9

Intersecting Biquadratic Bézier Surface Patches

Stéphane Chau[1], Margot Oberneder[2], André Galligo[1], and Bert Jüttler[2]

[1] Laboratoire J.A. Dieudonné, Université de Nice - Sophia-Antipolis, France
{chaus,galligo}@math.unice.fr
[2] Institute of Applied Geometry, Johannes Kepler University, Austria
{margot.oberneder,bert.juettler}@jku.at

Summary. We present three symbolic–numeric techniques for computing the intersection and self–intersection curve(s) of two Bézier surface patches of bidegree (2,2). In particular, we discuss algorithms, implementation, illustrative examples and provide a comparison of the methods.

9.1 Introduction

The intersection of two surfaces is one of the fundamental operations in Computer Aided Design (CAD) and solid modeling. Closely related to it, the elimination of self–intersections (which may arise. e.g., from offsetting) is needed to maintain the correctness of a CAD model. Tensor–product Bézier surface patches, which are parametric surfaces defined by vector–valued polynomials $\mathbf{x} : [0,1]^2 \to \mathbb{R}^3$ of certain bidegree (m,n), are extensively used to model surfaces in CAD and solid modeling. However, even for relatively small bidegrees $m,n \leq 3$, the intersection and self–intersection loci of such patches can be fairly complicated. Consequently, standard algorithms for surface–surface intersections [24, 28] generally do not take the properties of special classes of such tensor–product surfaces into account.

In the case of two general surfaces, a *brute–force approach* to compute the intersection curve(s) consists in (step 1) approximating the surface by triangular meshes and (step 2) intersecting the planar facets of these meshes. Clearly, in order to achieve high accuracy, a very fine approximation with a mesh may be needed. Alternatively, one may consider to choose another, more complicated representation, where the basic elements are capable of capturing more of the geometric features. For instance, one may choose quadratic triangular patches or biquadratic tensor–product patches[3]. Clearly, this approach would need robust intersection algorithms for the more complicated basic elements.

[3] In the same spirit, Reference [32] proposes to use triangular patches for efficient visualization.

In this paper we address the computation of the intersection curve of two surface patches of bidegree (2,2), i.e., biquadratic tensor–product patches. Our aim is to compute the intersection by using – as far as possible – *symbolic* techniques, in order to avoid problems with numerical robustness.

We chose the tensor–product representation, since it is more common in CAD environment. Approximations of general tensor–product surfaces by biquadratic ones can easily be generated by combining degree reduction techniques with subdivision. The techniques presented in this paper can immediately be extended to the case of triangular patches. Indeed, tringular patches can be seen as degenerate tensor–product patches, where one edge collapses into a single point.

The remainder of the paper is organized as follows. After some preliminaries, Sections 9.3 to 9.5 present three different techniques for computing the intersection curves, which are based on *resultants*, on *approximate implicitization* (which was one of the main research topics in the GAIA II project), and on *intersections of parameter lines*, respectively. Section 9.6 discusses the computation of self–intersections. We apply the three techniques to three representative examples and report the results in Section 9.7. Finally, we conclude this paper.

9.2 Intersection and self–intersection curves

We consider the intersection curves of two biquadratic Bézier surfaces $\mathbf{x}(u,v)$ and $\mathbf{y}(r,s)$, both with parameter domains $[0,1]^2$. They are assumed to be given by their parametric representations with rational coefficients (control points). More precisely, these representations have the form

$$\mathbf{x}(u,v) = \sum_{i=0}^{2}\sum_{j=0}^{2} \mathbf{c}_{i,j} B_i(u) B_j(v) \tag{9.1}$$

with certain rational control points $\mathbf{c}_{i,j} \in \mathbb{Q}^3$ and the quadratic Bernstein polynomials $B_j(t) = \binom{2}{i} t^i (1-t)^{2-i}$ (and similarly for the second patch $\mathbf{y}(r,s)$).

The intersection curve is defined by the system of three non–linear equations

$$\mathbf{x}(u,v) = \mathbf{y}(r,s) \tag{9.2}$$

which defines the intersection as a curve (in the generic case) in $[0,1]^4$. Similarly, self intersections of one of the patches are characterized by

$$\mathbf{x}(u,v) = \mathbf{x}(\bar{u},\bar{v}). \tag{9.3}$$

In this case, the set of solutions contains the 2–plane $u = u^*$, $v = v^*$ as a trivial component.

While these equations could be solved by using numerical methods, we plan to explore how far it is possible to compute the intersections by using *symbolic* computations, in order to avoid rounding errors and robustness problems.

The “generic” algorithm for computing the (self–) intersection curve(s), consists of three steps:

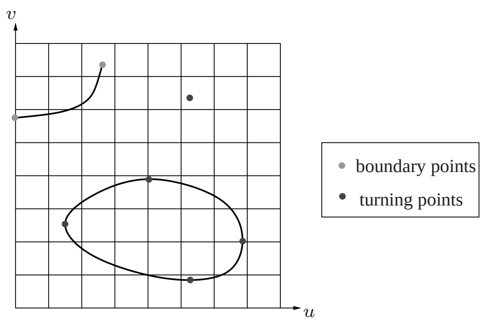

Fig. 9.1. Intersection curves in one of the parameter domains.

1. Find at least one point on each component of the intersection,
2. trace the segments of the intersection curve, and
3. collect and convert the segments into a format that is suitable for further processing (depending on the application).

We will focus on the first step, since the second step is a standard numerical problem, and step 3 depends on the specific background of the problem. Several parts of the intersection curve may exist. Some possible types are shown in Fig. 9.1 in the parameter domain of a Bézier surface $\mathbf{x}(u, v)$. Points with horizontal or vertical tangent are called *turning points* , and intersections with the boundaries of the patches generate *boundary points* . Note that also isolated points (where both surfaces touch each other) may exist.

9.3 A resultant–based approach

In this section, we will use the resultant to compute the intersection locus between $\mathbf{x}(u, v)$ and $\mathbf{y}(r, s)$. We consider the algebraic variety

$$\mathcal{C} = \{(u, v, r, s) \mid \mathbf{x}(u, v) = \mathbf{y}(r, s)\} \tag{9.4}$$

and we will suppose that $\mathcal{C} \cap [0, 1]^4$ is a curve.

9.3.1 Resultant basics

Let f_1, f_2 and f_3 be three polynomials in two variables with given monomial supports and N the number of terms of these 3 supports. For each $i \in \{1, 2, 3\}$ we denote by coeffs(f_i) the sequence of the coefficients of f_i. The resultant of f_1, f_2 and f_3 is, by definition, an irreducible polynomial R in N variables with the property, that

$$R(\text{coeffs}(f_1), \text{coeffs}(f_2), \text{coeffs}(f_3)) = 0 \tag{9.5}$$

if and only if these 3 polynomials have a common root in a specified domain $\mathcal{D}$. For a more precise description of resultants, see e.g. [2, 8, 9].

In our application to surface–surface–intersections, the resultant can be used as a projection operator. Indeed, if f_1, f_2 and f_3 are the three components of $\mathbf{x}(u,v) - \mathbf{y}(r,s)$ which are considered as polynomials in the two variables r and s, then the resultant of f_1, f_2 and f_3 is a polynomial $R(u,v)$ and it gives an implicit plane curve which corresponds to the projection of $\mathcal{C}$ in the (u,v) parameters. More precisely, if f_1, f_2 and f_3 are generic, then the two sets

$$\left\{(u,v) \in [0,1]^2 \mid R(u,v) = 0\right\} \tag{9.6}$$

and

$$\left\{(u,v) \in [0,1]^2 \mid \exists (r,s) \in \mathcal{D} : \mathbf{x}(u,v) = \mathbf{y}(r,s)\right\} \tag{9.7}$$

are identical. Several families of multivariate resultants have been studied and some implementations are available, see [5, 22].

9.3.2 Application to the intersection problem

A strategy to describe the intersection between $\mathbf{x}(u,v)$ and $\mathbf{y}(r,s)$ consists in projecting $\mathcal{C}$ on a plane (by using the resultant). Many authors propose to project $\mathcal{C}$ on the (u,v) (or (r,s)) plane and then the resulted plane curve is traced (see [16] and [20] for the tracing method) and is lifted to the 3D space by the corresponding parameterization. Note that this method can give some unwanted components (the so called "phantom components") which are not in $\mathbf{x}([0,1]^2) \cap \mathbf{y}([0,1]^2)$. So, another step is needed to cut off the extraneous branches. This last part can be done with a solver for multivariate polynomial systems (see [25]) or an inversion of parameterization (see [3]).

As an alternative to these existing approaches, we propose to project the set $\mathcal{C}$ onto the (u,r) space. Note that, in the equations $\mathbf{x}(u,v) = \mathbf{y}(r,s)$, the two variables v and s are separated, so they can be eliminated via a simple resultant computation. It turns out that such a resultant can be computed via the determinant of a Bezoutian matrix (see [15]). First, consider the $(3,3)$ determinant:

$$b = \det\left(\mathbf{x}(u,v) - \mathbf{y}(r,s), \frac{\mathbf{x}(u,v) - \mathbf{x}(u,v_1)}{v - v_1}, \frac{\mathbf{y}(r,s) - \mathbf{y}(r,s_1)}{s - s_1}\right). \tag{9.8}$$

The determinant b is a polynomial and its monomial support with respect to (v,s) is $\mathcal{S} = \{1, v, s, vs\}$ and similarly for (v_1, s_1), where $\mathcal{S}_1 = \{1, v_1, s_1, v_1 s_1\}$. So, a monomial of b is a product of an element of $\mathcal{S}$ and of an element of $\mathcal{S}_1$. Then, we form the 4×4 matrix whose entries are the coefficients of b indexed by the product of the two sets $\mathcal{S}$ and $\mathcal{S}_1$. This matrix contains only the variables u and r and is called the Bezoutian matrix. In our case, its determinant is a polynomial in (u,r) equal to the desired resultant $R(u,r)$ (deg(R)=24 and deg$_u(R)$=deg$_r(R)$=16) and it gives an implicit curve which corresponds to the projection of $\mathcal{C}$ in the (u,r) space.

Then, we analyse the topology of this curve (see [17] and [30]) and we trace it (see [16] and [20]). Finally, for each $(u_0, r_0) \in [0,1]^2$ such that $R(u_0, r_0) = 0$, we can determine if there exists a pair $(v_0, s_0) \in [0,1]^2$ such that $\mathbf{x}(u_0, v_0) = \mathbf{y}(r_0, s_0)$

(solve a polynomial system of three equations with two separated unknowns of bidegree (2,2)) and thus we can avoid the problem of the phantom components (see Fig. 9.2). We lift the obtained points in the 3D space to give the intersection locus. Note that this method can also give the projection of $\mathcal{C}$ in the (v, s) space by the same kind of computation.

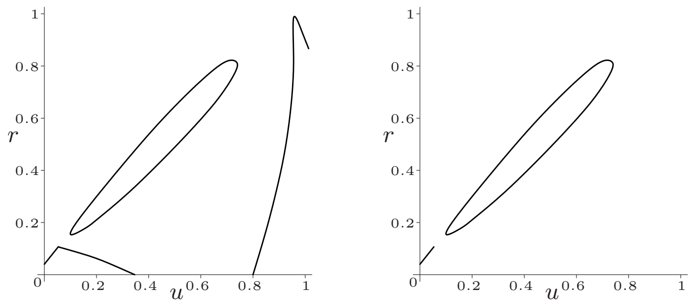

Fig. 9.2. Projection of $\mathcal{C}$ in the (u, r) space with (left) and without (right) phantom components. This curve corresponds to the example of Figure 9.6, page 175.

9.4 Approximate implicitization by a quartic surface

In this section, we apply the technique of approximate implicitization to compute the intersection of two biquadratic patches.

9.4.1 Approximate implicitization

The implicitization problem – which consists in finding an implicit equation (an algebraic representation) for a given parameterized rational surface – can be adressed by using several approaches, e.g., using resultants or Groebner bases [8, 9, 18]. However, the implicitization is very time consuming because of the degree of the implicit equation: for a generic parameterized surface of bidegree (n_1,n_2), the implicit equation has degree $2n_1n_2$. Also, all rational parametric curves and surfaces have an algebraic representation, but the reverse is not true; the relationship between the parametric and the algebraic representations can be very complex (problem of “phantom components”). Thus, we can try to find an algebraic approximation of a given parameterized surface for which the computation is more efficient and which contains less phantom components.

Consider a polynomial parameterized surface $\mathbf{x}(u, v)$ with the domain $[0, 1]^2$, and let d be a positive integer (the degree of the approximate implicit equation) and

$\epsilon \geq 0$ (the tolerance). Following [12], the approximate implicitization problem consists in finding a non–zero polynomial $P \in \mathbb{R}[x, y, z]$ of degree d such that

$$\forall (u, v) \in [0, 1]^2, P\left(\mathbf{x}(u, v) + \alpha(u, v)\, \mathbf{g}(u, v)\right) = 0 \tag{9.9}$$

with $|\alpha(u, v)| \leq \epsilon$ and $||\mathbf{g}(u, v)||_2 = 1$. Here, α is the error function and **g** is the direction for error measurement, e.g., the unit normal direction of the surface patch.

9.4.2 Approximate implicitization of a biquadratic surface

The main question of the approximate implicitization problem is how to choose the degree. A key ingredient for this choice seems to be the topology, especially if the initial surface has self–intersections. The use of degree 4 was suggested by Tor Dokken; after several experiments he concluded that the algebraic surfaces of degree 4 provide sufficiently many degrees of freedom to approximate most cases encountered in practice. In the case of a biquadratic surface, where the exact implicit equation has degree 8, using degree 4 seems to be a reasonable trade-off.

We describe two methods for approximate implicitization by a quartic for a biquadratic surface. The approximate implicit equation is

$$P(x, y, z) = \sum_{i=0}^{4} \sum_{j=0}^{4-i} \sum_{k=0}^{4-i-j} b_{ijk}\, x^i y^j z^k \tag{9.10}$$

with the unknown coefficients $b = (b_{000}, b_{100}, \ldots, b_{004}) \in \mathbb{R}^{35}$. Let $\beta(u, v)$ be the vector formed by the tensor–product Bernstein polynomials of bidegree (8,8).

Dokken's method.

This method, which is described in more detail in [12], proceeds as follows:

1. Factorize $P(\mathbf{x}(u, v)) = (Db)^T \beta(u, v)$ where D is a 81×35 matrix.
2. Generate a singular values decomposition (SVD) of D.
3. Choose b as the vector corresponding to the smallest singular value of D.

Note that this method is general and does not use the fact that we have a biquadratic surface. Hereafter, we use an adapted method based on the geometry of the surface of bidegree (2,2). Also, the computation of the singular value decomposition needs floating point numbers.

Geometric method using evaluation:

This approach consists in constructing some pertinent geometrical constraints to give a linear system of equations (with the unknowns $b_{000}, b_{100}, \ldots, b_{004}$), and then solving the resulting system by a singular values decomposition. In our method, we characterize some conics, especially the four border conics and two interior conics:

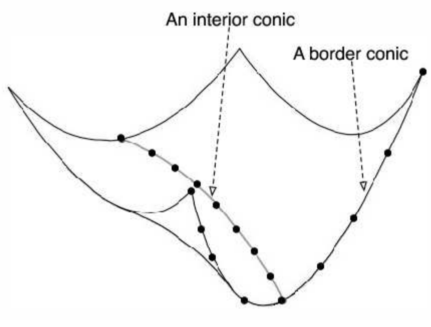

Fig. 9.3. Characterization of a conic in a biquadratic patch by 9 points

$$\begin{array}{ll} C_1 = \mathbf{x}([0,1] \times \{0\}), & C_2 = \mathbf{x}([0,1] \times \{1\}) \\ C_3 = \mathbf{x}(\{0\} \times [0,1]), & C_4 = \mathbf{x}(\{1\} \times [0,1]) \\ C_5 = \mathbf{x}(\{\frac{1}{2}\} \times [0,1]), & C_6 = \mathbf{x}([0,1] \times \{\frac{1}{2}\}) \end{array} \tag{9.11}$$

Lemma 1. *If the quartic surface $\{P = 0\}$ contains 9 points of any of the 6 conics C_i, then $C_i \subset \{P = 0\}$, see Fig. 9.3.*

Proof. C_i is of degree 2 and P is of degree 4, so by Bézout's theorem, if there are more than 8 elements in $C_i \cap \{P = 0\}$, then $C_i \subset \{P = 0\}$.

Using this geometric observation, we construct a linear system and solve it approximately via SVD; this leads to an algebraic approximation of $\mathbf{x}(u, v)$ by a degree 4 surface.

9.4.3 Application to the intersection problem

In order to compute the intersection curves, we apply the approximate implicitization to one of the patches and compose it with the second one. This leads to an implicit representation of the intersection curve in one of the parameter domains, which can then be traced and analyzed using standard methods for planar algebraic curves.

These two approximate implicitization methods are very efficient and suitable for general cases, but the results are not always satisfactory. When the given biquadratic patch is simple (i.e. with a certain flatness and without singularity and self–intersection) the approximation is very close to the initial surface. So, to use this method for a general biquadratic surface, we combine it, if needed, with a subdivision method (Casteljau's algorithm). The advantage is twofold, we exclude domains without intersections (by using bounding boxes) and avoid some unwanted configurations with a curve of self-intersection (use Hohmeyer's criterion [19]). For more complicated singularities, the results are definitively not satisfactory.

Note that even if we have a good criterion in the subdivision step, we still may have problems with phantom components (but in general fewer), so we have to cut off the extraneous branches as in the resultant method. This has to be done carefully

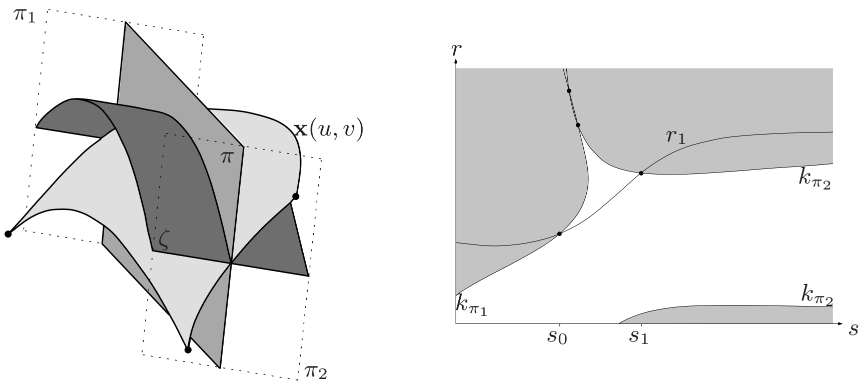

Fig. 9.4. Left: Representation of a parameter line as the intersection of a plane and a quadratic cylinder. Right: Identifying the intervals with feasible values of s.

in order to not discard points which do not correspond to phantom components. As another drawback – because of the various approximations – it is rather difficult to obtain certified points on the intersection locus. The use of approximate implicitization is clearly a numerical method, and it can give only approximate answers, even in the case of exact input.

9.5 Tracing intersections of parameter lines

In order to be able to trace the (self–) intersection curve(s), we have to find at least one point for each segment. We generate these points by intersecting the parameter lines of the first Bézier surface with the second one (see also [19]).

9.5.1 Intersection of a parameter line

A parameter line of $\mathbf{x}(u,v)$ for a constant rational value $u = u_0$ takes the form

$$\mathbf{p}(v) = \mathbf{x}(u_0, v) = \mathbf{a}_0(u_0) + \mathbf{a}_1(u_0)\,v + \mathbf{a}_2(u_0)\,v^2$$

with certain rational coefficient vectors $\mathbf{a}_i \in \mathbb{Q}^3$. It is a quadratic Bézier curve, hence we can represent it as the intersection of a *plane* and a *quadratic cylinder*, see Fig. 9.4, left. Since we are only interested in the intersection of these two surfaces in a certain region, we introduce two additional *bounding planes* π_1 and π_2. In the particular case that the parameter line is a straight line, we represent it as an intersection curve of two orthogonal planes.

In order to compute the intersection of the parameter line with the second surface patch $\mathbf{y}(r,s)$, we use the following algorithm.

1. Describe the parameter line as the intersection of a plane and a cylinder.

2. Intersect the plane with the second patch $\mathbf{y}(r,s)$ and compute the intersection $\mathcal{I}$.
3. Restrict the intersection curve(s) $\mathcal{I}$ to the region of interest.
4. Intersect the cylinder with the restricted intersection curve(s).

The four steps of the algorithm will now be explained in some more detail.

Defining the plane, the cylinder and the two bounding planes.

The parameter line and its three control points are coplanar. For computing the normal vector $\mathbf{n}$ of the plane, we have to evaluate the cross product of two difference vectors of the control points. The plane is given by the zero set of a linear polynomial

$$\pi(u_0)(x,y,z) = e_0(u_0) + n_1(u_0)\,x + n_2(u_0)\,y + n_3(u_0)\,z. \tag{9.12}$$

By extruding the parameter line in the direction of the normal vector of the plane, we obtain the parametric form of the quadratic cylinder, which intersects the plane orthogonally,

$$\mathbf{w}(u_0)(p,q) = \mathbf{x}(u_0,p) + q\cdot\mathbf{n}. \tag{9.13}$$

The implicitation of the cylinder is slightly more complicated. There exist two possibilities: we can either use Sylvester resultants or the method of comparing coefficients. In both cases we will get an equation of the form

$$\begin{aligned}\zeta(u_0)(x,y,z) := {}& a_0(u_0) + a_1(u_0)\,x + a_2(u_0)\,y + a_3(u_0)\,z\\ &+ a_4(u_0)\,x\,y + a_5(u_0)\,x\,z + a_6(u_0)\,y\,z\\ &+ a_7(u_0)\,x^2 + a_8(u_0)\,y^2 + a_9(u_0)\,z^2 \;=\; 0.\end{aligned} \tag{9.14}$$

Now we have both the plane and the cylinder in their implicit representation. Note that this is a semi-implicit representation in the sense of [6].

If the parameter line degenerates into a straight line, then we choose two planes through it which intersect orthogonally. Note that we use exact rational arithmetic, in order to avoid any robustness problems.

Finally, we create the two planes $\pi_1(x,y,z)$ and $\pi_2(x,y,z)$ which bound the parameter line. For instance, one may choose the two normal planes of the parameter line at its boundary points; this choice is always possible, provided that the curve segment is not too long (which can be enforced by using subdivision). Alternatively one may use the planes spanned by the boundary curves, but these planes may have an additional intersection with the parameter line in the region of interest.

Intersection of the plane and the second patch $\mathbf{y}(r,s)$.

Substituting the second Bézier surface $\mathbf{y}(r,s)$ into the equation (9.12) of the plane leads to a biquadratic equation in r and s. We can treat it as a quadratic polynomial in r with coefficients depending on s.

$$\pi(\mathbf{y}(r,s)) = a(s)\,r^2 + b(s)\,r + c(s) = 0. \tag{9.15}$$

For each value of s, we obtain two solutions $r_1(s)$ and $r_2(s)$ of the form

$$r_{1,2}(s) = -\frac{b(s)}{2\,a(s)} \pm \sqrt{d(s)} \quad \text{with} \quad d(s) = \frac{b(s)^2}{4\,a(s)^2} - \frac{c(s)}{a(s)}. \tag{9.16}$$

These solutions parameterize the two branches of the intersection curve $\mathcal{I}$ in the rs–parameter domain of the second patch. By solving several quadratic equations we determine the intervals $S_{i,j} \subset [0,1]$, where $d(s) \geq 0$ and $0 \leq r_i(s) \leq 1$ holds; this leads to a (list of) feasible domain(s) (i.e., intervals) for each branch of the intersection curve.

By composing (9.16) with $\mathbf{y}$ we obtain the two branches $\mathbf{k}_1(s)$ and $\mathbf{k}_2(s)$ of the intersection curve $\mathcal{I}$,

$$\mathbf{k}_{1,2}(s) = \mathbf{y}(r_{1,2}(s), s) = \frac{1}{a(s)^2}\mathbf{h}(s) \pm \frac{\sqrt{d(s)}}{a(s)}\mathbf{l}(s) + d(s)\mathbf{m}(s) \tag{9.17}$$

where the components of $\mathbf{h}(s)$, $\mathbf{l}(s)$ and $\mathbf{m}(s)$ are polynomials of degree 6, 4, and 2, respectively.

Restriction to the region of interest.

Since the region of interest is located between the planes $\pi_1(x, y, z)$ and $\pi_2(x, y, z)$, the two inequalities

$$\pi_1(x, y, z) \geq 0 \quad \text{and} \quad \pi_2(x, y, z) \leq 0 \tag{9.18}$$

have to be satisfied. By intersecting each bounding plane with the second Bézier surface $\mathbf{y}(r, s)$ in a similar way as described for $\pi(u_0)(x, y, z)$, we obtain

$$k_{\pi_1}(s) := \pi_1(\mathbf{y}(r(s), s)) \geq 0 \qquad \text{and} \qquad k_{\pi_2}(s) := \pi_2(\mathbf{y}(r(s), s)) \leq 0 \tag{9.19}$$

This leads to additional constraints for the feasible values of the parameter s. For each branch of the intersection curve we create the (list of) feasible domain(s) and store it. The bounds of the intervals can be computed by solving three systems of two biquadratic equations or – equivalently – by solving a system of three polynomials of degree 8, which are obtained after eliminating the parameter r. Here, we represent the polynomials in Bernstein–Bézier form and use a Bézier–clipping–type technique see [14, 25, 26, 28], applied to floating point numbers.

Example 2. For a parameter line $u = u_0$ of two biquadratic Bézier surface patches $\mathbf{x}(u, v)$ and $\mathbf{y}(r, s)$, Fig. 9.4, right, shows the rs–parameter domain of the second patch. Only the first branch $r_1(s)$ of the intersection curve is present. The bounds $0 \leq r \leq 1$ do not impose additional bounds on s in this case. However, the intersection with the bounding planes π_1 and π_2 produces two additional curves, which have to be intersected with the curve $s = r_1(s)$, leading to two bounds s_0 and s_1 of the feasible domain.

Intersection of the cylinder and the intersection curves.

We substitute the parametric representation of the intersection curve into the implicit equation (9.14) of the cylinder and obtain

$$\begin{aligned}\zeta(u_0)(s) = p_1(s) + p_2(s)\sqrt{d(s)} + p_3(s)\left(\sqrt{d(s)}\right)^2 + \\ + p_4(s)\left(\sqrt{d(s)}\right)^3 + p_5(s)\left(\sqrt{d(s)}\right)^4 = 0 \qquad (9.20)\end{aligned}$$

where the polynomials $p_j(s)$ are of degree 12. In order to eliminate the square root, we use the following trick. We split $\zeta(s, d(s)) = A - B$, where A and B contain all even and odd powers of $\sqrt{d}$, respectively. The equation $A - B = 0$ is then replaced with $A^2 \cdot d(s) - (B \cdot \sqrt{d(s)})^2 = 0$. This leads to a polynomial of degree 24 in one variable. After factoring out the discriminant, we obtain a polynomial of degree 16 in s. Note that this agrees with the theoretical number of intersections of a biquadratic surface, which has algebraic order 8, with a quadratic curve.

Finally, we solve this polynomial within all the feasible intervals of s, which were detected in the previous steps. Until this point we used symbolic computations. Now – after generating the Bernstein–Bézier representation – we change to floating-point numbers and use a Bézier–clipping–type method to find all roots within the feasible domain(s). These roots correspond to intersection points of the parameter line of the first patch with the second patch.

9.5.2 Global structure of the intersection curve

For each value $u = u_0$, the parameter line $\mathbf{x}(u_0, v)$ has a certain number of intersection points with the second patch. If u_0 varies continuously, then the number of intersection points may change only if

(1) one of the intersection points is at the boundary of one of the patches (boundary points) or
(2) the parameter line of the first patch touches the second patch (turning points) .

The algorithm for analyzing the global structure of the intersection curve proceeds in two steps: First we detect those values of u_0 where the number of intersection points changes, and order them. This leads to a sequence of critical u_0– values,

$$0 = u_0^{(0)} < u_0^{(1)} < \ldots < 1 = u_0^{(K)}. \qquad (9.21)$$

In the second step, we analyze the intersection of the parameter lines $u_0 = (u_0^{(i)} + u_0^{(i+1)})/2$ with the second patch. Since the number of intersection points between any two critical values remains constant, we can now either trace the segment using conventional techniques for tracing surface–surface intersections (see [20]) or generate more points by analyzing more intersections with parameter lines.

In the remainder of this section we address the computation of the critical u_0 values.

Boundary points.

Such points correspond to intersections of the boundary parameter lines of one surface with the other one. In order to compute them, we apply the algorithm for intersecting parameter lines with a biquadratic patch to the $2 \cdot 4$ boundary parameter lines of the two surfaces.

Turning points.

We consider the turning points of $\mathbf{x}(u, v)$ in respect to u. Let $\mathbf{y}_r$ and $\mathbf{y}_s$ denote the partial derivatives of $\mathbf{y}(r, s)$. Several possibilities for computing the turning points exist.

1. The two surfaces $\mathbf{x}(u_0, v)$ and $\mathbf{y}(r, s)$ intersect, $\mathbf{x}(u_0, v) = \mathbf{y}(r, s)$, and the tangent vector of the parameter line lies in the tangent plane of the second patch,
$$\mathbf{x}_u \cdot (\mathbf{y}_r \times \mathbf{y}_s) = 0. \tag{9.22}$$
These conditions lead to a system of four polynomial equations for four unknowns, which has to be solved for u.
2. By using the previous geometric result, we may eliminate the variable v, as follows. First, the plane spanned by the parameter line has to contain the point $\mathbf{y}(r, s)$,
$$\pi(u_0)(\mathbf{y}(r, s)) = 0, \tag{9.23}$$
which gives an equation of degree $(6, 2, 2)$ in (u_0, r, s). Second, the cylinder has to contain the point,
$$\zeta(u_0)(\mathbf{y}(r, s)) = 0, \tag{9.24}$$
which leads to an equation of degree $(16, 4, 4)$. Finally, the tangent vector of the parameter line has to be contained in the tangent plane of the second patch. Since the tangent of the parameter line is parallel to the cross product of the gradient of the plane and the gradient of the cylinder, the third condition gives an equation of degree $(18, 5, 5)$,
$$\det\left[\mathbf{y}_r,\ \mathbf{y}_s,\ \nabla\pi(u_0)(\mathbf{y}(r, s)) \times \nabla\zeta(u_0)(\mathbf{y}(r, s))\right] = 0. \tag{9.25}$$

For solving either of these two systems of polynomial equations, we use again a Bézier–clipping–type algorithm [14, 25, 28].

9.6 Self–intersections of biquadratic surface patches

In order to detect the self–intersection curves of any of the two patches, the methods for surface–surface intersections have to be modified. The computation of the self–intersection locus by using approximate implicitization is not discussed here, since it was already treated in [31]. Instead we focus on the other two techniques.

9.6.1 Resultant-based method

In the parameter domain $[0,1]^4$, the self–intersection curve of the first patch forms the set

$$\left\{(u_1,v_1,u_2,v_2)\in[0,1]^4 \mid (u_1,v_1)\neq(u_2,v_2) \text{ and } \mathbf{x}(u_1,v_1)=\mathbf{x}(u_2,v_2)\right\}. \tag{9.26}$$

This locus is the real trace of a complex curve. We assume that it is either empty or of dimension 0 or 1. We do not consider degenerate cases, such as a plane which is covered twice. In the examples presented below (see Section 9.7), the self–intersection locus is a curve in $\mathbb{R}^4$.

We use the following change of coordinates to discard the unwanted trivial component $(u_1,v_1)=(u_2,v_2)$. Let (u_2,v_1) be a pair of parameters in $[0,1]^2$, $(l,k)\in\mathbb{R}^2$ and let $u_1=u_2+l$, $v_2=v_1+lk$. If we suppose that we have $(u_1,v_1)\neq(u_2,v_2)$, then $l\neq 0$. Hence $\mathbf{x}(u_1,v_1)=\mathbf{x}(u_2,v_2)$ if and only if $\mathbf{x}(u_2+l,v_1)=\mathbf{x}(u_2,v_1+lk)$. We suppose now that (u_2,v_1,l,k) verifies this last relation.

Let $\tilde{T}(u_2,v_1,l,k)$ be the polynomial $\frac{1}{l}\left[\mathbf{x}(u_2+l,v_1)-\mathbf{x}(u_2,v_2+lk)\right]$, its degree in (u_2,v_1,l,k) is $(2,2,1,2)$ and the monomial support with respect to (l,k) contains only k^2l,k,l and 1. We can decrease the degree by introducing

$$T(u_2,v_1,m,k)=m\tilde{T}(u_2,v_1,\frac{1}{m},k). \tag{9.27}$$

Then in $T(u_2,v_1,m,k)$, the monomial support in (m,k) consists only of $1,m,k^2$ and km. So, we can write T in a matrix form:

$$T(u_2,v_1,m,k)=\begin{pmatrix} a_1(u_2,v_1) & b_1(u_2,v_1) & c_1(u_2,v_1) & d_1(u_2,v_1)\\ a_2(u_2,v_1) & b_2(u_2,v_1) & c_2(u_2,v_1) & d_2(u_2,v_1)\\ a_3(u_2,v_1) & b_3(u_2,v_1) & c_3(u_2,v_1) & d_3(u_2,v_1)\end{pmatrix}\begin{pmatrix}1\\ m\\ k^2\\ km\end{pmatrix} \tag{9.28}$$

By Cramer's rule, we get

$$m=\frac{D_2}{D_1},\quad k^2=\frac{D_3}{D_1},\quad \text{and}\quad km=\frac{D_4}{D_1} \tag{9.29}$$

with

$$D_1=\begin{vmatrix} b_1 & c_1 & d_1\\ b_2 & c_2 & d_2\\ b_3 & c_3 & d_3\end{vmatrix},\ D_2=\begin{vmatrix} -a_1 & c_1 & d_1\\ -a_2 & c_2 & d_2\\ -a_3 & c_3 & d_3\end{vmatrix},\ D_3=\begin{vmatrix} b_1 & -a_1 & d_1\\ b_2 & -a_2 & d_2\\ b_3 & -a_3 & d_3\end{vmatrix},\ D_4=\begin{vmatrix} b_1 & c_1 & -a_1\\ b_2 & c_2 & -a_2\\ b_3 & c_3 & -a_3\end{vmatrix}.$$

Let $Q(u_2,v_1)$ be the polynomial $Q=D_4^2D_1-D_2^2D_3$.

Lemma 3. *The implicitly defined curve $\left\{(u_2,v_1)\in[0,1]^2 \mid Q(u_2,v_1)=0\right\}$ is the projection of the self–intersection locus (given by the set (9.26) but in $\mathbb{C}^4$) into the parameters domain $(u_2,v_1)\in[0,1]^2$.*

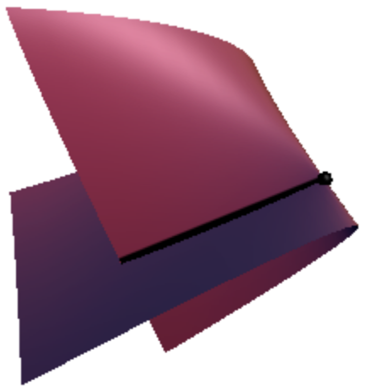

Fig. 9.5. A self–intersection of a surface with a cuspidal point

Proof. $Q(u_2, v_1) = 0$ is the only algebraic relation (of minimal degree) between u_2 and v_1 such that

$$\forall (u_2, v_1) \in [0,1]^2, Q(u_2, v_1) = 0 \Rightarrow \exists (m, k) \in \mathbb{C}^2, T(u_2, v_1, m, k) = 0.$$

This lemma provides a method to compute the self–intersection locus, we just have to trace the implicit curve $Q(u_2, v_1) = 0$ and for every point (u_2, v_1) on this curve, we obtain by continuation the corresponding point $(u_1, v_2) \in [0,1]^2$ if it exists (see the results on Fig. 9.9). So it suffices to characterize the bounds of these segments of curves.

9.6.2 Parameter-line-based method

For computing the self–intersection curves, we use the same algorithm as described in Section 9.5. We intersect the surface $\mathbf{x}(u_0, v)$ with itself $\mathbf{x}(r, s)$. In this case, both the "plane" equation (9.23) and the "cylinder" equation (9.24) contain the linear factor $(r - u_0)$, which has to be factored out. The computation of turning points as in section 9.5.2 leads us to two different types: the usual ones and cuspidal points (see Fig. 9.5).

9.7 Examples

The three methods presented in this paper (using resultants, via approximate implicitization, and by analyzing the intersections with parameter lines) work well for most standard situations usually encountered in practice. In this section, we present three representative examples. Additional ones are available at [21].

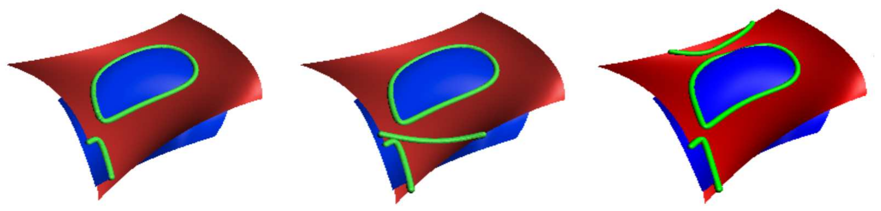

Fig. 9.6. First example. Left and center: Result of the resultant method after and before eliminating phantom branches. Right: result of the approach using approximate implicitization.

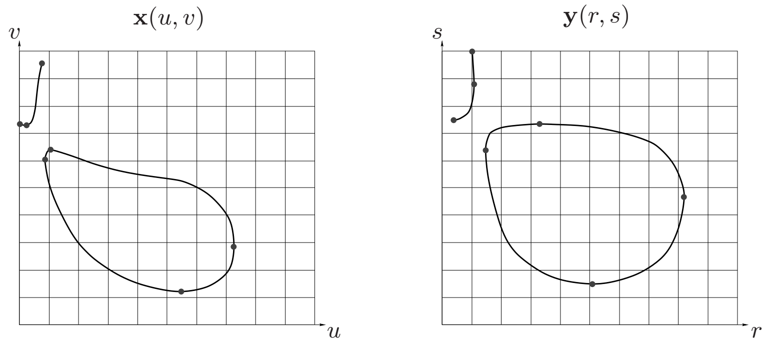

Fig. 9.7. First example. The intersection curves in the parameter domains of both surface patches, generated by the parameter–line based technique. Boundary points and turning points have been marked by grey circles.

9.7.1 First example

We consider two biquadratic surfaces with an open and a closed component of the intersection curve. The two surfaces have the control points

$$\underbrace{\begin{bmatrix} \left(\frac{1}{7},0,\frac{3}{5}\right) & \left(\frac{3}{5},\frac{1}{5},\frac{3}{4}\right) & \left(1,0,\frac{7}{10}\right) \\ \left(\frac{3}{8},\frac{4}{9},\frac{2}{3}\right) & \left(\frac{2}{3},\frac{3}{4},\frac{1}{3}\right) & \left(\frac{6}{7},\frac{3}{8},\frac{5}{7}\right) \\ \left(\frac{1}{5},\frac{6}{7},\frac{4}{7}\right) & \left(\frac{3}{4},\frac{7}{8},\frac{3}{4}\right) & \left(\frac{7}{8},\frac{7}{9},\frac{5}{8}\right) \end{bmatrix}}_{\mathbf{x}(u,v)} \text{ and } \underbrace{\begin{bmatrix} \left(\frac{2}{7},\frac{1}{7},\frac{2}{5}\right) & \left(\frac{3}{5},\frac{1}{10},\frac{2}{3}\right) & \left(1,0,\frac{4}{5}\right) \\ \left(\frac{3}{8},\frac{4}{9},\frac{2}{3}\right) & \left(\frac{1}{3},\frac{1}{2},1\right) & \left(\frac{5}{7},\frac{3}{8},\frac{2}{7}\right) \\ \left(\frac{1}{5},\frac{6}{7},\frac{3}{7}\right) & \left(\frac{3}{4},\frac{7}{8},\frac{5}{8}\right) & \left(\frac{7}{8},\frac{4}{7},\frac{1}{2}\right) \end{bmatrix}}_{\mathbf{y}(r,s)}.$$

By using the *resultant method*, a phantom component appears (see Fig. 9.6, center). It can be cut off as described in Section 9.3.2 (see Fig. 9.6, left).

Similar to the resultant method, the *approximate implicitization* produces a phantom component (see Fig. 9.6, right). However, when we cut it off, we obtain only very few certified points on the intersection locus as described in section 9.4.3.

The *parameter-line-based approach* finds both parts of the intersection curve, but no phantom components. One segment is closed and has two turning points with respect to each parameter u, v, r and s. The other segment has two boundary points

$u = 0$ and $s = 1$ and also possesses a turning point with respect to v and another one with respect to r (see Fig. 9.7).

9.7.2 Second example

The control points of the two biquadratic surfaces

$$\underbrace{\begin{bmatrix} \left(\frac{501}{775}, \frac{388}{775}, \frac{588}{775}\right) & \left(\frac{347}{775}, \frac{276}{775}, \frac{479}{775}\right) & \left(\frac{309}{775}, \frac{604}{775}, \frac{498}{775}\right) \\ \left(\frac{553}{775}, \frac{454}{775}, \frac{293}{775}\right) & \left(\frac{336}{775}, \frac{382}{775}, \frac{469}{775}\right) & \left(1, \frac{426}{775}, \frac{137}{775}\right) \\ \left(\frac{337}{775}, \frac{308}{775}, \frac{258}{775}\right) & \left(\frac{517}{775}, 0, \frac{367}{775}\right) & \left(\frac{533}{775}, \frac{492}{775}, \frac{564}{775}\right) \end{bmatrix}}_{\mathbf{x}(u,v)}$$

$$\underbrace{\begin{bmatrix} \left(\frac{492}{775}, \frac{67}{155}, \frac{522}{775}\right) & \left(\frac{543}{775}, \frac{322}{775}, \frac{117}{775}\right) & \left(\frac{346}{775}, \frac{13}{155}, \frac{4}{5}\right) \\ \left(\frac{113}{155}, \frac{392}{775}, \frac{58}{155}\right) & \left(\frac{632}{775}, \frac{469}{775}, \frac{413}{775}\right) & \left(\frac{307}{775}, \frac{514}{775}, \frac{564}{775}\right) \\ \left(\frac{602}{775}, \frac{129}{775}, \frac{274}{775}\right) & \left(\frac{669}{775}, \frac{692}{775}, \frac{53}{155}\right) & \left(\frac{488}{775}, \frac{219}{775}, \frac{412}{775}\right) \end{bmatrix}}_{\mathbf{y}(r,s)}$$

were generated by using a pseudo–random number generator.

The *resultant–based technique* leads to several phantom components (see Fig. 9.8, center), which can be cut off as described previously (see Fig. 9.8, left).

The combined use of *subdivision* and *approximate implicitization* produces even more phantom components (see Fig. 9.8, right). This is due to the fact that the subdivision generates more implicitly defined surfaces. Eventually we obtain sufficiently many points to draw the correct intersection curves.

We also computed the self–intersection curve (see Fig. 9.9) with the help of the method described in Section 9.6.1.

When using the *parameter–line based approach*, this example does not lead to any difficulties. The intersection curve consists of three segments (see Fig. 9.10). The first Bézier surface patch $\mathbf{x}(u, v)$ has one self–intersection curve, while the second one $\mathbf{y}(r, s)$ intersects itself three times and has two cuspidal points.

9.7.3 Third example

The two biquadratic surface patches with the control points

$$\underbrace{\begin{bmatrix} \left(0, \frac{1}{7}, \frac{4}{5}\right) & \left(\frac{3}{5}, \frac{1}{13}, \frac{1}{3}\right) & \left(1, 0, \frac{4}{5}\right) \\ \left(\frac{1}{8}, \frac{4}{9}, \frac{11}{40}\right) & \left(\frac{1}{3}, \frac{34}{65}, \frac{3}{4}\right) & \left(\frac{6}{7}, \frac{3}{8}, -\frac{16}{35}\right) \\ \left(\frac{1}{5}, \frac{6}{7}, \frac{4}{5}\right) & \left(\frac{3}{4}, \frac{443}{520}, \frac{3}{8}\right) & \left(\frac{7}{8}, 1, \frac{14}{15}\right) \end{bmatrix}}_{\mathbf{x}(u,v)} \text{ and } \underbrace{\begin{bmatrix} \left(0, \frac{1}{7}, \frac{1}{5}\right) & \left(\frac{3}{5}, \frac{1}{10}, \frac{1}{3}\right) & \left(1, 0, \frac{1}{5}\right) \\ \left(\frac{1}{8}, \frac{4}{9}, \frac{7}{8}\right) & \left(\frac{1}{3}, \frac{1}{2}, \frac{3}{4}\right) & \left(\frac{6}{7}, \frac{3}{8}, \frac{1}{7}\right) \\ \left(\frac{1}{5}, \frac{6}{7}, \frac{1}{5}\right) & \left(\frac{3}{4}, \frac{7}{8}, \frac{3}{8}\right) & \left(\frac{7}{8}, 1, \frac{1}{3}\right) \end{bmatrix}}_{\mathbf{y}(r,s)}$$

touch each other along a parameter line.

The *resultant-based approach* leads to an implicitly defined curve which describes the intersection. Due to the special situation, it contains the square of this

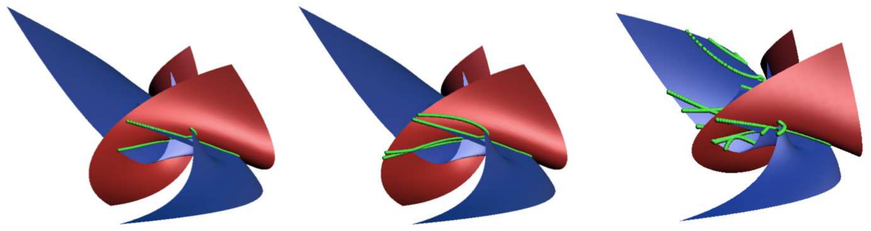

Fig. 9.8. Second example: Left and center: result of the resultant method after and before eliminating phantom branches. Right: result obtained by using approximate implicitization.

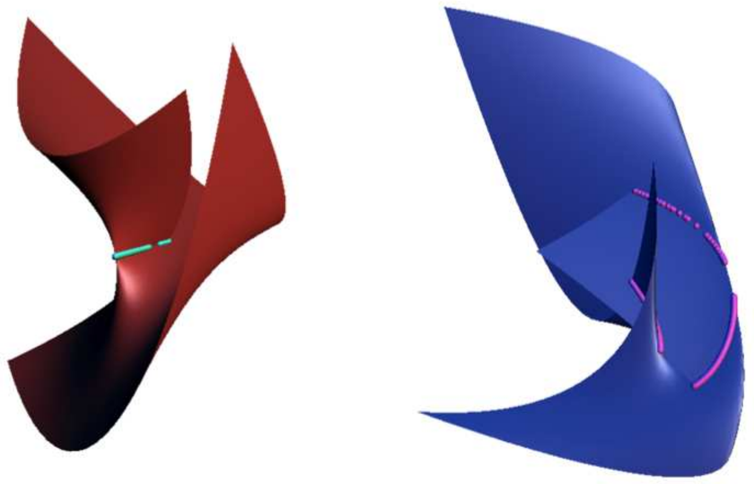

Fig. 9.9. Second example: Self intersections, computed with the method described in Section 9.6.1.

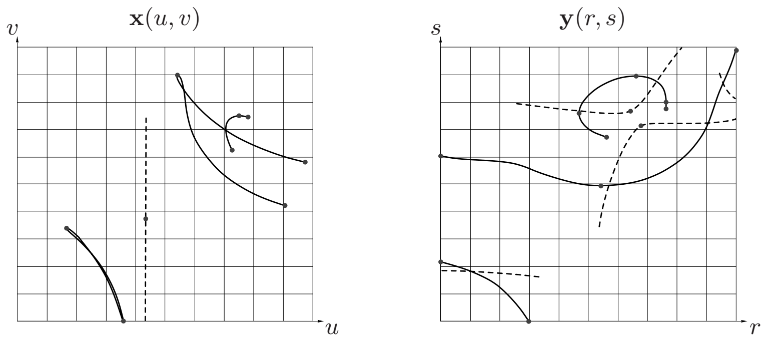

Fig. 9.10. Second example: Intersection (solid, black) and self–intersection (dashed, grey) curves in the parameter domains of both surface patches, generated by the parameter–line based technique. Boundary points and turning points have been marked by grey circles.

equation. A direct tracing of the curve is difficult, since the use of a standard predictor/corrector method is numerically unstable. However, one may factorize the equation, and apply the tracing to the individual factors without probems. This leads to the curve shown in Fig. 9.11, left.

The technique of *approximate implicitization* is not well suited to deal with this very specific situation: the approximation produces either an empty intersection or two curves which are close to each other (see Fig. 9.11, right).

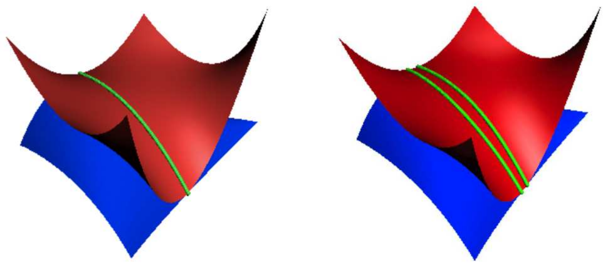

Fig. 9.11. Third example. Left: Result of the resultant method and of the parameter–line based approach. Right: result of the use of approximate implicitization.

The *parameter-line-based approach* finds two boundary points and it produces – for each value of $u = u_0$ – the correct intersection point of the parameter line with the other patch. The convergence of the Bézier clipping slows down to a linear rate, due to the presence of a double root. Also, it is difficult to trace the intersection curve by using a geometric predictor/corrector technique. Instead, we computed the intersection points for many values of u_0 and arrived at a result which is very similar to Fig. 9.11, left.

9.8 Conclusion

We presented three different algorithms for computing the intersection and self–intersection curves of two biquadratic Bézier surface patches. We implemented the methods and applied them to many test cases. Three of them have been presented in this paper.

The *resultant–based technique* was able to deal with all test cases. It may produce additional ‘phantom’ branches, which have to be eliminated by carefully analyzing the result of the elimination. As an advantage, one may – in the case of two surface patches that touch each other – factorize the implicit equation of the intersection curve, in order to obtain a stable representation, which can then be traced robustly.

After experimenting with *approximate implicitization* we arrived at the conclusion that this method is not to be recommended for biquadratic patches. On the one hand, it is not suited for avoiding problems with phantom branches. On the other hand, the use of an approximate technique introduces inaccuracies, which may cause problems with singular and almost singular situations. We feel that this price for using a lower degree implicit representation is too high.

The *parameter–line based approach* adds some geometric interpretations to the process of eliminating variables from the problem. As an advantage, it is possible to correctly establish the region(s) of interest. This avoids problems with unwanted branches of the (self–) intersection curves. In the case of two touching surfaces, using this approach becomes more expensive, since standard techniques for tracing the intersection cannot be applied.

Acknowledgment

This research was supported by the European Union through project IST 2001–35512 ‘Intersection algorithms for geometry based IT applications using approximate algebraic methods’ (GAIA II), by the Austrian Science Fund through the Joint Research Programme FSP S92 ‘Industrial Geometry’, and by Aim@Shape (IST NoE 506766).

References

1. L. Andersson, J. Peters, and N. Stewart, Self-intersection of composite curves and surfaces, *Computer Aided Geometric Design, 15 (1998)*, pp. 507–527.
2. L. Busé, Etude du résultant sur une variété algébrique, *PhD thesis, University of Nice*, December 2001.
3. L. Busé and C. D’Andrea, Inversion of parameterized hypersurfaces by means of subresultants, *Proceedings ACM of the ISSAC 2004*, pp. 65–71.
4. L. Busé, M. Elkadi, and B. Mourrain, Using projection operators in Computer Aided Geometric Design, *In Topics in Algebraic Geometry and Geometric Modeling*, pp. 321–342, Contemporary Mathematics, AMS, 2003.
5. L. Busé, I.Z. Emiris and B. Mourrain, *MULTIRES*, `http://www-sop.inria.fr/galaad/logiciels/multires.`
6. L. Busé and A. Galligo, Using semi-implicit representation of algebraic surfaces, *Proceedings of the SMI 2004 conference, IEEE Computer Society*, pp. 342–345.
7. E.W. Chionh and R.N. Goldman, Using multivariate resultants to find the implicit equation of a rational surface, *The Visual Computer 8 (1992)*, pp. 171–180.
8. D. Cox, J. Little and D. O’Shea, Ideals, Varieties and Algorithms, *Springer-Verlag*, New York, 1992 and 1997.
9. D. Cox, J. Little and D. O’Shea, Using Algebraic Geometry, *Springer-Verlag*, New York, 1998.
10. C. D’Andrea, Macaulay style formulas for sparse resultants, *Trans. Amer. Math. Soc., 354(7) (2002)*, pp. 2595–2629.
11. T. Dokken, Aspects of Intersection Algorithms and Approximation, *Thesis for the doctor philosophias degree*, University of Oslo, Norway 1997.
12. T. Dokken, Approximate implicitization, *Mathematical Methods for Curves and Surfaces*, T. Lyche and L.L. Schumaker (eds.), Vanderbilt University Press, 2001, pp. 81–102.

13. T. Dokken and J.B. Thomassen, Overview of Approximate Implicitization, *Topics in Algebraic Geometry and Geometric modeling*, ed. Ron Goldman and Rimvydas Krasauskas, AMS series on Contemporary Mathematics CONM 334, 2003, pp. 169–184.
14. G. Elber and M-S. Kim, Geometric Constraint Solver using Multivariate Rational Spline Functions, *The Sixth ACM/IEEE Symposium on Solid Modeling and Applications, 2001*, pp. 1–10.
15. M. Elkadi and B. Mourrain, Some applications of Bezoutians in Effective Algebraic Geometry, *Rapport de Recherche 3572*, INRIA, Sophia Antipolis, 1998.
16. G. Farin, J. Hoschek and M-S. Kim, Handbook of Computer Aided Geometric Design, *Elsevier*, 2002.
17. L. González-Vega and I. Necula, Efficient topology determination of implicitly defined algebraic plane curves, *Comput. Aided Geom. Design, 19(9) (2002)*, pp. 719–743.
18. C. M. Hoffmann, Implicit Curves and Surfaces in CAGD, *Comp. Graphics and Appl. (1993)*, pp. 79–88.
19. M. E. Hohmeyer, A Surface Intersection Algorithm Based on Loop Detection, *ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications, 1991*, pp. 197–207.
20. J. Hoschek and D. Lasser, Fundamentals of Computer Aided Geometric Design, *A.K. Peters*, 1993.
21. S. Chau and M. Oberneder, `http://www.ag.jku.at/˜margot/biquad`
22. A. Khetan, The resultant of an unmixed bivariate system, *J. of Symbolic Computation, 36 (2003)*, pp. 425–442. http://www.math.umass.edu/∼khetan/software.html
23. S. Krishnan and D. Manocha, An Efficient Surface Intersection Algorithm Based on Lower-Dimensional Formulation, *ACM Transactions on Graphics, 16(1) (1997)*, pp. 74–106.
24. L. Kunwoo, Principles of CAD/CAM/CAE Systems, *Addison-Wesley*, 1999.
25. B. Mourrain and J.-P. Pavone, Subdivision methods for solving polynomial equations, *Technical Report 5658*, INRIA Sophia-Antipolis, 2005.
26. T. Nishita, T.W. Sederberg and M. Kakimoto, Ray tracing trimmed rational surface patches, *Siggraph, 1990*, pp. 337–345.
27. N.M. Patrikalakis, Surface-to-surface intersections, *IEEE Computer Graphics and Applications, 13(1) (1993)*, pp. 89–95.
28. N. Patrikalakis and T. Maekawa, Chapter 25: Intersection problems, Handbook of Computer Aided Geometric Design (G. Farin and J. Hoschek and M.-S. Kim, eds.), *Elsevier*, 2002.
29. J.-P. Pavone, Auto-intersection des surfaces paramétrées réelles, *Thèse d'informatique de l'Université de Nice Sophia-Antipolis*, Décembre 2004.
30. J.P. Técourt, Sur le calcul effectif de la topologie de courbes et surfaces implicites, *PhD thesis in Computer Science at INRIA Sophia-Antipolis*, Décembre 2005.
31. J.B. Thomassen, Self-Intersection Problems and Approximate Implicitization, *Computational Methods for Algebraic Spline Surfaces, Springer*, pp. 155–170, 2005.
32. A. Vlachos, J. Peters, C. Boyd and J. L. Mitchell, Curved PN Triangles, Symposium on Interactive 3D Graphics, Bi-Annual Conference Series, ACM Press, 2001, 159–166.