ and $\mathbf{M}$ intersect, then the condition

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

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 quadric surface which is represented by the equation (6) that expresses the Plücker condition. This point will be called the *Plücker image* of the line, and the corresponding quadric surface (6), which is embedded in a 5-dimensional real projective space, is called the "Klein quadric" $\mathcal{K}$; and any point $\mathbf{L}$ that lies on it corresponds to a line in 3-space. Note that points for which $L_1 = L_2 = L_3 = 0$ correspond to lines at infinity.

3 The Normal Congruence of a Surface

Consider a surface in three-dimensional space. Each point on such a surface may be associated with a normal, i.e., a line through this point which is spanned by the normal vector. These normals form a two-dimensional family (called a "congruence") of lines, which is called the *normal congruence* of the given 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 a given surface has a parametric representation,

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

0-7695-1784-6/02 $17.00 © 2002 IEEE

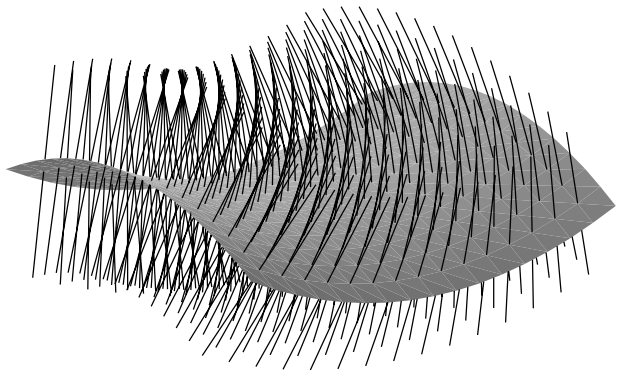


Figure 2: The normal congruence of a surface.

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 line coordinates (5) 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 again the normal congruence has a rational parametric representation.

Example 1: 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 in homogeneous coordinates it 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

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

[1]A classical source, including many related references, is "Die Normalenkongruenz der Flächen zweiter Ordnung" [32, No. 44].

with parameters $(u, v) \in \mathbb{R}^2$. (See Dietz et al. [6] for a detailed discussion of rational parameterizations of quadric surfaces.) From Equation (9) we can obtain 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,

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

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

By restricting both stereographic projections to parameters (u, v) from the unit disc, $u^2 + v^2 \leq 1$, we can cover both the upper and the lower hemisphere of the ellipsoid.

In order to obtain a robust implementation of our method, one should represent both the surfaces and the associated normal congruences in Bernstein–Bézier form, as this is more stable than standard representations (e.g., with respect to the power basis). The ellipsoid can be covered with two triangular Bézier patches. These patches can be found by applying the two stereographic projections to a circumscribed triangle of the unit circle, see Figure 3. The associated normal congruences are then described by quartic triangular Bézier patches.

3.2 Implicit Representation

An alternative is to describe the normal congruence in implicit form. This can be computed either by implicitizing a parametric form, such as Equation (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 point on the surface is the gradient

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

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

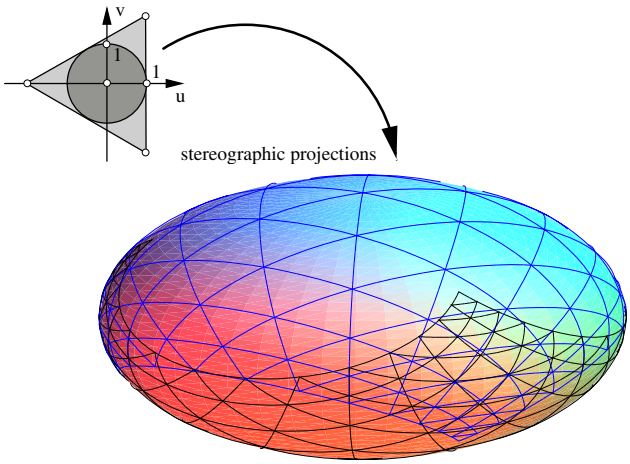


Figure 3: Covering the ellipsoid with triangular patches.

which 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

$$\begin{aligned} L_1 &= \lambda N_1, \\ L_2 &= \lambda N_2, \\ L_3 &= \lambda N_3, \\ L_4 &= \lambda(\underline{x}_2 N_3 - \underline{x}_3 N_2), \\ L_5 &= \lambda(\underline{x}_3 N_1 - \underline{x}_1 N_3), \\ L_6 &= \lambda(\underline{x}_1 N_2 - \underline{x}_2 N_1), \end{aligned} \tag{18}$$

for some homogenizing factor $\lambda \neq 0$. Thus, we have 1+3+6=10 equations (16), (17) and (18) for the 13 unknowns $\underline{x}_i, N_i, L_i, \lambda$. Using suitable algebraic techniques (direct elimination, resultants or both) we may eliminate the unknowns $\underline{x}_i$, N_i, and λ, yielding just 3 algebraic equations for the line coordinates $L_1, \ldots, L_6$. 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: 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 Equations (16), (17) and (18), as it is possible to solve for these variables and substitute the results into the remaining equations. This leads to the following 4 equations for the remaining 7 unknowns $L_1, \ldots, L_6$ 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.

The homogenizing factor λ remains 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 either 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 solve any two of the first three equations for λ and equate the results. This yields three quadratic equations:

$$L_2 L_5 (A^2 - B^2) + L_3 L_6 (A^2 - C^2) = 0, \tag{24}$$
$$L_1 L_4 (A^2 - C^2) + L_2 L_5 (B^2 - C^2) = 0, \tag{25}$$
$$L_1 L_4 (A^2 - B^2) + L_3 L_6 (C^2 - B^2) = 0. \tag{26}$$

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 equality in Equation (20), with the help of any of the first three equations. For example, using the third equation, 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 two disjoint solids in three-dimensional space, and let us assume that their boundaries are C^1 surfaces (orientable manifolds). We denote these boundary surfaces as

$\mathcal{L}$ and $\mathcal{M}$. The minimum distance between them 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 corresponds to $\mathbf{p}$ and $\mathbf{q}$, then the line spanned by $\mathbf{p}$ and $\mathbf{q}$ is normal to *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. This means finding the intersection of two two-dimensional surfaces embedded in a four-dimensional manifold; it will result in a finite number of joint normals.
3. Find the two foot-points of all joint normals and check the corresponding distances.

Surface-surface intersection (SSI) is a standard problem in geometric computing, and it can be attacked with various techniques [10, 26, 27]. The computations are greatly simplified if one of the surfaces is given in implicit form, and the other has a parametric representation. For intractable cases, many approximate techniques are available, based on the comparison of bounding volumes, such as bounding boxes, which may be derived with the help of the control structure. Many SSI algorithms can easily be generalized to higher-dimensional situations [27].

5 Distance between Two Ellipsoids

In order to compute the distance between two ellipsoids, we 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 and C are assumed to be different. Consequently, the normal congruence satisfies—in addition to Plücker's condition—the equations

$$F(\mathbf{L}) = L_2L_5(A^2-B^2)+L_3L_6(A^2-C^2)=0, \tag{30}$$

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

See Case 3 of Section 3.2. (If some of the principal diameters are identical, however, then the simpler equations corresponding to Cases 1 and 2 may be used.) Note that Equation (31) is obtained by summing up three alternative derivations of Equation (27), each from the first three equalities in Equation (20). This slightly more complicated formulation allows us to deal with degenerate cases more easily.

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 so-called *dual quaternions* [3, 17].

Quaternions are frequently used for the compact and efficient representation of 3D rotations in computer graphics and robot kinematics. In the following treatment, the quaternions $\mathcal{Q} \in \mathbb{H}$ are (formally) written as a sum of a scalar part and a three-dimensional vector,

$$\mathcal{Q} = q_0 + (q_1, q_2, q_3) = q_0 + \vec{\mathbf{q}} \tag{33}$$

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

$$\begin{aligned} & (q_0+\vec{\mathbf{q}}) \star (r_0+\vec{\mathbf{r}}) \\ = {} & (q_0r_0 - \vec{\mathbf{q}}\cdot\vec{\mathbf{r}}) + (q_0\vec{\mathbf{r}} + r_0\vec{\mathbf{q}} + \vec{\mathbf{q}}\times\vec{\mathbf{r}}). \end{aligned} \tag{34}$$

By introducing the dual unit ϵ satisfying $\epsilon^2 = 0$, the quaternion-based techniques for rotations can be extended to general rigid-body displacements. This approach is particularly useful in connection with line coordinates.

We identify ("$\sim$") the homogeneous line coordinates with dual vectors consisting of three real and three dual components,

$$\begin{aligned} & \vec{\mathbf{L}} = (L_1, \ldots, L_6) \\ \sim {} & \mathcal{L} = (L_1, L_2, L_3) + \epsilon(L_4, L_5, L_6). \end{aligned} \tag{35}$$

These vectors are then embedded into the ring of dual quaternions, $\mathbb{H}_\epsilon = \mathbb{H} + \epsilon\mathbb{H}$.

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 the product

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

Note that this transforms the pure[3] dual quaternion $\mathcal{L}$ into another pure dual quaternion, while continuing to satisfy Plücker's identity. Thus, one may again identify $\hat{\mathcal{L}}$ with the homogeneous line coordinates $\hat{\mathbf{L}}$.

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

5.3 Distance Computation

In order to find the 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:

$$f(u, v) = F(\hat{\mathbf{L}}(u, v)) = 0, \tag{38}$$
$$g(u, v) = G(\hat{\mathbf{L}}(u, v)) = 0. \tag{39}$$

The common roots of these polynomials correspond to the joint normals of the ellipsoids[4].

The minimum distance between two ellipsoids corresponds to 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 the initial solution, we repeat a point-ellipsoid distance computation a few times. Starting from the center of one ellipsoid, we generate foot-points on each surface in turn. Now consider the distance between a point $\underline{\mathbf{p}} = (\underline{p}_1, \underline{p}_2, \underline{p}_3)$ and an ellipsoid

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

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

[4] The degrees of both equations can be reduced to 6 and 12, by taking out common factors. They have only 36 solutions over $\mathbb{C}$, as their resultants with respect to u or v are polynomials of degree 36. The number of real solutions varies, depending on the relative position of the two ellipsoids. When the two ellipsoids are separate, typically one obtains between 4 and 8 joint normals, but we also found examples with 16 of them.

For some value of t, a foot-point $(\underline{x}_1, \underline{x}_2, \underline{x}_3)$ satisfies the following condition:

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

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 Equation (40), we obtain 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 arrive at a polynomial equation of degree 6 in t. The largest positive root of this equation corresponds to the foot-point realizing the minimum distance between the point $\underline{\mathbf{p}}$ and the ellipsoid.

6 Distances to Other Simple Surfaces

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

6.1 Cylinder

Consider a cylinder of radius R:

$$f = \underline{x}_1^2 + \underline{x}_2^2 - R^2 = 0, \tag{41}$$
$$\nabla f = (2\underline{x}_1, 2\underline{x}_2, 0).$$

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:

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

6.2 Cone

Consider a cone

$$f \;=\; \underline{x}_1^2 + \underline{x}_2^2 - c\underline{x}_3^2 = 0. \tag{42}$$

As in 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

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 of polynomial equations

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

of degree 8 and 4 respectively.

7 Experimental Results

We have implemented the algorithm proposed in this paper on a 500 MHz Windows PC using C++. Figure 4 shows an example in which an ellipsoid moves around another ellipsoid while smoothly changing its position and orientation. The figure is made up of four 'snapshots' of the motion, each of which corresponds to a distance computation. In each case, the distance between the ellipsoids was computed without using any information from previous computations. Nevertheless, each distance computation took less than 0.3 msec on average, which demonstrates the potential of our approach in real-time collision detection and avoidance.

In 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 (see 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.

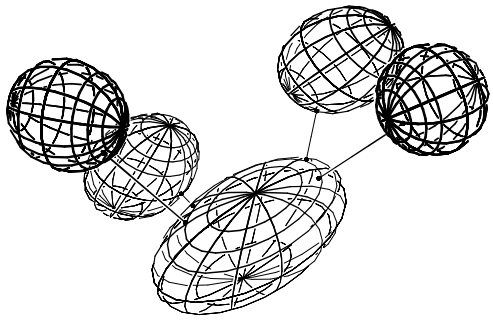


Figure 4: Distance between two ellipsoids.

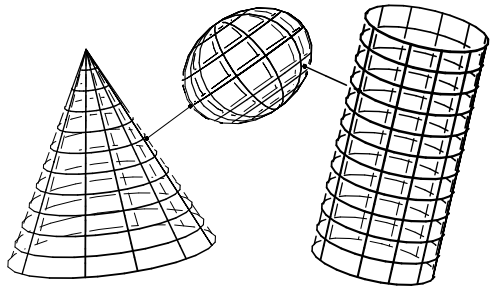


Figure 5: Distance between an ellipsoid and a cylinder/cone.

From 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; and they are the same 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 views of the moving ellipsoid in Figure 6. In this more difficult case, the iterative search may fall into a local minimum; for instance, the ellipsoid may pass through the center of the torus. In the worst case, it may be necessary 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[5]. We are currently investigating an efficient method for pruning redundant solutions in these computations.

Finding the distance between an ellipsoid and a free-form surface is the most difficult situation that we have ad-

[5]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.

dressed. In this case, we use a parametric representation for the line coordinates of the free-form surface, and an implicit representation for the line coordinates of the ellipsoid; the latter are then transformed using dual quaternions corresponding to 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 that of Johnson and Cohen [16] would generate candidate pairs of foot-points effectively. However, we are currently investigating a potentially more efficient method that is based on the special structure of an ellipsoid. It is known that an ellipsoid can be bounded by a discrete union of balls very compactly, and distances based on 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 creating good approximations of ellipsoids as unions of balls.

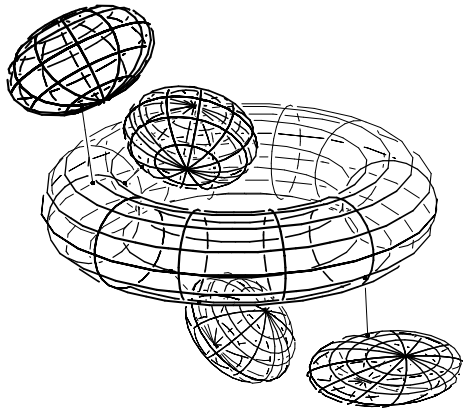


Figure 6: Distance between an ellipsoid and a torus.

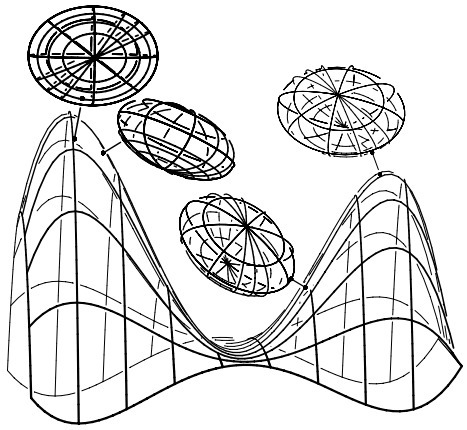


Figure 7: Distance between an ellipsoid and a free-form surface.

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 that situation, we were able to formulate the SSI problem in implicit/parametric form, leading to a system of only two polynomial equations in two variables, which are usually of degree 8 and 16 (but can easily be reduced to degree 6 and 12).

This technique 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 is substantially reduced.

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

On the basis of 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 helps to avoid potential problems with local minima, as we are able to detect all potential intersections of the two normal congruences. Clearly, this issue becomes even more important in considering 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 linear normal (LN) 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].) LN surfaces can be shown to have rational offset surfaces and normal congruences of relatively low degree.

Acknowledgements

The authors would like to thank the anonymous reviewers for their useful comments. This research was supported in part by the Austrian Science Fund (FWF) through the Special research programme (SFB) F013, project 15, in part by the Korean Ministry of Science and Technology (MOST) under the National Research Lab project, and in part by the Korean Ministry of Information and Communication (MIC)

under the program of IT Research Center for Computer Graphics and Virtual Reality.

References

[1] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics (Proc. of ACM SIGGRAPH'90)*, 24(4):19–28, 1990.

[2] S. Bischoff and L. Kobbelt. Ellipsoid decomposition of 3D-models. *Proc. of the First Symp. of 3D Data Processing, Visualization and Transmission,* IEEE Press, pp. 480–488, 2002.

[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*, pp. 123–128, Hong Kong, April 10–12, 2000.

[20] K.-J. Kim. Minimum distance between a canal surface and a simple surface. To appear in *Computer-Aided Design*, 2002.

[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*, 2000.

[23] C. Lennerz and E. Schömer. Efficient distance computation for quadratic curves and surfaces. *Proc. of Geometric Modeling and Processing*, pp. 60–69, Saitama, Japan, July 10–12, 2002.

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

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

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

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

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

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

[30] J.-K. Seong, M.-S. Kim, and K. Sugihara. The Minkowski sum of two simple surfaces generated by slope-monotone closed curves. *Proc. of Geometric Modeling and Processing*, pp. 33–42, Saitama, Japan, July 10–12, 2002.

[31] 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.

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

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

[34] 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.