An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics ⋆

Elmar Schömer

Johannes Gutenberg-Universität Mainz, Institut für Informatik, 55099 Mainz, Germany, schoemer@informatik.uni-mainz.de

Nicola Wolpert

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, wolpert@mpi-sb.mpg.de

Abstract

We present an approach for the exact and efficient computation of a cell in an arrangement of quadric surfaces. All calculations are based on exact rational algebraic methods and provide the correct mathematical results in all, even degenerate, cases. By projection, the spatial problem is reduced to the one of computing planar arrangements of algebraic curves. We succeed in locating all event points in these arrangements, including tangential intersections and singular points. By introducing an additional curve, which we call the Jacobi curve, we are able to find non-singular tangential intersections. We show that the coordinates of the singular points in our special projected planar arrangements are roots of quadratic polynomials. The coefficients of these polynomials are usually rational and contain at most a single square root. A prototypical implementation indicates that our approach leads to good performance in practice.

1 Introduction

Computing arrangements of curves and surfaces is one of the fundamental problems in different areas of computer science like solid modeling, computational geometry, and algebraic geometry. As long as arrangements of lines

⋆ Partially supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG – Effective Computational Geometry for Curves and Surfaces)

and planes defined by rational numbers are considered, all computations can be done over the field of rational numbers. This avoids numerical errors and leads to exact mathematical results as well as to good running time behaviors.

As soon as higher degree algebraic curves and surfaces are considered, instead of linear ones, things become more difficult. In general the intersection points of two planar curves or three surfaces in 3-space defined by polynomials with rational coefficients have irrational coordinates. That means instead of rational numbers one now has to deal with algebraic numbers. One way to overcome this difficulty is to develop algorithms that use floating point arithmetic. These algorithms are quite fast but in degenerate situations they can lead to completely wrong results because of approximation errors, rather than just slightly inaccurate outputs. Assume for example that for two planar curves one is interested in the number of intersection points. If the curves have tangential intersection points, numerical inaccuracies can lead to a wrong output.
A second approach besides using floating point arithmetic is to use computer algebra methods, based on exact arithmetic, and thus guarantee the correctness of the results. Algebraic geometry provides a rich theory for analyzing degenerate geometric situations, but one has to be careful in choosing only those techniques which perform well in the problem-specific context and which yield acceptable running times when compared to the floating point approach.

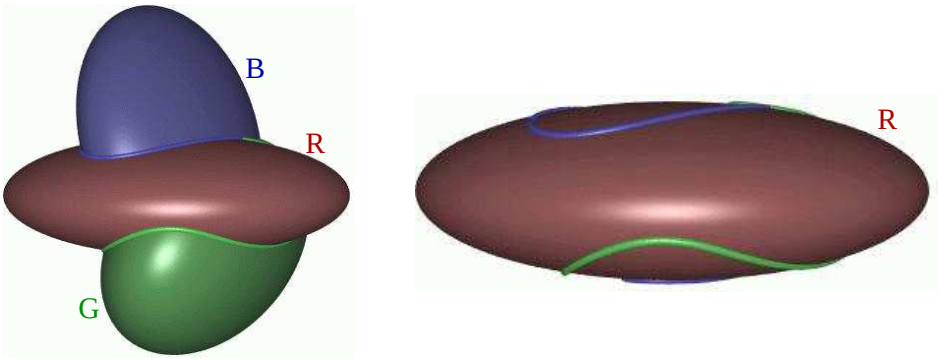

Fig. 1. An arrangement of three ellipsoids. The ellipsoids B and G intersect the ellipsoid R in the middle in two spatial curves running on the surface of R.

We consider arrangements induced by quadric surfaces in 3-dimensional space. Quadric surfaces, or quadrics for short, are defined as the set of roots of quadratic polynomials. For example, the ellipsoid R in Figure 1 is defined by the polynomial

$$R(x, y, z) \;=\; 27x^2 + 62y^2 + 249z^2 - 10.$$

A set of surfaces partitions the affine space in a natural way into four different types of maximal connected regions of dimensions 3, 2, 1, and 0 called cells, faces, edges, and vertices, respectively. We present an approach for computing

the mathematical correct topology of a cell in an arrangement of quadrics. Our algorithm is

(1) *exact* in the sense that it always computes the mathematical correct result, even for degenerate inputs, and
(2) *efficient* in practice concerning its running time.

As far as we know, we are the first who provide a solution to this problem [35]. For more details consider also [70]. Our algorithm uses exact rational algebraic computation and it can handle each degenerate input. A prototypical implementation shows that the theoretical results promise a good performance in practice.

Our approach operates similarly to the cylindrical algebraic decomposition [20]. On the surface of a given quadric p, the intersection curves of p with the remaining quadrics build a 2-dimensional subarrangement. In our example, the ellipsoids B and G intersect the ellipsoid R. This leads to two intersection curves on the surface of the ellipsoid R (right picture in Figure 1). Vertices of this subarrangement are common points of two intersection curves, that means intersection points of three quadrics.

Computing the 2-dimensional subarrangement on the surface of each quadric is the basic computation that has to be done independent of the special information about the arranged quadrics one may be interested in, in our case the topological description of a cell. The problem is particularly difficult because vertices are not expressible as nested square roots of rational numbers. We look for a method that conceptually is also extendible to more complicated surfaces. For intersection curves of surfaces of degree greater than 2 there is no hope to find a parameterization which can be manipulated symbolically in an easy way. Therefore we choose an approach that works by projection. For each quadric p we project all its intersection curves with the other quadrics and additionally its silhouette into the plane. This projection step applied to our example proceeds like shown in Figure 2. The main contribution of this work is that we show how to compute the planar arrangements resulting from the projection. Although we solve planar arrangements of a very special kind, some of the methods we present carry over to arrangements of arbitrary algebraic curves and therefore prepare the ground for computing with these curves.

All curves of the planar arrangements we consider turn out to be defined by polynomials of degree at most 4. So the reduction is algebraically optimal in the sense that it does not affect the algebraic degree of the curves we consider. In such arrangements of projected curves singular points and tangential intersections appear quite frequently as can be seen in the last picture of Figure 2. The main question with respect to exactness and efficiency is how to

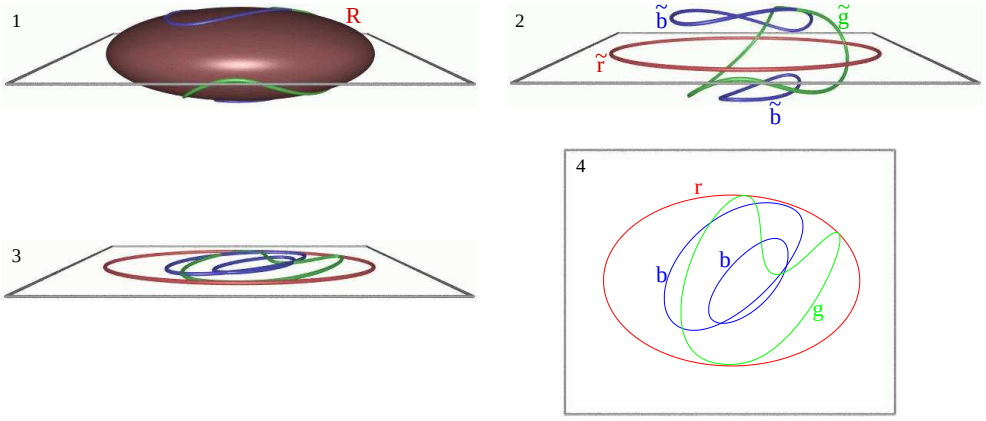

Fig. 2. Project the intersection curves $\tilde{b}$ and $\tilde{g}$ of the ellipsoid R with the ellipsoids B and G, respectively, and the silhouette $\tilde{r}$ of R into the plane. This leads to the planar arrangement of the curves b, g, and r.

locate these points. Our contribution, and what is new, is that we succeed in determining all event points in the planar arrangement efficiently, including tangential intersection points and singular points. This works for the following two reasons:

(1) We show that determining non-singular tangential intersection points can be reduced to the problem of locating transversal intersection points. For the latter we know that they can be located easily. The reduction is done by introducing a new curve to the arrangement.
(2) We succeed in factoring univariate polynomials in a way that the coordinates of singular points are roots of quadratic rational polynomials. Only in the case that a curve consists of four lines, computing the coordinates requires a second square root.

The organization of the remaining sections is as follows. In Section 2 we give an overview of previous work related to ours. Section 3 provides the notation and mathematical tools we need for our approach. In Chapter 4 we sketch the overall structure of our algorithm for computing a cell in an arrangement of n quadrics. We show how to reduce the 3-dimensional problem to n planar ones. We introduce simple box hit counting as a tool for determining transversal intersections of two curves. Section 5 first provides a method for distinguishing transversal intersection points from tangential intersection points and from singular points. Afterwards, we define an auxiliary curve, resulting in a new method for determining some non-singular intersections called extended box hit counting. Section 6 deals with the singular points of the planar curves we obtain from the reduction. We classify them in two different groups. We prove that one of the groups contains more than 2 singular points only if the spatial intersection curve of two quadrics consists of two lines and another conic curve. In Section 7 we prove our main theorem, namely that every event point in the planar arrangement can be determined. Throughout the previous

chapters some generality assumptions concerning the planar curves are made. In Section 8 we explain how to test and achieve these assumptions for general input. Finally, in Section 9 we discuss the results we obtained from a prototype implementation of determining the event points in the plane and give the prospects for further research.

2 Previous work

As mentioned, methods for the calculation of arrangements of algebraic curves and surfaces are an important area of research in different branches of computer science.

2.1 Solid Modeling

Arrangements of curved surfaces typically arise in solid modeling, see for example [38], 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. None of these systems are exact. Recently some efforts have been made towards exact and efficient implementations:

MAPC [42] is a library for exact computation and manipulation of algebraic points and curves. It includes an algorithm for computing the arrangement of curves in the plane. Degenerate situations like tangential intersections or singular points are explicitly not treated. ESOLID [23] performs exact boundary evaluation of low-degree curved solids. Also here it is stated that degenerate cases cannot be handled. For a more detailed description of MAPC and ESOLID consider the PhD thesis of Keyser [43].

2.2 Computational Geometry

Also in computational geometry there is a great focus on computing arrangements, but mainly on arrangements of linear objects. Consider the overview articles of Halperin [37] and Agarwal and Sharir [2]. Algorithms coping with

arrangements of lines can be implemented with exact rational arithmetic and with a good performance, because they only deal with linear algebraic primitives, see for example the fast filtered implementations in LEDA [47] and CGAL [34]. There are some geometric methods dealing with arbitrary curves and surfaces, see for example Mulmuley [54], Dobkin and Souvaine [25], Snoeyink and Hershberger [66], Bajaj and Kim [8], Nielsen and Yvinec [56], and Schwarzkopf and Sharir [64]. But all of them neglect the problem of exact computation in the way that they are based on an idealized real arithmetic provided by the real RAM model of computation [59]. The assumption is that all, even irrational, numbers are representable and that one can deal with them in constant time. This postulate is not in accordance with real computers.

Recently the exact computation of arrangements of non-linear objects has come into the focus of research. Several authors have looked into the question of using restricted predicates to report or compute segment intersections, [14], [13], and [17]. The restriction used in these papers is on the degrees of the predicates used by the algorithms. By restricting to low-degree predicates, one can generally achieve more robust computations. Predicates for arrangements of circular arcs are treated by Devillers et al. in [24]. Recent work by Emiris and Tsigardias [31] discusses some predicates on conics in this style. However, these approaches do not extend easily to more complicated curves.

Wein [69] extended the CGAL implementation of planar maps to conic arcs. Berberich et al. [11] made a similar approach for conic arcs based on the improved LEDA [47] implementation of the Bentley-Ottmann sweep-line algorithm [9]. Eigenwillig et al. [28] extended the sweep-line approach to cubic curves, see also [27]. A generalization of Jacobi curves (used below for locating tangential intersections) is described by Wolpert [71]. Finally there are efforts to extend CGAL with a kernel for curved objects [30] but this works only for circular arcs till now.

2.3 Algebraic Geometry

Computational real algebraic geometry studies algorithmic questions dealing with real solutions of a system of equalities and inequalities of polynomials over real numbers, see for example the overview article of Mishra [52].

Collins [20] introduced the cylindrical algebraic decomposition (CAD) as an improvement of the results obtained by Tarski [67] for quantifier elimination. The cylindrical algebraic decomposition is based on projection and partitions the d-dimensional Euclidean space $\mathbb{R}^d$ into connected subsets compatible with the zeros of a set of input polynomials. Some work has been done on improving

the result of Collins, see for example Arnon, Collins, and McCallum [4], [5], and [6], and Edelsbrunner et al. [18].

In principle the cylindrical algebraic decomposition can be implemented and our algorithm is based on this method. The problem is that after the projection steps one is left with roots of univariate polynomials. It is an open problem how to really implement the necessary algebraic primitives for the backwards construction in an exact and efficient way based on the computation with these algebraic numbers. Of course one could use the gap theorem introduced by Canny [16] or multivariate Sturm sequences discussed by Pedersen [58] and Milne [51] but the running time of both methods is quite high.

For non-singular curves in the plane, some specific work has been done by Arnborg and Feng [3] and Arnon and McCallum [7]. Based on the real RAM model Abhyankar and Bajaj [1] give a polynomial time algorithm that determines the genus of a plane algebraic curve. Sakkalis [62] uses rational arithmetic to compute the topological configuration of a single curve. He determines isolating boxes for the singular points with the help of negative polynomial remainder sequences. This last approach, although it is exact, is not very efficient, at least if singular points occur frequently. Hong [39] improves this approach by using floating point interval arithmetic.

Of course, in algebraic geometry and computer algebra some effort was made in developing software. For example, LiDIA [57] is a library for computational number theory. APU is [61] a tool for real algebraic numbers. Core [40] and LEDA [47] are libraries that address the issues of robust numerical and geometric computation.

2.4 Quadric surface intersection

Quadric surfaces are of great importance because they are the simplest of all curved surfaces and they are widely used in the design of mechanical parts. Levin [45], [46] introduced a pencil method for computing an explicit parametric representation of the intersection between two quadrics. Arguing that Levin's method does not take advantage of the fact that degenerate intersection curves admit a rational parameterization, Farouki, Neff, and O'Connor [33] made a complete study of degenerate cases for arbitrary quadric surfaces.

Based on Levin's method specific work has been done for natural quadrics, see for example Miller [49], Goldman and Miller [48], [50], and Shene and Johnstone [65].

Interval arithmetic is used by Geismann, Hemmer, and Schömer [35] to keep track of all occurring rounding and approximation errors that appear in Levin's

algorithm while computing a cell in an arrangement of quadrics. If the input does not lie too near to a degenerate configuration, the algorithm will succeed in predicting the correct topological structure of the intersection. Otherwise it can detect the existence of a critical situation.

Dupont, Lazard, Lazard, and Petitjean [26] and recently Lazard, Penaranda, and Petitjean [44] improved and implemented Levin's method for computing parameterizations for the intersection of two arbitrary implicit quadrics. Their parameterization is nearly optimal in the sense that its coefficients are contained in the smallest possible field extension of the rational numbers, up to a unique perhaps unnecessary square root. The lack of easy parameterizations for intersection curves of higher degree surfaces restricts their approach to quadric surfaces.

A very recent result on computing arrangements of quadric surfaces is by Mourrain et al. [53]. They use a space-sweep algorithm and therefore reduce the static 3-dimensional problem to a dynamic 2-dimensional one.

3 Notation

In this section we will shortly introduce the mathematical notation we will use in the following.

3.1 Surfaces and Curves

The objects we consider and manipulate in our work are algebraic surfaces and curves represented by rational polynomials. More generally, we define an *algebraic hypersurface* in the following way: Let f be a polynomial in $\mathbb{Q}[x_1, \ldots, x_d]$. We set

$$\text{ZERO}(f) \;:=\; \{(a_1, \ldots, a_d) \in \mathbb{R}^d \mid f(a_1, \ldots, a_d) = 0\}$$

and call $\text{ZERO}(f)$ the *algebraic hypersurface* defined by f. We reserve the terms *algebraic surface* and *algebraic curve* for the special cases $d = 3$ and $d = 2$, respectively. For example the ellipsoid R in Figure 1 is defined by the polynomial $R(x, y, z) = 27x^2+62y^2+249z^2-10$. If the context is unambiguous, we will often identify the defining polynomial of a hypersurface with its zero set.

The *total degree* of an algebraic hypersurface is the highest degree of all monomials of its defining polynomial. Thus, ellipsoids are degree 2 algebraic surfaces. We call degree 2 algebraic surfaces *quadric* surfaces, or *quadrics* for short.

A hypersurface f is called *squarefree* if there are no polynomials $f_1, f_2 \in \mathbb{Q}[x_1, \ldots, x_d]$ of positive total degrees with $f = f_1^2 \cdot f_2$. For a hypersurface f the *gradient vector* of f is defined to be

$$\nabla f \;=\; \left(\frac{\partial f}{\partial x_1}\,,\; \frac{\partial f}{\partial x_2}\,,\; \ldots\,,\; \frac{\partial f}{\partial x_d}\right) \;=:\; (f_{x_1}, f_{x_2}, \ldots, f_{x_d}) \in (\mathbb{Q}[x_1, \ldots, x_d])^d.$$

In the following we will always use the notation $\frac{\partial f}{\partial x} =: f_x$ for the partial derivative of f with respect to x. With the help of the gradient vector we characterize a point $a = (a_1, \ldots, a_d) \in \mathbb{R}^d$ lying on the hypersurface f. It is named a *singular point* of f if $\nabla f(a) = 0$, otherwise it is *non-singular*. The geometric interpretation is that singular points of a squarefree hypersurface f are exactly the ones that do not admit a unique tangential hyperplane to f. In a non-singular point a of f the tangential hyperplane is perpendicular to $(\nabla f)(a)$.

As we will see, our main task will be the computation of arrangements of curves in the plane. So for the moment let us consider $d = 2$. Let (a, b) be a non-singular point of a curve f in the plane, i.e. there exists a well defined tangent line in that point. Under certain assumptions we do a further classification of (a, b) (for illustration have a look at Figure 3):

(1) We speak of (a, b) having a *vertical tangent* in the case $\nabla f(a, b) = c \cdot (1, 0)$ with c being a non-zero constant.
(2) The point (a, b) is called a *turning point* of the curve if the tangent of f at (a, b) crosses f in (a, b). At a turning point the curvature of f changes sign. A necessary condition is that the polynomial $f_1(x, y) := (f_{xx} f_y^2 - 2 f_x f_y f_{xy} + f_{yy} f_x^2)(x, y) \in \mathbb{Q}[x, y]$ has a root at $(x, y) = (a, b)$: $f_1(a, b) = 0$.
(3) If (a, b) is a turning point that additionally has a vertical tangent, then we call it a *vertical turning point*. In our work we consider curves of degree at most 4. A point (a, b) is a vertical turning point of a curve f of degree at most 4 iff it is non-singular and has a 3-fold intersection with the vertical line $x = a$, i.e. iff

$$f_y(a, b) = 0 \quad \text{and} \quad f_{yy}(a, b) = 0 \quad \text{and} \quad f_{yyy}(a, b) \neq 0 \quad \text{and} \quad f_x(a, b) \neq 0.$$

(4) We call (a, b) an *extreme point* if it has a vertical tangent but is not a turning point.

A point $a = (a_1, \ldots, a_d) \in \mathbb{R}^d$ is called an *intersection point* of two hypersurfaces f and g if it lies on the hypersurface f as well as on the hypersurface g. It is called a *tangential intersection point* of f and g if additionally the two gradient vectors $\nabla f(a)$ and $\nabla g(a)$ are linearly dependent in a. Otherwise we

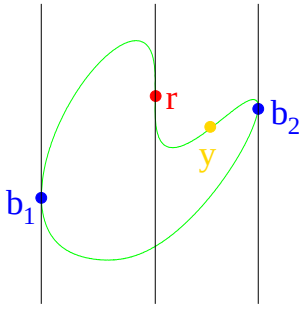

Fig. 3. The curve has two extreme points b_1, b_2, one vertical turning point r, and a turning point y.

speak of a *transversal intersection point*. If a is an intersection point of f and g and simultaneously a singular point of f, then of course $\nabla f(a) = 0$ and a is a tangential intersection point of f and g. We call an intersection point *non-singular*, if it is neither a singular point of f nor of g.

Two hypersurfaces $f, g \in \mathbb{Q}[x_1, \ldots, x_d]$ are *coprime* if they only share a common constant factor. The set of all intersection points of two coprime surfaces p and q is named *intersection curve*. In 3-space a point (a, b, c) on the intersection curve is a tangential intersection point of p and q if and only if the two gradient vectors $(f_x, f_y, f_z)(a, b, c)$ and $(g_x, g_y, g_z)(a, b, c)$ are linearly dependent, which can be expressed algebraically as:

$$(\nabla f \times \nabla g)\ (a,b,c) \ = \ \begin{pmatrix} f_y g_z - f_z g_y \\ f_x g_z - f_z g_x \\ f_x g_y - f_y g_x \end{pmatrix} (a,b,c) \ = \ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

For two planar curves f and g an intersection point (a, b) is tangential if and only if $(f_x g_y - f_y g_x)(a, b) = 0$.

We are interested in real singular, extreme, and vertical turning points of one curve f and also in real intersection points of two curves f and g. But $\mathbb{R}$ is not algebraically closed and most of the time we have to work over its algebraic closure $\mathbb{C}$. Therefore we transfer all notations and definitions we made for real points also to points in complex d-dimensional space.

3.2 Generality assumptions

After introducing the most important notation we will name properties of hypersurfaces that are, unlike the previous definitions, not intrinsic to the geometry of the arrangement induced by the hypersurfaces. They only depend

on the way the hypersurfaces are represented or on our chosen coordinate system.

We will establish squarefreeness for each hypersurface and coprimality for each pair of hypersurfaces we consider during our algorithm. Both conditions can be easily tested with resultants which we will introduce in the next section. If necessary, we split the curves with bivariate gcd-computation into squarefree and coprime subcurves, each defined by a rational polynomial. This operation does not change the topology of the arrangement. It just changes the way the curves are represented.

We call a polynomial $f = f_n x_d^n + \ldots + f_0 x_d^0 \in \mathbb{Q}[x_1, \ldots, x_d]$ with coefficients $f_i \in \mathbb{Q}[x_1, \ldots, x_{d-1}]$ *generally aligned with respect to* x_d if f_n is a non-zero constant: $0 \neq f_n \in \mathbb{Q}$. The polynomial f is named *generally aligned* if it is generally aligned with respect to every x_i, $1 \leq i \leq d$. Geometrically general alignment means the absence of asymptotes parallel to the coordinate axes.

We say that two curves defined by $f, g \in \mathbb{Q}[x, y]$ are in *general relation with respect to* x, if they have no two common roots with the same x-value. If two polynomials f and g are in general relation with respect to x as well as with respect to y, they are in *general relation*.

General alignment and general relation constitute no restriction on our surfaces and curves we consider. In Section 8 we will show how to test and realize them with a random shear for each kind of input. A shear has no effect on the topology of the arrangement.

We call a pair of curves f and g *well-behaved*, if both curves are generally aligned, squarefree, coprime and in general relation.

4 The basic algorithmic and algebraic ideas

Our aim is to compute the topology of a cell in an arrangement of quadrics. Let P be the set of all quadrics. The basic operation that has to be at our disposal is the following: For each quadric $p \in P$ we have to compute the 2-dimensional subarrangement on its surface, remember Figure 1. That means for all $p \neq q \in P$ we have to compute the intersection curve of p and q and we have to compute the interaction of all these curves on the surface of p. Our approach for computing the subarrangements is based on a *projection step* as it also occurs in the cylindrical algebraic decomposition [20].

We compute the projection of the intersection curve of two quadrics into the plane. Algebraically this means determining the (x, y)-coordinates of common roots of the two defining polynomials. This computation can be done with the help of resultants. For some further investigations we will also introduce subresultants. All propositions here are stated without proofs. For the proofs and further references on resultants consider for example [22], [72], for the ones on subresultants have a look at [15], [68].

Suppose we are given $f, g \in \mathbb{Q}[x_1, \ldots, x_d]$ with positive degree in x_d. We write

$$f \;=\; f_n x_d^n + \ldots + f_0 x_d^0 \quad , \quad g \;=\; g_m x_d^m + \ldots + g_0 x_d^0.$$

Here f and g are regarded as polynomials in x_d with coefficients $f_i, g_j \in \mathbb{Q}[x_1, \ldots, x_{d-1}]$, $1 \leq i \leq n$ and $1 \leq j \leq m$. We define the *Sylvester matrix* of f and g with respect to x_d to be the following $(m+n) \times (m+n)$ coefficient matrix with m rows of f-entries and n rows of g-entries:

$$\mathrm{Syl}(f, g, x_d) \;:=\; \begin{pmatrix} f_n & f_{n-1} & \cdots & f_0 & & & \\ & f_n & f_{n-1} & \cdots & f_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & f_n & f_{n-1} & \cdots & f_0 \\ g_m & g_{m-1} & \cdots & g_0 & & & \\ & g_m & g_{m-1} & \cdots & g_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_m & g_{m-1} & \cdots & g_0 \end{pmatrix}$$

where the empty places are filled with zeros. The determinant of the matrix is called the *resultant* of f and g with respect to x_d: $\mathrm{res}(f, g, x_d) := \det(\mathrm{Syl}(f, g, x_d)) \in \mathbb{Q}[x_1, \ldots, x_{d-1}]$. One can show that the resultant is a polynomial of total degree at most $n \cdot m$.

Here is the statement that the resultant of f and g with respect to x_d performs a projection of the common roots of f and g onto the $(x_1, \ldots, x_{d-1})$-plane:

Theorem 1 *Let $f, g \in \mathbb{Q}[x_1, \ldots, x_d]$ be generally aligned with respect to x_d. A complex point $(c_1, \ldots, c_{d-1}) \in \mathbb{C}^{d-1}$ is extendible to a common solution $(c_1, \ldots, c_{d-1}, c_d) \in \mathbb{C}^d$ of f and g if and only if $(c_1, \ldots, c_{d-1})$ is a root of the resultant $res(f, g, x_d)$.*

The resultant is equal to the zero polynomial iff f and g are not coprime.

We have just stated that a point $(c_1, \ldots, c_{d-1}) \in \mathbb{C}^{d-1}$ is a root of the resultant $\mathrm{res}(f, g, x_d)$ if and only if $f(c_1, \ldots, c_{d-1}, x_d) \in \mathbb{C}[x_d]$ and $g(c_1, \ldots, c_{d-1}, x_d) \in \mathbb{C}[x_d]$ have a common factor. Sometimes one is also interested in the degree of this common factor. For answering this questions we consider subresultants: Let S_l be the submatrix of the Sylvester matrix $\mathrm{Syl}(f, g, x_d)$ obtained by deleting the last $2l$ columns, the last l rows of f-entries, and the last l rows of g-entries. The determinant of this matrix is again a polynomial in $\mathbb{Q}[x_1, \ldots, x_{d-1}]$ and we call it the l-th *subresultant* of f and g with respect to x_d: $\mathrm{sres}_l(f, g, x_d) \in \mathbb{Q}[x_1, \ldots, x_{d-1}]$. One can prove the following:

Theorem 2 *Let $f, g \in \mathbb{Q}[x_1, \ldots, x_d]$ be generally aligned with respect to x_d. For a point $(c_1, \ldots, c_{d-1}) \in \mathbb{C}^{d-1}$ the polynomials $f(c_1, \ldots, c_{d-1}, x_d) \in \mathbb{C}[x_d]$ and $g(c_1, \ldots, c_{d-1}, x_d) \in \mathbb{C}[x_d]$ have a greatest common divisor of degree h if and only if h is the least index l for which $sres_l(f, g, x_d)$ does not vanish at $(c_1, \ldots, c_{d-1})$.*

4.2 The projection phase

In order to correctly interpret the resultant of two quadrics as the projected intersection curve, we assume throughout this and the next chapters that the quadratic input polynomials are *squarefree* and *generally aligned* and that each two of them are *coprime*. These assumptions constitute no restriction on the input quadrics, see also Section 8.

From the point of view of the (x, y)-plane a quadric p consists of three different parts: the *lower part*, the *silhouette*, and the *upper part*. The *lower (upper) part* of the quadric p consists of all points $(a, b, c) \in \mathbb{R}^3$ such that $p(a, b, z) \in \mathbb{R}[z]$ has two different real roots and c is the smaller (bigger) root. The *silhouette* of the quadric p consists of all points $(a, b, c) \in \mathbb{R}^3$ such that $p(a, b, z) \in \mathbb{R}[z]$ has one root of multiplicity 2 and c is this root.

For each input quadric p we want to compute the planar arrangement on its surface. So we perform a projection step for each quadric p. In our overall example the ellipsoid R is intersected by the ellipsoid G in one intersection curve $\tilde{g}$ consisting of one connected component and by the ellipsoid B in a second intersection curve $\tilde{b}$ consisting of two connected components, see Figure 4. With the help of resultants all intersection curves are projected onto the (x, y)-plane. This leads to the two planar curves g and b in our example. Besides the intersection curves we also project the silhouette of the underlying quadric p, in our example the silhouette $\tilde{r}$ of the ellipsoid R, leading to the curve r. It is easy to see that this last projection can be performed by computing $\mathrm{res}(p, p_z, z)$.

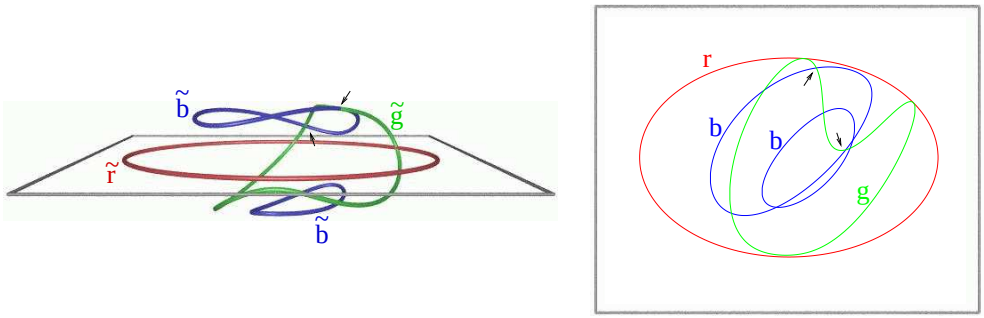

Fig. 4. For the quadric R we project its silhouette and all its intersection curves with the other quadrics into the plane.

In the planar arrangement we obtain for quadric p, there are two different types of planar curves and exactly one curve is of the first type:

silhouettecurve: The projection of the silhouette of p. The planar curve is the set of roots of $\mathrm{res}(p, p_z, z)$ and its algebraic degree is bounded from above by $\deg(p) \cdot \deg(p_z) = 2$. This is the curve r in our example.

cutcurve: The projection of the spatial intersection curve of p with another quadric q. The planar curve is the set of roots of $\mathrm{res}(p, q, z)$ and its algebraic degree is at most 4.

During the projection we lose the spatial information. Points of intersection curves on the upper part of p and on the lower part of p are projected on top of each other. This can cause singular points. For example the two branches of the curve $\tilde{b}$, one running on the upper and one on the lower part of the ellipsoid R, are projected on top of each other generating two self-intersections, see Figure 4.

Moreover, in space the curves $\tilde{b}$ and $\tilde{g}$ have 2 intersection points, marked by the small arrows. The projected curves b and g in comparison have 6 intersection points, 4 of them resulting from the loss of spatial information.

The important part of our algorithm with respect to exactness and efficiency is to compute the planar arrangements we obtain from the projection. Of course, afterwards we have to recover the spatial information in a postprocessing step in order to compute the arrangement of spatial intersection curves running on the surface of each quadric. We shortly sketch the main idea, for more details consider [12]: For each edge in the planar arrangement we have to decide whether it belongs to the upper or to the lower part of the underlying quadric p. For edges lying on the silhouettecurve nothing has to be done. Let us look at edges lying on a cutcurve. A cutcurve originates from an intersection curve of two quadrics p and q. Every edge has two adjacent cells in the plane. During the computation of the planar arrangement we easily obtain one rational point inside each of the two cells for free. We consider a line through each of these

points parallel to the z-axis and compute the order in which p and q are intersected along the lines. By comparing the two orderings we can make the decision. As soon as we have computed the arrangement on the surface of every quadric combining these results to the desired description of the whole arrangement or of one cell is only a problem of discrete combinatorics and data structures, not of exact algebraic computation.

4.3 Computing planar arrangements

We have to compute the planar arrangements we obtain from the projection phase. As we have seen, each arrangement consists of one silhouettecurve and a set of cutcurves. One way of representing the arrangement would be to store its trapezoidal decomposition [55]. The points at which a vertical attachment has to be added are the following, consider also Figure 5:

a) intersection points of two curves,
b) singular points of one curve, for example self-intersection points,
c) and extreme points.

The main part of computing the trapezoidal decomposition with respect to exactness and efficiency is to determine and locate all these points.

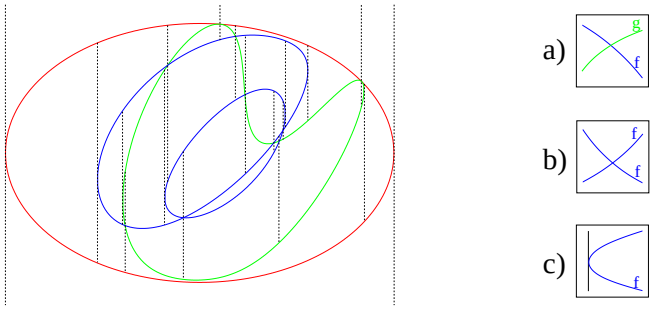

Fig. 5. A trapezoidal decomposition of the planar arrangement and points where a vertical attachment is added.

We would like to interpret singular and extreme points of a curve f as intersection points of two curves. Due to this aim we also defined vertical turning points in Section 3: The intersection points of f and f_y are exactly the singular, extreme, and vertical turning points of f. For illustration consider Figure 6. We make the following definition:

Definition 3 *The* event points *of a planar arrangement induced by a set F of planar curves are defined as the intersection points of each two curves $f, g \in F$ and the intersection points of f and f_y for all $f \in F$.*

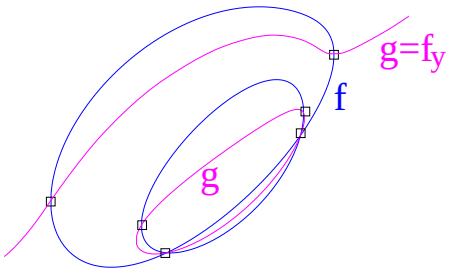

Fig. 6. Singular, extreme and vertical turning points of f are marked by small boxes. They are exactly the intersection points of f and $g = f_y$.

4.4 Computing intersection points of two curves

With our last observations we have reduced the problem of computing the event points in the planar arrangement to the question of determining intersection points of two curves f and g. Let F be the set of algebraic curves in the planar arrangement. Without loss of generality we assume that every pair of curves is well-behaved, see also Section 8. Due to our previous investigations, we can distinguish four different types of pairs of curves, the intersection points of which we want to locate:

(1) $f \in F$ and $g = f_y$, whereby f is the silhouettecurve.
(2) $f \in F$ and $g = f_y$, whereby f is a cutcurve.
(3) $f, g \in F$ and one of the two curves is the silhouettecurve and the other one is a cutcurve.
(4) $f, g \in F$ and both curves are cutcurves.

We face the problem that in general common points of two curves f and g will have irrational coordinates. Nevertheless we have to locate and characterize them exactly and unambiguously. Our solution again works in the spirit of the cylindrical algebraic decomposition. We compute the two univariate polynomials $X = \mathrm{res}(f, g, y) \in \mathbb{Q}[x]$ and $Y = \mathrm{res}(f, g, x) \in \mathbb{Q}[y]$. Let $\mathcal{R}(X)$ be the set of real roots of X and $\mathcal{R}(Y)$ be the ones of Y. Each real intersection point of f and g is a member of the grid

$$\textsc{Grid}(X, Y) \;:=\; \mathcal{R}(X) \times \mathcal{R}(Y) \;=\; \{(r_x, r_y) \mid r_x \in \mathcal{R}(X),\; r_y \in \mathcal{R}(Y)\}.$$

By definition, an *algebraic number* is a root of some polynomial $u \in \mathbb{Q}[x]$. If $\deg(u) > 2$ the real roots of u cannot always be expressed with a real expression with radicals. But we can compute an *isolating interval* for each real root α of u. That means we compute two rational numbers a and b such that α is the one and only real root of u in $[a, b]$. There are various methods of determining these isolating intervals [21], for example the algorithm of Uspensky.

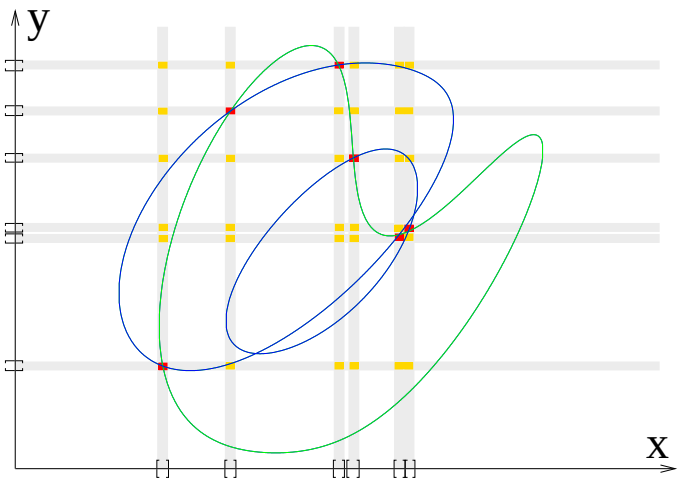

Fig. 7. Distinguish the empty light-colored boxes from the dark-colored ones

We cannot work directly on the grid $\textsc{Grid}(X, Y)$, but with the help of a root isolation algorithm we determine rational interval representations for the real algebraic numbers in $\mathcal{R}(X)$ and $\mathcal{R}(Y)$. This gives us rational intervals on the x- and y-axis, each containing one real root of X and Y, respectively. Every interval $[a, b]$ on the x-axis, $a, b \in \mathbb{Q}$, can be vertically extended to a stripe in the plane consisting of all points (x, y) with $a \leq x \leq b$ and $y \in \mathbb{R}$. In the same way each interval on the y-axis can be extended to a horizontal stripe. The intersection of the stripes yields disjoint boxes with rational corners. The real intersection points of f and g are contained in the boxes, at most one in each box.

For algorithmic reasons we furthermore want each box to contain at most one singular or extreme point of f and of g. In case a box contains an intersection point of f and g as well as a singular or extreme point, the coordinates of the two event points should be identical. This can easily be obtained by pairwise separating the real roots of X, $\text{res}(f, f_y, y)$, $\text{res}(g, g_y, y)$, and of Y, $\text{res}(f, f_y, x)$, $\text{res}(g, g_y, x)$ via gcd-computation and bisection by midpoints of the root isolating intervals.

It remains to test each box for a real intersection point. Unfortunately, the number of boxes is nearly quadratic in the number of intersection points. In the example in Figure 7 we have to distinguish the empty light-colored boxes from the dark-colored ones that contain an intersection point.

4.5 Testing a box for an intersection point

We have to answer the question whether a box with rational corners contains an intersection point of f and g or not. The problem is that we have no

information about what is happening inside the box. The only thing we can obtain is some information about the boundary of the box. Again with the help of a root isolating algorithm we compute the sequence of hits of the curves along the left edge of the box, counted with multiplicities. We analogously do the same for the upper, right, and lower edge of the box. This gives us a sequence of hits of f and g around the boundary of the box.

Sometimes this sequence can help us to determine the behavior of the curves inside the box. If there are exactly two hits with each curve, counted with multiplicities, and the hits alternate, then we can be sure that there is an intersection point inside the box at which the two curves cross each other, see Figure 8. This method of locating for example transversal intersection points we call *simple box hit counting*. It is also discussed in [42].

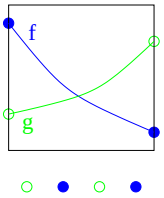

Fig. 8. Transversal intersections can be solved with simple box hit counting by examining the sequence of hits of the curves with the boundary of the box in clockwise order, staring at the lower left vertex.

Simple box hit counting:

```
determine sequence of hits of f and g with the box
while (#hits(f) > 2) or (#hits(g) > 2)
   shrink the box
   determine sequence of hits of f and g with the box
if (#hits(f) < 2) or (#hits(g) < 2) output: 0   // empty box
else
   if (hits alternate) output: 1 // intersection point
   else output: 0  // empty box
```

It is easy to see that simple box hit counting has the output `1` if and only if the box contains an intersection point at which f and g cross each other.

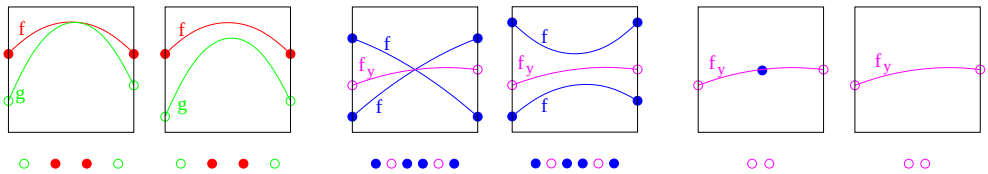

Fig. 9. Tangential intersection points and singular points cannot always be solved with simple box hit counting

The problem of simple box hit counting is that it sometimes cannot detect tangential intersection points of f and g, see Figure 9. In the first box f and g

have a non-singular tangential intersection, in the second box they have not. But the sequence of hits is identical in both cases. For non-singular tangential intersections our simple box hit counting algorithm ends up with the wrong output `empty box`.

Also singular points of f can be problematic, consider the last two pairs of boxes in Figure 9. Remember our definition that singular points of f are tangential intersection points of f and f_y. For self-intersections the first while-loop of the simple box hit counting algorithm runs forever. For the isolated point it gives the wrong answer `empty box`.

Concerning the examination of the boxes we have to solve two problems in the following:

(1) Find a method to avoid applying simple box hit counting to boxes that contain a tangential intersection in order to avoid infinite loops and wrong results.
(2) Find methods to solve these tangential intersections, that means find methods to solve (a) non-singular tangential intersections and (b) singular points.

The answer to these two questions is crucial, because, as one can see in Figure 10, non-singular tangential intersections and singular points appear quite often in our arrangements. This is the reason why classical methods like the gap theorem [16] or multivariate Sturm calculation [51] are too expensive. In the next two chapters we will develop a new method that treats these cases in a fast and robust way.

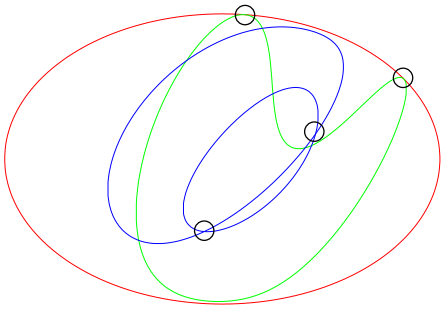

Fig. 10. Tangential intersections and self-intersections appear quite often

5 Non-singular tangential intersections

In this section we will answer the first question of how to avoid applying simple box hit counting to boxes that contain a tangential intersection. Again

remember our definition that also singular points of f can be interpreted as tangential intersection points, namely of f and f_y. The boxes are defined by the roots of the resultants on the x- and on the y-axis. The roots can have different multiplicities. There is a strong connection between the kind of intersection two planar curves f and g have at a common point (a, b) and the multiplicity of the root a of the resultant $X = \text{res}(f, g, y)$. Of course, the considerations symmetrically hold for a root b of the resultant $Y = \text{res}(f, g, x)$.

5.1 Multiple roots of the resultant

For two curves f and g we want to investigate the roots of $X = \text{res}(f, g, y) \in \mathbb{Q}[x]$ and especially their multiplicities.

Theorem 4 *Let f and g be two well-behaved polynomials. Then* every *multiple root of $X = res(f, g, y)$ is in 1-1 correspondence to one tangential intersection point of the curves defined by f and g.*

PROOF. The two curves f and g are well-behaved. From the definition of well-behavedness, which includes generally alignment and general relation, together with Theorem 1 ones derives that every root a of X is in 1-1 correspondence to one intersection point (a, b) of f and g.

Without loss of generality let us in the following assume $(a, b) = (0, 0)$. This is not a restriction because the multiplicities of the roots of X are invariant under translation of f and g: a translation of the two curves in x-direction only causes the same translation of the roots of the resultant. A translation in y-direction keeps the resultant unchanged.

In order to proof the theorem we have to show that 0 is a multiple root of X if and only if $(0, 0)$ is a tangential intersection point of f and g. Using partial derivatives it is easy to see that f and g can be written in the form

$$f(x,y) = \sum_{i=0}^{n}(\sum_{j=0}^{n-i} \frac{1}{i!j!} \underbrace{f_{x^j y^i}(0,0)}_{=:\tilde{f}_{x^j y^i}} \cdot x^j) \cdot y^i,\ g(x,y) = \sum_{i=0}^{m}(\sum_{j=0}^{m-i} \frac{1}{i!j!} \underbrace{g_{x^j y^i}(0,0)}_{=:\tilde{g}_{x^j y^i}} \cdot x^j) \cdot y^i.$$

With this new notation the resultant has the following form:

$$X \;=\; \mathrm{res}(f,g,y) \;=\; x\cdot\det\begin{pmatrix} \cdots\cdots\cdots\cdots\cdots\cdots \\ \ldots\; x^2(*)+\tilde{f}_x x \qquad 0 \\ \ldots \quad x(*)+\tilde{f}_y \quad x(*)+\tilde{f}_x \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \ldots\; x^2(*)+\tilde{g}_x x \qquad 0 \\ \ldots \quad x(*)+\tilde{g}_y \quad x(*)+\tilde{g}_x \end{pmatrix} \;=:\; x\cdot\det V$$

where all remaining entries in the last two columns of V are 0. We know that the resultant is a polynomial of degree at most $n\cdot m$ and because of that there are some rational numbers α_i, $1\le i\le mn$, with

$$X \;=\; x\cdot\det V \;=:\; x\cdot(\alpha_1+\alpha_2 x+\ldots+\alpha_{mn}x^{mn-1}).$$

The resultant has a root of multiplicity greater than 1 in $x=0$ if and only if the coefficient α_1 is equal to zero. By the definition of V and α_1 we have $\alpha_1=\det V(0)$. Substituting $x=0$ into V and applying the definition of the first subresultant we can explicitly compute α_1:

$$\begin{aligned}\alpha_1 = \det V(0) \;&=\; \det\begin{pmatrix} & \cdots\cdots \\ A & 0 \quad 0 \\ \ldots & \tilde{f}_y \;\; \tilde{f}_x \\ & \cdots\cdots \\ B & 0 \quad 0 \\ \ldots & \tilde{g}_y \;\; \tilde{g}_x \end{pmatrix} \\ &= (-1)^n\cdot(\tilde{f}_x\tilde{g}_y-\tilde{f}_y\tilde{g}_x)\cdot\det\begin{pmatrix} A \\ B \end{pmatrix} \\ &= (-1)^n\cdot(f_xg_y-f_yg_x)(0,0)\cdot\mathrm{sres}_1(f,g,y)(0).\end{aligned}$$

We conclude that 0 is a multiple root of X if and only if $(f_xg_y-f_yg_x)(0,0)=0$ or $\mathrm{sres}_1(f,g,y)(0)=0$. We want to prove that 0 is a multiple root of X if and only if $(0,0)$ is a tangential intersection point of f and g, that means if and only if $(f_xg_y-f_yg_x)(0,0)=0$. So it remains to show that $\mathrm{sres}_1(f,g,y)(0)=0$ implies $(f_xg_y-f_yg_x)(0,0)=0$.

By assumption the two curves f and g are well-behaved. By Theorem 2 we have $\mathrm{sres}_1(f,g,y)(0)=0$ if and only if the two polynomials $f(0,y), g(0,y)$ have

a common factor of degree at least 2. From general alignment we conclude that this common factor must have the form y^i for some $i \geq 2$. This implies $f_y(0,0) = g_y(0,0) = 0$ and therefore $(f_x g_y - f_y g_x)(0,0) = 0$. $\square$

Theorem 4 gives us a criterion for distinguishing boxes that can be correctly solved by simple box hit counting from the ones for which this tool is not suitable. The only thing we have to do is factoring the resultant $X = \mathrm{res}(f, g, y)$ of two curves f and g into one polynomial u_1 containing all simple roots and one polynomial u_2 containing all multiple roots: $X = u_1 \cdot u_2$. This factorization can be done by derivative- and gcd-computations. The same has to be done for $Y = \mathrm{res}(f, g, y)$: $Y = v_1 \cdot v_2$. Intersection points of f and g can only take place in boxes that are defined by roots of X and Y of the same multiplicity. The transversal intersections appear only in the boxes defined by the real roots of u_1 and v_1 and these boxes cannot contain tangential intersections. So these boxes can be correctly solved in all cases by simple box hit counting. This answers our first question we posed at the end of the last section. The remaining question is how to solve the boxes defined by the roots of u_2 and v_2.

5.2 The Jacobi curve

For well-behaved curves multiple roots correspond to tangential intersections. By definition, a point (a, b) is a tangential intersection of f and g if and only if $f(a, b) = g(a, b) = 0$ and (a, b) is a root of the polynomial $f_x g_y - f_y g_x$. This polynomial and the curve it defines will play an important role in our future investigations. Therefore we will give it a name:

Definition 5 *Let $f, g \in \mathbb{Q}[x, y]$ be two bivariate polynomials. We define a third polynomial $h \in \mathbb{Q}[x, y]$ by*

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

The set of real roots of this polynomial h we call Jacobi curve *of f and g.*

We remark that the algebraic degree of h is bounded from above by $\deg(f) + \deg(g) - 2$. With the help of the Jacobi curve we reformulate Theorem 4:

Corollary 6 *Let $f, g \in \mathbb{Q}[x, y]$ be well-behaved polynomials. The point $a \in \mathbb{C}$ is a root of multiplicity ≥ 2 of the resultant $res(f, g, y)$ if and only if there exists a number $b \in \mathbb{C}$ such that (a, b) is a common root of f, g, and $h = f_x g_y - f_y g_x$.*

For a non-singular tangential intersection point (a, b) of f and g there are two possibilities: either h cuts f and g transversally in (a, b) or tangentially,

consider Figure 11. Both situations lead to different multiplicities of the root $x = a$ of $\mathrm{res}(f, g, y)$:

Fig. 11. The Jacobi curve h either cuts tangentially or transversally through a tangential intersection point of f and g.

Theorem 7 *Let $f, g \in \mathbb{Q}[x, y]$ be two well-behaved bivariate polynomials, the planar curves of which have a non-singular tangential intersection in the point (a, b). Then the Jacobi curve h intersects f as well as g transversally in (a, b), or a is a root of multiplicity ≥ 3 of $res(f, g, y)$.*

PROOF. We again assume without loss of generality that $(a, b) = (0, 0)$. Further let f be a polynomial of total degree n and g be a polynomial of total degree m. With the notation introduced in the last subsection we obtain the following:

$$
\begin{aligned}
&\mathrm{res}(f, g, y) \\
&= x \cdot \det \begin{pmatrix}
\cdots & \cdots & \cdots & \cdots \\
\ldots & x(*) & 0 & 0 \\
\ldots & x(*) + \tilde{f}_y & x^2(*) + \tilde{f}_x x & 0 \\
\ldots & x(*) + \frac{1}{2!}\tilde{f}_{yy} & x^2(*) + \tilde{f}_{xy} x + \tilde{f}_y & x^2(*) + \frac{1}{2!}\tilde{f}_{xx} x + \tilde{f}_x \\
\cdots & \cdots & \cdots & \cdots \\
\ldots & x(*) & 0 & 0 \\
\ldots & x(*) + \tilde{g}_y & x^2(*) + \tilde{g}_x x & 0 \\
\ldots & x(*) + \frac{1}{2!}\tilde{g}_{yy} & x^2(*) + \tilde{g}_{xy} x + \tilde{g}_y & x^2(*) + \frac{1}{2!}\tilde{g}_{xx} x + \tilde{g}_x
\end{pmatrix} \\
&=: x \cdot \det V \\
&= x \cdot (\alpha_1 + \alpha_2 x + \ldots + \alpha_{mn} x^{mn-1}).
\end{aligned}
$$

Note that all other entries in the last three columns of the determinant are zero. We know from the previous section that $\alpha_1 = 0$ because f and g intersect tangentially in $(0, 0)$. It remains to show that $\alpha_2 = 0$ if h does not intersect f and g transversally. If h does not intersect f and g transversally, then all

three gradient vectors $(\tilde{f}_x, \tilde{f}_y), (\tilde{g}_x, \tilde{g}_y), (\tilde{h}_x, \tilde{h}_y)$ with

$$(\tilde{h}_x, \tilde{h}_y) = (\tilde{f}_{xx}\tilde{g}_y + \tilde{f}_x\tilde{g}_{xy} - \tilde{f}_{xy}\tilde{g}_x - \tilde{f}_y\tilde{g}_{xx}, \tilde{f}_{xy}\tilde{g}_y + \tilde{f}_x\tilde{g}_{yy} - \tilde{f}_{yy}\tilde{g}_x - \tilde{f}_y\tilde{g}_{xy})$$

are linearly dependent. We obtain the three properties

$$\begin{aligned}
1) \quad & 0 = \tilde{f}_x\tilde{g}_y - \tilde{f}_y\tilde{g}_x \\
2) \quad & 0 = \tilde{h}_x\tilde{f}_y - \tilde{h}_y\tilde{f}_x \\
& \quad = \tilde{f}_y(\tilde{f}_{xx}\tilde{g}_y - \tilde{f}_y\tilde{g}_{xx}) + 2\tilde{f}_y(\tilde{f}_x\tilde{g}_{xy} - \tilde{f}_{xy}\tilde{g}_x) - \tilde{f}_x(\tilde{f}_x\tilde{g}_{yy} - \tilde{f}_{yy}\tilde{g}_x) \\
3) \quad & 0 = \tilde{h}_x\tilde{g}_y - \tilde{h}_y\tilde{g}_x \\
& \quad = \tilde{g}_y(\tilde{f}_{xx}\tilde{g}_y - \tilde{f}_y\tilde{g}_{xx}) + 2\tilde{g}_y(\tilde{f}_x\tilde{g}_{xy} - \tilde{f}_{xy}\tilde{g}_x) - \tilde{g}_x(\tilde{f}_x\tilde{g}_{yy} - \tilde{f}_{yy}\tilde{g}_x).
\end{aligned}$$

We will show that under these three conditions $\alpha_2 = 0$ holds. We have $\det V \in \mathbb{Q}[x]$ and we know that $\alpha_2 = (\det V)'(0)$. Let V_i be the matrix we obtain from V by replacing the polynomials in the i-th column by their derivative. We obtain

$$\alpha_2 \;=\; \sum_{i=1}^{n+m} \det V_i(0).$$

Let us first have a look at $\det V_i(0)$ for $i = 1, \ldots, n+m-2$. For such a V_i we take the derivative of a column of V that is not one of the last two. For all these determinants there exist submatrices A_i and B_i with

$$\begin{aligned}
(\det V_i)(0) &= \det \begin{pmatrix}
 & & 0 & 0 \\
A_i & & \vdots & \vdots \\
 & & 0 & 0 \\
\ldots & * & \tilde{f}_y & \tilde{f}_x \\
 & & 0 & 0 \\
B_i & & \vdots & \vdots \\
 & & 0 & 0 \\
\ldots & * & \tilde{g}_y & \tilde{g}_x
\end{pmatrix} \\
&= (-1)^n \cdot (\tilde{f}_x\tilde{g}_y - \tilde{f}_y\tilde{g}_x) \cdot \det \begin{pmatrix} A_i \\ B_i \end{pmatrix} \\
&= 0
\end{aligned}$$

by our property 1). It follows

$$\alpha_2 = \det V_{m+n-1}(0) + \det V_{m+n}(0)$$

$$= \det \begin{pmatrix} \cdots\cdots\cdots\cdots \\ \cdots \quad \tilde{f}_y \quad \tilde{f}_x \quad 0 \\ \cdots \ \frac{1}{2!}\tilde{f}_{yy} \ \tilde{f}_{xy} \ \tilde{f}_x \\ \cdots\cdots\cdots\cdots \\ \cdots \quad \tilde{g}_y \quad \tilde{g}_x \quad 0 \\ \cdots \ \frac{1}{2!}\tilde{g}_{yy} \ \tilde{g}_{xy} \ \tilde{g}_x \end{pmatrix} + \det \begin{pmatrix} \cdots\cdots\cdots\cdots\cdots \\ \cdots \quad \tilde{f}_y \quad 0 \quad 0 \\ \cdots \ \frac{1}{2!}\tilde{f}_{yy} \ \tilde{f}_y \ \frac{1}{2!}\tilde{f}_{xx} \\ \cdots\cdots\cdots\cdots\cdots \\ \cdots \quad \tilde{g}_y \quad 0 \quad 0 \\ \cdots \ \frac{1}{2!}\tilde{g}_{yy} \ \tilde{g}_y \ \frac{1}{2!}\tilde{g}_{xx} \end{pmatrix}$$

with all other entries in the last 3 columns being 0. Let us have a look at these last 3 columns of V_{m+n-1} and V_{m+n}. For a $d \times d$ matrix M let $M(i,j,k)$ denote the 3×3 submatrix the entries of which are taken from the last 3 columns and the rows i, j, and k of M. If the determinant of each of these submatrices is equal to zero, that means if for each triple (i,j,k) with $1 \leq i < j < k \leq d$ we have $\det M(i,j,k) = 0$, then we can easily conclude $\det M = 0$. The two matrices V_{m+n-1} and V_{m+n} are of the same size and all columns are identical, except the last 2. So a similar argumentation about developing the two determinants with respect to the last 3 columns leads to the following: If $\det V_{m+n-1}(i,j,k) + \det V_{m+n}(i,j,k) = 0$ for all $1 \leq i < j < k \leq n+m$, then $\alpha_2 = \det V_{m+n-1} + \det V_{m+n} = 0$. In order to finish the proof it remains to show

$$\det V_{m+n-1}(i,j,k) + \det V_{m+n}(i,j,k) = 0 \quad \text{for all } 1 \leq i < j < k \leq n+m.$$

Remember our properties 1), 2), and 3) we made before.

(1) $\{i,j,k\} \not\subset \{m-1, m, n+m-1, n+m\}$
In this case we know that both matrices V_{m+n-1} and V_{m+n} have one row with only zero entries.

(2) $(i,j,k) = (m-1, m, n+m-1)$

$$\det \begin{pmatrix} \tilde{f}_y & \tilde{f}_x & 0 \\ \frac{1}{2}\tilde{f}_{yy} & \tilde{f}_{xy} & \tilde{f}_x \\ \tilde{g}_y & \tilde{g}_x & 0 \end{pmatrix} + \det \begin{pmatrix} \tilde{f}_y & 0 & 0 \\ \frac{1}{2}\tilde{f}_{yy} & \tilde{f}_y & \frac{1}{2}\tilde{f}_{xx} \\ \tilde{g}_y & 0 & 0 \end{pmatrix}$$

$$= -\tilde{f}_x(\tilde{f}_y\tilde{g}_x - \tilde{f}_x\tilde{g}_y) - \frac{1}{2}\tilde{f}_{xx}(\tilde{f}_y \cdot 0 - 0 \cdot \tilde{g}_y)$$

$$\stackrel{1)}{=} 0$$

(3) $(i,j,k) = (m-1, m, n+m)$

$$\det\begin{pmatrix} \tilde{f}_y & \tilde{f}_x & 0 \\ \frac{1}{2}\tilde{f}_{yy} & \tilde{f}_{xy} & \tilde{f}_x \\ \frac{1}{2}\tilde{g}_{yy} & \tilde{g}_{xy} & \tilde{g}_x \end{pmatrix} + \det\begin{pmatrix} \tilde{f}_y & 0 & 0 \\ \frac{1}{2}\tilde{f}_{yy} & \tilde{f}_y & \frac{1}{2}\tilde{f}_{xx} \\ \frac{1}{2}\tilde{g}_{yy} & \tilde{g}_y & \frac{1}{2}\tilde{g}_{xx} \end{pmatrix}$$
$$= \tilde{f}_y((\tilde{f}_{xy}\tilde{g}_x - \tilde{f}_x\tilde{g}_{xy}) - \frac{1}{2}\tilde{f}_x(\tilde{f}_{yy}\tilde{g}_x - \tilde{f}_x\tilde{g}_{yy}) + \frac{1}{2}\tilde{f}_y(\tilde{f}_y\tilde{g}_{xx} - \tilde{f}_{xx}\tilde{g}_y)$$
$$= -\frac{1}{2}\left(\tilde{f}_y(\tilde{f}_{xx}\tilde{g}_y - \tilde{f}_y\tilde{g}_{xx}) + 2\tilde{f}_y(\tilde{f}_x\tilde{g}_{xy} - \tilde{f}_{xy}\tilde{g}_x) - \tilde{f}_x(\tilde{f}_x\tilde{g}_{yy} - \tilde{f}_{yy}\tilde{g}_x)\right)$$
$$\stackrel{2)}{=} 0$$

(4) $(i,j,k) = (m-1, n+m-1, n+m)$

$$\det\begin{pmatrix} \tilde{f}_y & \tilde{f}_x & 0 \\ \tilde{g}_y & \tilde{g}_x & 0 \\ \frac{1}{2}\tilde{g}_{yy} & \tilde{g}_{xy} & \tilde{g}_x \end{pmatrix} + \det\begin{pmatrix} \tilde{f}_y & 0 & 0 \\ \tilde{g}_y & 0 & 0 \\ \frac{1}{2}\tilde{g}_{yy} & \tilde{g}_y & \frac{1}{2}\tilde{g}_{xx} \end{pmatrix}$$
$$= \tilde{g}_x(\tilde{f}_y\tilde{g}_x - \tilde{f}_x\tilde{g}_y) + 0$$
$$\stackrel{1)}{=} 0$$

(5) $(i,j,k) = (m, n+m-1, n+m)$

$$\det\begin{pmatrix} \frac{1}{2}\tilde{f}_{yy} & \tilde{f}_{xy} & \tilde{f}_x \\ \tilde{g}_y & \tilde{g}_x & 0 \\ \frac{1}{2}\tilde{g}_{yy} & \tilde{g}_{xy} & \tilde{g}_x \end{pmatrix} + \det\begin{pmatrix} \frac{1}{2}\tilde{f}_{yy} & \tilde{f}_y & \frac{1}{2}\tilde{f}_{xx} \\ \tilde{g}_y & 0 & 0 \\ \frac{1}{2}\tilde{g}_{yy} & \tilde{g}_y & \frac{1}{2}\tilde{g}_{xx} \end{pmatrix}$$
$$= -\tilde{g}_y(\tilde{f}_{xy}\tilde{g}_x - \tilde{f}_x\tilde{g}_{xy}) + \frac{1}{2}\tilde{g}_x(\tilde{f}_{yy}\tilde{g}_x - \tilde{f}_x\tilde{g}_{yy}) - \frac{1}{2}\tilde{g}_y(\tilde{f}_y\tilde{g}_{xx} - \tilde{f}_{xx}\tilde{g}_y)$$
$$= \frac{1}{2}\left(\tilde{g}_y(\tilde{f}_{xx}\tilde{g}_y - \tilde{f}_y\tilde{g}_{xx}) + 2\tilde{g}_y(\tilde{f}_x\tilde{g}_{xy} - \tilde{f}_{xy}\tilde{g}_x) - \tilde{g}_x(\tilde{f}_x\tilde{g}_{yy} - \tilde{f}_{yy}\tilde{g}_x)\right)$$
$$\stackrel{3)}{=} 0 \qquad \square$$

For illustration have a look at the silhouettecurve r and the cutcurve g in Figure 12. There are two tangential intersection points of r and g both causing roots of multiplicity 2 in the resultant $X = \mathrm{res}(r, g, y)$. And indeed, the Jacobi curve h has transversal intersections with r and g in both marked tangential intersection points.

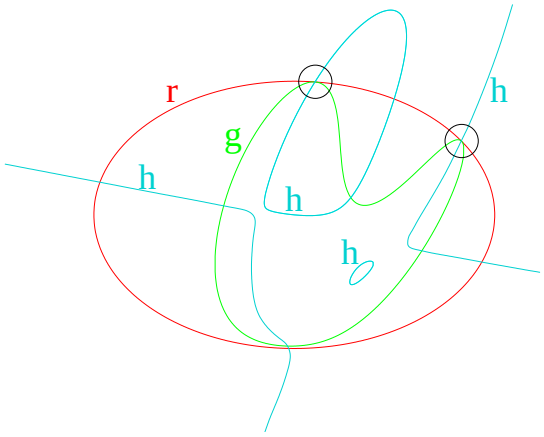

Fig. 12. Introduce the Jacobi curve in order to solve simple tangential intersections

5.3 Extended box hit counting

The Jacobi curve h leads to a new test for non-singular tangential intersection points that cause a root of multiplicity 2 in the resultant. The problem of this test is that it works only if we know in advance that an examined box contains no singular point. We address the problem of singular points and how to exclude that a singular point is contained in a box in the next two sections.

In order to determine the intersection points of two curves f and g, we partially factor their resultant $X = \mathrm{res}(f, g, y)$ over $\mathbb{Q}$ using partial derivatives and gcd-computation into one polynomial u_1 containing all simple roots, one polynomial u_2 containing all double roots, and one polynomial u_3 containing the rest: $X = u_1 \cdot u_2^2 \cdot u_3$. The same has to be done for $Y = \mathrm{res}(f, g, y)$: $Y = v_1 \cdot v_2^2 \cdot v_3$. The boxes defined by the real roots of u_1 and v_1 can be solved correctly by simple box hit counting. A box defined by u_2 and v_2 can be tested for a non-singular tangential intersection point with the help of the Jacobi-curve h. In order to get a correct result, we first have to make the box small enough to guarantee that there is exactly one intersection point between f, g, and h inside the box.

Extended box hit counting:

```
if (# different intersection points f,g,h >1)
  make box small enough
if (  simple box hit counting (f,h) = 1
    & simple box hit counting (g,h) = 1 )
  output: 1   // tangential intersection
else output: 0
```

As mentioned above, extended box hit counting only works correct if we know in advance that the box does not contain a singular point.

In the last sections we used partial factorization of univariate polynomials with respect to the multiplicities of their roots. For a clarification of our terms, we repeat and complete a notation we introduced in Section 4:

Definition 8

(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 $\textsc{Grid}(u, v)$ *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.*

Of course we can also apply other criteria than the multiplicity of the roots of the resultants in order to factor X and Y. Therefore our notation is in a more general way:

Definition 9 *Let $f, g \in \mathbb{Q}[x, y]$, $u_1, u_2 \in \mathbb{Q}[x]$ and $v_1, v_2 \in \mathbb{Q}[y]$. We call the expression $(u_1, v_1) \cdot (u_2, v_2)$ a* bi-factorization *of the bi-resultant $(X, Y) = (res(f, g, y), res(f, g, x))$ iff*

(1) $X = u_1 \cdot u_2$, $Y = v_1 \cdot v_2$,
(2) and all intersection points of f and g lie on $\textsc{Grid}(u_1, v_1) \cup \textsc{Grid}(u_2, v_2)$.

The pairs (u_1, v_1) and (u_2, v_2) are called bi-factors.

The problem we are still left with is that we do not know how to compute

(1) non-singular tangential intersections that lead to roots of multiplicity greater than 2 in the resultant and
(2) singular points

inside a box. Our two tools developed so far, namely simple and extended box hit counting, do not always reliably lead to a correct result in these cases. We next give a general criterion how to determine an intersection point of two curves, including the case that it can be a singular point of one curve, under the assumption that its defining bi-factor has a special form. We will see in the next section that for all points mentioned above such special bi-factors can always be computed [1].

Let again f and g be two bivariate polynomials. Let additionally (u, v) be a bi-factor of $(\text{res}(f, g, y), \text{res}(f, g, x))$, u and v being polynomials of degree at

[1] Of course all non-singular tangential intersections at which the branches of the two curves cross each other could also be determined by simple box hit counting. But to keep the algorithm simple we do not make this case distinction.

most 2:

$$u(x) = a_u x^2 + b_u x + c_u \quad , \quad v(x) = a_v y^2 + b_v y + c_v.$$

We assume that some kind of partial bi-factorization gave us these two rational polynomials u and v. In this case we can compute the real roots of u and v explicitly as one-root expressions:

$$\begin{aligned} x_{1,2} &= -\frac{1}{2a_u} \cdot (b_u \pm \sqrt{b_u^2 - 4a_u c_u})) =: -\frac{1}{2a_u} \cdot (b_u \pm \sqrt{a}) \\ y_{1,2} &= -\frac{1}{2a_v} \cdot (b_v \pm \sqrt{b_v^2 - 4a_v c_v})) =: -\frac{1}{2a_v} \cdot (b_v \pm \sqrt{b}). \end{aligned}$$

In order to determine whether f and g intersect in one of the points (x_i, y_j) we just have to test whether $f(x_i, y_j) = 0 = g(x_i, y_j)$. Testing simple square-root expressions for zero can be made by using root separation bounds, for example realized in the LEDA real class [47]. So if there are quadratic polynomials $u, v \in \mathbb{Q}$ the roots of which define boxes in the plane, then we can explicitly test each box for an intersection point. We call this method *explicit solutions*.

Assume for example we know a quadratic bi-factor (u, v) of (X, Y) describing the two self-intersection points in Figure 13. In this case we are able to compute the behavior of the curves directly on the grid $\textsc{Grid}(u, v)$.

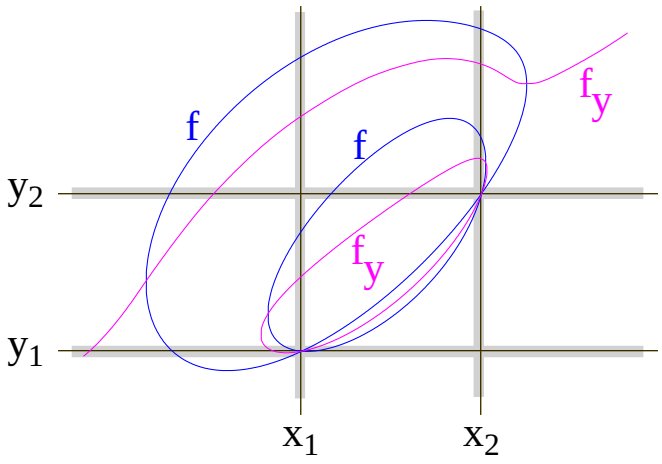

Fig. 13. We can explicitly compute the grid points $\textsc{Grid}(u, v)$

Of course this method of computing explicit solutions can also be applied to quadratic polynomials that are not rational but defined over a field extension $\mathbb{Q}(\sqrt{\rho})$ for some $\rho \in \mathbb{Q}$.

6 Singular points of cutcurves

In this section we will attack the problem of distinguishing in advance boxes that potentially contain singular points from the ones that cannot contain such a point. As an immediate result we will obtain that, in our special arrangements we obtain from projecting quadric intersection curves into the plane, every singular point can be determined using explicit solutions. In the following we will focus on singular points of cutcurves because, as we will see in the next section, silhouettecurves pose no problem.

As always we will assume that all pairs of quadrics we consider are generally aligned, squarefree, and coprime. Moreover we assume that all pairs of curves are well-behaved.

Every cutcurve $f = \mathrm{res}(p, q, z)$ is the result of projecting the intersection curve of two quadrics p and q into the plane. We assume p and q to be of total degree 2. Otherwise we are in the easy case that the cutcurve is a planar quadratic curve that can be treated the same way as a silhouettecurve.

If $(a, b) \in \mathbb{C}^2$ is a point on the cutcurve $f := \mathrm{res}(p, q, z)$, then general alignment of p and q guarantees the existence of a number $c \in \mathbb{C}$ with $p(a, b, c) = q(a, b, c) = 0$. We will show next that (a, b) is a singular point of f if and only if

(1) p and q share another common root $(a, b, c') \neq (a, b, c)$ which by Theorem 2 mathematically means $\mathrm{sres}_1(p, q, z)(a, b) = 0$
(2) or p and q intersect tangentially in (a, b, c) which can be expressed by $0 = (p_x q_z - p_z q_x)(a, b, c) = (p_y q_z - p_z q_y)(a, b, c) = (p_x q_y - p_y q_x)(a, b, c)$, remember Section 3.

For illustration of the first case have a look at the left picture of Figure 14. The spatial curve running on the ellipsoid is the intersection curve of the quadrics R and B in our overall example. It consists of two branches, one on the upper and one on the lower part of the ellipsoid R. The two branches are projected on top of each other causing two singular points, namely self-intersection points. The second case is illustrated in the right picture. The two ellipsoids have a tangential intersection in space. In this point already the spatial intersection curve has a singular point. This singular point will be projected into the plane.

Definition 10 *Let $f = res(p, q, z)$ be a cutcurve defined by the quadratic polynomials $p, q \in \mathbb{Q}[x, y, z]$. Let $(a, b) \in \mathbb{C}^2$ be a point on the cutcurve that originates from the intersection point (a, b, c) of p and q. If $sres_1(p, q, z)(a, b) = 0$, we call (a, b) a* top-bottom point. *If $0 = (p_x q_z - p_z q_x)(a, b, c) = (p_y q_z - p_z q_y)(a, b, c) = (p_x q_y - p_y q_x)(a, b, c)$, we call (a, b)* genuine.

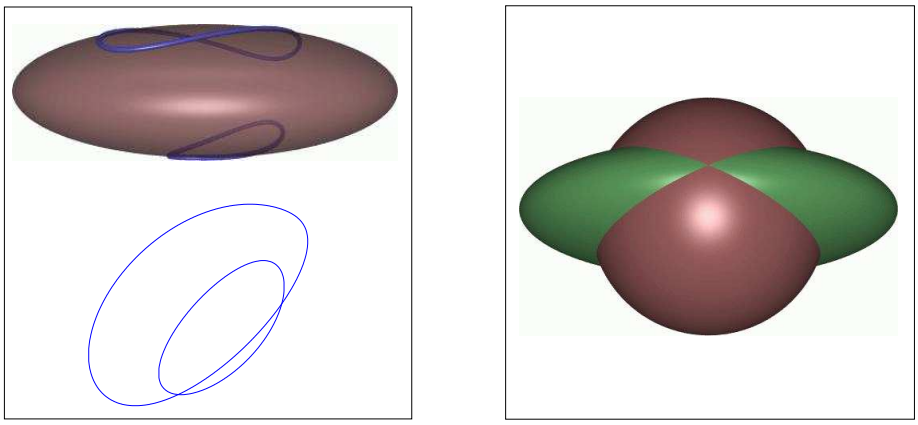

Fig. 14. Top-bottom points result from the projection. Genuine points are caused by tangential intersections in space.

Theorem 11 *Let f be a cutcurve that originates from two generally aligned quadrics p and q. The singular points of f are exactly the top-bottom and genuine points of f.*

PROOF. Without loss of generality let $(0,0)$ be a point of f which originates from the intersection point $(0,0,0)$ of p and q. The resultant computation is invariant under translation along the z-axis. A translation of p and q along the x- or y-axis just causes the same translation of the resultant.

Without loss of generality the polynomials p and q have the form $p = z^2+p_1z+p_0$ and $q = z^2 + q_1z + q_0$ where $p_i \in \mathbb{Q}[x,y]$ and $q_i \in \mathbb{Q}[x,y]$ are polynomials of degree at most $2-i$. Taking partial derivatives we obtain the equalities

$$\begin{aligned} p_0 &= p(x,y,0) \;\; =: p|_{z=0} & \qquad q_0 &= q(x,y,0) \;\; =: q|_{z=0} \\ p_1 &= p_z(x,y,0) =: p_z|_{z=0} & \qquad q_1 &= q_z(x,y,0) =: q_z|_{z=0}. \end{aligned}$$

Computing the resultant of p and q leads to the expression

$$f \;=\; \mathrm{res}(p,q,z) \;=\; (pq_z - p_zq)|_{z=0} \cdot \mathrm{sres}_1(p,q,z) + ((p-q)|_{z=0})^2.$$

For a polynomial $p \in \mathbb{C}[x,y,z]$ it is obvious that taking the partial derivative with respect to a variable $x \neq z$ and then substituting $z = 0$ is the same as first substituting $z = 0$ and then taking the partial derivative: $(p_x)|_{z=0} = (p|_{z=0})_x$. Due to our assumption we have $p(0,0,0) = q(0,0,0) = 0$. This leads to

$$\begin{aligned} f_x(0,0) &= ((p_xq_z - p_zq_x)|_{z=0} \cdot \mathrm{sres}_1(p,q,z))(0,0) \\ f_y(0,0) &= ((p_yq_z - p_zq_y)|_{z=0} \cdot \mathrm{sres}_1(p,q,z))(0,0). \end{aligned}$$

We immediately obtain that a top-bottom or genuine point of f is a singular point of f. For the other inclusion it remains to show the following: If $(0,0)$

is a singular point of f but not a top-bottom point, then also $0 = (p_x q_y - p_y q_x)(0,0,0)$.

If at least one of q_z or p_z does not vanish at $(0,0,0)$, without loss of generality q_z, this is easy to see: From $q_z(0,0,0) \neq 0$, $0 = (p_x q_z - p_z q_x)(0,0,0)$, and $0 = (p_y q_z - p_z q_y)(0,0,0)$ it follows $p_x(0,0,0) = \left(\frac{p_z}{q_z} q_x\right)(0,0,0)$ and $p_y(0,0,0) = \left(\frac{p_z}{q_z} q_y\right)(0,0,0)$. This leads to

$$(p_x q_y - p_y q_x)(0,0,0) \;=\; \left(\frac{p_z}{q_z} q_x q_y - \frac{p_z}{q_z} q_y q_x\right)(0,0,0) = 0.$$

Under our assumption that $(0,0)$ is not a top-bottom point, $p_z(0,0,0) = 0 = q_z(0,0,0)$ cannot occur. Otherwise $p(0,0,z)$ and $q(0,0,z)$ would have a common factor of degree 2, namely z^2, and therefore the first subresultant of p and q would vanish at $(0,0)$, leading to a contradiction. □

Projecting two intersection points in space on top of each other or projecting a tangential intersection point in space are exactly the events that cause singular points of a cutcurve. Of course, a singular point can be top-bottom as well as genuine.

6.1 Top-bottom points

We will prove that under our conditions of general alignment and squarefreeness a cutcurve f can have at most 2 top-bottom points. These 2 points can be determined using explicit computation as described in the previous section.

Theorem 12 *Let $f \in \mathbb{Q}[x,y]$ be a generally aligned and squarefree polynomial which defines a cutcurve. Then f can have at most 2 top-bottom points. Moreover, one can compute two at most quadratic polynomials $u_{tb} \in \mathbb{Q}[x]$ and $v_{tb} \in \mathbb{Q}[y]$ such that the top-bottom points lie on* $\textsc{Grid}(u_{tb}, v_{tb})$.

PROOF. Let $p, q \in \mathbb{Q}[x,y,z]$ be two quadratic polynomials and f the squarefree part of $\mathrm{res}(p,q,z)$. Without loss of generality we denote $p = z^2 + p_1 z + p_0$ and $q = z^2 + q_1 z + q_0$ where $p_i, q_i \in \mathbb{Q}[x,y]$ are polynomials of degree at most $2 - i$.

The first subresultant of p and q with respect to z is of the form $l := \mathrm{sres}_1(p,q,z) = q_1 - p_1$. That means l is a polynomial of degree at most 1. By definition, all intersection points of f and l are top-bottom and therefore by Theorem 11 singular points of f. The curve f is assumed to be squarefree. That means f has only finitely many singular points and because of that l

cannot be the zero-polynomial. If l is constant and non-zero, then there are no intersection points of f and l. In this case f has no top-bottom point and we are done.

So let us consider the case that l is a polynomial of total degree 1 defining a line. We assume without loss of generality that l is generally aligned. The resultant $\mathrm{res}(f, l, y) \in \mathbb{Q}[x]$ is not the zero polynomial (f is squarefree) and has degree at most 4. By Theorem 4 each root of the resultant $\mathrm{res}(f, l, y)$ has multiplicity ≥ 2 because it results from a singular point of f through which also l cuts and this by definition is a tangential intersection point of f and l. That means f and l intersect in at most 2 points. We finally conclude that the polynomial

$$u_{\mathrm{tb}} \;=\; \gcd(\mathrm{res}(f, l, y), \frac{d}{dx}\mathrm{res}(f, l, y))$$

has degree at most 2 and contains the x-coordinates of the top-bottom points. Analogously one can compute the polynomial v_{tb} using $\mathrm{res}(f, l, x)$. $\square$

6.2 Genuine points

Next we will consider genuine points. We will prove that their number is bounded by 4 and we will give the algorithmic ideas how to determine them.

Theorem 13 *Let $p, q \in \mathbb{Q}[x, y, z]$ be generally aligned with respect to z, coprime, and squarefree. Furthermore let f be generally aligned and the squarefree part of $res(p, q, z)$.*
The cutcurve defined by f can have at most 4 genuine points. If it has more than 2 genuine points, f consists of two distinct lines and another quadratic curve, all of them not necessarily rational.

PROOF. Let $A = (\alpha_x, \alpha_y, \alpha_z)$, $B = (\beta_x, \beta_y, \beta_z)$, and $C = (\gamma_x, \gamma_y, \gamma_z)$ be three distinct tangential intersection points of p and q. We will show that $f = \mathrm{res}(p, q, z)$ consists of two lines and another quadratic curve and has at most 4 genuine points.

We will first prove by contradiction that the three points A, B, and C cannot be collinear. So assume A, B, C to be collinear and let l be the line in space passing through them. We will prove that in this case each point on l is a tangential intersection point of p and q. That means f has infinitely many singular points, contradicting its squarefreeness.

For our investigations the location of l in space is not important. So we assume without loss of generality $A = (\alpha_x, 0, 0)$, $B = (\beta_x, 0, 0)$, and $C = (\gamma_x, 0, 0)$. The polynomials $p(x, 0, 0), q(x, 0, 0)$ have degree at most 2 and we know three

different roots, namely α_x, β_x, and γ_x. We conclude that $p(x,0,0)$ and $q(x,0,0)$ both are equal to the zero polynomial and therefore every point $D = (x,0,0)$ of l is an intersection point of p and q. We define $p_1(x) := (p_x q_y - p_y q_x)(x,0,0)$, $p_2(x) := (p_x q_z - p_z q_x)(x,0,0)$, and $p_3(x) := (p_y q_z - p_z q_y)(x,0,0)$, remember the definition of a tangential intersection point in Section 3. Every $p_i \in \mathbb{Q}[x]$ is the zero-polynomial because by construction p_i has degree at most 2 and we know the three different roots α_x, β_x, and γ_x. We derive that every point $D = (x,0,0)$ of l is a tangential intersection point of p and q.

We know that A, B, and C are not collinear and therefore there exists a uniquely defined plane h through these three points. Let us first consider the case that h is a factor of p or q. It cannot be a factor of both because by assumption p and q have disjoint factorizations. Assume without loss of generality $p = h \cdot \tilde{h}$. Then the spatial intersection curve c of p and q consists of two conics s and $\tilde{s}$ embedded on h and $\tilde{h}$, respectively. The points of c that cause genuine points in the plane are its singular ones. These are exactly the intersection points of s and $\tilde{s}$ plus the singular points of s plus the singular points of $\tilde{s}$.

There can be at most two intersection points of s and $\tilde{s}$: They lie on the line $l = \{(x,y,z) \in \mathbb{C}^3 | h(x,y,z) = \tilde{h}(x,y,z) = 0\}$ and we know that no 3 tangential intersection points of p and q are collinear. The curve f has 3 genuine points, so we conclude that at least one of s or $\tilde{s}$ must have a singular point. A planar quadratic curve has a singular point if and only if it consists of two (maybe complex) intersecting lines. We conclude that c consists of two intersecting lines and another quadratic curve and so does its projection f: $f = f_1 \cdot f_2$. The two quadratic curves f_1 and f_2 are not necessarily rational because they are the projected intersection curves of h and q and of $\tilde{h}$ and q, respectively, and h and $\tilde{h}$ are not necessarily rational. Furthermore we deduce that c has at most 4 singular points and therefore f has at most 4 genuine points.

The case we have not discussed so far is that the tangential intersection points A, B, and C of p and q are not collinear and h is neither a factor of p nor of q. Then the spatial intersection curves of h and p and of h and q are quadratic. So on h there are two quadratic planar curves that have 3 tangential intersection points. This can only happen if both curves are identical. Let us denote this curve by s. We conclude that there exists another spatial quadratic curve $\tilde{s}$ such that p and q intersect in s and $\tilde{s}$. Due to our assumption of f being squarefree we have $s \neq \tilde{s}$. That means there exists another plane $\tilde{h} \neq h$ such that s and $\tilde{s}$ are embedded on h and $\tilde{h}$, respectively. Let $r := h \cdot \tilde{h}$. By construction the spatial intersection curve of p and q equals the one of p and r. That means we have reduced this case to the previous one where h is a factor of one of the quadrics. $\square$

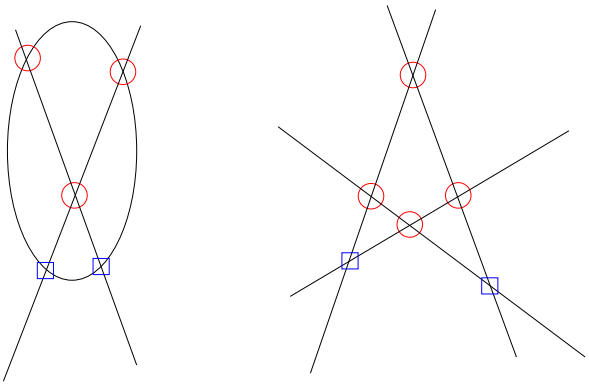

Fig. 15. The cutcurve consists of two intersecting lines and another conic. There are at least 3 genuine points, marked by the circles. The remaining singular points are top-bottom points and marked by the squares.

The statement of the theorem is that in most cases a squarefree cutcurve has at most 2 genuine points. Bad things only happen if the cutcurve consists of two lines and another quadratic curve as shown in Figure 15. A close look at the proof shows that in these cases the underlying spatial intersection curve consists of two lines and another quadratic spatial curve, each embedded in a plane. Let h and $\tilde{h}$ be these two planes and $r := h \cdot \tilde{h}$. At most 2 tangential intersection points lie on the intersection line l of h and $\tilde{h}$ and at most 2 do not.

It would be quite useful to know r because the line defined by the polynomial $\mathrm{res}(r, r_z, z)$ is the projection of l. With the help of this line the genuine points could be classified: the ones that lie on the line and the ones that do not. Each group has at most two members. As we did for top-bottom points we could factor the resultants on the x- and y-axis according to this distinction leading to quadratic polynomials as desired.

Theorem 14 *Let p and q be two quadrics in space.*

(1) If $f = res(p, q, z)$ consists of two lines and another quadratic curve not equal to two lines, then there exists a polynomial $r \in \mathbb{Q}[x, y, z]$ defining two planes such that $f = res(p, r, z)$.
(2) If $f = res(p, q, z)$ consists of four lines, then there exists a polynomial $r \in \mathbb{Q}(\sqrt{\rho})[x, y, z]$, for some $\rho \in \mathbb{Q}$, defining two planes such that $f = res(p, r, z)$.

In both cases we can compute r.

PROOF. We omit a detailed proof which can be found in [70]. This result is not new. It is part of the theory of quadric pencils and already discussed in [33]. Using the classification of intersection curves made there, one derives that in our first situation exactly one quadric r defining two planes is in the

quadric pencil of p and q. So necessarily r is rational. In the second case there are two such quadrics r_1 and r_2 in the pencil and because of this r_1 and r_2 are polynomials over a field extension $\mathbb{Q}(\sqrt{\rho})$. $\square$

7 Computing the planar arrangements

Now we have finished all preliminary considerations and we will prove our main theorem: Given a set of n input quadrics, we can exactly determine the event points in each of the n planar arrangements that arise from the projection phase.

Theorem 15 *Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be a set of trivariate quadratic polynomials. For $1 \leq i \leq n$ let $\mathcal{F}_i$ be the set of curves in the i-th planar arrangement we obtain from the projection phase: $\mathcal{F}_i = \{res(p_i, (p_i)_z, z)\} \cup \bigcup_{i \neq j} \{res(p_i, p_j, z)\}$. Let furthermore f and g be a pair of polynomials with either $f, g \in \mathcal{F}_i$ or $f \in \mathcal{F}_i$ and $g = f_y$.*

For f and g we can compute a set of k rational boxes that is in 1-1 correspondence to the set of real intersection points of the curves defined by f and g. The jth real intersection point (α_j, β_j) of f and g is the only one inside the jth box. Moreover we can determine whether the intersection point inside a box is transversal or tangential, whether the two curves cross each other, and whether it is a singular point of one of the curves.

In the following we will prove the theorem: for all curves $f \in \mathcal{F}_i$ in a planar arrangement the intersection points of f and $g = f_y$ and the ones of $f \in \mathcal{F}_i$ and $g \in \mathcal{F}_i$ can be determined. We do this either by applying box hit counting arguments or by partially factoring the bi-resultant of f and g and computing explicit solutions. The second method is used to compute non-singular tangential intersections that lead to roots of multiplicity greater than 2 in the resultant and to determine singular points. For these points box hit counting does not always reliably lead to a correct result. According to the distinction of the curves in $\mathcal{F}_i$ into silhouettecurve and cutcurves there are four different kinds of pairs of curves f and g that have to be considered, each treated in one subsection.

We assume without loss of generality that all pairs of curves are well-behaved. That means all curves are generally aligned, squarefree, coprime, and in general relation. The way these conditions are tested and realized is described in the next section.

Let in the following $X := \text{res}(f, g, y)$ and $Y := \text{res}(f, g, x)$.

7.1 f is the silhouettecurve and $g = f_y$

If f is the silhouettecurve and $g = f_y$, then we know that both resultants X and Y have degree at most 2. That enables us to compute explicit solutions. Consider also the left picture in Figure 16.

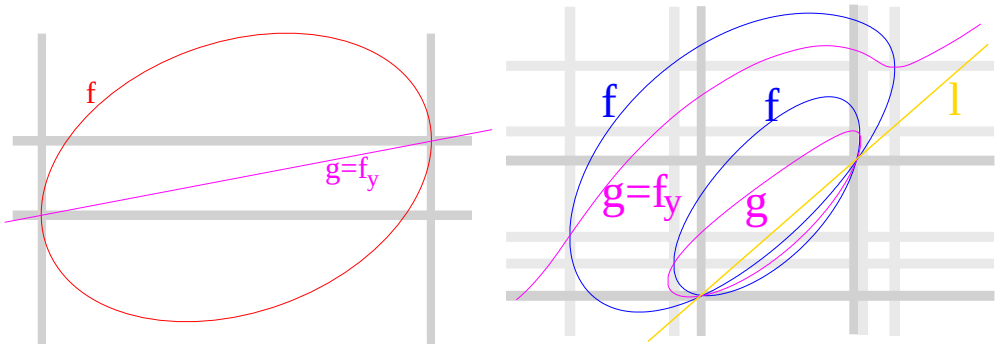

Fig. 16. For the extreme and singular points of the silhouettecurve f (left picture) we can compute explicit solutions. If f is a cutcurve (right picture) we do a multiplicity bi-factorization. Transversal intersections lie inside light grey boxes, tangential intersection points lie inside dark grey boxes. The line l cuts through the top-bottom points.

7.2 f is a cutcurve and $g = f_y$

In the case f is a cutcurve and $g = f_y$ is its partial derivative, 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 multiplicity 1, i.e. all transversal intersection points, lie on $\textsc{Grid}(u_1, v_1)$. All intersection points with multiplicity ≥ 2, i.e. all tangential intersection points, lie on $\textsc{Grid}(u_2, v_2)$. For illustration have a look at the right picture of Figure 16.

The light grey boxes around $\textsc{Grid}(u_1, v_1)$ can be handled with simple box hit counting as shown in Chapter 5.

The dark grey boxes around $\textsc{Grid}(u_2, v_2)$ are the candidate boxes for tangential intersections. Unfortunately, a tangential intersection point of f and $g = f_y$ is not necessarily a singular point of f. By definition, an intersection point $(a, b) \in \mathbb{C}^2$ of f and g is tangential if and only if $(f_x g_y - f_y g_x)(a, b) = (f_x f_{yy} - f_y f_{xy})(a, b) = (f_x f_{yy})(a, b) = 0$. The last but one equality holds because (a, b) is a point of $g = f_y$ and therefore $f_y(a, b) = 0$. We conclude that there are two kinds of tangential intersection points:

(1) Singular points of f, that means $f_x(a, b) = 0$, and

(2) non-singular points of f with $f_{yy}(a,b) = 0$. We call these points *vertical flat points*. A vertical flat point is either a vertical turning point of f ($f_{yyy}(a,b) \neq 0$) or an extreme point of f ($f_{yyy}(a,b) = 0$).

The existence of vertical flat points is not caused by the geometry of our curves but by our chosen coordinate system. In the remaining part of this subsection we will provide an algorithm that factors (u_2, v_2) in absence of vertical flat points such that each of its bi-factors (u_f, v_f) is of degree at most 2. If one of the resulting bi-factors has a higher degree, we know that this is caused by vertical flat points. In this case we shear f and g to get rid of the situation and restart, see also Section 8.

For every quadratic bi-factor (u_f, v_f) we can compute explicit solutions. Of course it can happen that a point (a,b) we computed explicitly this way is a vertical flat point of f instead of a singular point. In order to recognize this, we substitute (x,y) by (a,b) in f_x and explicitly test $f_x(a,b)$ for zero. If $f_x(a,b) = 0$, (a,b) is a singular point of f. Otherwise it is a vertical flat point and we explicitly test $f_{yyy}(a,b)$ for zero in order to distinguish vertical turning points of f from extreme points of f.

We promised to provide an algorithm that factors (u_2, v_2) into bi-factors which are at most quadratic. Using gcd-computation we first make u_2 and v_2 squarefree. According to Theorem 12 we compute the first bi-factor (u_{tb}, v_{tb}) of (u_2, v_2) splitting off the top-bottom points of f: $(u_2, v_2) = (u_{tb}, v_{tb}) \cdot (u_g, v_g)$. Both polynomials in (u_{tb}, v_{tb}) have degree at most 2. In our example all tangential intersection points are top-bottom. The yellow line l, which is the first subresultant of the involved spatial quadrics, cuts through them, see Figure 16.

In the case that u_g as well as v_g are at most quadratic polynomials, as in our example, everything is fine and we compute explicit solutions also for (u_g, v_g).

Now consider the case that u_g or v_g or both have degree > 2. We assumed that all curves we consider are squarefree. We conclude according to Theorem 13 that (in the absence of vertical flat points) the cutcurve consists of two intersecting lines and a conic. Let p and q be the quadrics with $f = \mathrm{res}(p, q, z)$. We compute the quadric pencil of p and q and look for a pair of planes h and $\tilde{h}$ in this pencil. If we cannot find such a pair of planes, we know that we are in degenerate situation caused by vertical flat points and we proceed as described before. Otherwise we can compute a polynomial $r = h \cdot \tilde{h}$ with either $r \in \mathbb{Q}[x,y,z]$ or $r \in \mathbb{Q}(\sqrt{\rho})[x,y,z]$, $\rho \in \mathbb{Q}$. In the discussion after the proof of Theorem 13 we noticed that at most 2 genuine points lie on the line $l = \mathrm{res}(r, r_z, z)$ and at most 2 genuine points do not lie on l. As in the case of top-bottom points, we use l to factor off these points: $(u_g, v_g) = (u_{g1}, v_{g1}) \cdot (u_{g2}, v_{g2})$. In the absence of vertical flat points all polynomials in this partial bi-factorization are at most quadratic.

7.3 f is the silhouettecurve and g is a cutcurve

Let f be the silhouettecurve and g be a cutcurve. The polynomials X and Y have degree at most 8. This implies that there are at most two roots of multiplicity ≥ 3. We compute a bi-factorization $(u_1, v_1) \cdot (u_2, v_2)^2 \cdot (u_3, v_3)$ of (X, Y) such that u_1, v_1 contain all simple roots, u_2, v_2 all roots of multiplicity 2, and u_3, v_3 all roots of multiplicity ≥ 3.

All transversal intersections points lie on $\text{GRID}(u_1, v_1)$ and can be solved with simple box hit counting.

The ones lying on $\text{GRID}(u_2, v_2)$ either are singular points of f or g or they are transversal intersections of the Jacobi curve $h = f_x g_y - f_y g_x$ and f and of h and g, according to Theorem 7.

We would like to apply extended box hit counting to these boxes, but first we have to be sure that there is no singular point inside the tested box. In the last section we have shown how to compute quadratic bi-factors (u_{tb}, v_{tb}), (u_{g1}, v_{g1}), and (u_{g2}, v_{g2}) for all singular points and compute explicit solutions. If any of these bi-factors has a common bi-factor with (u_2, v_2), we split off this common bi-factor. What remains is a bi-factor (u_2', v_2') with only non-singular tangential intersections of f and g on its grid. We apply extended box hit counting to the boxes defined by u_2' and v_2'.

Because of the degree of X and Y the bi-polynomial (u_3, v_3) has at most two different roots. With the help of gcd-computation we compute two at most quadratic polynomial u_3' and v_3' containing them and apply explicit solutions.

7.4 f and g both are cutcurves

Let f and g both be cutcurves. They are the result of intersecting a quadric p with other quadrics q and r, respectively. Each cutcurve has algebraic degree ≤ 4 and therefore the polynomials X and Y have degree at most 16. We would like to compute a bi-factorization $(X, Y) = (u_s, v_s) \cdot (u_a, v_a)$ such that all polynomials u_s, v_s, u_a, v_a have degree at most 8 and the polynomials u_s and u_a and the polynomials v_s and v_a share no common factor.

Let us assume we have such a bi-factorization. Then for (u_s, v_s) and for (u_a, v_a) we can proceed exactly like in the case of a silhouettecurve and a cutcurve described in the previous section. We perform a bi-factorization according to the multiplicities 1, 2, and ≥ 3. Again a polynomial of degree 8 can have at most two roots of multiplicity ≥ 3. According to the assumption that u_s, u_a and v_s, v_a have no common roots, the boxes belonging to multiplicity

1 can be handled with simple box hit counting. The boxes defined by roots of multiplicity 2 are solved with extended box hit counting, after we split off the singular points of f or g. The remaining bi-factor belonging to roots of multiplicity ≥ 3 defines at most 4 grid points that are solvable with explicit solutions.

What remains to do is to establish the bi-factorization $(X, Y) = (u_s, v_s) \cdot (u_a, v_a)$ such that each involved polynomial has degree ≤ 8. As for singular points, we can distinguish two different types of intersection points for f and g: *spatial* and *artificial*.

Definition 16 *Let $(a, b) \in \mathbb{C}^2$ be an intersection point of two curves $f = res(p, q, z)$ and $g = res(p, r, z)$ with p, q, r being quadratic trivariate polynomials. We call (a, b)* spatial*, if for a root c of $p(a, b, z) \in \mathbb{C}[z]$ we have $p(a, b, c) = q(a, b, c) = r(a, b, c) = 0$. If $p(a, b, z) \in \mathbb{C}[z]$ has the two roots c and c' and it holds $p(a, b, c) = q(a, b, c) = 0$ and $p(a, b, c') = r(a, b, c') = 0$, then we call (a, b)* artificial*.*

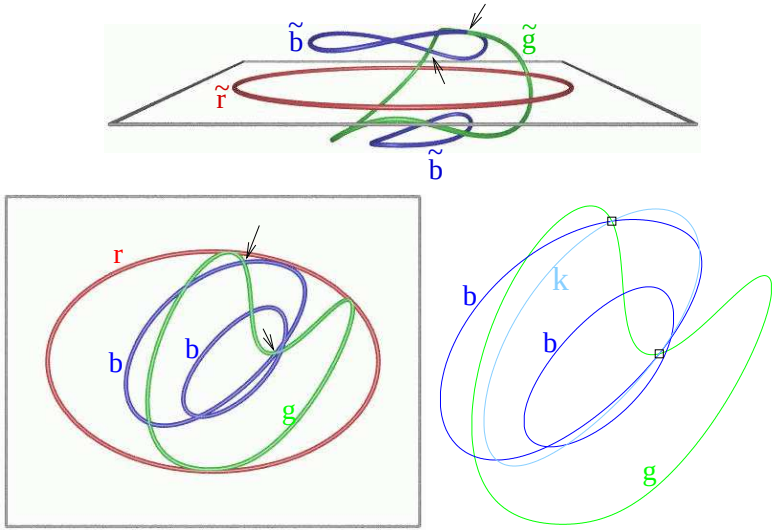

Fig. 17. Projections of common intersection points of the curves $\tilde{b}$ and $\tilde{g}$ are called spatial.

Spatial points are projected common intersection points of p, q, and r. Artificial points are a result of the projection phase. For illustration have a look at Figure 17. Intersection points of the spatial curves $\tilde{b}$ and $\tilde{g}$ are common points of the quadrics R, B, and G in our permanent example. There are two such intersection points, marked by the arrows. Projecting both curves into the plane results in the cutcurve b and g that have 6 real intersection points: 2 spatial and 4 artificial. The curve k is the projection of the intersection curve of the ellipsoids B and G. By definition we know that it cuts through the spatial points. Of course it can happen that points are both spatial and artificial.

Theorem 17 *Two cutcurves have at most 8 spatial and at most 8 artificial intersections, counted with multiplicities.*

PROOF. Let as before $f = \text{res}(p, q, z)$ and $g = \text{res}(p, r, z)$ be the cutcurves. The bound for their spatial intersections immediately follows by the theorem of Bézout. Three quadrics in space can have at most 8 discrete common intersection points, counted with multiplicities.

For proving the second bound we proceed as follows: We mirror q parallel to the z-axis at the plane $p_z = 0$. Without loss of generality let the polynomial p be of the form $p = z^2 + p_1 z + p_0$ with $p_i \in \mathbb{Q}[x, y]$ of degree $2 - i$, $i = 1, 0$. The function $f : \mathbb{C}^3 \to \mathbb{C}^3$, $f(a, b, c) = (a, b, -p_1(a, b) - c)$ mirrors points (a, b, c) vertically at the plane $p_z = 2z + p_1(x, y)$. If we apply this function to every point of a quadric $q = z^2 + q_1 z + q_0$ this leads to the quadratic polynomial $\tilde{q} = z^2 + (2p_1 - q_1)z + (p_1^2 - p_1 q_1 + q_0)$.

It is easy to verify that $\text{res}(p, \tilde{q}, z) = \text{res}(p, q, z)$. That means all intersection points of p and $\tilde{q}$ have the same (x, y)-coordinates as the intersection points of p and q. But the intersection points of p and q that lie on the top of p now lie on its bottom and vice versa. So the spatial and artificial intersections have changed place and we can again apply the theorem of Bézout to p, r and $\tilde{q}$. □

Now we know that there are at most 8 spatial and artificial intersection points, we would like to compute a bi-factorization $(u_s, v_s)\cdot(u_a, v_a)$ of (X, Y) according to this distinction. We want the roots of u_s to be the x-coordinates of common intersection points of p, q, and r. One way would be to additionally compute the resultant $k = \text{res}(q, r, z)$ and perform a greatest common divisor computation between X, $\text{res}(f, k, y)$, and $\text{res}(g, k, y)$. Caused by the projection from space to the plane it can happen that k cuts through an artificial intersection point of f and g. Then the x-coordinate of this artificial point would be a root of u_s, contradicting our goal. This would not disturb our following algorithm as long as the degree of u_s would still be at most 8. Otherwise, similar to the methods described in the next section, we could shear the spatial arrangement in order to remove this effect.

An alternative way to compute u_s would be to use the results of [19], [29]. There a method for computing u_s directly from the spatial quadrics p, q, and r with the help of multivariate resultants is provided.

8 Establishing the generality assumptions

In the previous sections we made assumptions on the location of the quadrics in space and the curves in the plane in order to simplify the argumentation: general alignment and general relation. We will present a method how to detect the lack of the conditions and how to establish them with a random shear.

8.1 General alignment

Let $P \subset \mathbb{Q}[x, y, z]$ be a set of n quadrics. We want each trivariate polynomial to have a constant non-zero coefficient of z^2. This of course is easy to test by just examining the coefficient of z^2 for each $p \in P$. If the coefficient is non-constant for at least one p, we will shear all quadrics. The main idea of a shear method is described in [60].

We randomly choose a rational vector $u = (u_1, u_2) \in \mathbb{Q}^2$ and consider the shear function $\phi(x, y, z) = (x + u_1 \cdot z, y + u_2 \cdot z, z)$. We compute $p \circ \phi$ for each $p \in P$. For nearly all choices of (u_1, u_2) we obtain generally aligned quadrics. The geometry of the spatial arrangement is not effected by the shear. Intersection points of quadrics remain intersection points. They only change their x- and y-coordinate. The drawback is that a shear causes a larger coefficient size. This can have an impact on all following resultant and root isolation computations. One way of choosing the shear parameter would be to bound the number r of forbidden directions and then randomly choose the shear parameter from a range $-r \ldots r$. With this strategy the probability of choosing a bad direction would be at most $1/2$. But as already stated in [27] this approach overestimates the failure probability in almost all cases. The strategy proposed and implemented there is to start with a small parameter range that does not increase the coefficient size too much and increase it depending on the number of past failures.

We also want to have the property that all bivariate polynomials we consider have a constant non-zero leading coefficient with respect to each variable. Let F be the set of bivariate polynomials defining the curves in a planar arrangement. Let us look at the polynomials in F as polynomials in the variable y. Like in the case for the trivariate polynomials, we randomly choose a rational number $v \in \mathbb{Q}$. Applying the affine transformation $\psi(x, y) = (x + v \cdot y, y)$ to each polynomial $f \in F$ will result in a set of polynomials that have a constant leading coefficient of y with very high probability.

We also want to have a constant leading coefficient of x. So in the same way, we randomly choose a rational number w and apply the shear $\tilde{\psi}(x, y) = (x, y +$

$w \cdot x$) to each polynomial. It is easy to see that the constant leading coefficient of y is not effected by this shear and so the second shear does not destroy the effect of the first shear.

8.2 General relation of two planar curves

In our algorithm we want each pair of curves f and g to be in general relation. Remember that f and g are defined to be in general relation with respect to x or y if there are no two common roots with the same x- or y-value, respectively. As vertical asymptotes, common coordinates of intersection points are not intrinsic to the two curves. We can avoid it by choosing a different direction of projection or by equivalently shearing the curves.

Let us look how to test general relation with respect to x. The test for y is symmetric. A similar algorithm to the one we we will explain is described in [63].

For the two curves f and g we look at their sheared versions at the time v:

$$f^v(x,y) := f(x+vy,y) \quad , \quad g^v(x,y) := g(x+vy,y).$$

Of course we can interpret f^v and g^v as polynomials in $\mathbb{Q}[x,y,v]$. The resultant $\mathrm{res}(f^v,g^v,y) \in \mathbb{Q}[x,v]$ defines an arrangement of lines, the x-coordinates of the intersection points of f and g walk along when we change the shear parameter v, see Figure 18. That means the resultant factors over $\mathbb{C}$ in linear parts $l_1,\ldots,l_k \in \mathbb{C}[x,v]$ with k being an upper bound on the number of intersection points of f and g: $\mathrm{res}(f^v,g^v,y) = l_1^{i_i} \cdot \ldots \cdot l_k^{i_k}$.

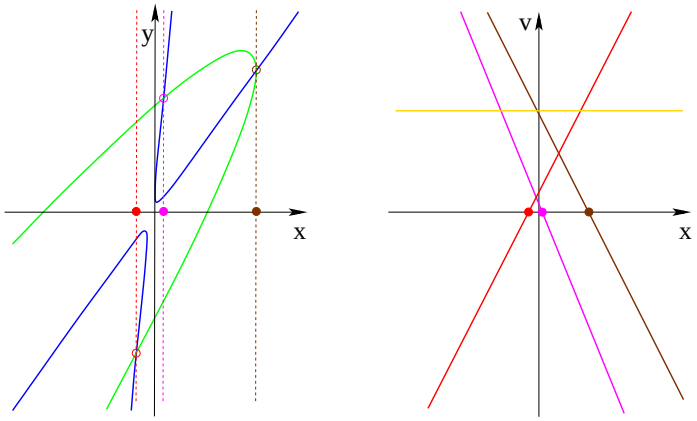

Fig. 18. The x-coordinates of intersection points of two curves change linearly when we shear.

During the shear each intersection point $(a,b) \in \mathbb{C}^2$ of f and g moves along one of the lines l_j, $1 \le j \le k$. Each intersection point moves along a different line. So the two curves $f = f^0$ and $g = g^0$ have no two intersection points

with common x-coordinates if and only if there is no intersection point of two of the lines on the x-axis. This is equivalent to the statement that there are no factors l_a and l_b in the complex factorization of $\text{res}(f^v, g^v, y)$, $a \neq b$, with $l_a(x, 0) = l_b(x, 0)$. The polynomials f and g are both generally aligned and because of that $(\text{res}(f^v, g^v, y))|_{v=0} = \text{res}(f, g, y)$. There is an intersection point on the x-axis if and only if substituting $v = 0$ in the factorization of the resultant $(l_1^{i_i} \cdot \ldots \cdot l_k^{i_k})|_{v=0}$ differs from the factorization over $\mathbb{C}$ of $\text{res}(f, g, y)$.

In practice we do not want to perform a complex factorization of $\text{res}(f^v, g^v, y)$. But we can compute its multiplicity factorization over $\mathbb{Q}$ analogously to the one for univariate polynomials and then substitute $v = 0$. Here only gcd-computations are performed and they can also be realized for bivariate polynomials using pseudo-division [41]. And of course we can do a multiplicity factorization of the univariate polynomial $\text{res}(f, g, y)$. With our previous remarks it is easy to see that f and g are in general relation if and only if both rational factorizations are equal.

9 Experimental results and outlook

We have developed a method for computing a cell in an arrangement of quadric surfaces. It uses exact algebraic computation and provides the correct mathematical result in every case, even in a degenerate one.

We claimed that our theoretical results for computing arrangements of quadric surfaces promise a good performance in practice. In order to justify our statement, we made some experiments in implementing and testing our ideas. Until now we have not mentioned the asymptotic time-complexity of our algorithm which is $O(n^3 \log n)$ for computing the n planar arrangements. Considering the subsequent running times, the classical method of analyzing the running time behavior for large n does not seem appropriate here. At this stage of research the main problem when computing with curved objects is to realize the algebraic predicates in an efficient way because these are the ones that influence the running time most.
An approach of obvious importance to measure the complexity of our algorithm would be to analyze its bit-complexity. However, a complete worst-case analysis is impractical, see the number of case distinctions, and furthermore, we expect no promising result from the known separation bounds that we would need to apply for the (cascaded) root isolations. Our approach is not tied to the worst case. Our methods benefit whenever a particular instance does not require the isolating intervals to approach the separation bounds limit but can stop earlier. Not only does the iteration stop earlier, e.g., in the Uspensky method, but also the bit complexity of the interval boundaries becomes smaller and subsequent steps are faster.

A first rather prototypical implementation determines event points in the planar arrangements induced by three quadrics. It uses the basic data types of LEDA [47] and the rational polynomial class as well as the resultant and Sturm sequence computation of MAPC [42].

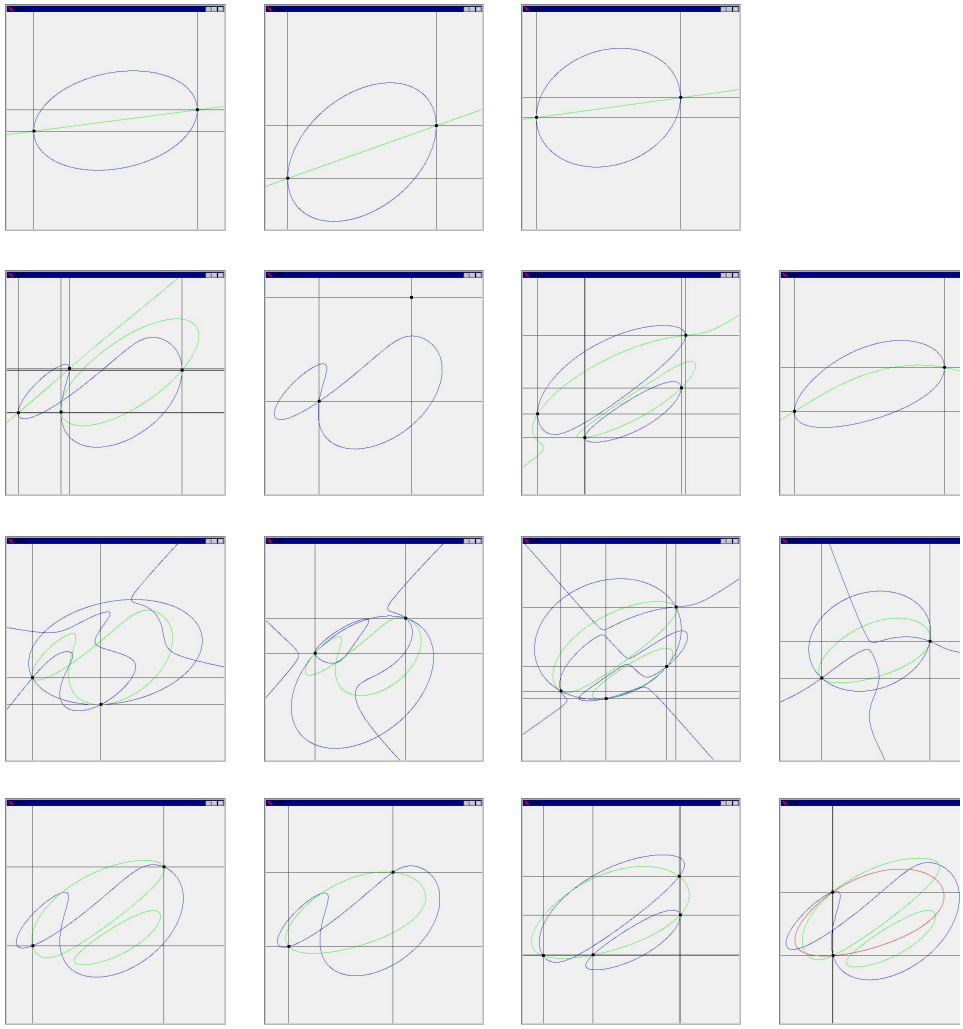

Fig. 19. Screen shots for computing all event points in the three planar arrangements for the quadrics p, q, and r.

Consider the screen shots in Figure 19 we made from the output for computing the event points for the three input quadrics

$$\begin{aligned} p(x,y,z) = {} & 7216x^2 - 11022xy - 12220xz + 15624y^2 + 15168yz \\ & +11186z^2 - 1000 \\ q(x,y,z) = {} & 4854x^2 - 3560xy + 4468xz + 658x + 5040y^2 + 32yz + 1914y \\ & +10244z^2 + 3242z - 536 \\ r(x,y,z) = {} & 8877x^2 - 10488xy + 9754xz + 1280x + 16219y^2 - 16282yz \\ & -808y + 10152z^2 - 1118z - 796. \end{aligned}$$

The ellipses in the first row are the silhouettecurves of the input ellipsoids p, q, and r. The line cutting through the extreme points is the respective partial derivative. All extreme points are determined correctly and marked by small boxes. In the second row first the extreme and then the singular points are determined for each of the three cutcurves. Again the additional curves are the partial derivatives. In the third row all tangential intersection points between silhouettecurves and cutcurves are computed. In each picture there is a third curve: the Jacobi-curve. In the last row we first compute all artificial intersection points of pairs of cutcurves. The spatial points are the common intersection points of all three cutcurves. They are computed in the last picture.

The running time of our implementation for this special example on an Intel Pentium 700 is about 18 seconds. Of course the running time mainly depends on the number of decimal digits of the three input quadrics as can be seen in the following table:

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

The only mathematical tools that are used during the calculation are resultants and subresultants, root separation, gcd of univariate and bivariate polynomials, and solving quadratic univariate polynomials. The size of the coefficients of the polynomials has a great impact on the behavior of all these computations. In our example about half of the running time is spent on computing all necessary resultants. Isolating the real roots on the coordinate axes with Uspensky's algorithm is quite fast. The rest of the time is needed to test the more than 100 boxes for intersection points.

The first experimental implementation was just designed as a proof of concept of our ideas. The resulting running times were promising enough to further work on our approach. Very recently Eric Berberich [10] finished a second implementation, see also [12]. This implementation is, although still ongoing work, much more elaborated than the first prototypical one. It is based on the Exacus software library [32] which provides all necessary mathematical tools mentioned above. The main ideas of the underlying algorithm for computing the event points, for example introducing the Jacobi curve to determine tangential intersections and computing explicit solutions for the singular points, are exactly the ones developed in this paper. It only differs and is improved regarding the combinatorial approach. Based on the work and the implementation in Exacus of computing arrangements of cubic curves by Eigenwillig [27] it directly performs a modified Bentley-Ottmann sweep-line algorithm for the planar arrangements instead of inspecting quadratically many boxes. Besides computing the projected planar arrangements it additionally performs

the step described in Section 4 of recovering the spatial information. It divides each planar arrangement into two subarrangements, one running on the upper and one on the lower part of the underlying quadric p. So at the end we have the full information about the arrangement of the spatial intersection curves on the surface of p. For example have a look at Figure 20.

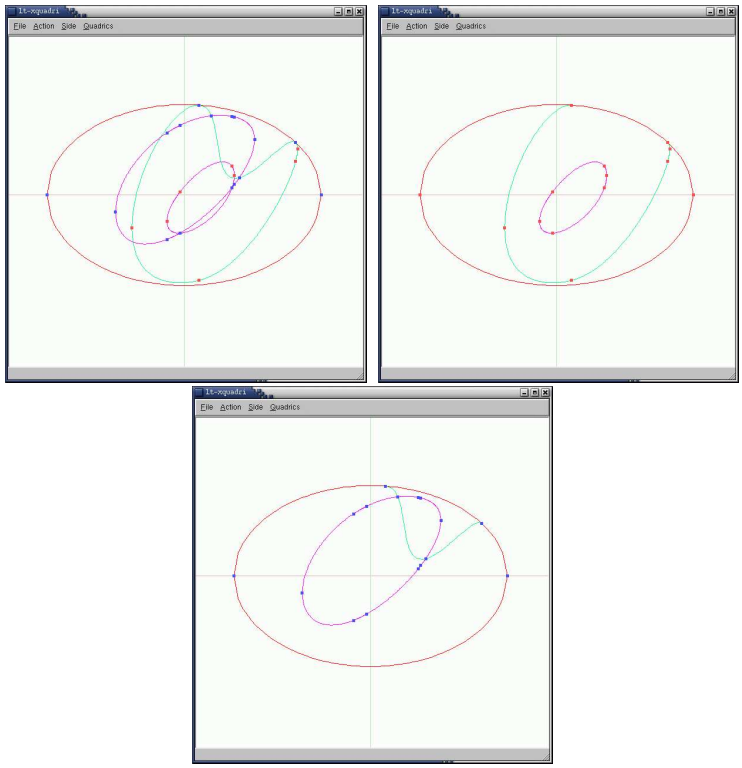

Fig. 20. The arrangement of our overall example and split into upper and lower part

We only want to mention some of the benchmarks obtained for this second implementation. For a detailed discussion consider [10]. The running times are measured on a Pentium III Mobile 800. The used arithmetic number types are the ones provided by LEDA [47]. The input tested with our first experimental program now leads to the following running times:

number of digits	5	10	15	20	25	30
running time in seconds	1.1	2.7	5.0	7.8	12.1	16.1

One can observe that the running times have decreased by more than a factor of 10 which does not only depend on the slightly newer computer. We want to mention two additional benchmarks. The first one is concerning random quadrics. The random quadrics have a bit size of 70. They are interpolated through 9 points the coordinates of which are integers chosen from the range $[-48, 48]$. In the table below one can see the increase of the running time depending on n together with the increase of the vertices and edges that have to be computed for all planar arrangements. One can observe that already for

$n = 12$ the latter is really large.

number of quadrics	4	8	12	16
number of vertices	474	2552	7544	13592
number of edges	1032	6702	21980	42168
running time in seconds	20.7	148.0	457.4	876.2

Also degenerate situations, for example where more than three quadrics intersect in a common point and share the same tangential planes, were systematically created and tested. The degenerate quadrics in our benchmark again have a bit size of 70 and the coordinates of the interpolation points are chosen from the range $[-48, 48]$. All n quadrics are chosen in such a way that their intersection curves intersect at common points with multiplicity one, two or three.

number of quadrics	4	8	12
number of vertices	378	2330	5594
number of edges	836	5956	15480
running time in seconds	35.8	293.7	1138.0

The implementations show that our algorithm is a first and important step towards exact and efficient computation of arrangements of curves and surfaces. Until now we have made no special efforts to optimize the running time of our implementation. Making use of filtering techniques and applying more sophisticated algebraic methods will surely lead to a better performance. Above that, of course, there is still some work in practical as well as in theoretical sense. So far our implementation only determines the arrangements of intersection curves on the surface of each quadric. Part of our future work will be to combine these results in order to describe a single cell or the overall arrangement of the quadrics in space.

Until now we do not have any results concerning the algebraic degree of our predicates. We want to investigate this in the future although it is not quite clear whether this will lead to satisfying results due to the large number of case distinctions.

Our approach provides an efficient and exact algorithm for computing a cell in an arrangement of quadric surfaces, even in degenerate cases. It is general in the sense that it could be applied to every kind of spatial surfaces defined by rational polynomials. So another topic of our future research will be to extend our approach to more general algebraic surfaces.

Acknowledgements

The authors thank Raimund Seidel for useful discussions and suggestions and Michael Hemmer for providing the spatial images he rendered for the video [36]. We want to express our special gratitude to our anonymous referees for their valuable comments which have helped a lot to improve the paper.

References

[1] S. Abhyankar and C. Bajaj. Computations with algebraic curves. In *Proc. Internat. Sympos. on Symbolic and Algebraic Computation*, volume 358 of *Lecture Notes Comput. Sci.*, pages 279–284. Springer-Verlag, 1989.

[2] P. K. Agarwal and M. Sharir. Arrangements and their applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[3] S. Arnborg and H. Feng. Algebraic decomposition of regular curves. *J. Symbolic Comput.*, 15(1):131–140, 1988.

[4] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM J. Comput.*, 13(4):865–877, 1984.

[5] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition II: The adjacency algorithm for the plane. *SIAM J. Comput.*, 13(4):878–889, 1984.

[6] D. S. Arnon, G. E. Collins, and S. McCallum. An adjacency algorithm for cylindrical algebraic decomposition in three-dimensional space. *J. Symbolic Comput.*, 5(1–2):163–187, 1988.

[7] D. S. Arnon and S. McCallum. A polynomial time algorithm for the topological type of a real algebraic curve. *J. Symbolic Comput.*, 5:213–236, 1988.

[8] C. Bajaj and M. S. Kim. Convex hull of objects bounded by algebraic curves. *Algorithmica*, 6:533–553, 1991.

[9] J. L. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28:643–647, 1979.

[10] E. Berberich. *Exact Arrangements of Quadric Intersection Curves.* Universität des Saarlandes, Saarbrücken, 2004. Master Thesis.

[11] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, and E. Schömer. A computational basis for conic arcs and boolean operations on conic polygons. In *ESA 2002, Lecture Notes in Computer Science*, pages 174–186, 2002.

[12] E. Berberich, M. Hemmer, L. Kettner, E. Schömer, and N. Wolpert. An exact, complete and efficient implementation for computing planar maps of quadric intersection curves. In *Proc. 21st Annu. ACM Sympos. Comput. Geom.*, pages 99–106, 2005.

[13] J.-D. Boissonat and J. Snoeyink. Efficient algorithms for line and curve segment intersection using restricted predicates. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 370–379, 1999.

[14] J. Boissonnat and F. P. Preparata. Robust plane sweep for intersecting segments. *SIAM Journal on Computing*, 23:1401–1421, 2000.

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

[16] J. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1987.

[17] T. Chan. Reporting curve segment intersection using restricted predicates. *Computational Geometry*, 16:245–256, 2000.

[18] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoretical Computer Science*, 84:77–105, 1991.

[19] 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.

[20] 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.

[21] 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.

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

[23] T. Culver, J. Keyser, M. Foskey, , S. Krishnan, and D. Manocha. Esolid - a system for exact boundary evaluation. *Computer-Aided Design (Special Issue on Solid Modeling)*, 36, 2003.

[24] O. Devillers, A. Fronville, B. Mourrain, and M. Teillaud. Exact predicates for circle arcs arrangements. In *Proc. 16th Annu. ACM Symp. Comput. Geom.*, 2000.

[25] D. P. Dobkin and D. L. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5:421–457, 1990.

[26] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. A new algorithm for the robust intersection of two general quadrics. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 246–255, 2003.

[27] A. Eigenwillig. *Exact Arrangement Computation of Cubic Curves*. Universität des Saarlandes, Saarbrücken, 2003. Master Thesis.

[28] A. Eigenwillig, L. Kettner, E. Schömer, and N. Wolpert. Complete, exact, and efficient computations with cubic curves. In *Proc. 20th Annu. ACM Symp. Comput. Geom.*, pages 409–418, 2004.

[29] D. Eisenbud and F.-O. Schreyer. Resultants and chow forms via exterior syzygies. *Journal of the American Mathematical Society*, 16:537–579, 2003.

[30] I. Emiris, A. Kakargias, S. Pion, M. Teillaud, and E. Tsigaridas. Towards an open curved kernel. In *Proc. 20th Annu. ACM Symp. Comput. Geom.*, pages 438–446, 2004.

[31] I. Emiris and E. Tsigaridas. Comparing real algebraic numbers of small degree. In *ESA 2004, Lecture Notes in Computer Science*, pages 652–663, 2004.

[32] The EXACUS project. www.mpi-sb.mpg.de/projects/EXACUS/.

[33] R. T. Farouki, C. A. Neff, and M. A. O'Connor. Automatic parsing of degenerate quadric-surface intersections. *ACM Trans. Graph.*, 8:174–203, 1989.

[34] E. Flato, D. Halperin, I. Hanniel, and O. Nechushtan. The design and implementation of planar maps in cgal. In *Proceedings of the 3rd Workshop on Algorithm Engineering*, Lecture Notes Comput. Sci., pages 154–168, 1999.

[35] N. Geismann, M. Hemmer, and E. Schömer. Computing a 3-dimensional cell in an arrangement of quadrics: Exactly and actually! In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 264–271, 2001.

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

[37] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, second edition*, chapter 24, pages 529–562. CRC Press LLC, Boca Raton, FL, 2004.

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

[39] H. Hong. Efficient method for analyzing topology of plane real algebraic curves. In *Proceedings of IMACS-SC 93, (Lille, France)*, June 1993.

[40] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A core library for robust numeric and geometric computation. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 351–359, 1999.

[41] G. L. Keith O. Geddes, Stephen R. Czapor. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, 1992.

[42] 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.

[43] J. C. Keyser. *Exact boundary evaluation for curved solids.* Univ. of North Carolina at Chapel Hill, Chapel Hill, 2000. Ph.D. dissertation.

[44] S. Lazard, L. M. Penaranda, and S. Petitjean. Intersecting quadrics: An efficient and exact implementation. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, pages 419–428, 2004.

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

[46] J. Levin. Mathematical models for determining the intersections of quadric surfaces. *Comput. Graph. Image Process.*, 11:73–87, 1979.

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

[48] J. Miller and R. Goldman. Combining algebraic rigor with geometric robustness for the detection and calculation of conic sections in the intersection of two quadric surfaces. In *Proceedings of the Symposium on Solid Modeling and Applications*, pages 221–231, 1991.

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

[50] J. R. Miller and R. Goldman. Geometric algorithms for detecting and calculating all conic sections in the intersection of any two natural quadric surfaces. *Graphical Models and Image Processing*, 57:55–66, 1995.

[51] P. S. Milne. On the solutions of a set of polynomial equations. In *Symbolic and Numerical Computation for Artificial Intelligence*, pages 89–102. 1992.

[52] B. Mishra. Computational real algebraic geometry. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, second edition*, chapter 33, pages 743–764. CRC Press LLC, Boca Raton, FL, 2004.

[53] B. Mourrain, J.-P. Técourt, and M. Teillaud. On the computation of an arrangement of quadrics in 3d. *Computational Geometry: Theory and Applications*, 30:145–164, 2005. Special issue, 19th European Workshop on Computational Geometry.

[54] K. Mulmuley. A fast planar partition algorithm, II. *J. ACM*, 38:74–103, 1991.

[55] K. Mulmuley. *Computational Geometry.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[56] F. Nielsen and M. Yvinec. Output-sensitive convex hull algorithms of planar convex objects. *Internat. J. Comput. Geom. Appl.*, 8(1):39–66, 1998.

[57] T. Papanikolaou. *LiDIA Manual - A Library for Computational Number Theory*. Universität des Saarlandes, Saarbrücken, 1995.

[58] P. Pedersen. Multivariate sturm theory. In *Proceedings of Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 318–323, 1991.

[59] F. P. Preparata and M. I. Shamos. *Computational geometry and introduction*. Springer-Verlag, New York, 1985.

[60] D. Prill. On approximations and incidence in cylindrical algebraic decomposition. *Siam J. Comput.*, 15(4):972–993, 1986.

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

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

[63] T. Sakkalis and R. T. Farouki. Singular points of algebraic curves. *Journal of Symbolic Computation*, 9:405–421, 1990.

[64] O. Schwarzkopf and M. Sharir. Vertical decomposition of a single cell in a three-dimensional arrangement of surfaces and its applications. *Discrete Comput. Geom.*, 18:269–288, 1997.

[65] C.-K. Shene and J. K. Johnstone. On the planar intersection of natural quadrics. In *Proc. ACM Sympos. Solid Modeling Found. CAD/CAM Appl.*, pages 233–242. Springer-Verlag, 1991.

[66] J. Snoeyink and J. Hershberger. Sweeping arrangements of curves. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 6:309–349, 1991.

[67] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Univ. of California Press, Berkely, 1951. second ed., rev.

[68] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[69] R. Wein. High level filtering for arrangements of conic arcs. In *ESA 2002, Lecture Notes in Computer Science*, pages 884–895, 2002.

[70] N. Wolpert. *An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics*. Universität des Saarlandes, Saarbrücken, 2002. Ph.D. Thesis.

[71] N. Wolpert. Jacobi curves: Computing the exact topology of arrangements of non-singular algebraic curves. In *ESA 2003, Lecture Notes in Computer Science*, pages 532–543, 2003.

[72] C. K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, Oxford, 2000.