Journal for Geometry and Graphics
Volume 9 (2005), No. 1, 11–29.

Comparison of implicitization methods

Bohumír Bastl, František Ježek

Dept. of Mathematics, Fac. of Applied Sciences, Univ. of West Bohemia
Universitní 22, 306 14 Plzeň, Czech Republic
email: bastl@kma.zcu.cz

Abstract. This paper overviews several methods for implicitization of an algebraic variety (finding an implicit representation of a rational algebraic variety given by its parametric equations): the classical implicitization using Gröbner bases, the implicitization using resultants, polynomial interpolation, or moving curves and surfaces, and the direct implicitization method. All these methods are used for finding the implicit equation of NURBS curves and surfaces, which are typical objects in geometric modeling. At the end, the computational costs for finding the implicit equation needed by implementations of these different methods are compared.

Key Words: Implicitization methods, NURBS curves and surfaces
MSC 2000: 68W30, 13P10, 14Q05, 14Q10

1. Introduction

There are two standard ways of representing algebraic varieties, the *implicit representation* and the *parametric representation*. The needs for either an implicit or parametric representation of an algebraic variety depends on the operations to be perform with the variety. The parametric representation is appropriate for generating points of the variety and for plotting it with the computer. The implicit representation is convenient for checking whether a given point lies on the variety. That's why the transition from one representation to the other is important.

Typical objects of geometric modeling are *rational Bézier curves and surfaces* and *NURBS (Non-Uniform Rational B-Spline) curves and surfaces* which are represented by parametric equations. That's why we will concentrate on *implicitization problem*, that is on finding an implicit representation of a rational algebraic variety given by parametric equations.

In recent years, methods of implicitization of algebraic varieties have been intensively studied. Basic approach lies in using variable elimination methods, such as resultants (see [7, 9, 16]) or Gröbner bases (see [3, 4]). By clearing denominators, the parametrization is converted into a system of nonlinear algebraic equations from which parameters are eliminated. But the implicit representation obtained by this method needs not be the smallest variety (in the sense of inclusion) containing the given rational parametrization. When Gröbner bases are

ISSN 1433-8157/$ 2.50

used, the problem can be solved by including an additional equation and variable which ensure that the denominators do not vanish. Thus the smallest variety containing the parametric representation is obtained using the Gröbner bases method ([4]). Another approach to solve this problem without including any additional variable is presented in [1].

On the other hand, even if we add the additional equation and variable which ensure that the denominators do not vanish, the determinant of the resultant matrix needs not provide exactly the implicit representation; it can contain extraneous factors. We only obtain the so-called *projection operator* which always contains the resultant (and therefore the implicit representation of the variety) but also extraneous factors which have to be eliminated.

Implicitization algorithms based on Gröbner bases and resultants suffer from problems with parameterizations which involve base points (for details see Section 2.1). In such a case the Dixon resultant (and any other resultant as the determinant of the resultant matrix) vanishes identically and hence fails to produce the implicit equation. Then the projection operator can be extracted from the resultant matrix by the RSC (Rank Submatrix Construction) method (see [7, 8, 9, 16]). But this projection operator often contains extraneous factors.

Another method for the implicitization of algebraic varieties is the method of moving curves and surfaces (see [17, 18]). This method allows to write the implicit equation in a more compact form than the implicitization using resultants. For surfaces without base points, this method expresses the implicit equation again as the determinant of a special matrix which can be smaller than the resultant matrix. If base points exist, then the method actually simplifies: the degree of some matrix elements decreases, or even the matrix reduces its size.

Based on the method of moving lines (planes), [5] and [6] present a method for the implicitization of rational curves (surfaces). In this method the implicit equation is given by the determinant of a special implicitization matrix with the structure of a Sylvester matrix (sparse matrix, simple entries) and the order of the Bézout (Dixon) matrix.

In recent years an interesting approach to the implicitization of curves and surfaces based on numerical methods appeared. [12] and [13] present a method for the implicitization of curves and surfaces based on the classical polynomial interpolation. The determinant of a resultant matrix is interpolated by the classical Lagrange interpolation method (for curves) or by extension of this method to the multivariate case (for surfaces).

A new implicitization method has appeared recently in [19]. This method is based on a degree estimation of the implicit polynomial F with indeterminate coefficients u_i which are computed from the corresponding system of linear equations.

The paper under study is organized as follows: Section 2 overviews the implicitization methods for curves and surfaces with rational parametrizations. Sections 3 and 4 are devoted to the main results of the paper, the application of implicitization methods to NURBS curves and surfaces and mainly to the comparison of implementations of these methods in MATHEMATICA and MATLAB.

2. Methods of implicitization-theory

2.1. Implicitization using Gröbner bases and resultants

The basic approach to the implicitization of algebraic varieties lies in using variable elimination methods, such as Gröbner bases or resultants, because we want to eliminate parameters from the parametrization of an algebraic variety in order to obtain an implicit equation.

Let

$$C(t) = \left(\frac{X(t)}{W(t)}, \frac{Y(t)}{W(t)} \right) \tag{1}$$

be a rational parametrization of a plane curve. Then, by clearing denominators, the system of equations

$$\begin{aligned} x \cdot W(t) - X(t) &= 0, \\ y \cdot W(t) - Y(t) &= 0 \end{aligned} \tag{2}$$

is obtained. If the parameter t is eliminated from the system (2) using the Gröbner basis or resultants, a polynomial $F(x, y)$ is obtained. This polynomial contains the implicit representation of the plane curve C but it can also contain some extraneous factors[1]. Hence, in this case it is not guaranteed that $F(x, y)$ represents the smallest variety containing the curve C given by the parametrization (1). This can be assured by adding one variable s and one equation

$$1 - s \cdot W(t) = 0 \tag{3}$$

which ensures that the denominator of the parametrization (1) does not vanish. The resultant $R(x, y)$ obtained by the elimination of the variables s and t from both (2) and (3) represents the smallest variety containing the curve C (see [4, Theorem 2 in Chapter 3, § 3]). Thus, $R(x, y)$ is the implicit representation of the curve C given by the parametrization (1).

A similar process can be used to implicitize rational surfaces. Let

$$S(u, v) = \left(\frac{X(u, v)}{W(u, v)}, \frac{Y(u, v)}{W(u, v)}, \frac{Z(u, v)}{W(u, v)} \right) \tag{4}$$

be a rational parametrization of a surface. By clearing the denominators the system of equations

$$\begin{aligned} x \cdot W(u, v) - X(u, v) &= 0, \\ y \cdot W(u, v) - Y(u, v) &= 0, \\ z \cdot W(u, v) - Z(u, v) &= 0 \end{aligned} \tag{5}$$

is obtained. After eliminating the parameters[2] u and v a polynomial $F(x, y, z)$ is obtained which contains an implicit representation of the surface S; but it can contain extraneous factors, too. Hence, the system of equations (5) can be modified by adding a variable s and the equation

$$1 - s \cdot W(u, v) = 0 \tag{6}$$

to ensure that the denominator of the parametrization (4) does not vanish. The resultant $R(x, y, z)$ obtained by the elimination of variables s, u, v from both (5) and (6) represents the smallest variety containing the surface S (see [4, Theorem 2 in Chapter 3, §3]). Thus, $R(x, y, z)$ is the implicit representation of the surface S.

There is also another problem concerning the implicitization of surfaces, namely base points. A *base point* is a common solution $(u_0, v_0) \in \mathbb{C}^2$ of a system of equations

$$X(u, v) = 0, \quad Y(u, v) = 0, \quad Z(u, v) = 0, \quad W(u, v) = 0.$$

[1] *Extraneous factors* are factors of $F(x, y)$ which do not belong to the implicit representation. More precisely, only factors of $F(x, y)$ for which $F\left(\frac{X(t)}{W(t)}, \frac{Y(t)}{W(t)}\right) = 0$ form the implicit representation of the given algebraic variety. Factors which do not fulfil this condition are called extraneous.

[2] If a surface parametrization contains base points, then the elimination of parameters is not possible by this method. A detailed explanation will be given later.

If the surface parametrization contains any base point, then both Gröbner bases and resultants fail to provide an implicit representation of the surface S. In such a case the resultant equals zero identically thus providing no condition on the initial system of equations (5) to have a common solution. The reason is that base points represent non-trivial solutions of the system (5) independent of x, y, z. Gröbner bases for the implicitization of surfaces are based on the fact that the implicit representation is contained in the ideal I generated by the equations (5). However, if the surface parametrization contains base points, the ideal I does not contain any polynomial independent of u, v (besides 0); so the ideal I does not contain any implicit representation of S.

There are several possibilities how to solve the problem with base points:

- We can add the equation (6) to ensure that the denominator $W(u, v)$ of the parametrization does not vanish and hence base points are excluded;
- we use a perturbation method to modify the initial system (5) (for more details see [10, 11]);
- when we use resultants for the elimination of parameters, we can use the *RSC (Rank Submatrix Construction) method* to extract the projection operator from the resultant matrix even though the determinant of the resultant matrix is identically zero (for more details see [7, 16]).

2.2. Implicitization using moving curves and surfaces

The method of moving curves and surfaces expresses an implicit equation of a curve or surface given by parametric equations as the determinant of a matrix. Only a brief introduction will be presented here; details can be found in [17, 18].

The most commonly used moving curves are moving lines and moving conics. A *moving line of degree* $m-1$ can be defined equivalently in two forms:

$$L_{m-1}(x,y)t^{m-1} + \ldots + L_1(x,y)t + L_0(x,y) = 0 \quad \text{or} \quad A(t)x + B(t)y + C(t) = 0$$

where the $L_i(x, y)$ are linear polynomials in variables x, y and $A(t)$, $B(t)$, $C(t)$ are polynomials of degree $m-1$ in the variable t (at least one of them). For each t_0 the moving line represents an implicit equation of a line in the xy-plane.

A moving line is said to *follow a rational curve* (1) if

$$A(t)\frac{X(t)}{W(t)} + B(t)\frac{Y(t)}{W(t)} + C(t) \equiv 0 \quad \text{or equivalently} \quad A(t)X(t) + B(t)Y(t) + C(t)W(t) \equiv 0$$

holds true. Geometrically, a moving line follows a rational curve if the implicit line corresponding to the parameter t passes through the point on the rational curve corresponding to the parameter t.

Moving lines have an interesting connection to the Bézout resultant. Each row in the Bézout matrix corresponds to a moving line following the given curve.

For a rational curve $C(t)$ of given degree m, we begin by seeking moving lines of degree $m-1$

$$L_{m-1}(x,y)t^{m-1} + \ldots + L_1(x,y)t + L_0(x,y) = 0 \tag{7}$$

that follow the curve. Since each $L_i(x, y)$ is linear in x and y, eq. (7) can be rewritten as

$$(A_{m-1}x + B_{m-1}y + C_{m-1})t^{m-1} + \ldots + (A_1x + B_1y + C_1)t + (A_0x + B_0y + C_0) = 0. \tag{8}$$

If we substitute x and y by rational functions $X(t)/W(t)$ and $Y(t)/W(t)$, resp., and multiply it with $W(t)$, we obtain a polynomial of degree $2m-1$ in the variable t

$$(A_{m-1}X(t) + B_{m-1}Y(t) + C_{m-1}W(t))t^{m-1} + \ldots + (A_0X(t) + B_0Y(t) + C_0W(t)) = 0. \quad (9)$$

In order to guarantee that eq. (8) represents a moving line which follows the rational curve, the polynomial (9) has to vanish identically. This leads to a system of $2m$ homogeneous linear equations in $3m$ unknowns A_k, B_k, C_k, $k = 0, \ldots, m-1$, which has at least m linearly independent solutions

$$\begin{aligned} p_1(t) &= L_{1,m-1}(x,y)t^{m-1} + \ldots + L_{1,1}(x,y)t + L_{1,0}(x,y) = 0, \\ &\vdots \\ p_m(t) &= L_{m,m-1}(x,y)t^{m-1} + \ldots + L_{m,1}(x,y)t + L_{m,0}(x,y) = 0. \end{aligned} \quad (10)$$

Then

$$\det R(x,y) = \begin{vmatrix} L_{1,0} & \ldots & L_{1,m-1} \\ \vdots & & \vdots \\ L_{m,0} & \ldots & L_{m,m-1} \end{vmatrix} = 0$$

is the implicit equation of the rational curve, provided that there are no base points.

Theorem 1 *The method of moving lines always generates the correct implicit equation of a rational curve, provided the rational curve has no base points.*

Proof: See [18]. □

In the presence of base points the method has to be modified because a rational curve of degree m with r base points is represented by an algebraic curve of degree $m-r$. Details of this modification of the moving lines method can be found in [18].

An interesting way of representing the implicit equation of a rational curve lies in using moving conics. A *moving conic of degree* $m-1$ can be written in two ways:

$$C_{m-1}(x,y)t^{m-1} + \ldots + C_1(x,y)t + C_0(x,y) = 0 \quad (11)$$

or

$$A(t)x^2 + B(t)xy + C(t)y^2 + D(t)x + E(t)y + F(t) = 0 \quad (12)$$

where $C_j(x,y)$ are polynomials of degree two in x, y and $A(t), \ldots, F(t)$ are polynomials of degree $m-1$ in t.

Similarly to the case of moving lines, the moving conic (12) follows any rational curve $C(t)$ if it vanishes on the curve, i.e., if

$$A(t)X^2(t) + B(t)X(t)Y(t) + C(t)Y^2(t) + D(t)X(t)W(t) + E(t)Y(t)W(t) + F(t)W^2(t) \equiv 0.$$

Geometrically, a moving conic follows a rational curve if the implicit conic corresponding to the parameter t passes through the point on the rational curve corresponding to the parameter t.

In (11), each coefficient $C_j(x,y)$ is quadratic in x, y. Hence, we can rewrite (11) as

$$\begin{gathered} (A_{m-1}x^2 + B_{m-1}xy + C_{m-1}y^2 + D_{m-1}x + E_{m-1}y + F_{m-1})t^{m-1} + \\ \vdots \\ + (A_0x^2 + B_0xy + C_0y^2 + D_0x + E_0y + F_0) = 0. \end{gathered} \quad (13)$$

To implicitize the rational curve $C(t)$ of degree $2m$ we seek for moving conics of degree $m-1$ which follow $C(t)$. Hence, we substitute for x and y rational functions $X(t)/W(t)$ and $Y(t)/W(t)$, resp., and we multiply the obtained equation by $W^2(t)$. This yields

$$\begin{gathered}(A_{m-1}X(t)^2 + B_{m-1}X(t)Y(t) + \ \dots\ + E_{m-1}Y(t)W(t) + F_{m-1}W^2(t))t^{m-1} + \\ \vdots \\ +(A_0X(t)^2 + B_0X(t)Y(t) + C_0Y(t)^2 + D_0X(t)W(t) + E_0Y(t)W(t) + F_0W^2(t)) = 0.\end{gathered} \tag{14}$$

Since $X(t)$, $Y(t)$ and $W(t)$ are polynomials of degree $2m$ in t, (14) represents a polynomial of degree $5m-1$ in t. (14) represents a moving conic following the rational curve $C(t)$ if and only if the polynomial (14) equals zero identically. This condition generates a system of $5m$ homogeneous linear equations in $6m$ unknowns A_k, $\dots$, F_k, $k = 0, \dots, m-1$ which has at least m linearly independent solutions

$$\begin{aligned} q_1(t) &= C_{1,m-1}(x,y)t^{m-1} + \ \dots\ + C_{1,1}(x,y)t + C_{1,0}(x,y) = 0, \\ &\vdots \\ q_m(t) &= C_{m,m-1}(x,y)t^{m-1} + \ \dots\ + C_{m,1}(x,y)t + C_{m,0}(x,y) = 0. \end{aligned} \tag{15}$$

The coefficients of the moving conics (15) form a $m \times m$ matrix $C(x,y) = (C_{ij}(x,y))$. Then $\det(C(x,y)) = 0$ is a good candidate to be an implicit equation of the given rational curve. However, in some cases $\det(C(x,y))$ can be identically zero, even if there are no base points; and then the method of moving conics does not yield an implicit equation of the rational curve. A necessary and sufficient condition for the method of moving conics to generate the implicit equation of an even degree rational curve is given by following theorem:

Theorem 2 *The method of moving conics generates the implicit equation for a rational curve of degree $2m$ without base points if and only if there is no moving line of degree $m-1$ that follows the curve. Moreover, when there is a moving line of degree $m-1$ that follows the curve, any determinant generated by the method of moving conics is identically zero.*

Proof: See [18]. □

Let $S(u,v) = (X(u,v), Y(u,v), Z(u,v), W(u,v))$ be a rational parametric surface in the projective extension of $\mathbb{E}_3$ and

$$X(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} a_{ij}u^iv^j, \qquad Y(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} b_{ij}u^iv^j,$$

$$Z(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} c_{ij}u^iv^j, \qquad W(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} d_{ij}u^iv^j.$$

Among moving surfaces only moving planes and moving quadrics are usually used. A *moving plane of bi-degree* (σ_1, σ_2) is an implicit equation of the form

$$\sum_{i=0}^{\sigma_1}\sum_{j=0}^{\sigma_2}(A_{i,j}x + B_{i,j}y + C_{i,j}z + D_{i,j}w)\cdot u^iv^j = 0. \tag{16}$$

For fixed values of u and v, the expression (16) represents the implicit equation of a plane. The moving plane is said to *follow the rational surface* $S(u,v)$ if

$$\sum_{i=0}^{\sigma_1}\sum_{j=0}^{\sigma_2}(A_{i,j}X(u,v)+B_{i,j}Y(u,v)+C_{i,j}Z(u,v)+D_{i,j}W(u,v))\cdot u^i v^j \equiv 0. \tag{17}$$

Eq. (17) represents a polynomial of degree $m+\sigma_1$ in u and degree $n+\sigma_2$ in v. By setting the coefficients of the monomials $u^i v^j$ for $i=0,\ldots,m+\sigma_1$ and $j=0,\ldots,n+\sigma_2$ to zero, we obtain a system of $(m+\sigma_1+1)(n+\sigma_2+1)$ homogeneous linear equations in $4(\sigma_1+1)(\sigma_2+1)$ unknowns $\{A_{i,j},B_{i,j},C_{i,j},D_{i,j}\}$, $i=0,\ldots,\sigma_1$, $j=0,\ldots,\sigma_2$. The solution of this system provides a collection of moving planes which follow the surface $S(u,v)$.

For moving planes we usually choose $\sigma_1=2m-1$ and $\sigma_2=n-1$. Then from (17) we obtain a system of $6mn$ homogeneous linear equations in $8mn$ unknowns. Such a system has at least $2mn$ linearly independent solutions which represent moving planes following the surface $S(u,v)$:

$$\begin{aligned} L_1 &\equiv \sum_{i=0}^{2m-1}\sum_{j=0}^{n-1}(A^1_{i,j}x+B^1_{i,j}y+C^1_{i,j}z+D^1_{i,j}w)\cdot u^i v^j = 0,\\ &\vdots \\ L_{2mn} &\equiv \sum_{i=0}^{2m-1}\sum_{j=0}^{n-1}(A^{2mn}_{i,j}x+B^{2mn}_{i,j}y+C^{2mn}_{i,j}z+D^{2mn}_{i,j}w)\cdot u^i v^j = 0. \end{aligned} \tag{18}$$

The determinant of the coefficients of $u^i v^j$ of moving planes (18), i.e.,

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

vanishes whenever (x,y,z,w) lies on the surface $S(u,v)$. Hence if this determinant does not vanish identically, then it is a multiple of the implicit equation of surface $S(u,v)$. This formulation generates a matrix of the same size as the implicitization using the Dixon resultant, and therefore both formulations are equivalent; each row of the Dixon matrix represents one moving plane following the rational surface $S(u,v)$.

An interesting way how the matrix size can be reduced is the usage of moving quadrics. A *moving quadric of bi-degree* (σ_1,σ_2) is an implicit equation of the form

$$\begin{aligned} \sum_{i=0}^{\sigma_1}\sum_{j=0}^{\sigma_2}(&A_{i,j}x^2+B_{i,j}y^2+C_{i,j}z^2+D_{i,j}xy+E_{i,j}xz+F_{i,j}yz+\\ &G_{i,j}xw+H_{i,j}yw+I_{i,j}zw+J_{i,j}w^2)\cdot u^i v^j = 0. \end{aligned} \tag{19}$$

For fixed values u and v the expression (19) represents the implicit equation of a quadric. The moving quadric is said to follow rational surface $S(u,v)$ if

$$\sum_{i=0}^{\sigma_1}\sum_{j=0}^{\sigma_2}(A_{i,j}X(u,v)^2+B_{i,j}Y(u,v)^2+\ \ldots\ +I_{i,j}Z(u,v)W(u,v)+J_{i,j}W(u,v)^2)\cdot u^i v^j = 0. \tag{20}$$

Eq. (20) represents a polynomial of degree $2m+\sigma_1$ in u and degree $2n+\sigma_2$ in v. By setting the coefficients of the monomials $u^i v^j$ for $i = 0, \ldots, 2m+\sigma_1$ and $j = 0, \ldots, 2n+\sigma_2$ to zero, we obtain a system of $(2m+\sigma_1+1)(2n+\sigma_2+1)$ homogeneous linear equations in $10(\sigma_1+1)(\sigma_2+1)$ unknowns $\{A_{i,j}, B_{i,j}, \ldots, I_{i,j}, J_{i,j}\}$, $i = 0, \ldots, \sigma_1$, $j = 0, \ldots, \sigma_2$. Solving this system gives a collection of moving quadrics which follow the surface $S(u, v)$.

For a moving quadric we usually choose $\sigma_1 = m-1$ and $\sigma_2 = n-1$. Then we obtain from (20) a system of $9mn$ homogeneous linear equations in $10mn$ unknowns. Such a system has at least mn linearly independent solutions which represent moving quadrics following the surface $S(u, v)$:

$$\begin{aligned} Q_1 &\equiv \sum_{i=0}^{\sigma_1}\sum_{j=0}^{\sigma_2}(A^1_{i,j}x^2 + B^1_{i,j}y^2 + \ldots + I^1_{i,j}zw + J^1_{i,j}w^2)\cdot u^i v^j = 0, \\ &\vdots \\ Q_{mn} &\equiv \sum_{i=0}^{\sigma_1}\sum_{j=0}^{\sigma_2}(A^{mn}_{i,j}x^2 + B^{mn}_{i,j}y^2 + \ldots + I^{mn}_{i,j}zw + J^{mn}_{i,j}w^2)\cdot u^i v^j = 0. \end{aligned} \tag{21}$$

The determinant of the coefficients of $u^i v^j$ of moving quadrics (21), i.e.,

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

vanishes whenever (x, y, z, w) lies on the surface $S(u, v)$. Hence, if this determinant does not vanish identically, then it is a multiple of the implicit equation of surface $S(u, v)$.

2.3. Implicitization using polynomial interpolation

This method uses an efficient computation of the resultant by means of classical bivariate or multivariate Lagrange interpolation to find the implicit equation of a curve or surface given by any rational parametrization.

Let (1) be a proper parametrization of a rational plane algebraic curve, i.e., $\mathrm{GCD}(X, W) = \mathrm{GCD}(Y, W) = 1$. Since GCD is the greatest common divisor of the given polynomials, the condition $\mathrm{GCD}(X, W) = \mathrm{GCD}(Y, W) = 1$ means that there is no common factor of X and W (and similarly for Y and W). For such a parametrization, an implicit equation $F(x, y)$ of curve C is given by

$$\mathrm{Res}(xW(t) - X(t),\ yW(t) - Y(t)) = 0.$$

The condition $\mathrm{GCD}(X, W) = \mathrm{GCD}(Y, W) = 1$ can be removed, but in this case also parametrizations with base points are included for which the resultant matrix is singular (so, the zero polynomial will be interpolated). Similarly to the implicitization using resultants, any submatrix of the resultant matrix with full rank can be taken to interpolate the projection operator (which contains the resultant, i.e., the implicit equation).

In this method the concept of an *interpolation space* is essential. The following theorem says which is the most suitable interpolation space.

Theorem 3 *Let $C(t) = (X(t)/W(t),\ Y(t)/W(t))$ be a proper rational parametrization of the irreducible curve C defined by $F(x, y)$ and let $\mathrm{GCD}(X, W) = \mathrm{GCD}(Y, W) = 1$. Then*

$$\begin{aligned} m = \max\{\deg_t(X), \deg_t(W)\} &= \deg_y(F) \\ n = \max\{\deg_t(Y), \deg_t(W)\} &= \deg_x(F). \end{aligned}$$

Proof: See [12]. □

This theorem states that each polynomial $F(x,y)$ defining an implicit equation of the curve C belongs to the polynomial space $\Pi_{n,m}(x,y)$.

The polynomial $F(x,y)$ can be computed by classical Lagrange interpolation. If the interpolation nodes (x_i, y_j), $i = 0,\dots,n$, $j = 0,\dots,m$, and interpolation data $f_{ij} \in K$, $i = 0,\dots,n$, $j = 0,\dots,m$ are given, we want to find a polynomial

$$F(x,y) = \sum_{(i,j)\in I} c_{ij} x^i y^j \in \Pi_{n,m}(x,y), \quad I = \{(i,j) \mid i = 0,\dots,n,\ j = 0,\dots,m\}$$

such that

$$F(x_i, y_j) = f_{ij} \quad \forall (i,j) \in I. \tag{22}$$

The interpolation conditions (22) can be written as a system of linear equations

$$Ac = f \tag{23}$$

where the coefficient matrix A is given by the Kronecker product[3] $A = V_x \otimes V_y$, and where V_x, V_y are Vandermonde matrices, c is the column vector of unknown coefficients of the implicit equation $F(x,y)$ and f is the column vector containing the interpolation data.

The interpolation nodes (x_i, y_j) can be chosen such that $x_i = i$ for $i = 0,\dots,n$ and $y_j = j$ for $j = 0,\dots,m$ which ensures non-singularity of matrix A. Then the interpolation data f_{ij} correspond to values $F(i,j)$ of the resultant at points (i,j). That is, if M is the symbolic Bézout matrix for the equations $xW(t) - X(t)$ and $yW(t) - Y(t)$ with respect to t and if M_{ij} is the matrix M where i, j are substituted for x, y, resp., then $f_{ij} = \det M_{ij}$.

At the end, the special structure of matrix A is used for a fast solving of the system (23). The problem of solving the linear system (23) with $A = V_x \otimes V_y$ can be reduced to the solution of $n+1$ systems with the same matrix V_y followed by solving $m+1$ systems with the same matrix V_x (for details see, e.g., [2]). Since V_x and V_y are Vandermonde matrices, a special method for solving linear systems with such a coefficient matrix can be used (for details see [12]).

The generalization for rational surfaces is very straightforward. Let S be the surface with the parametrization (4). Clearing denominators in the parametrization yields three polynomials

$$X(u,v) - xW(u,v), \quad Y(u,v) - yW(u,v), \quad Z(u,v) - zW(u,v) \tag{24}$$

with total degrees d_0, d_1, d_2.

Again, the first step is to determine an appropriate interpolation space $\Pi_{l,m,n}(x,y,z)$ such that the Macaulay resultant $R(x,y,z)$ for the polynomials (24) with respect to u and v is included in $\Pi_{l,m,n}(x,y,z)$. By [13, Proposition 3] holds

$$\begin{aligned} l = \deg_x(R(x,y,z)) &\le d_1 d_2, \\ m = \deg_y(R(x,y,z)) &\le d_0 d_2, \\ n = \deg_z(R(x,y,z)) &\le d_0 d_1. \end{aligned}$$

Then the interpolation problem is solved for the interpolation nodes (x_i, y_j, z_k) and the interpolation data r_{ijk}, $i = 0,\dots,l$, $j = 0,\dots,m$, $k = 0,\dots,n$ in the interpolation space

[3]The *Kronecker product* $B \otimes D$ is defined by blocks as $(b_{kl}D)$, with $B = (b_{kl})$.

$\Pi_{l,m,n}(x,y,z)$ to interpolate the resultant $R(x,y,z)$. This leads to a system of linear equations

$$Ac = r \tag{25}$$

where the coefficient matrix is given by the Kronecker product $A = (V_x \otimes V_y) \otimes V_z$, where V_x, V_y and V_z are Vandermonde matrices, c is the column vector of unknown coefficients of the implicit equation $R(x,y,z)$, and r is the column vector containing the interpolation data.

Usually the interpolation nodes (x_i, y_j, z_k) are chosen as grid points, that is, $(x_i,\ y_j,\ z_k) = (i,j,k)$ for $i = 0,\dots,l$, $j = 0,\dots,m$, $k = 0,\dots,n$. The interpolation data r_{ijk} correspond to the values $R(i,j,k)$ of the resultant at the grid points (i,j,k). That is, if M is the symbolic Macaulay matrix (or any other resultant matrix) for the equations (24) with respect to u and v and if M_{ijk} is the matrix M where i,j,k are substituted for x,y,z, resp., then $r_{ijk} = \det M_{ijk}$.

Again, the special structure of the coefficient matrix A is used for a fast solving of the system (25). Since matrix A is a Kronecker product of matrices V_x, V_y and V_z, the procedure of solving the system of linear equations with coefficient matrix A of order $(l+1)(m+1)(n+1)$ can be replaced by solving $(n+1)$ system with the coefficient matrix V_z followed by solving $(m+1)$ systems with the coefficient matrix V_y and then $(l+1)$ systems with the coefficient matrix V_x. Reader can find more details in [13].

2.4. Implicitization using direct method

This simple method for the conversion of rational parametric equations of curves and surfaces into implicit equations has been presented recently in [19]. The method is based on solving a system of linear algebraic equations for the coefficients of the implicit equation F of estimated degree.

The method is formulated for rational surfaces but it works in the similar way for rational curves. Let a rational surface be defined by (4). The main task is to determine a polynomial $F(x,y,z)$ such that

$$F(x,y,z) = F\left(\frac{X(u,v)}{W(u,v)}, \frac{Y(u,v)}{W(u,v)}, \frac{Z(u,v)}{W(u,v)}\right) \equiv 0.$$

There are two possibilities how the degree of polynomial F can be estimated:

1. the total degree of the polynomial F,
2. individual degrees of the polynomial F in x, y and z.

If the total degree n of F is given, then F can be set up in the form

$$F = \sum_{\substack{i\ge 0, j\ge 0, k\ge 0\\ i+j+k\le n}} a_{ijk} x^i y^j z^k \quad \text{with} \quad \sum_{\substack{i\ge 0, j\ge 0, k\ge 0\\ i+j+k= n}} a_{ijk}^2 > 0 \tag{26}$$

where the coefficients a_{ijk} are unknown and must be found. After substituting (4) into (26) a rational expression

$$\sum_{\substack{i\ge 0, j\ge 0, k\ge 0\\ i+j+k\le n}} a_{ijk} \frac{X^i Y^j Z^k}{W^i W^j W^k} = \frac{g}{h} = 0 \tag{27}$$

is obtained. Since g has to vanish identically, the coefficients of all monomials $u^\alpha v^\beta$ in g must equal zero. Each such nonzero coefficient is linear in the variables a_{ijk} and represents an equation of a system of linear algebraic equations whose nontrivial solution represents the coefficients of an implicit equation F. The surface (4) has an implicit equation of degree

$\leq n$ if and only if this linear system has a nontrivial solution for a_{ijk}, say $\bar{a}_{ijk}$. The implicit polynomial of the rational surface (4) is obtained as a nonconstant factor of $F|_{a_{ijk}=\bar{a}_{ijk}}$ only in the variables x, y, z.

In case that the individual degrees of F in the variables x, y, z are used, F can be written in the form

$$F = \sum_{i=0}^{n_x} \sum_{j=0}^{n_y} \sum_{k=0}^{n_z} a_{ijk} x^i y^j z^k. \tag{28}$$

For degree estimation the method described in Section 2.3 can be used. Then, the algorithm of the direct method continues in the same way as for the polynomial F given by its total degree. Reader can find more details in [19].

3. Application to NURBS curves

A *NURBS curve of degree p* is defined by

$$C(t) = \frac{\sum_{i=0}^{n} N_{i,p}(t) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(t) w_i}, \qquad a \leq t \leq b, \tag{29}$$

where P_i are control points (forming a control polygon), w_i are weights of control points and $N_{i,p}(t)$, $i = 0, \ldots, n$, are the B-spline basis functions of degree p defined on a non-periodic (and non-uniform) knot vector

$$T = \{\underbrace{a, \ldots, a}_{p+1}, t_{p+1}, \ldots, t_{m-p-1}, \underbrace{b, \ldots, b}_{p+1}\}$$

by the recurrent formula

$$\begin{aligned} N_{i,0}(t) &= \begin{cases} 1 & \text{for } t_i \leq t \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ N_{i,p}(t) &= \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t). \end{aligned}$$

Reader can find more details on NURBS curves in [14].

3.1. Examples and comparison of methods for curves

Now several testing examples will be presented: The implicitization methods described in Section 2 will be used to solve the implicitization problem for several NURBS curves. All implicitization methods were implemented by the authors in MATLAB 6.5, including the computation of the Dixon resultant, the Dixon dialytic resultant and the RSC method for the extraction of the projection operator from the resultant matrix. Only Gröbner basis computations are performed using external calls of the Computer Algebra System COCOA 4.2[4] (see [15]). Almost all methods were also implemented by the authors in MATHEMATICA 5.0, except the computation of the Dixon dialytic resultant (this implementation will be done soon) and the implicitization where the Dixon resultant for the system of equations (2) and

[4]External calls of COCOA are done by the MATLAB command "!"

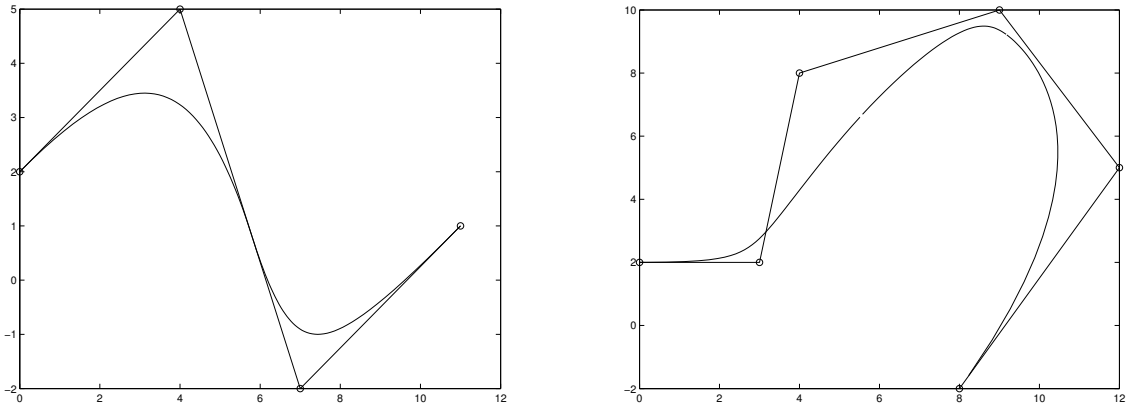

Figure 1: NURBS curves for the Examples 1 (left) and 2 (right)

	Matlab 6.5		Mathematica 5.0	
	Part 1	Part 2	Part 1	Part 2
Gröbner Basis	0.0788	0.0850	0.002	0.002
Dixon Resultant[1]	0.5388	0.5450	0.001	0.002
Dixon Resultant[2]	1.0225	1.0413	—	—
Dixon Dialytic	0.8787	0.9038	—	—
Interpolation	0.8063	0.8188	0.004	0.004
Moving Lines	0.2050	0.2050	0.001	0.001
Moving Conics	0.2737	0.2725	0.001	0.
Direct	0.1438	0.1437	0.	0.001

Table 1: Computational costs (in seconds) for Example 1

(3) is used. In Mathematica 5.0 the Gröbner basis is computed using the built-in function GroebnerBasis. For both these mathematical software, the same algorithms were used for each developed function.

The computational costs (in seconds, obtained by the function Cputime in Matlab 6.5[5] and by the function Timing in Mathematica 5.0[6]) needed by each method for solving the implicitization problem for each segment of NURBS curves in the Examples 1–4 are summarized in the Tables 1–4. There are two rows denoted by "Dixon resultant" differing only in superscript. "Dixon Resultant[1]" means that the Dixon resultant for the system (2) with respect to t was computed to find the implicit equation of a given curve. "Dixon Resultant[2]" means that the Dixon resultant for the system (2) together with the additional equation (3) with respect to s and t was computed to find the implicit equation. The row denoted by "Dixon Dialytic" means that the Dixon dialytic resultant for the system (2) with respect to t was computed to find the implicit equation. Each time value in the tables was acquired from 10 initiations of a corresponding program for implicitization. The smallest and the biggest value were dropped and the average of other values is included in the table.

[5]Matlab 6.5 was running on AMD Athlon XP 2500+, 512MB RAM

[6]Mathematica 5.0 was running on Intel Pentium M 1.6GHz, 512 MB RAM (CPU speed similar to AMD Athlon XP 2500+)

	MATLAB 6.5			MATHEMATICA 5.0		
	Part 1	Part 2	Part 3	Part 1	Part 2	Part 3
Gröbner Basis	0.0950	0.1050	0.0975	0.004	0.004	0.004
Dixon Resultant[1]	1.1000	1.1525	1.1712	0.016	0.0151	0.014
Dixon Resultant[2]	2.3312	2.4250	2.4263	—	—	—
Dixon Dialytic	1.6688	1.7150	1.7213	—	—	—
Interpolation	1.7912 Wrong result	1.8725 Wrong result	1.8800 Wrong result	0.01	0.01	0.09
Moving Lines	0.3088	0.3112	0.3163	0.002	0.002	0.002
Moving Conics	0.3700	0.3813	0.3812	0.003	0.004	0.003
Direct	0.2275	0.2350	0.2400	0.005	0.007	0.0051

Table 2: Computational costs (in seconds) for Example 2

	MATLAB 6.5		MATHEMATICA 5.0	
	Part 1	Part 2	Part 1	Part 2
Gröbner Basis	0.3850	0.4150	0.007	0.007
Dixon Resultant[1]	2.0688	2.0987	0.0191	0.02
Dixon Resultant[2]	4.5425	4.6387	—	—
Dixon Dialytic	2.0212	2.6913	—	—
Interpolation	3.4650 Wrong result	3.5200 Wrong result	0.022	0.024
Moving Lines	0.4413	0.4450	0.002	0.003
Moving Conics	0.3825	0.3950	0.003	0.004
Direct	0.3675	0.3688	0.008	0.01

Table 3: Computational costs (in seconds) for Example 3

Example 1 A NURBS curve is given by 4 control points with corresponding weights $[1, 2, 3, 1]$ and by the knot vector $T = \{0, 0, 0, 1/2, 1, 1, 1\}$ (see Fig. 1 (left)). The computational costs needed for finding implicit equations of both parts of the NURBS curve are summarized in Table 1. ■

Example 2 A NURBS curve is given by 6 control points with the corresponding weights $[1, 3, 1, 5, 2, 1]$ and by the knot vector $T = \{0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1\}$ (see Fig. 1 (right)). The different computational costs are summarized in Table 2. ■

Example 3 A NURBS curve is given by 6 control points with corresponding weights $[1, 2, 4, 6, 2, 1]$ and by the knot vector $T = \{0, 0, 0, 0, 0, 1/2, 1, 1, 1, 1, 1\}$ (see Fig. 2 (left)). The computational costs are summarized in Table 3. ■

Example 4 A NURBS curve is given by 8 control points with corresponding weights $[1, 2, 6, 5, 8, 2, 4, 1]$ and by the knot vector $T = \{0, 0, 0, 0, 0, 0, 0, 1/2, 1, 1, 1, 1, 1, 1, 1\}$ (see Fig. 2 (right)). The computational costs are summarized in Table 4. ■

	MATLAB 6.5		MATHEMATICA 5.0	
	Part 1	Part 2	Part 1	Part 2
Gröbner Basis	83 326	106 836	2.3394	2.8861
Dixon Resultant[1]	5.7363	5.7862	0.0771	0.078
Dixon Resultant[2]	13.7725	13.9913	—	—
Dixon Dialytic	6.0575	6.2550	—	—
Interpolation	9.7412 Wrong result	9.9088 Wrong result	0.0781	0.0791
Moving Lines	0.9200	1.0325	0.015	0.016
Moving Conics	0.6913	0.7675	0.013	0.015
Direct	Wrong result	Wrong result	0.1002	0.1352

Table 4: Computational costs (in seconds) for Example 4

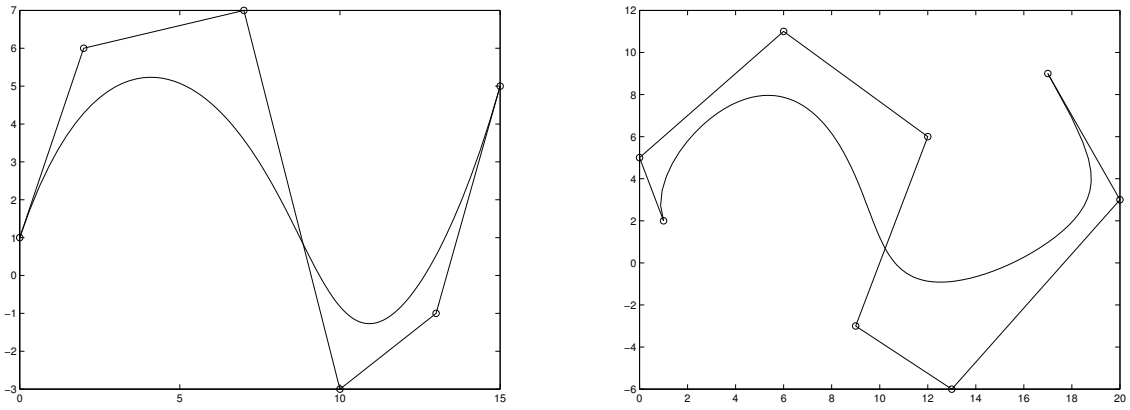

Figure 2: NURBS curves for the Examples 3 (left) and 4 (right)

Comparing the times used for implicitization methods in MATLAB, we can see that the method based on the usage of Gröbner bases is very fast but only for low degree parametrizations. It is very hard to estimate in advance from a given parametrization whether the computation of Gröbner basis will be very fast or very slow. For very similar parametrizations the computational costs can be very different. That's why Gröbner bases are not the best choice to implicitize curves given by a rational parametrization. One of the reasons why Gröbner bases are fast in some testing examples is that although all other methods were implemented by authors in MATLAB without using any special algorithms and programming techniques. Gröbner bases are computed using the specialized computer algebra system CoCoA.

For low degree parametrizations, the direct method seems to be suitable. This simple method produces the implicit equation for simple parametrizations very fast. But for parametrizations of higher degrees the computation cost increases faster than for other methods. An interesting problem concerning this method was also discovered: Although our implementation produces a correct implicit equation for parametrizations of all tested degrees (up to 10) which was generated randomly (random coefficients of the polynomials of given degree for rational parametrization), the method gives a wrong answer for rational parametrizations of NURBS curves of degree 5 and higher, probably due to rounding errors.

As for the methods based on resultants, first of all it should be mentioned that all three methods give the same implicit equation in all tested cases. So, it does not matter if we add

the additional equation (3) or not, actually it only increases the computation cost. In case of curves, the Dixon resultant seems to be faster than the Dixon dialytic resultant, but in comparison with other presented methods the resultants seems to be quite slow for low degree parametrizations (only the interpolation method is slower). However, the computational costs are less dependent on the degree of the parametrization than in some other methods (mainly the direct method, the method based on Gröbner bases and the method based on interpolation). Hence, the implicitization method based on resultants can be much faster for high degree parametrization than these methods.

In our opinion, the worst implicitization method in MATLAB (among methods presented in this paper) is the interpolation method. This method is the slowest for all tested examples, except the usage of the Dixon resultant for the system with the additional equation (3) which is used very rarely (almost never) and is very slow for all types of parametrizations independently on the degree of parametrization. A direct implementation of the algorithm in MATLAB also suffers from rounding errors because of huge intermediate numbers appearing in the computation, and that's why it yields a wrong answers to the implicitization problem.

On the other hand, the best implicitization methods are the methods of moving lines and moving conics (generally the methods of moving curves). These methods are very fast for low degree and also for high degree parametrizations. For low degree parametrizations the method of moving conics is not so effective and the method of moving lines is a little bit faster. The full strength of methods of moving conics occurs for high degree parametrizations where this method is very fast, the fastest among all presented methods.

The main observation following from the computation of implicit representations of NURBS curves in the Examples 1–4 in MATHEMATICA is that MATHEMATICA is incredibly fast in comparison with MATLAB. The computation costs for different implicitization methods used in MATHEMATICA are very similar, the differences are very small, but similarly to the case of computation in MATLAB as the fastest implicitization method seems to be method of moving curves. The speed of the interpolation method is quite similar to the method which uses the Dixon resultant, sometimes the interpolation method is faster, sometimes not. The problem concerning the interpolation method in MATLAB — a wrong answer of the algorithm because of rounding errors — was removed in MATHEMATICA. The direct method for curves is also very fast in MATHEMATICA, in all examples even faster than the interpolation method or the method which uses the Dixon resultant. For the direct method, the problem with Example 4 in MATLAB where the method returned a completely wrong answer also disappeared in MATHEMATICA.

Quite interesting are computation costs for the implicitization method based on Gröbner bases. For low degree parametrizations, the method is very fast (as in MATLAB (COCOA)), but in MATHEMATICA the implicit representation of NURBS curve from Example 4 can also be found in a very short time (compare the computation time needed by MATHEMATICA to obtain the implicit representation with that needed by MATLAB (COCOA)).

We also tested examples with floating point numbers in coordinates and weights of control points, not only with integers. In such a case the floating point numbers are approximated by rational expressions and we obtain very similar computational costs as for integer coefficients (since it is necessary to do this approximation only once, at the start of the algorithm, to rationalize given parametrization). Of course, the computations are a little bit slower (but similarly in both mathematical software), but it is probably caused by computations with more complicated fractions. Thus, we decided not to study this situation in more detail.

4. Application to NURBS surfaces

A *NURBS surface of degree p in u and degree q in v* is defined by

$$S(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}}, \qquad 0 \le u,v \le 1, \tag{30}$$

where $P_{i,j}$ are control points (forming a control net), $w_{i,j}$ are weights of control points and $N_{i,p}(u), i = 0,\dots,,m$, and $N_{j,q}(v), j = 0,\dots,n$, are the B-spline basis functions defined on non-periodic (and non-uniform) knot vectors

$$U = \{\underbrace{0,\dots,0}_{p+1}, u_{p+1},\dots,u_{r-p-1}, \underbrace{1,\dots,1}_{p+1}\}, \qquad V = \{\underbrace{0,\dots,0}_{q+1}, v_{q+1},\dots,v_{s-q-1}, \underbrace{1,\dots,1}_{q+1}\}$$

by the same recurrent formula as in the case of NURBS curves. Reader can find more details on NURBS surfaces in [14].

4.1. Examples and comparison of methods for surfaces

As in the case of curves, several testing examples will be presented now. The implicitization methods described in Section 2 can also be used to find an implicit equation of a rational surface, and all these methods were implemented by the authors for the case of surfaces also in MATLAB 6.5. Similarly to the case of curves, almost all methods (except the method which uses the Dixon dialytic resultant and the method which uses the Dixon resultant for the system of equations (5) and (6)) were implemented by the authors also in MATHEMATICA 5.0.

The computation cost (in seconds, obtained by the function CPUTIME in MATLAB 6.5[7] and by the function TIMING in MATHEMATICA 5.0[8]) needed by each method to solve the implicitization problem for each segment of the NURBS surfaces in the Examples 5–6 are summarized in Tables 5–6. There are two rows denoted by "Dixon resultant" differing only in superscript. "Dixon Resultant[1]" means that the Dixon resultant for the system of equations (5) with respect to u and v was computed to find an implicit equation of the given surface. "Dixon Resultant[2]" means that the Dixon resultant for the system (5) together with the additional equation (6) with respect to s, u and v was computed to find the implicit equation. The row denoted by "Dixon Dialytic" means that the Dixon dialytic resultant for the system (5) with respect to u and v was computed to find an implicit equation. Each time value in the table was acquired from several initiations of a corresponding program for implicitization.

Example 5 A NURBS surface is given by a control net (4×3 points) with all weights equal to 1 and by the knot vectors $U = \{0,0,0,1/2,1,1,1\}$, $V = \{0,0,0,1,1,1\}$ (see Fig. 3 (left)). Computational costs needed for finding implicit equations of all parts of the NURBS surface are summarized in Table 5. ■

[7] MATLAB 6.5 was running on AMD Athlon XP 2500+, 512MB RAM

[8] MATHEMATICA 5.0 was running on Intel Pentium M 1.6GHz, 512 MB RAM (CPU speed similar to AMD Athlon XP 2500+)

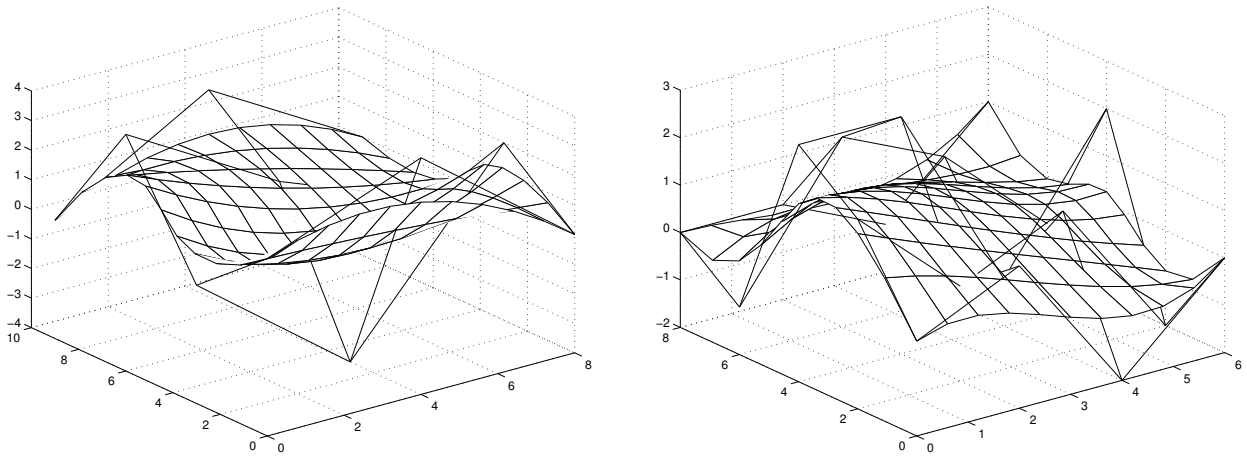

Figure 3: NURBS curves for Examples 5 (left) and 6 (right)

	Matlab 6.5		Mathematica 5.0	
	Patch 1	Patch 2	Patch 1	Patch 2
Gröbner Basis	0.0867	0.0933	0.00333	0.00667
Dixon Resultant[1]	19.5233	18.4467	0.03333	0.03333
Dixon Resultant[2]	19.6300	18.4967	—	—
Dixon Dialytic	3.0000	3.1033	—	—
Interpolation	1364.8333 Wrong result	Integer too large in context	0.08333	0.08667
Moving Planes	4.5133	3.1100	0.10033	0.097
Moving Quadrics	6.1833	30.1233	0.26367	0.27633
Direct	375.0633	5898.77	16.417	18.0993

Table 5: Computational costs (in seconds) for Example 5

	Matlab 6.5		Mathematica 5.0	
	Patch 1	Patch 2	Patch 1	Patch 2
Gröbner Basis	More than 24 hours	More than 24 hours	Abnormal program termination	Abnormal program termination
Dixon Resultant[1]	1713.9	1743.87	0.985	0.95467
Dixon Resultant[2]	1720.3	1748.13	—	—
Dixon Dialytic	10.3633	10.3367	—	—
Interpolation	Out of memory	Out of memory	1.699	1.67233
Moving Planes	46.32	161.243	0.80433	0.79767
Moving Quadrics	2025.3	Integer too large in context	945.156	950.146
Direct	18056	19309	1266.86	1353.85

Table 6: Computational costs (in seconds) for Example 6

Example 6 A NURBS surface is given by a control net (5×4 points) with all weights equal to 1 and by the knot vectors $U = \{0, 0, 0, 0, 1/2, 1, 1, 1, 1\}$, $V = \{0, 0, 0, 0, 1, 1, 1, 1\}$ (see Fig. 3 (right)). Computational costs needed for finding implicit equations of all parts of the NURBS surface are summarized in Table 6. ■

From Tables 5–6 it is obvious that the conclusion about the implicitization methods for surfaces will be similar to the case of curves. In case that MATLAB 6.5 was used, the method based on a Gröbner bases is again fast for low degree parametrizations but has significant problems with the bicubic patches in Example 6 where the computation of a Gröbner basis using CoCoA does not end even after 24 hours. The method which uses the Dixon resultant gives results in both examples but the computations last quite long. It is interesting that the computational costs are very similar independently on the used system of equations for the Dixon resultant computation (with or without the additional equation (6)). The method which uses the Dixon dialytic resultant seems to be very fast for surfaces — this method is surprisingly the fastest of all implicitization methods used for surfaces. As in the case of curves, the method of moving surfaces is fast. For the low degree surfaces the method of moving planes is faster than the method of moving quadrics which would be probably faster for surfaces of higher degree. The biggest problems arose again with the interpolation method (in MATLAB). The function $F(x, y, z)$ returned by the interpolation method does not represent in most cases the implicit equation of the given patch. The reason lies in rounding errors because the method is very sensitive to rounding errors. Sometimes the method does not return anything because MATLAB crashes after several hours of computation with the "Out of memory" error message.

MATHEMATICA 5.0 is — as in the case of curves — much faster than MATLAB 6.5 for the implicitization of surfaces. A problem appears only when the implicitization using a Gröbner basis is applied to Example 6. After several hours of computation, MATHEMATICA always crashes with "Runtime error: Abnormal program termination" error message. Thus, MATHEMATICA is not able to find the Gröbner basis in this case. The direct method is much slower than all other methods in both examples. Thus, the direct method is more suitable for curves than for surfaces. The method of moving quadrics is not so fast for these low degree surfaces and has a considerable problem to implicitize the NURBS surface given in Example 6. The rest of the methods — method based on the Dixon resultant, method of moving planes and interpolation method — are very fast in both examples. For surfaces of higher degree it seems that the method of moving planes will be faster than other methods. Surprisingly, the interpolation method is very fast in MATHEMATICA providing a correct implicit equation in both examples (no problems with rounding errors as in MATLAB).

The main conclusion of this comparison is that *the fastest implicitization method is the method of moving curves and surfaces.*

Acknowledgements

Both authors were supported by the Research Plan MSM 4977751301. This support has been highly appreciated.

References

[1] C. Alonso, J. Gutierrez, T. Recio: *An Implicitization Algorithm with Fewer Variables*. Comput.-Aided Geom. Design **12**, 3, 251–258 (1995).

[2] B. Bastl: *Symbolic manipulation in geometric modeling*. Dissertation. Dept. of Mathematics, Fac. of Applied Sciences, Univ. of West Bohemia, Pilsen 2004.

[3] T. Becker, V. Weispfenning: *Gröbner bases — A Computational Approach to Commutative Algebra*. Graduate Texts in Mathematics, Springer-Verlag, New York 1993.

[4] D. Cox, J. Little, D. O'Shea: *Ideals, Varieties, and Algorithms*. Undergraduate Texts in Mathematics, Springer-Verlag, New York 1992.

[5] E. Chionh, M. Zhang, R. Goldman: *Implicitization Matrices in the Style of Sylvester with the Order of Bezout*. Proceedings of Curve & Surface Design: Internat. Conference Saint-Malo, 17–26, 2000.

[6] E. Chionh, M. Zhang, R. Goldman: *Efficient Implicitization of Rational Surfaces by Moving Planes*. Proceedings of ASCM 2000, 142–151, 2000.

[7] A.-D. Chtcherba: *A new Sylvester-type Resultant Method based on the Dixon-Bézout Formulation*. Dissertation, Dept. of Computer Science, Univ. of New Mexico, 2003.

[8] D. Kapur, T. Saxena, L. Yang: *Algebraic and Geometric Reasoning Using Dixon Resultants*. Proceedings Internat. Symposium on Symbolic and Algebraic Computation 1994 (ISSAC '94), Oxford, United Kingdom, 99–107, 1994.

[9] D. Kapur, T. Saxena: *Comparison of Various Multivariate Resultant Formulations*. Proc. Internat. Symposium on Symbolic and Algebraic Computation 1995 (ISSAC '95), Montreal, Canada, 187–194, 1995.

[10] D. Manocha, F. Canny: *Implicit Representation of Rational Parametric Surfaces*. J. Symbolic Computation **13**, 5, 485–510, Academic Press, 1992.

[11] D. Manocha, F. Canny: *Algorithm for Implicitizing Rational Parametric Surfaces*. Comput.-Aided Geom. Design **9**, 1, 25–51 (1992).

[12] A. Marco, J.J. Martínez: *Using Polynomial Interpolation for Implicitizing Algebraic Curves*. Comput.-Aided Geom. Design **18**, 4, 309–319 (2001).

[13] A. Marco, J.J. Martínez: *Implicitization of Rational Surfaces by Means of Polynomial Interpolation*. Comput.-Aided Geom. Design **19**, 5, 327–344 (2002).

[14] L. Piegl, W. Tiller: *The NURBS Book*. Monographs in Visual Communications. Springer, Berlin 1997.

[15] L. Robbiano: *CoCoA System* [computer programm]. Vers. 4.2 for Windows, Italy, 2002, [cited 2003-04-15]. Available on <http://cocoa.dima.unige.it>. Computer Algebra System. Freeware.

[16] T. Saxena: *Efficient Variable Elimination using Resultants*. Doctoral Thesis, Dept. of Computer Science, State Univ. of New York at Albany, 1996.

[17] T. Sederberg, F. Chen: *Implicitization using Moving Curves and Surfaces*. Computer Graphics Proc., Annual Conference Series, 301–308, ACM SIGGRAPH 1995.

[18] T. Sederberg, R. Goldman, H. Du: *Implicitizing Rational Curves by the Method of Moving Algebraic Curves*. J. Symbolic Computation **23**, 2-3, 153–175 (1997).

[19] D. Wang: *A simple method for implicitizing rational curves and surfaces*. J. Symbolic Computation **38**, 1, 899–914 (2004).

Received February 4, 2005; final form June 1, 2005