Algebraic Geometry and Geometric Modeling: Insight and Computation

Ron Goldman

Department of Computer Science
Rice University
6100 Main Street
Houston TX 77005
rng@rice.edu

Summary. Recent trends in algebraic geometry emphasize effective computation over transcendent theory. The theme of this paper is that from the perspective of geometric modeling this trend is largely misguided – that for the purpose of geometric modeling the true role of algebraic geometry is insight not computation.

1 Introduction

The conventional wisdom expressed at several recent joint conferences and workshops on algebraic geometry and geometric modeling – *Algebraic Geometry and Geometric Modeling 2002*, Vilnius, Lithuania, *The MSRI Workshop on Real Algebraic Geometry in Applications*, Berkeley, 2004, and *Algebraic Geometry and Geometric Modeling 2004*, Nice, France – is that algebraic geometry has much to offer geometric modeling. Algebraic geometry and geometric modeling both deal with polynomials: Algebraic geometry investigates the algebraic and geometric properties of polynomials; geometric modeling uses polynomials to build computer models for industrial design and manufacture. Since algebraic geometry is by far the older, more mature discipline, geometric modeling should have a lot to learn from algebraic geometry.

Contemporary trends in algebraic geometry emphasize effective computation. This current fashion in algebraic geometry is largely an outgrowth of two recent developments from outside the discipline: the ubiquity and utility of modern digital computers and the tastes and predilections of the funding agencies.

Computers demand effective computation, but the ideal of effective computation was present at the creation. Descartes first merged geometry with algebra; Descartes' dream was to invoke algebra to mechanize geometry. Descartes envisioned a day when a machine using algebra could take over all of the difficult and tedious analysis of geometry [18]. Today we are still far from realizing Descartes' dream, but modern computers certainly accentuate this ambition.

Funding agencies emphasize applications. There are many reasons for this emphasis. To justify their work within contemporary capitalist economies, funding agencies need to show practical returns for their disbursements. The public and private donors who put up the money for research also demand immediate returns. Bureaucrats and donors both prefer safe short term gains over risky long term investments. Thus the funding culture is naturally inclined to believe that with computers in hand, mathematicians should concentrate on developing immediately useful computational tools to solve pressing engineering problems rather than pursue difficult long term theoretical goals.

Here I shall argue against this trend toward effective computation in algebraic geometry. My experience is that *the main contribution of algebraic geometry to geometric modeling is insight not computation.* I shall give many examples to support this thesis, and I shall argue for theoretical and practical reasons that despite the ubiquity of computers and the aspirations of funding agencies, this relationship between algebraic geometry and geometric modeling is not likely to change.

The claims and examples in this paper are based mostly on my personal research. Perhaps my experience is too narrow, or perhaps the current imbalance between insight and computation is about to change due to recent improvements in computational techniques. I am aware of at least some of these developments and I will return to these hopeful possibilities in the final section of this paper, for despite what I am about to say in the body of the text the outlook for constructive algebraic geometry is not unremittingly negative. Nevertheless, I believe that currently the burden of proof still lies overwhelmingly with those who insist on the practical utility of effective computational techniques from algebraic geometry. Geometric modeling would certainly benefit a great deal if these constructive techniques ever turn out to be practical. At present I remain skeptical of this prospect, but lest this paper seem relentlessly negative, I will revisit this contingency in the final section of this paper.

Section 2 begins the discussion with the canonical computational problem in both algebraic geometry and geometric modeling: solving systems of polynomial equations. Although algebraic geometry offers several constructive tools such as Grobner bases, resultants, and homotopy continuation, for solving systems of polynomial equations, these techniques are not currently competitive with numerical methods. What algebraic geometry offers is insight into the difficulty of finding the solutions rather than practical computational techniques for calculating solutions.

Intersection algorithms for algebraic curves and surfaces are the focus of Section 3. For low degree curves and surfaces algebraic geometry does indeed provide the most effective computational tools for calculating intersections. But such low degree curves and surfaces are not the major focus of geometric modeling. For the curves and surfaces most common in geometric modeling, numerical methods are, at present, far superior to algebraic techniques.

Section 4 investigates implicitization. Recently the implicitization problem has received a good deal of attention from the algebraic geometry community, ostensibly because of its practical applications. We shall see that the practical advantages of implicitization are less than generally supposed and that, in any event, the computational tools available from algebraic geometry are not particularly efficient. Again what algebraic geometry offers, at present, is insight into the nature of implicitization rather than practical computational techniques for calculating the implicit equation.

Resultants are the focus of Section 5. Though algebraic geometry provides lots of insight into different resultant formulations, almost nothing is currently known about the relative speed or numerical stability of various resultant techniques. Thus the utility and viability of resultants in practical applications is still an open question.

The method of moving surfaces is discussed in Section 6. This implicitization technique is an area of conscious collaboration between algebraic geometry and geometric modeling: The method – based on linear algebra – was introduced in geometric modeling, the proofs – based on sheaf cohomology – were provided by algebraic geometry. Thus algebraic geometry supplies the insight underlying the computations.

Efficient resolvents are the subject of Section 7. In principle, resolvents can be used to intersect non-planar curves and to find base points on rational surfaces. In practice, current resolvent methods are way too inefficient to be applied in practical geometric modeling applications. Thus, like resultants, resolvents are an engaging and effective theoretical tool, but much work remains to be done on efficiency and numerical stability before resolvents can be used in practical applications.

Section 8 is devoted to surface classification and feature extraction. Much insight is gained in geometric modeling from the classification of different surface types in algebraic geometry. But when we want to extract geometric features – axes, centers of symmetry, foci – from rational parameterizations, algebraic geometry seems to lack effective computational tools. Nor does algebraic geometry supply any intuition into curvature formulas for algebraic or rational surfaces. Geometric modeling is often interested in metric properties of surfaces; algebraic geometry is mainly concerned with affine or projective invariants. Thus, whereas algebraic geometry provides insight into the different kinds of surfaces that are available as tools for geometric modeling, the computational interests of the two disciplines concerning surfaces do not at all coincide.

All of my concerns and conclusions are summarized in Section 9, where I provide as well some hope and some challenges for future research that may help to resolve the role played by effective algebraic geometry in geometric modeling.

2 The Canonical Problem: Solving Systems of Polynomial Equations

Since we want to investigate the efficacy of computational algebraic geometry in geometric modeling, it is natural to ask: what important computational problems in geometric modeling can algebraic geometry solve more efficiently than standard numerical methods?

Many computational problems in geometric modeling can be reduced to solving systems of polynomial equations. For example, in geometric modeling curves and surfaces are typically represented by polynomial equations; therefore intersection algorithms for curves and surfaces reduce to finding common roots of polynomials. Thus finding common roots of polynomial equations is one of the canonical problem in geometric modeling. Polynomial root finding is also basic to many other branches of computational science and engineering, including robotics, computational biology, and coding theory.

What then is the best way to find the roots for a system of polynomial equations? There are many variants of this problem, depending on which roots are of interest in the application. For example, we might be interested in finding all the complex roots, or all the real roots, or all the real roots inside a bounding box, or even in just one real root inside a bounding box.

In geometric modeling, we are usually interested in all the real roots inside some bounded domain. What does algebraic geometry have to say about the solution to this problem? The main insight from algebraic geometry concerning roots of polynomial equations is provided by Bernstein's theorem [15, 47].

Bernstein's Theorem

Consider a collection of polynomials $F_1(x_1, ..., x_N), ..., F_{N+1}(x_1, ..., x_N)$. If the number of roots is finite, then

$$\#Complex\ Roots =$$
$$N! \times (Mixed\ Volume\ of\ the\ associated\ Newton\ Polygons) \quad (1)$$

If there are infinitely many roots, then the right hand side of Equation (1) provides a bound on the number of connected components of the roots.

As a consequence of Bernstein's theorem, even if both the degree and the number of variables is of moderate size, say of order 10, the number of isolated complex roots can be very large, roughly of order 10^8. Moreover, connected components may also appear. Thus algebraic geometry does indeed provide a good deal of insight into the complexity of the root finding problem.

What about computation? Effective algebraic geometry offers several techniques for solving systems of polynomial equations, including Grobner bases, resultants, and homotopy continuation. To judge the effectiveness of these tools, we need to compare and contrast these techniques with standard numerical methods such as binary search, Bezier subdivision, and Newton iteration.

The current state of the art in solving for the bounded real roots of a large system of polynomial equations is due to Zhang, who can solve for the real roots inside a unit hypercube of 9 equations in 9 variables, each variable of degree at most 2 [61]. Zhang can solve these equations in less than 10 minutes on a standard PC using Bezier subdivision. Problems of this size arise in computer aided drug design, where the short term goal is to solve 15 equations in 15 variables, where each variable is of degree at most 2. Much larger polynomial systems with hundreds of equations in hundreds of unknowns appear in computational biology in protein folding problems.

At present no technique from computational algebraic geometry comes even close to solving such systems of equations. The problem with current methods from effective algebraic geometry such as Grobner bases, resultants, and homotopy continuation is that these methods are built to find all the complex roots; at present these techniques, unlike Bezier subdivision, cannot be fine tuned to look only for bounded real roots. Since we know from Bernstein's theorem that the number of complex roots is often huge, these methods either consume way too much memory or they must run for a very long time.

Thus while algebraic geometry provides insight into the complexity of solving systems of polynomial equations, the computational tools developed by algebraic geometry are nowhere near competitive with numerical methods such as Bezier subdivision already available in geometric modeling. In effect, the computational approaches of algebraic geometry are in reality theoretical tools for proving theorems by concrete methods rather than abstract techniques.

We should not confuse concrete theoretical tools with effective computational methods. To reconfirm this conclusion, let us look at more examples.

3 Intersecting Rational Curves and Surfaces

A large part of geometric modeling depends on intersecting rational curves and surfaces. Yet, although a good deal of work has been done on this intersection problem both in algebraic geometry and in geometric modeling, very few comparative studies are available.

About twenty years ago, Sederberg and Parry compared resultant methods to Bezier subdivision for intersecting planar rational curves [52]. What they found was that for degree ≤ 3 resultant techniques are faster than Bezier subdivision, but for degree ≥ 4 Bezier subdivision is faster than resultants. These results are consistent with what we have already discussed about solving systems of polynomials equations, but here the conclusions are even more dramatic: algebraic techniques are more efficient than numerical methods for intersecting rational curves only for curves of very low degree. Perhaps in twenty years improvements in constructive algebraic methods have shifted these bounds and algebraic methods are now superior for higher degrees, but I know of no recent studies that either confirm or contradict these earlier conclusions.

The simplest curved surfaces that appear in geometric modeling are quadric surfaces, especially spheres, cylinders and cones. Many research groups have implemented algebraic methods to intersect quadric surfaces. From algebraic geometry (Bernstein's theorem), we know that three quadric surfaces can have up to eight real intersection points. In the early 1980's General Motors implemented homotopy continuation to compute these intersection points [44]. In the early 1990's [11] applied Macaulay's resultant to calculate the intersections of three quadric surfaces. Though both groups were successful, neither group reports any comparisons with numerical methods. Nor despite recent improvements in algebraic techniques such as homotopy continuation or Grobner bases have I heard or seen any contemporary comparisons of algebraic techniques vs. numerical methods for intersecting three quadric surfaces.

Many individual researchers and research groups [40, 48, 43, 45, 25, 22, 55] have applied tools from algebraic geometry to compute the intersection curve between two arbitrary quadric surfaces. Here a complete classification of the geometry and topology of the intersection curve along with degenerate cases is available from algebraic geometry, and explicit parameterizations of the intersection curve are also provided. Thus the analysis of the intersection curve between two quadric surfaces is one of the great successes of algebraic geometry in geometric modeling.

Nevertheless, even this success has drawbacks. Quadric surfaces form a closed system. Techniques that work well for quadric surfaces do not generally extend to higher order surfaces. But quadric surfaces by themselves are far from sufficient for the purposes of geometric modeling. The geometric coverage of quadric surfaces is quite limited; higher order surfaces are typically required to model industrial parts.

For this reason many companies – Boeing, Volkswagen, SDRC, Alpha-1 – have developed geometric modeling systems based on rational Bezier or rational B-spline surfaces. Bicubic patches are the most popular surfaces in these systems. Generic bicubic patches have implicit degree 18, so the intersection curve between two generic bicubic patches has degree $18^2 = 324$. To my knowledge, no industrial or academic geometric modeling system has attempted to implement algebraic techniques to compute intersection curves for bicubic patches; the degrees are just too large. Typically to compute intersections, these systems use numerical methods such as subdivision for Bezier surfaces or knot insertion for B-splines. Thus once again algebraic geometry provides insight into the complexity of the intersection problem rather than practical computational tools for calculating these intersections.

4 Implicitization

One problem that has received a good deal of attention lately is implicitization [2, 3, 6, 4, 13, 14, 16, 20, 33, 50]: given a rational curve or surface, find the corresponding implicit equation. Applications of implicitization are purported to include algorithms for intersecting rational curves and surfaces, ray casting procedures for rendering rational surfaces, and inside-outside tests for solid modeling. We have already seen that intersection algorithms for rational curves and surfaces based on algebraic techniques are generally much less efficient than numerical methods, but the other two applications may have some merit.

Algebraic geometry provides several implicitization methods, including resultants – classical, sparse, and residual – Grobner bases, syzygies (known in geometric modeling as the method of moving surfaces [49, 62]), and the method of undetermined coefficients. Algebraic geometry also provides insight by telling us that if there are no base points, then the expected degree of the implicit equation is the mixed volume of the Newton polygons of the parameterizing polynomials. Base points are parameter values where all the rational coordinate functions evaluate to $0/0$. Base points reduce the expected implicit degree, and may also cause an implicitization method to fail.

In terms of actual computation, Grobner bases are generally way too slow (at least for bicubic patches) and undetermined coefficients are way too inefficient. Thus we are left with resultants for which there are many possible formulations, and syzygies for which the theoretical foundations are not yet complete. Approximate implicitization methods have also been considered [20, 33], but these methods are numerical rather than algebraic.

Though several computational tools for implicitization are provided by algebraic geometry, precise quantitative measurements concerning the relative speed and numerical stability of these techniques are not yet available. Apparently much work remains to be done to determine the reliability and utility of these implicitization methods. Nevertheless, in the next two sections we shall look more closely both at resultants and at syzygies in order to determine what algebraic geometry has contributed to geometric modeling concerning these two potential computational techniques.

5 Resultants

A resultant is a polynomial in the coefficients of a polynomial system that vanishes if and only if the polynomials in the system have a common root. Resultants have many formulations, but typically resultants are represented either by determinants [57] or by ratios of determinants [17] or by GCD's of determinants [15, 27].

The two most common representations for the resultant of two polynomials in one variable are the Sylvester resultant and the Bezout resultant [9]. Each entry of the Sylvester resultant is either zero or a coefficient of one of the original polynomial equations; the entries of the Bezout resultant are more complicated, but the Bezout resultant employs a much smaller matrix. Resultants for two polynomials in one variable can also be represented by hybrids of the Sylvester and Bezout resultants [51] as well as by companion matrices [58].

Resultants for multivariate polynomials have at least a half dozen different representations many of which are valid only in special cases. The Sylvester formulation represents the resultant by a single determinant each of whose entries is either zero or one of the coefficients of the original polynomial equations [38]. The Dixon formulation uses smaller matrices than the Sylvester representation, but employs more complicated Bezoutian entries [19]. Macaulay represents the resultant as the ratio of two determinants whose entries are similar to the Sylvester resultant [41]. There are also hybrids representations that consist of a mixture of Sylvester and Bezout type entries [60, 37]. For low degree polynomial systems there are special formulations such as the Jacobian representation [10]. And there exist other residual formulations as well to handle other general cases including base points [5].

The polynomial representing the resultant is often quite large. Therefore to compute the resultant, Manocha and Canny provide a term expansion technique involving multivariate polynomial interpolation together with modular arithmetic [42]. Matrices representing the resultant are also often quite large. Explicit formulas as well as efficient recursive computations for the entries of the Bezout and Dixon resultants are provided by [12].

Despite all this theory, there are often more questions than answers regarding actual computations with resultants. Natural questions include: what are the relative speed and numerical stability of different resultant formulations? Speed and stability are clearly key issues for practical implementations. Specific resultant formulations also raise theoretical questions. For example, it is known that when the Dixon determinant vanishes identically, the maximal minor is a multiple of the resultant [23]. But it is not known how to find the maximal minor; nor is it known how to predict the extraneous factors in the maximal minor.

The only work so far on numerical stability for resultants is [59]. Winkler and Goldman show that in the univariate setting Sylvester's resultant is numerically unstable; Bezout's resultant is inherently more reliable than Sylvester's resultant. These facts about numerical stability are crucial in practice for geometric modeling applications.

Conditions under which the Dixon determinant generates the exact resultant are provided by [8]. In cases where the Dixon matrix does not generate the exact resultant, the maximal minor of the Dixon resultant for some simple Newton polygons is found by [26]. Foo and Chionh can also predict the Dixon extraneous factors for these simple Newton polygons.

These three papers – one on numerical stability in the univariate setting, one on specific conditions for when the Dixon determinant generates the exact resultant, and one on finding the maximal minor and the extraneous factor for the Dixon resultant in the multivariate setting – together with the paper of Manocha and Canny on term expansion seem to be all that is known about numerical computations with resultants. Surely a lot more study on computational issues needs to be done before resultants can be considered as a practical tool for numerical computation in geometric modeling applications. Yet these investigations are not part of the research agenda of algebraic geometry.

Rather than an analysis of the numerical properties of resultants, what algebraic geometry provides is insight into when certain specific resultant formulations actually exist. For example, [37] uses sheaf cohomology to establish that a determinant formulation for the resultant always exists for unmixed systems of bivariate polynomials with arbitrary Newton polygons. These resultants are hybrids with complicated structures: some rows and columns are in the style of Sylvester, whereas others are in the style of Bezout.

Pure Sylvester style resultants – resultants where each entry of the resultant matrix is either zero or a coefficient of one of the polynomial equations – do not always exist, even in the case of polynomials of a fixed total degree where the Newton polygon is a triangle. Nevertheless, because of their simple structure, it would be desirable to know when such pure Sylvester style resultants do exist. Moreover, whenever pure Sylvester style resultants exist, pure Bezout style resultants – resultants generated by Bezoutian matrices (see, for example, [5]) – must also exist. Bezout resultants are more compact than the hybrid resultants of Khetan, and their structure may be simpler as well. Thus we seek Newton polygons with pure Sylvester style resultants.

For which Newton polygons do Sylvester style resultants exist? The general answer is not known even in the bivariate setting. What is known is that for triangles (total degree) the answer is never for degree>1; for rectangles (bidegree), the answer is always [19]; and for hexagons the answer is sometimes [38].

To construct a Sylvester style resultant for three bivariate polynomials f, g, h with the same Newton polygon A, we typically seek a collection of monomials M such that the determinant of the matrix formed by the coefficients of the polynomials Mf, Mg, Mh is the resultant. The monomials M are called a m*ultiplying set* for the Newton polygon A. Thus we ask: for which Newton polygons A do multiplying sets M exist?

The multiplying set must satisfy the following three constraints [38]:

$$\#M = 2Area(ConvexHull(A))$$
$$\#(M \oplus A) = 3(\#M) = 6Area(ConvexHull(A))$$
$$(\oplus\ denotes\ Minkowski\ sum)$$
$$Det(Mf, Mg, Mh) \neq 0.$$

The first constraint says that the degree of the resultant in the coefficients of any of the original polynomials must be $2Area(ConvexHull(A))$. The second constraint is simply that the matrix formed by the coefficients of the polynomials Mf, Mg, Mh must be a square matrix (#rows=#columns). The final constraint is that the determinant does not vanish identically. When the Newton polygon of M has the same shape as the Newton polygon of f, g, h – that is, when the sides of the Newton polygon of M are parallel to the sides of the Newton polygon A – the first two constraints reduce to two quadratic Diophantine equations [Khetan et al 2004]. These equations can often be solved by a computer using a simple search. The difficult part is finding explicit examples to show that $Det(Mf, Mg, Mh) \neq 0$. Using sheaf cohomology, [38], show that $Det(Mf, Mg, Mh) \neq 0$ can be verified without constructing canonical examples. Thus once again, algebraic geometry provides insight into a theoretical question that may have practical applications in geometric modeling.

6 Implicitization Revisited: The Method of Moving Surfaces

Resultants can be applied to find the implicit equation of a rational surface. But resultants vanish in the presence of base points. The method of moving surfaces was introduced by [49] to find the implicit equation for a rational surface in the presence of base points. We shall give a brief description of this method here; for further details and insights, see [13, 14, 62].

A moving plane of bidegree (m, n) is an expression of the form

$$(A_{m,n}s^m t^n + \cdots + A_{0,0})x + (B_{m,n}s^m t^n + \cdots + B_{0,0})y$$
$$+(C_{m,n}s^m t^n + \cdots + C_{0,0})z + (D_{m,n}s^m t^n + \cdots + D_{0,0})w = 0 \tag{2}$$

or equivalently

$$(A_{m,n}x+B_{m,n}y+C_{m,n}z+D_{m,n}w)s^m t^n+\cdots+(A_{0,0}x+B_{0,0}y+C_{0,0}z+D_{0,0}w) = 0.$$

Similarly a moving quadric of bidegree (m, n) is an expression of the form

$$(A_{m,n}s^m t^n + \cdots + A_{0,0})x^2 + \cdots + (J_{m,n}s^m t^n + \cdots + J_{0,0})w^2 = 0. \tag{3}$$

or equivalently

$$(A_{m,n}x^2 + \cdots + J_{m,n}w^2)s^m t^n + \cdots + (A_{0,0}x^2 + \cdots + J_{0,0}w^2) = 0.$$

For each value of (s, t), a moving plane is the implicit equation of a plane; different values of (s, t) represent different planes. Similarly for each value of (s, t), a moving quadric is the implicit equation of a quadric surface; again different values of (s, t) represent different quadric surfaces.

Moving surfaces are related to syzygies. A moving plane (2) is said to follow a rational surface

$$x = \frac{x(s,t)}{w(s,t)}, y = \frac{y(s,t)}{w(s,t)}, z = \frac{z(s,t)}{w(s,t)}, \tag{4}$$

if

$$(A_{m,n}s^m t^n + \cdots + A_{0,0})x(s,t) + (B_{m,n}s^m t^n + \cdots + B_{0,0})y(s,t)$$
$$+(C_{m,n}s^m t^n + \cdots + C_{0,0})z(s,t) + (D_{m,n}s^m t^n + \cdots + D_{0,0})w(s,t) \equiv 0$$

Thus a moving plane that follows a rational surface is equivalent to a syzygy of the polynomials $x(s,t), y(s,t), z(s,t), w(s,t)$ representing the rational parameterization. Similarly a moving quadric (3) is said to follow the rational surface (4) if

$$(A_{m,n}s^m t^n + \cdots + A_{0,0})x^2(s,t) + \cdots + (J_{m,n}s^m t^n + \cdots + J_{0,0})w^2(s,t) \equiv 0.$$

Thus a moving quadric that follows a rational surface is a syzygy of the ten polynomials $x^2(s,t), \ldots, w^2(s,t)$.

To implicitize a rational surface of bidegree (m,n), consider moving planes of bidegree $(2m-1, n-1)$ that follow the rational surface:

$$\begin{aligned}(A_{2m-1,n-1}s^{2m-1}t^{n-1} + \cdots + A_{0,0})x(s,t) + \\ (B_{2m-1,n-1}s^{2m-1}t^{n-1} + \cdots + B_{0,0})y(s,t) + \\ (C_{2m-1,n-1}s^{2m-1}t^{n-1} + \cdots + C_{0,0})z(s,t) + \\ (D_{2m-1,n-1}s^{2m-1}t^{n-1} + \cdots + D_{0,0})w(s,t) \equiv 0.\end{aligned}$$

Treating the parameters $A_{ij}, B_{ij}, C_{ij}, D_{ij}$ as undetermined coefficients leads to $6mn$ homogeneous linear equations in $8mn$ unknowns. Hence from linear algebra there exist at least $2mn$ linearly independent solutions. Placing these solutions in a matrix and taking the determinant generates

$$\begin{vmatrix} A^1_{2m-1,n-1}x + \cdots + D^1_{2m-1,n-1}w & \cdots & A^1_{0,0}x + \cdots + D^1_{0,0}w \\ \vdots & \vdots & \vdots \\ A^{2mn}_{2m-1,n-1}x + \cdots + D^{2mn}_{2m-1,n-1}w & \cdots & A^{2mn}_{0,0}x + \cdots + D^{2mn}_{0,0}w \end{vmatrix} = 0$$

a polynomial equation of degree $2mn$ in x, y, z, w. When the rational surface has no base points, this polynomial is the implicit equation of the rational surface.

When base points are present, we can apply the method of moving quadrics to find the implicit equation. To implicitize a rational surface of bidegree (m,n), consider moving quadrics of bidegree $(m-1, n-1)$ that follow the rational surface:

$$\begin{aligned}(A_{m-1,n-1}s^{m-1}t^{n-1} + \cdots + A_{0,0})x^2(s,t) + \cdots \\ +(J_{m-1,n-1}s^{m-1}t^{n-1} + \cdots + J_{0,0})w^2(s,t) \equiv 0\end{aligned}$$

Now treating the parameters $A_{ij}, \cdots, J_{ij}$ as undetermined coefficients leads to $9mn$ homogeneous linear equations in $10mn$ unknowns. Hence there exist at least mn linearly independent solutions. Placing these solutions in a matrix and taking the determinant generates

$$\begin{vmatrix} A^1_{m-1,n-1}x^2+\cdots+J^1_{m-1,n-1}w^2 & \cdots & A^1_{0,0}x^2+\cdots+J^1_{0,0}w^2 \\ \vdots & \vdots & \vdots \\ A^{mn}_{m-1,n-1}x^2+\cdots+J^{mn}_{m-1,n-1}w^2 & \cdots & A^{mn}_{0,0}x^2+\cdots+J^{mn}_{0,0}w^2 \end{vmatrix} = 0,$$

a polynomial equation of degree $2mn$ in x, y, z, w. When the rational surface has no base points, this polynomial is the implicit equation of the rational surface. When base points are present, some of the quadratic rows reduce to linear expressions – that is, some of the moving quadrics become moving planes – and in many cases the determinant still represents the implicit equation of the rational surface.

The method of moving quadrics has certain advantages over resultant techniques. First, the method employs smaller matrices than resultants – $mn \times mn$ vs. $2mn \times 2mn$ for surfaces of bidegree (m, n). Second, the method often works even in the presence of base points. Here for each base point counted with proper multiplicity, one row of the moving quadric matrix reduces to a row of moving planes. Moreover, the method of moving quadrics is valid for total degree as well as for bidegree rational surfaces, though for surfaces of a fixed total degree some of the rows must always be filled in with moving planes. Finally, the method of moving quadrics uses only straightforward tools from linear algebra. Therefore the method of moving quadrics is easier for members of the geometric modeling community to understand than competing techniques such as residual resultants [5] or approximation complexes [6].

The method of moving quadrics is a method of undetermined coefficients. But moving quadrics is faster than naive undetermined coefficients, since to use the methods of moving quadrics we need to solve $O(mn)$ equations in $O(mn)$ unknowns, whereas to use naive undetermined coefficients we would have to solve $O(m^3n^3)$ equations in $O(m^3n^3)$ unknowns.

Why do these moving surface implicitization techniques work? Again algebraic geometry provides the key insights. When there are no base points, both methods – moving planes and moving quadrics – generate a polynomial in x, y, z of the same degree as the implicit equation, and by construction this polynomial vanishes on the rational surface. The key point is that this expression is not identically zero.

The method of moving planes can be shown to be equivalent to implicitization by resultants [51]. The method of moving quadrics is more subtle. When there are no base points, [16] use sheaf cohomology to show that the method will always generate the implicit equation provided there are no moving planes of bidegree $(m-1, n-1)$ that follow the surface. Moreover, [4, 6, 14] extend these arguments to show that the method of moving quadrics works for surfaces of fixed total degree even in the presence of base points provided that the base points are local complete intersections. Thus algebraic geometry provides the insight underlying the computations developed by geometric modeling.

7 Resolvents

A *resultant* is a single polynomial condition that vanishes when $N+1$ polynomials in N variables have a common root. *Resolvents* are a collection of polynomial conditions that vanish when M polynomials in N variables ($M > N+1$) have a common root.

Resolvents have several potential applications in geometric modeling. For example, we could use resolvents to represent non-planar rational curves

$$x = \frac{x(t)}{w(t)}, \quad y = \frac{y(t)}{w(t)}, \quad z = \frac{z(t)}{w(t)}$$

as the intersection of implicit algebraic surfaces by computing the resolvents of the three polynomials $x(t) - xw(t)$, $y(t) - yw(t)$, $z(t) - zw(t)$. Thus resolvents could be applied to implicitize and intersect non-planar rational curves [29, 31]. In the bivariate setting, resolvents could be employed to detect base points on a rational surface

$$x = \frac{x(s,t)}{w(s,t)}, \quad y = \frac{y(s,t)}{w(s,t)}, \quad z = \frac{z(s,t)}{w(s,t)}$$

by detecting common roots of the polynomials $x(s,t), y(s,t), z(s,t), w(s,t)$ representing the parameterization. For example, if we want to know if a rational quadratic parameterization represents a quadric surface, then we need to determine if the parameterization has two base points.

Kronecker's method is a technique that uses resultants for computing resolvents. To compute the resolvents for three univariate polynomials $F(t)$, $G(t)$, $H(t)$ of degree D, Kronecker computes the resultant $Res(F(t), G(t) + uH(t))$, a polynomial in u of degree D. This polynomial in u vanishes identically if and only if $F(t), G(t), H(t)$ have a common root. Hence the coefficients of $u^k, k = 0, \cdots, D$, are resolvents for the polynomials $F(t), G(t), H(t)$. Thus Kronecker's method generates $D+1$ resolvent conditions for three univariate polynomials of degree D. Similarly Kronecker's method can be applied to find resolvents in the multivariate setting. Kronecker's methods is effective, but not efficient. Alternative resolvent methods due to [34, 21] and others often lead to fewer conditions.

Consider, for example, the simple case of three quadratic polynomials

$$F(t) = a_2 t^2 + a_1 t + a_0,$$

$$G(t) = b_2t^2 + b_1t + b_0,$$
$$H(t) = c_2t^2 + c_1t + c_0.$$

Kronecker's method generates three conditions from $Res(F(t), G(t)+uH(t)) \equiv 0$:

1. $Res(F(t), G(t)) = 0 \quad \Leftrightarrow \quad F$ and G have a common root
2. $Res(F(t), H(t)) = 0 \quad \Leftrightarrow \quad F$ and H have a common root
3. $2a_2^2b_0c_0 - a_1a_2b_1c_0 + a_1^2b_2c_0 - 2a_0a_2b_2c_0 - a_1a_2b_0c_1$
   $+2a_0a_2b_1c_1 - a_0a_1b_2c_1 + a_1^2b_0c_2 - 2a_0a_2b_0c_2 - a_0a_1b_1c_2 + 2a_0^2b_2c_2 = 0.$

On the other hand, two conditions suffice to determine if three quadratic polynomials F, G, H have a common root.

1. $Res(F(t), G(t)) \equiv Det \begin{pmatrix} a_2 & 0 & b_2 & 0 \\ a_1 & a_2 & b_1 & b_2 \\ a_0 & a_1 & b_0 & b_1 \\ 0 & a_0 & 0 & b_0 \end{pmatrix} = 0 \quad \Leftrightarrow \quad F$ and G have a common root
2. $Det(F(t), G(t), H(t)) \equiv Det \begin{pmatrix} a_2 & b_2 & c_2 \\ a_1 & b_1 & c_1 \\ a_0 & b_0 & c_0 \end{pmatrix} = 0 \quad \Leftrightarrow \quad F, G, H$ are linearly dependent

In this simple example, it may be possible to recover the two simpler conditions from the three Kronecker conditions, but, in general, algebraic geometry does not seem to provide efficient computational methods for expressing resolvents. Surely we do not need $D + 1$ algebraic surfaces to implicitize a non-planar rational curve of degree D. What then is the minimum number of surfaces required? What are their minimal degrees and how do we find them? The existence of resolvents is guaranteed by the theory, but before resolvents can be used effectively in geometric modeling applications, additional research needs to be done to determine the fastest, most efficient, most compact, most numerically stable approach to resolvents.

8 Surface Classifications and Surface Features

Algebraic geometry distinguishes many different types of surfaces: rational, total degree, bidegree, toric, del Pezzo, and many others. Each of these surface types incorporates special geometric properties that have potential applications in geometric modeling. Tensor product bidegree surface patches are the mainstay of geometric modeling because these surface patches are easily integrated with rectangular meshes [24, 28]. Triangular total degree surface patches are the next most common surfaces in geometric modeling because these surface patches are well suited for handling scattered data [24, 28]. Recently toric surface patches with arbitrary Newton polygons have entered geometric modeling in order to build multisided patches and to fill n-sided holes [36, 39]. Geometric insights – Segre, Veronese, and toric embeddings – and algebraic tools – Macaulay, Dixon, and toric resultants – originate from within algebraic geometry, but the computational techniques – Bezier representations as well as subdivision, evaluation, and differentiation algorithms – arise from inside geometric modeling.

Surfaces have many geometric features such as curvatures, axes and centers of symmetry, and foci. For quadric surfaces, schemes are known for extracting standard features such as the type, center, axes and foci from their implicit representation [46], but in geometric modeling the standard representation is most often rational parametric rather than implicit polynomial. To extract these quadric surface features from rational representations, we must implicitize and then compute. But as we have seen, implicitization for surfaces with base points can be complicated. Recently, straightforward algorithms have been developed for extracting the geometric features of conic sections – type, center, axes, foci – from their rational parameterizations [55], but direct computational tools for extracting geometric features for quadric surfaces from rational representations are at present unavailable.

Gaussian and mean curvature formulas for surfaces both for implicit representations and for parametric representations are known from differential geometry [53], [54]. Algebraic geometry provides neither geometric insight nor computational tools for extracting these basic features.

Thus, whereas algebraic geometry provides insight into the different surface types that are available as tools for geometric modeling, the computational interests of the two disciplines concerning surfaces are often distinct. What is the reason for this dichotomy? Geometric modeling is frequently interested in metric properties of surfaces, but algebraic geometry is concerned primarily with affine or projective invariants. Perhaps this distinction helps to explain why algebraic geometry does not provide effective tools for computing many of the surface features of interest in geometric modeling, even though algebraic geometry does indicate effectively which surface types are worth further study.

9 Conclusions and Some Open Problems for Future Research

The thesis of this paper is that it is a mistake to confound constructive theoretical tools with effective computational techniques. I have endeavored to provide many examples to substantiate this premise. We have seen that for problems of even moderate size constructive tools from algebraic geometry such as Grobner bases, resultants, and homotopy continuation for solving systems of polynomial equations are not competitive with numerical methods from geometric modeling such as Bezier subdivision.

Rather than efficient computation, the main contribution of algebraic geometry to geometric modeling has been insight into the complexity of the computational problems at hand. For example, Bernstein's theorem provides the number of complex roots of a system of polynomial equations. This result bounds the number of real intersections of two rational curves and the degree of the real intersection curve of two rational surfaces. Bernstein's theorem also furnishes the degree of the implicit equation of a rational curve or surface.

Algebraic geometry has also provided a service to geometric modeling by supplying proofs for numerical techniques such as implicitization by the method of moving surfaces developed from within geometric modeling. What algebraic geometry has not provided to geometric modeling is efficient numerical techniques. Fashions in the philosophy of mathematics vary from transcendent to constructive, but constructive philosophical trends in mathematics do not necessarily translate into effective computational tools for engineers: there is a big difference between constructive theoretical tools and effective computational techniques.

Nevertheless, these conclusions should not be taken to mean that constructive methods are not at all useful. To the contrary, computers are potentially powerful theoretical tools. Computation can be allied to intuition; even inefficient computations can lead to novel theoretical insights.

There may also be isolated areas where algebraic geometry furnishes computation as well as insight to geometric modeling. Finding singularities [7] and rendering algebraic curves and surfaces based on an analysis of singularities [1] are two such potential applications. Perhaps too the proof of the validity of the method of moving quadrics for the special case presented in [4] (total degree surfaces with base points that are local complete intersections) provides a more efficient implicitization algorithm than the method of moving quadrics based on standard linear algebra.

Moreover, new, more efficient computational approaches to Grobner bases or homotopy continuation or resultants may yet prove to be effective in geometric modeling. Much of the original work and many of the original comparative studies in geometric modeling are now dated – resultants [52], homotopy continuation [44], Grobner bases [32] – and it may well be time to reevaluate newer, more efficient computational methods embodied in more modern software. In addition, almost no comparative studies exist on the efficiency or numerical stability of the various resultant and resolvent formulations.

I will close then with some problems for future research that may help to fill in some holes in the current theory as well as bridge the gap between the constructive tools of algebraic geometry and the effective numerical methods required by geometric modeling.

1. *Generic Resultants.* Explore the relative speed, efficiency, and numerical stability of the different resultant formulations, especially in the bivariate setting.
2. *Sylvester Resultants.* Classify the Newton polygons for which exact Sylvester resultants (and exact Bezout resultants) exist.
3. *Dixon Resultants.* When the Dixon resultant vanishes identically, determine both the maximal minor and the extraneous factor.
4. *Resolvents.* Find an efficient, compact, numerically stable representation for resolvents. Show how to use resolvents to determine efficiently the number and multiplicity of the base points for a rational surface.
5. *Implicitization.* Investigate the relative speed, efficiency, and numerical stability of the different implicitization techniques for rational surfaces, including Grobner bases, resultants and syzygies.
6. *Extraction of Surface Invariants.* Develop techniques for finding axes, centers of symmetry, foci and other geometric features of rational surfaces directly from their rational parameterizations.
7. *Floating Point Computations.* For speed, most computations in geometric modeling are performed using floating point rather than exact arithmetic. Analyze the effect of floating point computations on various techniques from constructive algebraic geometry, including resultants, resolvents, Grobner bases, and homotopy continuation. Determine whether these floating point computations are sufficiently stable for geometric modeling applications.
8. *Comparative Studies.* Compare constructive techniques based on the latest implementations of constructive algebraic geometry – Grobner bases, homotopy continuation, resultants – to numerical methods such as Bezier subdivision from geometric modeling. Determine which type and which size problems are most amenable to techniques from constructive algebraic geometry and which problems are best suited to numerical methods from geometric modeling. Consider, in particular, implicitizing and intersecting rational curves and surfaces, as well as finding the real roots of a system of polynomials inside a bounded domain.

Acknowledgements:

This work is partially supported by NSF grants CCF-0203318 and INT-0421771. I would also like to thank David Cox and Falai Chen for reading a preliminary version of this manuscript and making some helpful comments and suggestions. All the opinions expressed in this paper as well as any errors that still remain are, of course, entirely my own.

References

1. Bodnar, G. and Schicho, J. (2000), A computer program for the resolution of singularities, in H. Hauser, editor, *Resolution of Singularities*, Vol. 181 of Progr. Math., pp. 231-238. Birkhauser.
2. Busé, L. (2001), Residual resultant over the projective plane and the implicitization problem, Proceedings of ISSAC 2001, pp. 48-55.
3. Busé, L. and Chardin, M. (2004), Implicitizing rational hypersurfaces using approximation complexes, submitted to *Journal of Symbolic Computation*.
4. Busé, L., Cox, D., and D'Andrea, C. (2003), Implicitization of surfaces in P^3 in the presence of base points, *Journal of Algebra and its Applications*, Vol. 2, pp. 189-214.
5. Busé, L., Elkadi, M., and Mourrain, B. (2001), Resultant over the residual of a complete intersection, *Jour. of Pure and Applied Algebra*, Vol. 164, pp. 35-57.
6. Busé, L. and Jouanolou, J.P. (2003), On the closed image of a rational map and the implicitization problem, *Journal of Algebra*, Vol. 265, pp. 312-357
7. Chen, F. and Wang, W. (2004), Computing the singular points of a planar rational curve using the μ-basis, preprint.
8. Chtcherba, A. and Kapur, D. (2000), Conditions for exact resultants using the Dixon formulation, *International Symposium on Symbolic and Algebraic Computation (ISSAC 2000)*, St. Andrews, Scotland.
9. Chionh, E. and Goldman, R. (1995a), Elimination and resultants part 1: Elimination and bivariate resultants, *IEEE Computer Graphics and Applications*, Vol. 15, pp. 69-77.
10. Chionh, E. and Goldman, R. (1995b), Elimination and resultants part 2: Multivariate resultants, *IEEE Computer Graphics and Applications*, Vol. 15, pp. 60-69.
11. Chionh, E., Miller, J. and Goldman, R. (1991), Using multivariate resultants to find the intersection of three quadric surfaces, *Transactions on Graphics*, Vol. 10, pp. 378-400.
12. Chionh, E., Zhang, M. and Goldman, R. (2002), Fast computation of the Bezout and Dixon resultant matrices, *Journal of Symbolic Computation*, Vol. 33, pp. 13-29.
13. Cox, D. (2001), Equations of parametric curves and surfaces via syzygies, in *Symbolic Computation: Solving Equations in Algebra, Geometry and Engineering*, AMS Contemporary Mathematics, Vol. 286, pp. 1-20.
14. Cox, D. (2003), Curves, surfaces and syzygies, in *Topics in Algebraic Geometry and Geometric Modeling*, edited by R. Goldman and R. Krasauskas, AMS Contemporary Mathematics, Vol. 334, pp. 131-150.
15. Cox, D., Little, J. and O'Shea, D. (1998), *Using Algebraic Geometry*, Springer, New York.
16. Cox, D., Zhang, M. and Goldman, R. (2000), On the validity of implicitization by moving quadrics for rational surfaces with no base points, *Journal of Symbolic Computation*, Vol. 29, pp. 419-440.
17. D'Andrea, C. and Dickenstein, A. (2001), Explicit formulas for the multivariate resultant, *Jour. Pure Appl. Algebra*, Vol. 164, pp. 59-86.
18. Davis, P. and Hersh, R. (1986). *Descartes' Dream: The World According to Mathematics*, Harcourt Brace Jovanovich, San Diego.
19. Dixon, A. (1908), The eliminant of three quantics in two independent variables, *Proc. London Mathematical Society*, Vol. 6, pp. 49-69.

20. Dokken, T. (2001), Approximate implicitization, in *Mathematical Methods in CAGD*, T. Lyche and L. Schumaker (eds.), Vanderbilt University Press, Nashville.
21. Du, H. (1991), *New Resolvent Methods with Applications to Curves and Surfaces in Geometric Modeling*, Ph.D. Thesis, Department of Computer Science, University of Waterloo.
22. Dupont, L., Lazard, D., Lazard, S., and Petitjean, S. (2003), Near-optimal parameterization of the intersection of quadrics, *Symposium on Computational Geometry 2003*, pp. 246-255.
23. Emiris, I. and Mourrain, B. (1999), Matrices in elimination theory, *Journal of Symbolic Computation*, Vol. 28, pp. 3-44.
24. Farin, G. (2002), *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, Fifth Edition, Academic Press, Inc., San Diego.
25. Farouki, R., Neff, C., and O'Connor, M. (1989), Automatic parsing of degenerate quadric surface intersections, *ACM Transactions on Graphics*, Vol. 8, pp. 174-203.
26. Foo, M. and Chionh, E. (2004), Corner edge cutting and Dixon A-resultant quotients, *Journal of Symbolic Computation*, Vol. 37, pp. 101-119.
27. Gelfand, I. M., Kapranov, M. M. and Zelevinsky, A. V. (1994), Discriminants, Resultants and Multidimensional Determinants, Boston, Birkhauser.
28. Goldman, R. (2002), *Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling*, Morgan Kaufmann.
29. Goldman, R. (1985), The method of resolvents: A technique for the implicitization, inversion, and intersection of non-planar, parametric, rational cubic curves, *Computer Aided Geometric Design*, Vol. 2, pp. 237-255.
30. Goldman, R. and Sederberg, T. (1986), Algebraic geometry for computer-aided geometric design, *IEEE Computer Graphics and Applications*, Vol. 6, pp. 52-59.
31. Goldman, R. and Sederberg, T. (1987), An analytic approach to the intersection of all piecewise parametric rational cubic curves, *Computer-Aided Design*, Vol. 19, pp. 282-292.
32. Hoffmann, C. (1989), *Geometric and Solid Modeling: An Introduction*, Morgan Kaufmann.
33. Juettler, B. and Wurm, E. (2003), Approximate implicitization via curve fitting, in L. Kobbelt, P. Schroder, H. Hoppe (eds.), *Symposium on Geometry Processing, Eurographics/ACM Siggraph*, New York, pp. 240-247.
34. Kakie, K. (1976), The resultant of several homogeneous polynomials in two indeterminates, *Proceedings of AMS*, Vol. 54, pp. 1-7.
35. Karciauskas, K. (2003), Rational M-patches and tensor border patches, in *Topics in Algebraic Geometry and Geometric Modeling*, edited by R. Goldman and R. Krasauskas, AMS Contemporary Mathematics, Vol. 334, pp. 101-128.
36. Karciauskas, K. and Krasauskas, R. (2000), Comparison of different multisided patches using algebraic geometry, *Curve and Surface Design: Saint-Malo 1999*, P-J. Laurent, P. Sablonniere, and L. Schumaker (eds.), pp. 163-172, Vanderbilt University Press, Nashville.
37. Khetan, A. (2003), The resultant of an unmixed bivariate system, *Journal of Symbolic Computation*, Vol. 36, pp. 425-442.
38. Khetan, A., Song, N., and Goldman, R. (2004), Sylvester A-resultants for bivariate polynomials with planar Newton polygons, *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation (ISSAC)*, Santander, Spain.

39. Krasauskas, R. (2000), Toric surface patches, *Advances in Computational Mathematics*, Vol. 21, pp. 1-25.
40. Levin, J. (1979), Mathematical models for determining the intersections of quadric surfaces, *Computer Graphics and Image Processing*, Vol. 1, pp. 73-87.
41. Macaulay, F. (1902), Some formulae in elimination, *Proceedings London Math. Soc.*, Vol. 1, pp. 3-27.
42. Manocha, D. and Canny, J. (1993), Multipolynomial resultant algorithms, *Journal of Symbolic Computation*, Vol. 15, pp. 99-122.
43. Miller, J. (1987), Geometric approaches to nonplanar quadric surface intersection curves, *ACM Transactions on Graphics*, Vol. 6, pp. 274-307.
44. Morgan, A, (1983), A method for computing all solutions to systems of polynomial equations, *ACM Trans. on Math. Software*, Vol. 9, pp. 1-17.
45. Ocken, S., Schwartz, J. and Sharir, M. (1987), Precise implementation of CAD primitives using rational parametrizations of standard surfaces, Schwartz, Hopcroft, and Sharir, eds., *Planning, Geometry, and Complexity of Robot Motion*, Ablex Publishing Corporation, pp. 245-266.
46. Roe, J. (1993), *Elementary Geometry*, Oxford University Press.
47. Rojas, J. M. (2003), Why polyhedra matter in non-linear equation solving, in *Topics in Algebraic Geometry and Geometric Modeling*, edited by R. Goldman and R. Krasauskas, AMS Contemporary Mathematics, Vol. 334, pp. 293-320.
48. Sarraga, R. (1983), Algebraic methods for intersections of quadric surfaces in GMSOLID, *Computer Vision, Graphics and Image Processing*, Vol. 22, pp. 222-238.
49. Sederberg, T. and Chen, F. (1995), Implicitization using moving curves and surfaces, *Proceedings of Siggraph '95.*
50. Sederberg, T., Goldman, R., and Anderson, D. (1984), Implicit representation of parametric curves and surfaces, *Computer Vision, Graphics, and Image Processing*, Vol. 28, pp. 72-84.
51. Sederberg, T., Goldman, R., and Du, H. (1997), Implicitizing rational curves by the method of moving algebraic curves, *Journal of Symbolic Computation*, Vol. 23, pp. 153-175.
52. Sederberg, T. and Parry, S. (1986), A comparison of curve-curve intersection algorithms, *Computer-Aided Design*, Vol. 18, pp. 58–63.
53. Spivak, M. (1975), *A Comprehensive Introduction to Differential Geometry*, Vol. 3, Publish or Perish, Inc., Boston.
54. Stoker, J. (1969), *Differential Geometry*, John Wiley and Sons, New York.
55. Wang, W. and Goldman, R. (2004), Using invariants to extract geometric characteristics of conic sections from rational quadratic parametrizations, to appear in *The International Journal of Computational Geometry and Applications*.
56. Wang, W., Joe, B. and Goldman, R. (2003), Computing quadric surface intersections based on an analysis of planar cubic curves, *Graphical Models*, Vol. 64, pp. 335-367.
57. Weyman, J. and Zelevinsky, A. (1994), Determinantal formulas for multigraded resultants, *Jour. Algebraic Geometry*, Vol. 3, pp. 569-597.
58. Winkler, J. (2003), A companion matrix resultant for Bernstein polynomials, *Linear Algebra and Its Applications*, 362, pp. 153-175.
59. Winkler, J. and Goldman, R. (2003), The Sylvester resultant matrix for Bernstein polynomials, in *Curve and Surface Design: Saint-Malo 2002*, T. Lyche, M. Mazure and L. Schumaker (eds.), Nashboro Press, Brentwood, Tennessee, pp. 407-416.

60. Zhang, M., Chionh, E. and Goldman, R. (1998), Hybrid Dixon resultants, *Proceedings of the Eighth IMA Conference on the Mathematics of Surfaces*, August 1998, pp. 193-212.
61. Zhang, M., White, R., Wang, L., Kavraki, L., Goldman, R. and Hassett, B. (2005), Improving conformational searches by geometric screening, accepted to appear in the *Journal of Bioinformatics*.
62. Zheng, J., Sederberg, T., Chionh, E. and Cox, D. (2003), Implicitizing rational surfaces with base points using the method of moving surfaces, in *Topics in Algebraic Geometry and Geometric Modeling*, edited by R. Goldman and R. Krasauskas, AMS Contemporary Mathematics, Vol. 334, pp. 151-168.

Summary. This paper describes a method, exploiting approximation complexes, for computing the implicit equation of a parameterized hypersurface. The fundamental ingredients and properties used in this approach are recalled and illustrated on simple examples.