Computing a 3-dimensional Cell in an Arrangement of Quadrics: Exactly and Actually!

Nicola Geismann
Fachrichtung 6.2 Informatik
Universität des Saarlandes
66123 Saarbrücken, Germany
nicola@cs.uni-sb.de

Michael Hemmer Elmar Schömer
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
hemmer | schoemer@mpi-sb.mpg.de

ABSTRACT

We present two approaches to the problem of calculating a cell in a 3-dimensional arrangement of quadrics. The first approach solves the problem using rational arithmetic. It works with reductions to planar arrangements of algebraic curves. Degenerate situations such as tangential intersections and self-intersections of curves are intrinsic to the planar arrangements we obtain. The coordinates of the intersection points are given by the roots of univariate polynomials. We succeed in locating all intersection points either by extended local box hit counting arguments or by globally characterizing them with simple square root expressions. The latter is realized by a clever factorization of the univariate polynomials. Only the combination of these two results facilitates a practical and implementable algorithm.

The second approach operates directly in 3-space by applying classical solid modeling techniques. Whereas the first approach guarantees a correct solution in every case the second one may fail in some degenerate situations. But with the help of verified floating point arithmetic it can detect these critical cases and is faster if the quadrics are in general position.

1. INTRODUCTION

We consider 3-dimensional arrangements of quadric surfaces, for example ellipsoids, paraboloids, and hyperboloids. Our goal is to develop an exact, implementable, and fast algorithm that computes the topological description of a cell.

Efficient methods for the calculation of arrangements of algebraic surfaces are an important area of research in different branches of computer science. For example, the problem typically arises in solid modeling (see [11]) when performing boolean operations for quadric surfaces, which play an important role in the design of mechanical parts. The algorithms in CAD systems have the advantage that they are quite fast. They profit from floating point arithmetic and often use numerical procedures for tracing the intersection curves and then approximate them as spline curves. But just this makes them very sensitive to approximation and rounding errors. Thus they achieve the good running time at the expense of exactness in degenerate situations which are nevertheless frequent in the design of geometric objects.

Also in computational geometry there is a great focus on computing arrangements. For a good and brief overview see [10]. The geometric methods have the drawback that nearly all of them are based on an idealized real arithmetic. The assumption is that all, even irrational, numbers are representable and that one can deal with them in constant time per operation. This postulate is not in accordance with real computers. However, algorithms coping with arrangements of hyperplanes can be implemented with exact rational arithmetic and with a good performance, because they only deal with linear algebraic primitives. The situation becomes more difficult if the hyperplanes are replaced by arbitrary algebraic surfaces. There are many theoretical results about constructing arrangements of curved surfaces which rely on idealized real arithmetic, see for example [5], and a few which discuss the topological description of a single planar algebraic curve using rational arithmetic [19]. But it is still a wide area of research how to implement the necessary algebraic primitives in a robust way without accepting an extremely high running time. Since the coordinates of the intersection points of three surfaces will in general be irrational numbers, one has to compute with algebraic numbers. This often causes a dramatic loss of speed.

Thus the challenge lies in finding an implementable method for calculating a cell in an arrangement of curved surfaces that uses exact algebraic computation and also has an acceptable running time. We are the first who provide such an algorithm for a set of quadrics. Of course we could exploit the fact that a single cell of the arrangement has a smaller complexity than that of the whole arrangement, but we want to concentrate on the handling of the algebraic primitives, because they have a very strong impact on the practicability and robustness of the whole algorithm. There are a lot of degenerate cases that can appear. Our method is of universal nature in the sense that it can deal with each kind of degenerate input. Especially it can cope with all situations described in [16] and [8].

There are some geometric and algebraic libraries that concentrate on similar implementation issues. For example LEDA [15], CGAL [3], APU [18], and MAPC [13]. LEDA provides basic data types and a lot of geometric algorithms, CGAL does the same with more emphasis on the geomet-

SCG'01, June 3-5, 2001, Medford, Massachusetts, USA.

ric aspect, APU is a tool for real algebraic numbers, and MAPC is a library for exact computation and manipulation of algebraic points and curves.

We have developed and implemented two different approaches. The first one (section 2) uses exact algebraic computation and operates similar to the cylindrical algebraic decomposition [5]. We reduce the problem to computing planar arrangements of algebraic curves of degree up to 4 (see Figure 1). In these arrangements self-intersections and tangential intersection points are very common. During the calculation we use algebraic techniques like resultants and root separation of univariate polynomials. Root separation can be done for example with the help of a Sturm sequence computation or the algorithm of Uspensky [6]. We further apply a method called box hit counting, which was also used in [13]. With its help we can detect a lot of event points of the planar arrangement, namely transversal intersection points of two curves (Figure 1, point c) and points of a curve that have a vertical tangent (Figure 1, point a). But it fails for tangential intersection points of two curves (Figure 1, point d) and for self-intersections (Figure 1, point b). Our contribution, and what is new, is that we succeed in fixing all event points including tangential intersection points and self-intersections while keeping the running time low. This works for two reasons: First we succeed in factorizing all univariate polynomials describing the x- and y-coordinates of the event points into polynomials of small degree. Second for tangential intersection points of two curves we locally introduce a new curve of small degree to the arrangement. The auxiliary curve enables us to determine nearly all event points of the arrangement by an extended box hit counting argument. Only a small fraction cannot be handled in this way. In these cases the factorization enables us to switch to a global argument. We have to decide whether a simple expression involving square roots of rational numbers is zero.

A prototype implementation of the main part of our algorithm including the computation of all relevant points of the two dimensional arrangements shows a good performance in practice. Our implementation uses the basic data types of LEDA [15] and the rational polynomial class as well as the resultant computation of MAPC [13].

The second approach (section 3) operates directly in 3-space and pursues a classical solid modeling technique in order to calculate the intersection cell of a set of quadrics. In contrast to many existing CAD-systems we keep track of all occurring rounding and approximation errors by using interval arithmetic (see [17]). In interval arithmetic each numerical value is represented by an interval of floating-point numbers, which enclose the true value. The use of interval arithmetic is greatly simplified thanks to the existence of an explicit parametric representation of the spatial intersection curves between two quadrics (see [14]). If the input does not lie too near to a degenerate configuration our algorithm will succeed in predicting the correct topological structure of the intersection, otherwise it can detect the existence of a critical situation and switch to the first approach.

In the last part of this paper (section 4) we show how to apply our results via duality to the convex hull problem of a set of ellipsoids, see also [9]. Finally we discuss further research.

2. AN EXACT APPROACH

From now on we use the notation $\tilde{q}$ for the zero set of a polynomial $q \in \mathbb{Q}[x_1, \ldots, x_n]$:

$$\tilde{q} = \{(x_1, \ldots, x_n) \mid q(x_1, \ldots, x_n) = 0\}.$$

A set of quadrics forms a 3-dimensional arrangement and partitions affine space in a natural way into four different types of maximal connected regions: *cells* are on either side of each quadric, *faces* lie on exactly one quadric, *edges* are on the intersection curve of two quadrics and *vertices* are intersection points of three or more quadrics or rather intersection points of at least two intersection curves.

We want to compute the topology of a cell in an arrangement of quadrics. Therefore we independently consider each quadric $\tilde{q}$ and calculate the sub-arrangement of all edges and vertices of the 3d arrangement that lie on the surface of $\tilde{q}$. Afterwards we decide which of these edges and vertices belong to the desired cell and join together all separate topological information to a unified description of the cell. It is easy to see that this last reunion step only involves some combinatorial and administrative effort. We omit the details here.

We want to compute the sub-arrangement on the surface of a quadric $\tilde{q}$ that is build by the intersection curves of $\tilde{q}$ with all other quadrics. In general there is no rational parameterization of the intersection curve of two quadrics in space. Therefore it is hard work on the surface of $\tilde{q}$ directly if we do not want to give up our aim to develop an exact and fast algorithm. But we can project the sub-arrangement into the (x, y)-plane and thereby reduce the problem to the question of calculating a planar arrangement of algebraic curves of degree up to 4. Our main contribution consists in an implementable, exact, and fast method to compute the planar arrangements that arise from this projection. We will describe our approach in details in the next three sections.

2.1 Computing planar arrangements – the ideas

The projection of the sub-arrangement on the surface of a quadric $\tilde{q}$ is realized by projecting the silhouette of $\tilde{q}$ and all its intersection curves with other quadrics into the (x, y)-plane. This yields a set of planar algebraic curves of degree at most 4. Figure 1 illustrates this: We can see an arrangement of three ellipsoids. There are three pairwise intersection curves. The pale and the dark one are intersection curves of the middle ellipsoid $\tilde{q}$ with the two other ellipsoids. Next the planar arrangement that appears as a result from the projection phase for $\tilde{q}$ is shown. The bordering ellipse is the projection of the silhouette of $\tilde{q}$. The other two curves are the projections of the pale and the dark spacecurve.

Next our task is to determine a topological description of the planar arrangement we get from the projection phase. Let $F = \{\tilde{f}_1, \ldots, \tilde{f}_k\}$ be the set of curves of this planar arrangement. We want to sweep parallel to the y-axis. What are the event points in the plane the sweep line has to stop at? It has to stop at intersections of two curves, at self-intersections, and at points with a vertical tangent because there a branch of the curve starts or ends. We call points with a vertical tangent *extreme*. For a curve $\tilde{f} \in F$ its extreme and self-intersection points are exactly the intersection points of $\tilde{f}$ and $\tilde{f}_y$. Thus each event point can be seen as an intersection point of two curves. Consequently

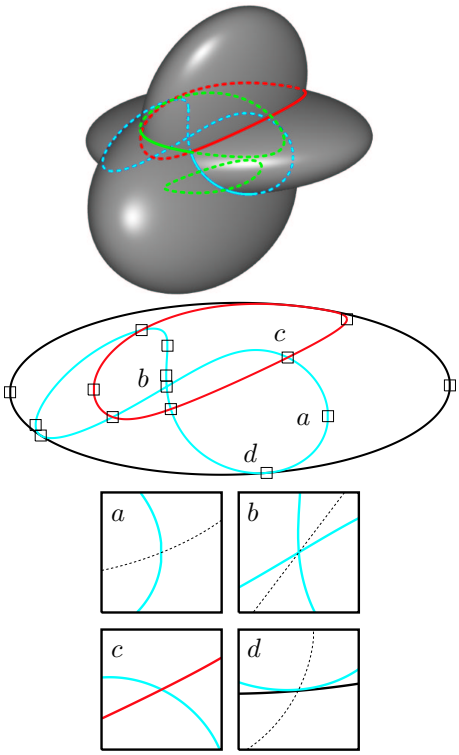

Figure 1: Planar arrangement of the middle ellipsoid computed by our algorithm

we have to locate the intersection points of $\tilde{f}_i$ and $\tilde{f}_j$ for all $1 \le i < j \le k$ and the intersection points of $\tilde{f}_i$ and $(\tilde{f}_i)_y$ for all $1 \le i \le k$.

The most serious problem we have to cope with is that in general the intersection points of two curves $\tilde{f}$ and $\tilde{g}$ have irrational x- and y-coordinates. That is why we cannot deal with them directly, instead we have to handle algebraic numbers. But we are able to determine small stripes with rational endpoints parallel to the y-axis with the following two properties: All real intersection points of $\tilde{f}$ and $\tilde{g}$ lie inside these stripes, and we can achieve that there is at most one intersection point per stripe. In this part of our algorithm we work over the complex numbers which causes some trouble. For example, there can be stripes with no real valued intersection point inside and we cannot detect this algebraically. We also compute stripes parallel to the x-axis and the intersection of the stripes yields boxes with rational corners the number of which is roughly quadratic in the number of real intersection points of $\tilde{f}$ and $\tilde{g}$. Each real intersection point lies inside such a box, but it is not clear which boxes contain such a point and which ones do not. This part of the algorithm is realizable using the algebraic tools resultants and root separation.

We have to identify the boxes that contain an intersection point of the two curves $\tilde{f}$ and $\tilde{g}$ in its interior. This turns out to be quite difficult because we only have at our disposal the discrete information of what happens on the boundary of the box. We know nothing about what happens inside. We can only determine the sequence of hits of $\tilde{f}$ and $\tilde{g}$ in order along the boundary of the box with the help of a root separation algorithm. This information may be sufficient. For example,

number of digits	5	10	15	20	25	30
running time in seconds	18	33	56	92	126	186

Figure 2: Running time for computing the planar arrangements of three quadrics.

if there are no hits or only hits from one curve then we know that there cannot be an intersection point inside. Or, if the two curves $\tilde{f}$ and $\tilde{g}$ intersect in order $\tilde{f}, \tilde{g}, \tilde{f}, \tilde{g}$ (see for example box number c in Figure 1) then we can be sure that an intersection must have taken place. But for a lot of event points this method called box hit counting fails. For example if we consider tangential intersection points or self-intersection points through which the partial derivation also cuts through, then the scenario on the boundary of the box is the same independently of whether there is an event point inside or not. In the next section (section 2.2) we introduce some mathematical tools and then we present our results how to overcome these difficulties (section 2.3).

After we have located all event points, we sweep the arrangement connecting the boxes, or rather event points, in the right way. This can easily be done and we will not go into further details.

We have implemented the algorithm described above up to and including the computation of all event points. The planar arrangement in Figure 1 appears during the run of our algorithm and is part of the visualization of our implementation. Here one can see that tangential intersections and self-intersections are not exceptional but intrinsic to the planar arrangements we obtain. We have measured the running time of our algorithm for three input quadrics on an Intel Pentium 700 varying the number of decimal digits of the coefficients of the input polynomials, see the table in Figure 2.

In the planar arrangement of Figure 1 one can see that our implementation has discovered all event points inside the boxes and has thrown away all empty boxes.

In the following we assume that all event points have different x- and y-coordinates. This is not a real restriction because on the one hand our algorithm can detect the violation of this assumption and on the other hand the coordinates depend on the direction of projection and with a "small number" of random affine transformations in space we can achieve that our assumption is fulfilled.

2.2 The main mathematical tools

In this section we briefly describe the main mathematical ingredients we use in our algorithm.

Resultants: One important concept is that of resultants and subresultants. For further information see for example [7] and [2]. Let

$$f(x_1, \ldots, x_n) = \alpha_l x_n^l + \alpha_{l-1} x_n^{l-1} + \cdots + \alpha_0, \quad \alpha_l \neq 0$$
$$g(x_1, \ldots, x_n) = \beta_m x_n^m + \beta_{m-1} x_n^{m-1} + \cdots + \beta_0, \quad \beta_m \neq 0,$$

be polynomials with coefficients $\alpha_i, \beta_i \in \mathbb{Q}[x_1, \ldots, x_{n-1}]$. The *resultant* $\mathrm{res}(f, g, x_n)$ of f and g with respect to x_n is a polynomial in $\mathbb{Q}[x_1, \ldots, x_{n-1}]$ defined by the α_i and β_i such that the following proposition holds: the vanishing of the resultant at a point $(c_1, \ldots, c_{n-1}) \in \mathbb{C}^{n-1}$ is a necessary condition for the extendibility of this point to a common root of f and g. Additionally the vanishing of the k-th *principal subresultant coefficients* $\mathrm{psc}^k(f, g, x_n) \in \mathbb{Q}[x_1, \ldots, x_{n-1}]$ for

$1 \leq k < j$ is a necessary condition that $f(c_1, \ldots, c_{n-1}, x_n) \in \mathbb{C}[x_n]$ and $g(c_1, \ldots, c_{n-1}, x_n) \in \mathbb{C}[x_n]$ have a common factor of degree j. Again the $\text{psc}^k(f, g, x_n)$ are defined by the α_i and β_i. If α_l and β_m both are constant and nonzero the conditions for extendibility are necessary as well as sufficient.

Multiplicity of intersection points: Let $f, g \in \mathbb{Q}[x, y]$. f and g define algebraic curves in the complex plane and common roots of the polynomials are intersections of the curves. An intersection point (a, b) yields $\text{res}(f, g, y)(a) = 0$. Remember our assumption that no two intersection points have common x- or y-coordinates. Then we state that the multiplicity of the intersection at (a, b) is the same as the multiplicity of the root a of $\text{res}(f, g, y)$. For example, if $\tilde{f}$ and $\tilde{g}$ intersect transversally at (a, b) then a is a root of the resultant of multiplicity 1. If they intersect tangentially the multiplicity of the root a of the resultant is at least 2. The same holds for $\text{res}(f, g, x)$ and b.

Multiplicity factorization: When we deal with the resultant of two bivariate polynomials f and g we can factor it according to the multiplicity of its roots by using multiple differentiation and division of polynomials. For example the polynomial

$$p(x) = x^9 - x^7 - x^5 + x^3 = (x - i)(x + i)(x + 1)^2(x - 1)^2x^3$$

can be factored into three polynomials $p_1(x) = x^2 + 1 = (x - i)(x + i)$, $p_2(x) = (x + 1)^2(x - 1)^2$, and $p_3(x) = x^3$.

Let $u \in \mathbb{Q}[x]$ and $v \in \mathbb{Q}[y]$ be the factors of $\text{res}(f, g, y)$ and $\text{res}(f, g, x)$ belonging to multiplicity i, respectively. Then all coordinates of intersection points of multiplicity i are roots of u and v.

Explicit solutions: Let again f and g be two bivariate polynomials. Let additionally $u \in \mathbb{Q}[x]$ and $v \in \mathbb{Q}[y]$ be factors of $\text{res}(f, g, y)$ or $\text{res}(f, g, x)$, respectively, each of degree at most 2. The roots of u and v define four points in the plane and we can give explicit terms for them involving only one square-root per coordinate. Substituting these terms into f and g reduces the question of whether $\tilde{f}$ and $\tilde{g}$ intersect at one of the points to the question whether two terms simultaneously become zero.

2.3 Computing planar arrangements - the details

The projection phase for a 3-dimensional quadric $\tilde{q}$ can be realized with the help of resultants. The projection of the silhouette of $\tilde{q}$ is the set of roots of $\text{res}(q, q_z, z)$. We call this planar curve *projectioncurve*. The projection of the spatial intersection curve of $\tilde{q}$ with another quadric $\tilde{p}$ is the set of roots of $\text{res}(q, p, z)$. Such a planar curve we name *cutcurve*. The algebraic degree of the projectioncurve is 2 and the degree of each cutcurve is ≤ 4.

Let F be the set of all algebraic curves of this planar arrangement. Now we have to locate as described before the intersection points of all pairs of algebraic curves $\tilde{f}$ and $\tilde{g}$, whereby either $\tilde{f}, \tilde{g} \in F$ or $\tilde{f} \in F$ and $\tilde{g} = \tilde{f}_y$. In general the intersection points of $\tilde{f}$ and $\tilde{g}$ will have irrational (even complex) coordinates. But again with the help of resultants we compute two univariate polynomials $X = \text{res}(f, g, y) \in \mathbb{Q}[x]$ and $Y = \text{res}(f, g, x) \in \mathbb{Q}[y]$ the roots of which contain the x- and y-coordinates of all intersection points, respectively. Thus the roots of X and Y together define a grid the intersection points of $\tilde{f}$ and $\tilde{g}$ lie on. For the same reason as before we cannot deal with the points on the grid directly but a root isolation algorithm can determine rational interval representations for the real algebraic roots of X and Y. The intervals on the x- and on the y-axis define disjoint boxes with rational corners. The real grid points are contained in these boxes. What remains to do is testing each box for a real intersection point of $\tilde{f}$ and $\tilde{g}$. In some cases this can easily be done using a method we call *simple box hit counting* in contrast to our extended method we will introduce soon:

Simple box hit counting argument: With the help of a root separation algorithm we determine the hits of $\tilde{f}$ and $\tilde{g}$ along the boundary of the box. We shrink the box until there are at most two hits from each curve. If there are exactly two hits from each curve we have to look whether they alternate or not. In the case that there are exactly two hits and they alternate the test is successful: $\tilde{f}$ and $\tilde{g}$ have a transversal intersection point inside the box.

One problem of this method is that it cannot detect tangential intersection points of $\tilde{f}$ and $\tilde{g}$. In most cases this difficulty can be solved by introducing a new curve to the box, see the dotted curve in Figure 1, picture d). To our knowledge we are the first who additionally consider an auxiliary curve in order to solve degeneracies:

Jacobi curve: Let f and g be two bivariate polynomials the planar curves of which intersect tangentially in the point (a, b). We define a third polynomial

$$h := f_x g_y - f_y g_x.$$

The set of roots of this polynomial h we call Jacobi curve.

With the help of the Jacobi curve we can detect most tangential intersection points.

THEOREM 1. *Let f and g be two bivariate polynomials the planar curves of which intersect tangentially in the point (a, b). Then either the Jacobi curve given by $h = f_x g_y - f_y g_x$ intersects $\tilde{f}$ as well as $\tilde{g}$ transversally in (a, b) or a and b are roots of multiplicity ≥ 3 of $\text{res}(f, g, y)$ and $\text{res}(f, g, x)$, respectively.*

Proof (The idea) $\tilde{f}$ and $\tilde{g}$ intersect tangentially in (a, b). This implies that a is a root of $\text{res}(f, g, y)$ of multiplicity at least 2. The same holds for $\text{res}(f, g, y)$ and b. Further (a, b) is a root of the Jacobi determinant $f_x g_y - f_y g_x$.

If either (f_x, f_y) or (g_x, g_y) become zero for (a, b) then (a, b) additionally is a self-intersection point of $\tilde{f}$ or $\tilde{g}$ and the theorem follows immediately from the multiplicity of intersection points.

Otherwise there exists $\lambda \in \mathbb{R}$, $\lambda \neq 0$, such that $f_x(a, b) = \lambda g_x(a, b)$ and $f_y(a, b) = \lambda g_y(a, b)$. If the curve of $h = f_x g_y - f_y g_x$ does not intersect $\tilde{f}$ and $\tilde{g}$ transversally in (a, b) then (a, b) is also a root of $h_x f_y - h_y f_x$. Substituting the definition of h into this term and making use of $f_x(a, b) = \lambda g_x(a, b)$ and $f_y(a, b) = \lambda g_y(a, b)$ leads after some calculation to the statement that in this case $\tilde{f}$ and $\tilde{g}$ have the same curvature in (a, b). Therefore the intersection has multiplicity at least 3 and we are done. □

This auxiliary curve leads to a new test for tangential intersection points of $\tilde{f}$ and $\tilde{g}$:

Extended box hit counting argument: We additionally consider the Jacobi curve h defined by f and g. We shrink the box until it contains at most one intersection

point between $\tilde{f}$, $\tilde{g}$, and $\tilde{h}$. This can again be done with the help of resultants and and a root separation computation. Now we apply simple box hit counting arguments to $\tilde{f}$ and $\tilde{h}$ and to $\tilde{g}$ and $\tilde{h}$. If both tests are successful the extended box hit counting argument is successful. $\tilde{f}$ and $\tilde{g}$ have a tangential intersection point inside the box.

Till now we are able to locate boxes that either contain a transversal intersection point of $\tilde{f}$ and $\tilde{g}$ or that contain a tangential intersection point the multiplicity of which is 2. What remains to do is locating self-intersection points and tangential intersection points of multiplicity > 2. We would like to do this using explicit solutions described in the previous section. So what we need is factorizing X and Y into products of polynomials of smaller degree. Before explaining the ideas how to factorize the polynomials we need some notation:

DEFINITION 1. *1. Let $u \in \mathbb{Q}[x]$ and $v \in \mathbb{Q}[y]$. By $\mathcal{R}(u)$ we denote the set of real roots of u. By $\#(u, v)$ we mean the grid $\mathcal{R}(u) \times \mathcal{R}(v)$.*

2. Let $f, g \in \mathbb{Q}[x, y]$, $X = res(f, g, y)$ and $Y = res(f, g, x)$. We call the pair (X, Y) the bi-resultant of f and g.

3. Let $u_1, u_2 \in \mathbb{Q}[x]$ and $v_1, v_2 \in \mathbb{Q}[y]$. The expression $(u_1, v_1) \cdot (u_2, v_2)$ is called bi-factorization of the bi-resultant (X, Y) iff $X = u_1 \cdot u_2$, $Y = v_1 \cdot v_2$, and all intersection points of $\tilde{f}$ and $\tilde{g}$ lie on $\#(u_1, v_1) \cup \#(u_2, v_2)$. The pairs (u_1, v_1) and (u_2, v_2) are called bi-factors.

For example, a factorization $X = \text{res}(f, g, y) = u_1 u_2$ and $Y = \text{res}(f, g, x) = v_1 v_2$ according to multiplicities 1 and ≥ 2 yields two bi-factors (u_1, v_1) and (u_2, v_2) of (X, Y). As before we can compute boxes around the grid points of $\#(u_1, v_1)$ for the bi-factor (u_1, v_1) of (X, Y). All transversal intersection points of $\tilde{f}$ and $\tilde{g}$ are then contained in these boxes.

Just multiplicity factorization is not enough. Let us consider a cutcurve $\tilde{f}$. Its extreme and self-intersection points are exactly the intersection points of $\tilde{f}$ and $\tilde{g} = \tilde{f}_y$. Let (X, Y) be the bi-resultant of $\tilde{f}$ and $\tilde{g}$. The coordinates of extreme points of $\tilde{f}$ are roots of multiplicity 1 of X and Y and the coordinates of self-intersection points are roots of multiplicity ≥ 2. Thus a multiplicity bi-factorization gives us a bi-factor (u, v) describing all self-intersections. But it is possible that a cutcurve has more than 2 self-intersections. That is why the degree of u and v can be more than quadratic and this prevents the use of explicit solutions. We are able to distinguish two different kinds of self-intersections, both of which are describable by quadratic polynomials. *Top-bottom* intersections are a result from the projection phase because points on the top and on the bottom of $\tilde{q}$ are projected on top of each other. *Genuine* intersections are self-intersections that already existed in space. They appear when two quadrics additionally have the same tangential plane at an intersection point.

THEOREM 2. *Either a cutcurve has at most two top-bottom intersection points or all points of the curve are of this kind. In the first case we can compute a line that passes through these points.*

Proof Let $\tilde{q}$ and $\tilde{p}$ be the quadrics that caused the cutcurve. Every top-bottom point (a, b) is also a root of $\text{psc}^1(q, p, z)$. An easy calculation shows that $\text{psc}^1(q, p, z)$ is a polynomial of degree at most one. If $\text{psc}^1(q, p, z)$ is the zero polynomial every point of the cutcurve is top-bottom. Otherwise $\text{psc}^1(q, p, z)$ either is constant and nonzero in which case there is no top-bottom point, or it defines a line. A line intersects an algebraic curve of degree 4 at most four times. Since it cuts through the self-intersection points each intersection has multiplicity 2. □

THEOREM 3. *Either a cutcurve has at most two genuine self-intersection points or all points of the curve are of this kind.*

Proof Assume that there are three different tangential intersection points $a, b, c \in \mathbb{C}^3$ of two quadrics $\tilde{q}$ and $\tilde{p}$. One can construct a complex hyperplane through these points. The hyperplane intersects $\tilde{q}$ and $\tilde{p}$ in space-curves q_1 and p_1. From the point of view of the hyperplane q_1 and p_1 are conics which have the same tangent in their common points a, b, and c. Because of that $q_1 = p_1$ must hold. In space all tangential planes to the points on q_1 intersect in one point, namely the apex of the cone the tangential planes define. This point is uniquely defined by the three tangential planes in a, b, and c. Therefore the surfaces of q and p have the same tangential plane for each point on q_1. A brief investigation on the degree of a cutcurve yields the desired result that in this case all points on the cutcurve are genuine. □

For ease of exposition we assume that each cutcurve only has finitely many self-intersections.

Now we are ready to prove our main theorem that we are able to locate every event point in our arrangement.

THEOREM 4. *Let F be the set of curves of the planar arrangement we obtain from the projection phase. Let $\tilde{f}, \tilde{g} \in F$ or $\tilde{f} \in F$ and $\tilde{g} = \tilde{f}_y$. All intersection points of $\tilde{f}$ and $\tilde{g}$ can be determined either by locally applying simple or extended box hit counting arguments or by factorizing the bi-resultant of $\tilde{f}$ and $\tilde{g}$ and computing explicit solutions.*

Proof Remember our assumption that all event points have different x- and y-coordinates. Let in the following $X := \text{res}(f, g, y)$ and $Y := \text{res}(f, g, x)$.

If $\tilde{f}$ is the projectioncurve and $g = f_y$ then we apply a simple box hit counting argument to every box.

The case that $\tilde{f}$ and $\tilde{g}$ both are projectioncurves has not to be considered, because there is exactly one projectioncurve in every arrangement.

Let $\tilde{f}$ be a cutcurve and $g = f_y$. The resultants X and Y have degree at most 12. We compute a multiplicity bi-factorization $(u_1, v_1) \cdot (u_2, v_2)$ of (X, Y) such that all intersection points with muliplicity 1, i.e. all extreme points, lie on $\#(u_1, v_1)$ and all intersection points with multiplicity ≥ 2, i.e. all self-intersection points, lie on $\#(u_2, v_2)$. The boxes around $\#(u_1, v_1)$ can be handled with simple box hit counting arguments. The second bi-factor (u_2, v_2) can be factored again into bi-factors (u_3, v_3) and (u_4, v_4) of (X, Y) according to the top-bottom and genuine distinction of self-intersections. This can be done with the help of $\text{psc}^1(f, g, y)$ according to theorem 2, see also the dotted line in Figure 1 picture b). The details are omitted. From theorem 2 and

3 it follows that u_3, u_4, v_3 and v_4 are at most quadratic polynomials and we can compute explicit solutions.

Let $\tilde{f}$ be the projectioncurve and $\tilde{g}$ a cutcurve. It is clear that both curves can have at most tangential intersection points. That is why each root of X and Y either has degree 2 or ≥ 4. X and Y have degree at most 8. This yields that they can have at most two roots of multiplicity 4. We compute a bi-factorization $(u_1, v_1) \cdot (u_2, v_2)$ of (X, Y). All intersections points with multiplicity 2 lie on $\#(u_1, v_1)$ and the boxes around $\#(u_1, v_1)$ can be handled with extended box hit counting arguments. The bi-polynomial (u_2, v_2) consists of two at most quadratic polynomials for which we can determine explicit solutions.

The last case is that $\tilde{f}$ and $\tilde{g}$ both are cutcurves. They are the result of the intersection of $\tilde{q}$ with two other quadrics $\tilde{p_1}$ and $\tilde{p_2}$. We distinguish *spatial* and *artificial* intersection points. Spatial intersections are projections of common intersection points of $\tilde{q}$, $\tilde{p_1}$ and $\tilde{p_2}$. Artificial intersections take place because one intersection curve runs on the top of $\tilde{q}$ and the other on the bottom of $\tilde{q}$ and both space curves are projected on top of each other causing an intersection point. The resultants X and Y have degree at most 16. We compute a bi-factorization $(u_1, v_1) \cdot (u_2, v_2)$ of (X, Y) according to the distinction of spatial and artificial intersection points. This bi-factorization can be done by additionally computing the resultant $k = \text{res}(p_1, p_2, z)$ and a greatest common divisor computation between X, $\text{res}(f, k, y)$, and $\text{res}(g, k, y)$ and between Y, $\text{res}(f, k, x)$, and $\text{res}(g, k, x)$. An alternative way would be to use the results of [4] and the method described in the proof of the next theorem. In the next theorem we additionally show that u_1, u_2, v_1 and v_2 all have degree ≤ 8. Then we refine the bi-factorization according to the multiplicities of the intersection points. The boxes belonging to multiplicity 1 or 2 are handled by simple or extended box hit counting arguments, respectively. Both polynomials of the remaining bi-polynomials are at most quadratic and we compute explicit solutions. □

THEOREM 5. *There are at most 8 spatial and at most 8 artificial intersections.*

Proof (The idea) The first part follows by the theorem of Bezout. For the second part we mirror $\tilde{p_1}$ parallel to the z-axis at the plane which separates the top of $\tilde{q}$ from the bottom of $\tilde{q}$. Let $\tilde{p_3}$ be the new surface. We can show that $\tilde{p_3}$ is again a quadric. This transformation has the effect that all intersection points of $\tilde{q}$ and $\tilde{p_3}$ have the same (x, y)-coordinates as the intersection points of $\tilde{q}$ and $\tilde{p_1}$, i.e. $\text{res}(q, p_1, z) = \text{res}(q, p_3, z)$. But a root (a, b) of $\text{res}(q, p_1, z) = \text{res}(q, p_3, z)$ that corresponds to an intersection point of $\tilde{q}$ and $\tilde{p_1}$ lying on the top of $\tilde{q}$ now corresponds to an intersection point of $\tilde{q}$ and $\tilde{p_3}$ lying on the bottom of $\tilde{q}$ and vice versa. In this way the spatial and artificial intersections have changed place and we can again apply the theorem of Bezout to q, p_2 and p_3. □

3. A SOLID MODELING APPROACH

In this section we briefly describe our approach to calculate the intersection of a set of oriented quadric surfaces assuming that no degenerate situation exists. We use an incremental algorithm to compute the boundary representation of the intersection body. This body may be composed of disjoint lumps each having several shells. The whole boundary consists of faces, edges and vertices which are embedded in quadric surfaces, intersection curves and intersection points, respectively. First of all we have to point out how to adequately represent the intersection curves (section 3.1) and their intersection points (section 3.2). Then we explain the data structure (section 3.3) and the necessary topological operations to intersect a new quadric with an intermediate intersection body (section 3.4). Last but not least we deal with the detection of all situations, in which our algorithm may fail because of the rounding and approximation errors (section 3.5).

3.1 Parameterization of the intersection curve between two quadrics

Let A and B be two quadric surfaces. Suppose A is a ruled quadric, i.e. a plane, a cylinder, a cone, a hyperboloid of one sheet or a hyperbolic paraboloid. In this case A can be generated by a family of lines, which has a simple parametric representation. In order to obtain a parametric representation of the intersection curve $A \cap B$, we can simply intersect each line of this family with B. This only requires the solution of a quadratic equation.

The key observation to handle the case when A is an arbitrary quadric, is to reduce it to the special case above. This can be achieved by looking for a ruled quadric in the pencil of quadrics $A + \lambda B$ ($\lambda \in \mathbb{R}$) and intersecting this ruled quadric instead of A with B. This substitution is feasible, since any two quadrics in the pencil have the same intersection curve in common.

A theorem of Levin [14] guarantees that a ruled surface can always be found in the pencil of quadrics:

THEOREM 1 (LEVIN). *The intersection of two quadric surfaces lies in a ruled quadric.*

In appendix B we review and simplify the proof of this theorem for the most general case when the ruled quadric in the pencil is a hyperbolic paraboloid. In this case the intersection curve $\mathbf{s}(t)$ can be parameterized as follows

$$\mathbf{s}(t) = \mathbf{U} \cdot [t\, b(t), a(t) \pm \sqrt{c(t)}, t\,(a(t) \pm \sqrt{c(t)}), b(t)]^T$$

where $\mathbf{s}(t) \in \mathbb{R}^4$ and $\mathbf{U} \in \mathbb{R}^{4\times 4}$. $a(t)$ and $b(t)$ are polynomials of degree 2 and $c(t)$ is a polynomial of degree 4. Unfortunately the coefficients of these polynomials are irrational numbers (roots of polynomials of degree 3).

3.2 Intersection of three quadrics

Let A, B and C denote the three quadrics we want to intersect. We are interested in all common points on their boundary. First of all we determine the intersection curve $\mathbf{s}(t)$ between A and B and put this formula into $\mathbf{s}(t)^T \mathbf{C} \mathbf{s}(t) = 0$. In this way we obtain an equation of the following form:

$$f(t)\, c(t) \pm g(t)\sqrt{c(t)} + h(t) = 0$$

where $f(t)$, $g(t)$ and $h(t)$ are polynomials in t of degree 2, 4 and 6, respectively. We eliminate the square root by squaring and get a polynomial of degree 12. The real roots of this polynomial provide all parameter values at which a triple intersection point can occur.

$$f^2(t)c^2(t) + (2f(t)h(t) - g^2(t))c(t) + h^2(t) = 0$$

3.3 The data structure

We consider all input quadrics as solid bodies. The result of a sequence of intersection operations on these bodies can again be regarded as a solid body. We represent such a solid body by describing its boundary, which may consist of several connected components. The topological entities of the body are its lumps, shells, faces, loops, edges and vertices. The geometric entities are the quadric surfaces, the intersection curves between two surfaces and the common intersection points of three or more surfaces. The two-dimensional boundary of the body consists of a set of faces. Each face is embedded in a quadric surface. The one-dimensional boundary of a face is characterized by its loops. Each loop is composed of a set of oriented edges. The geometric information associated with an edge is given by the space curve the edge lies on. These space curves arise when two quadrics intersect. The zero-dimensional boundary of an edge consists of two vertices whose Cartesian coordinates correspond to multiple intersection points between the input quadrics.

In order to facilitate an efficient handling of the entities of the boundary representation we store some additional information about the neighborhood of the topological entities. It is sufficient to arrange the lumps of the body, the shells of a lump, the faces of a shell, and the loops of a face in simply linked lists. Every entity also has a pointer to its superior entity. A loop is represented as a doubly linked list of its oriented edges and it knows about the face it belongs to. In a two-manifold each edge is adjacent to exactly two faces. We store edges as two opposite oriented edges with a mutual reference. In addition each edge has a pointer to the unique loop it occurs in and pointers to its two vertices.

3.4 Topological operations

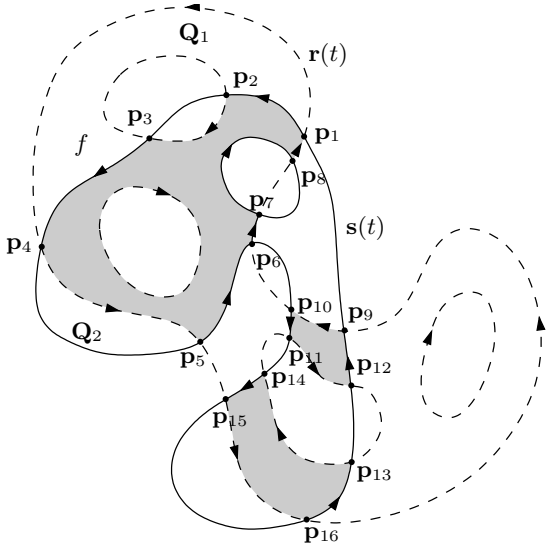

Figure 3: The intersection of a face f embedded in Q_1 with the second quadric Q_2.

We want to intersect a face f, which is embedded in a quadric Q_1, with a second quadric Q_2. See Figure 3! The boundary of face f may consist of several loops (broken lines). For the sake of lucidity the different edges and vertices of the loops are not shown. The orientation of the edges within a loop is chosen such that the interior of f always lies to the left when moving in forward direction of the edges and looking opposite to the surface normal. This means that the vector $\mathbf{n}(\mathbf{p}) \times \frac{d\,\mathbf{r}}{d\,t}(\mathbf{p})$ points into the interior of f for each point $\mathbf{p} = \mathbf{r}(t)$ of the edge. Here $\mathbf{n}(\mathbf{p})$ is the normal vector of Q_1 and $\frac{d\,\mathbf{r}}{d\,t}(\mathbf{p})$ is the tangential vector of $\mathbf{r}(t)$ at the point $\mathbf{p}$.

The first step in the computation of $f \cap Q_2$ is the determination of the parametric intersection curve $\mathbf{s}(t)$ (solid lines) between Q_1 and Q_2. This curve may consist of one or two loops. Next we calculate all intersection points (in our example $p_1, \ldots, p_{16}$) between $\mathbf{s}(t)$ and all edges of the loops of f. This calculation has not only to provide the Cartesian coordinates of each intersection point but also its parameter values with respect to $\mathbf{s}(t)$ and with respect to the curves $\mathbf{r}(t)$ of the intersected edges. These parameter values enable us to sort the intersection points along the curves they belong to.

In the next step we identify the new loops of $f \cap Q_2$. We start at an arbitrary intersection point $\mathbf{p}$ and follow the leftmost curve until we arrive at the next intersection point. We continue in this fashion until we return to our starting point. The decision which curve to follow at an intersection point $\mathbf{p}$ can be taken by looking at the surface normal and at the tangent vectors of the intersecting oriented curves. Simply calculate the sign of $\det(\mathbf{n}(\mathbf{p}), \frac{d\,\mathbf{r}}{d\,t}(\mathbf{p}), \frac{d\,\mathbf{s}}{d\,t}(\mathbf{p}))$. This procedure produces an ordered sequence of edges of a new loop alternating between the original boundary curves of f and the newly created intersection curve. As Figure 3 illustrates more than one loop may arise, so that $f \cap Q_2$ decomposes into several connected components (the gray shaded regions).

Up to now we have ignored all the loops of f and the intersection curve, which do not contain an intersection point. We still have to determine the nesting of all these loops in order to form the final new faces of $f \cap Q_2$. The question whether a loop l_1 lies within another loop l_2 can be answered by following a path $\mathbf{w}(t)$ on the surface Q_1 from an arbitrary point of l_1 to the first point $\mathbf{p}$ where this path hits l_2. The simplest choice for $\mathbf{w}(t)$ is a conic. Again the sign of $\det(\mathbf{n}(\mathbf{p}), \frac{d\,\mathbf{r}}{d\,t}(\mathbf{p}), \frac{d\,\mathbf{w}}{d\,t}(\mathbf{p}))$ tells us, whether l_1 lies inside or outside of l_2.

The operations for calculating $f \cap Q_2$ have to be performed for every face of the body. As a result we obtain all new faces except those faces which lie on the surface of Q_2. The boundary of these faces automatically ensues from sewing up all edges with their opposite directed counterparts. The connected components of the face set form the shells and the nesting of these shells yields the lumps of the intersection body. The nesting of the shells can be determined by ray-shooting.

3.5 Verified arithmetic

During the whole process of intersecting quadric surfaces we have to perform critical arithmetic operations over and over again: We have to avoid divisions by zero, determine the correct sign of expressions and reliably predict the exact number of real roots of a univariate polynomial in a given interval. If we simply used floating-point arithmetic, we would be confronted with rounding and approximation errors. That is the reason why we use interval arithmetic (see [17]) as a technique to self-validate all numerical operations. Each quantity such as the value of an expression or the root of a polynomial is represented by an interval of

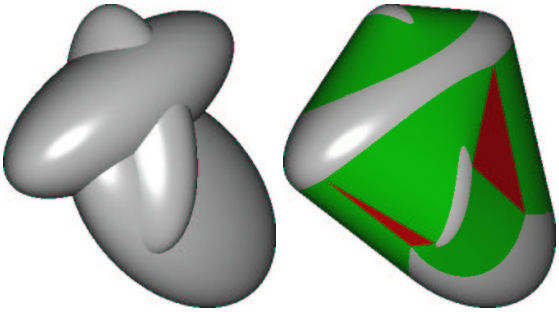

Figure 4: The convex hull of three ellipsoids.

floating-point numbers. This interval is guaranteed to contain the true value of the quantity it represents.

In this way we can detect all (near-) degenerate geometric situations, which we can not handle by the approach discussed in this section. Whenever ambiguous results occur, we break off the complete procedure and switch to the algorithm described in the section 2.

4. APPLICATIONS AND FURTHER RESEARCH

Efficient and robust algorithms for the calculation of the convex hull of a set of points or a set of spheres are well known ([1],[3]). Our algorithm to compute the cell in an arrangement of quadrics can via duality be used to compute the convex hull of a set of ellipsoids. For further information about the dualization step see the appendix A.

Figure 4 illustrates the topological structure of the convex hull. A point in the dual space dualizes to a plane in the primal space. A vertex dualizes to a plane that touches each of the three involved ellipsoids in exactly one point. The three points define a triangle and the area lying on the tangential plane bounded by the triangle will be part of the convex hull. Analogously, each point on an edge of the intersection cell defines a line connecting the two involved ellipsoids. All lines together form a developable surface. Thus the convex hull consists of three different types of surfaces: parts of the original quadric surfaces, triangles which arise from the dual vertices, and developable surfaces which arise from the dual edges.

The problem of computing the convex hull of quadric surface patches as discussed in [12] can also be solved exactly. In accordance with [12] the boundary of the patches has to consist of segments of conic curves such that the dual set only comprises parts of quadric surfaces and conic curves.

Our approach to compute the intersection or the convex hull of quadrics provides the basic techniques for implementing a robust constructive solid geometry (CSG) system, which could not only handle polyhedral solids but also solids with second order surfaces. In a CSG system complex objects can be produced from simple primitives by combining multiple levels of CSG-operators. In addition to intersection also union and difference come into question as operators. In our case the primitive solids could be arbitrary quadrics with rational coefficients.

5. ACKNOWLEDGMENTS

We would like to thank Raimund Seidel for many useful discussions and his help in preparing this paper.

6. REFERENCES

[1] J.-D. Boissonnat, A. Cérézo, O. Devillers, J. Duquesne, and M. Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension *d*. *Comput. Geom. Theory Appl.*, 6:123–130, 1996.

[2] W. Brown and J. F. Traub. On Euclid's algorithm and the theory of subresultants. *Journal of the ACM*, 18:505–514, 1971.

[3] http://www.cgal.org.

[4] E. Chionh, R. Goldman, and J. Miller. Using multivariate resultants to find the intersection of three quadric surfaces. *Transactions on Graphics*, 10:378–400, 1991.

[5] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conf. on Automata Theory and Formal Languages*, volume 6, pages 134–183. Lecture Notes in Computer Science, Springer, Berlin, 1975.

[6] G. E. Collins and R. Loos. Real zeros of polynomials. In B. Buchberger, G. E. Collins, and R. Loos, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 83–94. Springer-Verlag, New York, NY, 1982.

[7] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer, New York, 1997.

[8] Farouki, Neff, and O'Connor. Automatic parsing of degenerate quadric-surface intersections. *ACM Trans. Graph.*, 8:174–203, 1989.

[9] N. Geismann, M. Hemmer, and E. Schömer. The convex hull of ellipsoids. *SOCG video track*, 2001.

[10] D. Halperin. Arrangements. In *Handbook of Discrete and Computational Geometry*, pages 389–412, 1997.

[11] C. Hoffmann. *Geometric and Solid Modeling*. Morgan-Kaufmann, San Mateo, CA, 1989.

[12] C.-K. Hung and D. Ierardi. Constructing convex hulls of quadratic surface patches. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 255–260, 1995.

[13] J. Keyser, T. Culver, D. Manocha, and S. Krishnan. MAPC: A library for efficient and exact manipulation of algebraic points and curves. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 1999.

[14] J. Levin. A parametric algorithm for drawing pictures of solid objects composed of quadric surfaces. *Commun. ACM*, 19(10):555–563, Oct. 1976.

[15] K. Mehlhorn and S. Näher. *LEDA – A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[16] J. R. Miller. Geometric approaches to nonplanar quadric surface intersection curves. *ACM Trans. Graph.*, 6:274–307, 1987.

[17] R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966.

[18] A. Rege. *A Toolkit for Algebra and Geometry*. Univ. of California at Berkely, Berkely, California, 1996. Ph.D. dissertation.

[19] T. Sakkalis. The topological configuration of a real algebraic curve. *Bulletin of the Australian Mathematical Society*, 43:37–50, 1991.

APPENDIX

A. DUALIZATION OF QUADRICS

LEMMA 1. *Let Q be a quadric in homogeneous coordinates:*

$$\mathbf{x}^T\mathbf{Q}\mathbf{x} = 0 \quad \textit{with} \quad \mathbf{Q} = \mathbf{Q}^T \in \mathbb{R}^{4\times 4} \quad \textit{and} \quad \mathbf{x} \in \mathbb{R}^4$$

The set of points which are dual to the set of the tangential planes of Q lie on the quadric Q^{-1}, i.e. they fulfill the equation $\mathbf{x}^T\mathbf{Q}^{-1}\mathbf{x} = 0$.

Proof The set of the tangential planes of Q is given by

$$\{\mathbf{x}_0^T\mathbf{Q}\mathbf{x} = 0 \,|\, \mathbf{x}_0^T\mathbf{Q}\mathbf{x}_0 = 0\}.$$

The set of the dual points can thus be characterized as

$$\{\mathbf{Q}\mathbf{x}_0 \,|\, \mathbf{x}_0^T\mathbf{Q}\mathbf{x}_0 = 0\}.$$

All these points lie on the quadric Q^{-1} because

$$\mathbf{x}_0^T\mathbf{Q}\mathbf{Q}^{-1}\mathbf{Q}\mathbf{x}_0 = \mathbf{x}_0^T\mathbf{Q}\mathbf{x}_0 = 0.$$

□

Let Q be an ellipsoid with center $\mathbf{c}$ and the equation $(\mathbf{x}-\mathbf{c})^T\mathbf{M}(\mathbf{x}-\mathbf{c}) = 1$ where $\mathbf{x}, \mathbf{c} \in \mathbb{R}^3$ and $\mathbf{M} = \mathbf{M}^T \in \mathbb{R}^{3\times 3}$. It is easy to verify that the corresponding homogeneous matrices $\mathbf{Q}$ and $\mathbf{Q}^{-1}$ are given by

$$\mathbf{Q} = \begin{bmatrix} \mathbf{M} & -\mathbf{M}\mathbf{c} \\ -\mathbf{c}^T\mathbf{M} & \mathbf{c}^T\mathbf{M}\mathbf{c} - 1 \end{bmatrix}, \mathbf{Q}^{-1} = \begin{bmatrix} \mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T & -\mathbf{c} \\ -\mathbf{c}^T & -1 \end{bmatrix}.$$

The form of Q^{-1} depends on the sign of $\mathbf{c}^T\mathbf{M}\mathbf{c} - 1$. We discuss three different cases:

case 1: $\mathbf{c}^T\mathbf{M}\mathbf{c} - 1 \neq 0$

The translation $\mathbf{y} = \mathbf{x} + \dfrac{\mathbf{M}\mathbf{c}}{\mathbf{c}^T\mathbf{M}\mathbf{c} - 1}$ transforms Q^{-1} to the central form:

$$\mathbf{y}^T(\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T)\mathbf{y} + \frac{1}{\mathbf{c}^T\mathbf{M}\mathbf{c} - 1} = 0$$

case 1.1: $\mathbf{c}^T\mathbf{M}\mathbf{c} - 1 < 0$

Q^{-1} is an ellipsoid, since $\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T$ is positive definite. We can show this by examining the inverse matrix

$$(\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T)^{-1} = \mathbf{M} - \frac{\mathbf{M}\mathbf{c}\mathbf{c}^T\mathbf{M}}{\mathbf{c}^T\mathbf{M}\mathbf{c} - 1}$$

For all $x \in \mathbb{R}^3 \setminus \{\mathbf{0}\}$ it holds:

$$\mathbf{x}^T(\mathbf{M} - \frac{\mathbf{M}\mathbf{c}\mathbf{c}^T\mathbf{M}}{\mathbf{c}^T\mathbf{M}\mathbf{c} - 1})\mathbf{x} = \mathbf{x}^T\mathbf{M}\mathbf{x} - \frac{(\mathbf{c}^T\mathbf{M}\mathbf{x})^2}{\mathbf{c}^T\mathbf{M}\mathbf{c} - 1} > 0.$$

case 1.2: $\mathbf{c}^T\mathbf{M}\mathbf{c} - 1 > 0$

Q^{-1} is a two-sheet hyperboloid, since

$$\det(\mathbf{Q})^{-1} = \det(\mathbf{Q}^{-1}) = \frac{1}{\mathbf{c}^T\mathbf{M}\mathbf{c} - 1}\det(\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T) < 0.$$

As a consequence $\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T$ has exactly one negative eigenvalue. (Three negative eigenvalues can not occur, because that would be an imaginary surface.)

case 2: $\mathbf{c}^T\mathbf{M}\mathbf{c} - 1 = 0$

Q^{-1} is an elliptic paraboloid, since $\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T$ is singular because

$$(\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T)\mathbf{x} = \mathbf{0} \quad \text{for} \quad \mathbf{x} = \mathbf{M}\mathbf{c}.$$

Apart from the eigenvalue 0 there exist two positive eigenvalues, i.e. $\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T$ is positive semidefinite. Since $\mathbf{M}$ is positive definite, it suffices to show that $\mathbf{M}(\mathbf{M}^{-1} - \mathbf{c}\mathbf{c}^T)\mathbf{M} = \mathbf{M} - \mathbf{M}\mathbf{c}\mathbf{c}^T\mathbf{M}$ is positive semidefinite. It holds:

$$\begin{aligned} \mathbf{x}^T(\mathbf{M} - \mathbf{M}\mathbf{c}\mathbf{c}^T\mathbf{M})\mathbf{x} &= \mathbf{c}^T\mathbf{M}\mathbf{c} \cdot \mathbf{x}^T\mathbf{M}\mathbf{x} - \mathbf{x}^T\mathbf{M}\mathbf{c} \cdot \mathbf{c}^T\mathbf{M}\mathbf{x} \\ &= (\mathbf{c} \times \mathbf{x})^T(\mathbf{M}\mathbf{c} \times \mathbf{M}\mathbf{x}) \\ &= \det(\mathbf{M})(\mathbf{c} \times \mathbf{x})^T\mathbf{M}^{-1}(\mathbf{c} \times \mathbf{x}) \geq 0 \end{aligned}$$

B. PROOF OF LEVIN'S THEOREM

Proof Let $\mathbf{A}$ and $\mathbf{B} \in \mathbb{R}^{4\times 4}$ denote the homogeneous, symmetric matrices of the quadrics A and B. We look for a matrix $\mathbf{P}(\lambda) = \mathbf{A} + \lambda\mathbf{B}$ with $\det(\overline{\mathbf{P}}(\lambda)) = 0$, where $\overline{\mathbf{P}}(\lambda)$ denotes the upper left 3×3 -submatrix of $\mathbf{P}(\lambda)$. Such a λ exists, because $\det(\overline{\mathbf{P}}(\lambda))$ is a polynomial of degree 3 in λ, and thus has at least one real zero λ_1.

Let $\mathbf{P}_1 = \mathbf{A} + \lambda_1\mathbf{B}$. We perform a principal axes transformation for $\overline{\mathbf{P}}_1$:

$$\begin{aligned} \overline{\mathbf{P}}_2 &= \operatorname{diag}(\mu_1, \mu_2, 0) = \overline{\mathbf{V}}_1^T\,\overline{\mathbf{P}}_1\,\overline{\mathbf{V}}_1 \\ \mathbf{P}_2 &= \mathbf{V}_1^T\,\mathbf{P}_1\,\mathbf{V}_1 \quad \text{with} \quad \mathbf{V}_1 = \operatorname{diag}(\overline{\mathbf{V}}_1, 1) \end{aligned}$$

The columns of $\overline{\mathbf{V}}_1$ contain the eigenvectors for the eigenvalues μ_1, μ_2 and 0. The eigenvalues can be determined via the characteristic polynomial $\chi_{\overline{\mathbf{P}}_1}(\mu) = \det(\overline{\mathbf{P}}_1 - \mu\mathbf{I})$. For 3×3-matrices $\chi_{\overline{\mathbf{P}}_1}(\mu)$ can be expressed as

$$-\mu^3 + \operatorname{trace}(\overline{\mathbf{P}}_1)\mu^2 - \operatorname{trace}(\operatorname{adj}(\overline{\mathbf{P}}_1))\mu + \det(\overline{\mathbf{P}}_1) \qquad (1)$$

Since $\det(\overline{\mathbf{P}}_1) = 0$, the eigenvalues μ_1 and μ_2 can be calculated as the roots of a quadratic equation. Now we distinguish between two cases on the basis of the signs of the eigenvalues:

case 1: $\mu_1 \cdot \mu_2 < 0$

This is the simpler case. If there is no degeneration, we have already found a hyperbolic paraboloid in the pencil of quadrics. We only have to transform it into its normal form. The necessary transformations will be given after the discussion of the second case.

case 2: $\mu_1 \cdot \mu_2 > 0$

In this case we generally have to deal with an elliptic paraboloid. We further transform matrix $\mathbf{P}_2$ and combine it with one of the input quadrics in order to find a suitable quadric like in the first case.

If $\mu_1 < 0$ and $\mu_2 < 0$, we substitute $\mathbf{P}_2$ by $-\mathbf{P}_2$.

The transformation

$$\mathbf{P}_3 = \mathbf{V}_2^T\,\mathbf{P}_2\,\mathbf{V}_2 \quad \text{with} \quad \mathbf{V}_2 = \operatorname{diag}(\frac{1}{\sqrt{|\mu_1|}}, \frac{1}{\sqrt{|\mu_2|}}, 1, 1)$$

guarantees, that $\overline{\mathbf{P}}_3 = \operatorname{diag}(1, 1, 0)$ holds.

Let $\mathbf{Q} = \mathbf{V}_2^T\mathbf{V}_1^T\mathbf{B}\mathbf{V}_1\mathbf{V}_2$. We transform $\mathbf{P}_3$ and $\mathbf{Q}$ with the help of the orthonormal matrix

$$\mathbf{V}_3 = \operatorname{diag}\left(\frac{1}{\sqrt{Q_{13}^2 + Q_{23}^2}}\begin{bmatrix} Q_{23} & Q_{13} \\ -Q_{13} & Q_{23} \end{bmatrix}, 1, 1\right)$$

This transformation leaves the upper left 3×3-matrix of $\mathbf{P}_4 = \mathbf{V}_3^T\,\mathbf{P}_3\,\mathbf{V}_3$ unchanged, and the upper left 3×3-matrix of $\mathbf{R} = \mathbf{V}_3^T\,\mathbf{Q}\,\mathbf{V}_3$ becomes:

$$\overline{\mathbf{R}} = \begin{bmatrix} R_{11} & R_{12} & 0 \\ R_{12} & R_{22} & R_{23} \\ 0 & R_{23} & R_{33} \end{bmatrix}$$

Provided that $R_{33} \neq 0$, we set $\mathbf{S} = \frac{1}{R_{33}}\mathbf{R}$ and choose α such that $\det(\overline{\mathbf{S}} - \alpha\,\overline{\mathbf{P}}_4) = 0$. α can be found as the root of a quadratic equation. Let $\beta = S_{22} - S_{23}^2$, $\gamma = \sqrt{(S_{11}-\beta)^2 + 4S_{12}^2}$ and $\alpha = \frac{1}{2}(S_{11} + \beta \pm \gamma)$. We now show that $\mathbf{P}_5 = \mathbf{S} - \alpha\,\mathbf{P}_4$ is a hyperbolic paraboloid unless there is a degenerate situation:

Since $\det(\overline{\mathbf{P}}_5) = 0$ holds, 0 is an eigenvalue of $\overline{\mathbf{P}}_5$. In order to proof that there still exist a negative as well as a positive eigenvalue, we show according to (1) that $\mathrm{trace}(\mathrm{adj}(\overline{\mathbf{P}}_5)) < 0$ is true. Using the fact that $\det(\overline{\mathbf{P}}_5) = 0$, it is easy to verify that

$$\mathrm{trace}(\mathrm{adj}(\overline{\mathbf{P}}_5)) = \frac{1}{2}S_{23}^2(S_{23}^2 + S_{11} - S_{22}) \mp (1 + \frac{1}{2}S_{23}^2)\gamma.$$

Since $(1+\frac{1}{2}S_{23}^2)\gamma^2 - \frac{1}{4}S_{23}^4(S_{23}^2+S_{11}-S_{22})^2 = \gamma^2(1+S_{23}^2) + S_{12}^2S_{23}^4 \geq 0$, it is guaranteed that for $\alpha = \frac{1}{2}(S_{11}+\beta+\gamma)$ the inequality $\mathrm{trace}(\mathrm{adj}(\overline{\mathbf{P}}_5)) \leq 0$ is fulfilled.

We have now reached the point, where $\mathbf{P}_5$ has the following form:

$$\mathbf{P}_5 = \begin{bmatrix} \mu_1 & 0 & 0 & p_1 \\ 0 & \mu_2 & 0 & p_2 \\ 0 & 0 & 0 & p_3 \\ p_1 & p_2 & p_3 & p_0 \end{bmatrix},$$

where $\mu_1\,\mu_2 < 0$.

We perform a translation given by the matrix

$$\mathbf{V}_5 = \begin{bmatrix} \mathbf{I} & \mathbf{v} \\ 0 & 1 \end{bmatrix}.$$

If we set

$$v_1 = -\frac{p_1}{\mu_1}, \quad v_2 = -\frac{p_2}{\mu_2}, \quad v_3 = -\frac{p_1v_1 + p_2v_2 + p_0}{2p_3},$$

this yields

$$\mathbf{P}_6 = \mathbf{V}_5^T\,\mathbf{P}_5\,\mathbf{V}_5 = \mathrm{diag}\left(\begin{bmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{bmatrix}, \begin{bmatrix} 0 & t_3 \\ t_3 & 0 \end{bmatrix}\right)$$

We divide $\mathbf{P}_6$ by $-t_3$:

$$\mathbf{P}_7 = -\frac{1}{t_3}\mathbf{P}_6 = \mathrm{diag}\left(\begin{bmatrix} \nu_1 & 0 \\ 0 & \nu_2 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}\right)$$

After that we use another transformation $\mathbf{V}_7$

$$\begin{aligned} \mathbf{P}_8 &= \mathbf{V}_7^T\,\mathbf{P}_7\,\mathbf{V}_7 \quad \text{with} \quad \mathbf{V}_7 = \mathrm{diag}(\frac{1}{\sqrt{|\nu_1|}}, \frac{1}{\sqrt{\nu_2|}}, 1, 1) \\ &= \mathrm{diag}\left(\begin{bmatrix} \pm 1 & 0 \\ 0 & \mp 1 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}\right) \end{aligned}$$

Finally we perform the transformation $\mathbf{V}_8$.

$$\begin{aligned} \mathbf{P}_9 &= \mathbf{V}_8^T\,\mathbf{P}_8\,\mathbf{V}_8 \text{ with } \mathbf{V}_8 = \mathrm{diag}(\begin{bmatrix} \frac{1}{\sqrt{2}} & \pm\frac{1}{\sqrt{2}} \\ \mp\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}, 1, 1) \\ &= \mathrm{diag}\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}\right) \end{aligned}$$

I.e. P_9 is a hyperbolic paraboloid with the equation $x_3 = x_1\,x_2$.

A combination of all transformations yields the transformation matrix $\mathbf{U}$. In the first case we set $\mathbf{U} = \mathbf{V}_1\,\mathbf{V}_5\,\mathbf{V}_7\,\mathbf{V}_8$ and in the second case $\mathbf{U} = \mathbf{V}_1\,\mathbf{V}_2\,\mathbf{V}_3\,\mathbf{V}_5\,\mathbf{V}_7\,\mathbf{V}_8$. We transform the quadric $\mathbf{B}$ by means of $\mathbf{U}$. Then we intersect the resulting quadric $\mathbf{C} = \mathbf{U}^T\,\mathbf{B}\,\mathbf{U}$ with the family of lines which lie on the hyperbolic paraboloid with the equation $x_3 = x_1\,x_2$: Putting $\mathbf{x} = [x_1, x_2, x_1\,x_2, 1]^T$ into $\mathbf{x}^T\mathbf{C}\,\mathbf{x} = 0$ yields a multivariate polynomial in x_1 and x_2 which we solve for x_2:

$$x_2(x_1) = \frac{a(x_1) \pm \sqrt{c(x_1)}}{b(x_1)},$$

where $a(x_1)$ and $b(x_1)$ are polynomials of degree 2 and $c(x_1)$ is a polynomial of degree 4. This already proves that the postulated parameterization exists.

If $c(x_1)$ does not have any real roots, the quadrics A and B do not intersect. If $c(x_1)$ has two simple roots, the intersection curve consists of one connected component and if $c(x_1)$ has four simple roots, the intersection curve is decomposed into two connected components. The existence of multiple roots is regarded as a degenerate situation. □