A general and efficient approach to implicitization *

Olivier Ruatta[†]and Philippe Trebuchet [‡]
email : ruatta@mail.dm.unipi.it
trebuche@calfor.lip6.fr

April 15, 2003

Abstract

In this paper we propose a general algorithm for rational hypersurfaces implicitization even if there is base points. Efficiency of the method is illustrated by experiements.

1 Introduction

In this paper, we propose a general and efficient algorithm to implicitize rational hypersurfaces in $\mathbb{K}^n$, where $\mathbb{K}$ is an effective field (possibly of finite characteristic but we are particularly interested in the case where $\mathbb{K} = \mathbb{Q}$). Our motivations come from computer aided geometric design (CAGD) and therefore we will only consider rational surfaces in $\mathbb{K}^3$ but the approach can easily generalized to the case of general hypersurfaces in K^n. The results described here were presented for the ACA conference in 2002 in Volos (Greece).

Currently, in CAGD, a rational surface (S) can be represented in two different ways :

- the parametric representation $(S) : \begin{cases} x = \frac{h_1(s,t)}{h_0(s,t)} \\ y = \frac{h_2(s,t)}{h_0(s,t)} \\ z = \frac{h_3(s,t)}{h_0(s,t)} \end{cases}$ where h_0, h_1, h_2 and $h_3 \in \mathbb{K}[s,t]$
- the implicit representation $(S) = \mathcal{Z}(P) = \left\{(x,y,z) \in \mathbb{K}^3 \mid P(x,y,z) = 0\right\}$ where $P \in \mathbb{K}[x,y,z]$

The two representations have their own computational interests. In this paper we are interested in finding a general and efficient algorithm to recover the implicit representations (i.e. the implicit equations) of rationales surfaces from their parametric representations. This problem is a real computational challenge since, more and more, CAGD software work with rational surfaces (as splines) for solid modeling.

---

*Work partially supported by the COLOR project SIMPLES
†Università di Pisa
‡Laboratoire d'Informatique de Paris 6

Several methods are known to treat this problem. The two early methods are based on resultants and Grœbner bases (see [24]). The method based on Grœbner bases leads to untractable computations and the method based on classical resultant formulations gives only a multiple of the the implicit equation and therefore a factorization is required (that leads to real computational difficulties). Furthermore, this two methods failed when there is base points (i.e. when $\mathcal{Z}(h_0, h_1, h_2, h_3) = \{(s,t) \in \mathbb{K}^2 \mid h_i(s,t) = 0, \forall i \in \{0, \ldots, 3\}\} \neq \emptyset$). The Grœbner bases techniques can theoretically treat problems with base points by adding a new variable, but this increase the complexity of the computations. More recently, Jouanolou give a general formula to compute the implicit equation of rational surfaces based on (anisotropic) resultant (see [25]). This technique were found independently and later by Aries and Senoussi ([2]) in a more algorithmic way. Jouanolou shown in [25] that this methods only work for surface in $\mathbb{K}^3$ and not for hypersurfaces in arbitrary dimension. In a other way, a new technique appear developed by Sederberg ([32]) and several coauthors. The method work in some cases with base points but need geometric conditions on the base points [12, 6]. A technique of residual resultant were developed to treat problems with base points but also need conditions on those points (see [5]). An other method based on the computation of the algebraic residue were initiated by Gonzales-Vega and Trujillo in [22, 21] and generalized in [16, 30]. This method is general (i.e. allow to treat base points and work in every dimension) but its complexity is still unknown.

We propose here a method based in Grœbner bases techniques. Our approach is an algorithmic improvement of the method proposed by Liccardi and Mora in [29] and later more effectively by Kalkbrener in [27]. We use the wide generality of Grœbner bases to reduce the problem to linear algebra computations. The high complexity of Grœbner bases is partially avoided thanks to the use of modular arithmetic and term order conversion. The computation of a Grœbner bases and a linear algebra computation reduce the problem to a well posed interpolation problem. The base points problem is treated thanks to an algebraic manipulation call localization. We obtain a very efficient algorithm for implicitization.

The paper is organized as follows : In section 2, we expose the use of the Grœbner bases and linear algebra computation we do in the treatment of our problem. We also quickly describe the algebraic and geometric ideas to treat the base points setting. In section 3, we describe explicitly the algorithm and study its complexity. In section 4, we describe practical experiments to illustrate the efficiency of the method. In the last section, we conclude and we give some perspectives.

2 Grœbner bases and linear algebra for implicitization

In this section, we describe the theoretical background of our approach. We begin in recalling some basic results those show how a Grœbner basis allows to compute the implicit equation with or without base points. This approach is based on a linear algebra reduction process to convert monomial term order. In a second part we show how to use modular arithmetic to reduce the problem to an interpolation one.

2.1 Theoretical background

The two key points that we will use are the fact that a Grœbner basis of an ideal is a generating set of the ideal and that the ideal associated to an hypersurface is a principal ideal. Let us consider the following parameterized surface :

$$(S) : \begin{cases} x = \frac{h_1(s,t)}{h_0(s,t)} \\ y = \frac{h_2(s,t)}{h_0(s,t)} \\ z = \frac{h_3(s,t)}{h_0(s,t)} \end{cases}$$

We want to recover a minimal algebraic relation between x, y and z, i.e. the equation of the algebraic set defined as the Zariski closure of the set of point parameterized by $(s,t) \mapsto \left(\frac{h_1(s,t)}{h_0(s,t)}, \frac{h_2(s,t)}{h_0(s,t)}, \frac{h_3(s,t)}{h_0(s,t)}\right)$, i.e. the implicit equation of the surface (S). We denote $H_1(x,s,t) = h_0(s,t)x - h_1(s,t)$, $H_2(s,t) = h_0(s,t)y - h_2(s,t)$ and $H_3(s,t) = h_0(s,t)z - h_3(s,t)$. We denote by $\mathcal{I} = (H_1, H_2, H_3) \subset \mathbb{K}[s,t,x,y,z]$. The classical approach, when there is no base point, consists in computing the generator of the ideal $\mathcal{I} \cap \mathbb{K}[x,y,z]$ which is the wanted equation defined up to the scalar multiplication. It is to say that $(S) = Z(\mathcal{I} \cap \mathbb{K}[x,y,z])$ in this case.

A problem occurs when there is base points. We say that $\zeta \in \mathbb{K}^2$ is a base point of the parameterization if $h_0(\zeta) = \cdots = h_3(\zeta) = 0$. So if $\zeta \in \mathbb{K}^2$ is a base point, then, for all $(x,y,z) \in \mathbb{K}^3$ we have $H_1(x,\zeta) = H_2(y,\zeta) = H_3(z,\zeta) = 0$. This implies that $\mathcal{I} \cap \mathbb{K}[x,y,z] = \mathbb{K}^3$. To avoid this difficulty, we introduce a new variable w and we consider the new equation $H_0(w,s,t) = h_0(s,t)w - 1$. We denote by $\tilde{\mathcal{I}} = (H_0, H_1, H_2, H_3) \subset \mathbb{K}[s,t,w,x,y,z]$. It is not difficult to show that $(S) = Z(\tilde{\mathcal{I}} \cap \mathbb{K}[x,y,z])$. Then we need to compute a generator of $\tilde{\mathcal{I}} \cap \mathbb{K}[x,y,z]$. This method work even if there is no base point and we will use it systematically despite it implies higher complexity.

2.2 Term order conversion

From here-above remarks, we see that a natural framework to recover the implicit equation is to consider the ideal $\tilde{\mathcal{I}}$. Indeed as this ideal describes only (S) out of the base points of the given parameterization, nothing tricky can appear in this context.
In the sequel we will denote by A the quotient algebra

$$A = \mathbb{C}[x,y,z,w,s,t]/\tilde{\mathcal{I}}$$

Actually, a polynomial belonging to $\tilde{\mathcal{I}}$ is sent to 0 in A, so we can see the implicit equation, in A, as a linear dependence relation between the monomials of its support. Thus our problem amounts to work in the quotient algebra A and to find there a particular relation between the monomials in x, y, z. A well known way to work in a quotient algebra of polynomials is by using Grœbner bases. Grœbner bases are a powerful tool in computational algebraic geometry appeared in 1965 [4]; that, given a polynomial p, allows to find a canonical representative for the class $\bar{p}$ of p in A. We will not develop any further here theory and refer the interested reader to [9] for a more comprehensive description of this theory. The feature of interest here is that they provide a way to compute a canonical expression for every element in A.

Using Grœbner bases, we are now able to find the linear relation between the monomials in x, y, z we are looking for:

- compute a Grœbner basis for $\mathcal{I}$.
- Let k be an integer inferior or equal to the degree of the implicit equation. Construct the list L_k of all monomials of degree inferior or equal to k, enumerated in the Degree lexicographical order. Construct the set of all normals forms of all monomial of L_k with respect to the computed Grœbner base. Construct the list C_k of all monomials which appear in the support of the normals forms. Construct M_k a matrix with $l = \#L_k$ lines and $c = \#C_k$ columns (where $\#$ is the number of elements). Set the entry $M_k[i, j]$ of the matrix ($1 \leq i \leq l$ and $1 \leq j \leq c$) with the value of the coefficient of the monomial $C_k[j]$ in the normal form of $L_k[i]$. If the normal forms are linearly dependent (i.e. M_k has not the maximal rank), stop. Else increase k and go stepforward.

Theorem 2.1 *The process stops and at the end, the vector of the kernel of M_k^t is formed by the coefficients of the implicit equation.*

Proof :

Assume that the process stop. Then, the vector of the kernel of M_k^t is a linear dependency relation between the rows of M_k (i.e. the normal forms of the monomials of L_k) in A. So it is also a linear dependency relation between the monomials in L_k in A, i.e. it is a multiple of the implicit equation. Remark now that this relation has minimal degree. Otherwise there is a relation of lower degree lower than k and the matrix has not a maximal rank for integer higher than the degree of this relation and the process has stopped before, which lead to a contradiction. So the vector of the kernel of M_k^t is the implicit equation if the process stop. Furthermore the kernel of M_k^t has dimension 1 since there is only one implicit equation up to the multiplication by a scalar.

Now the process always stop since such linear dependency relation exists and has a finite degree, the degree of the implicit equation. ♣

Finally we must emphasize here that the monomial order chosen for computing the Grœbner basis **need not be an elimination order**, in fact it can be any monomial ordering. The construction of the matrix and the reduction we propose is similar to the FGLM algorithm for elimination in codimension one setting (in fact it generalizes an algorithm proposed by Faugère in [17]). It is an alternative to Grœbner walks as exposed in [27]. The simplicity of our method allows simple and powerful implementation as we will see in the section 3.

2.3 Using modular arithmetic

We have now a general method for recovering the implicit equation from the parameterized expression of our surface. Though this result seems satisfying it faces one of the main drawbacks of exact Grœbner bases computations, namely the tremendous growth of the coefficients, see [18] for further explanations on this phenomenon. In fact modular computations are at least one order faster than integer ones. And we could be interested in determining exactly what kind of information such computations could provide. In fact the whole method hereabove described can apply over finite fields, and with a very high probability we obtain the modular trace of R over the finite field we are working in. Also with a very high probability the support of this modular trace is the actual support of R. By very high probability we mean that there is finitely many primes that would make such a computation fail, and as there are infinitely many primes the probability of failure is 0.

This modular computation allows to deduce the monomial support of the implicit equation since we know its modular trace. Hereafter, we describe how recover the true implicit equation from its monomial support.

2.3.1 Reduction to an interpolation problem

Recovering the implicit equation when we know its monomial support is easy because we know the parameterization of (S). The knowledge of the parameterization allows us to generate points lying in (S). We can then reduce the computation of the coefficient of the implicit equation to an interpolation problem.

We assume to know the set $E \subset \mathbb{N}^3$ of multi-index such that $\{m_\alpha = x^{\alpha_1} y^{\alpha_2} z^{\alpha_3} | \alpha = (\alpha_1, \alpha_2, \alpha_3) \in E\}$, it is the monomial support of the implicit equation P (i.e. $P(x,y,z) = \sum_{\alpha \in E} p_\alpha m_\alpha$). We consider $\mathbb{C}[x,y,z]_E = \bigoplus_{\alpha \in E} \mathbb{C} m_\alpha$, that is to say the $\mathbb{C}$ vector space generated by the monomial with exponent in E. We denote by $d = \#(E) = \dim(\mathbb{C}[x,y,z]_E)$.

Proposition 2.2 *If E is the set of multi-index of monomials of the support of P and if $F \in \mathbb{C}[x,y,z]_E$ is such that $F(\zeta) = 0$ for all $\zeta \in (S)$ then F is a generator of $\mathcal{I} \cap \mathbb{C}[x,y,z]$.*

Remark 2.3 *This proposition underlines the fact that the generator of $\mathcal{I} \cap \mathbb{C}[x,y,z]$ is defined up to a scalar multiplication.*

Definition 2.4 *Let $E = \{\alpha_1, \ldots, \alpha_d\}$ be a set of multi-index with $\#(E) = d$ and $\mathcal{Z} = \{\zeta_1, \ldots, \zeta_d\} \subset (S)$ be a set of d distinct points of (S), we define the pseudo-Vandermonde matrix associated with E and $\mathcal{Z}$ as follow :*

$$V_{E,\mathcal{Z}} = \begin{pmatrix} m_{\alpha_1}(\zeta_1) & \ldots & m_{\alpha_d}(\zeta_1) \\ \vdots & \ddots & \vdots \\ m_{\alpha_1}(\zeta_d) & \ldots & m_{\alpha_d}(\zeta_d) \end{pmatrix}$$

Proposition 2.5 *If E is the set of multi-index of monomials of the support of P and $\mathcal{Z}$ is a set of d distinct points of the surface (S) such that $V_{E,\mathcal{Z}}$ has rank $d-1$, then the polynomial associated with a vector of the kernel of $V_{E,\mathcal{Z}}$ is a generator of $\mathcal{I} \cap \mathbb{C}[x,y,z]$.*

Proof : The fact that $V_{E,\mathcal{Z}}$ has exactly rank $d-1$ reveals the fact that the points in $\mathcal{Z}$ do not satisfy relation with less monomials than all those are indexed by E. Suppose that $F \in \mathbb{C}[x,y,z]_E$ satisfy $F(\zeta_1) = \cdots = F(\zeta_d) = 0$ (i.e. the vector formed with the coefficient of F is in the kernel of $V_{E,\mathcal{Z}}$) and that F is not a generator of $\mathcal{I} \cap \mathbb{C}[x,y,z]$. Then this relation is maximal in the sense that it contains all the monomials indexed by E (because $V_{E,\mathcal{Z}}$ has rank $d-1$). In this case the fact that $F(\zeta_i) = P(\zeta_i) = 0$ for all $i \in \{1,\ldots,d\}$ implies that there exists $\lambda \in \mathbb{C}$ such that $P = \lambda F$ and so $(P) = (F)$. This is in contradiction with our assumptions and so $(P) = (F) = \mathcal{I} \cap \mathbb{C}[x,y,z]$. ♣

This proposition allows to recover the implicit equation from the monomial support of the implicit equation with the computation of a vector of the kernel of $V_{E,\mathcal{Z}}$.

3 Algorithm

In this section we present our algorithm and give some indications on the complexity of your approach.

The algorithm is directly derived of the results of the previous section.

Algorithm 3.1
INPUT:$S = (h_0(s,t), h_1(s,t), h_2(s,t), h_3(s,t)),\ \ s,t \in \mathbb{R}$

- *Compute $H_0,\ldots,H_3$ from S*
- *choose a prime p*
- *Compute a Grœbner basis of $\tilde{\mathcal{I}}$ modulo p for a monomial order that need not be the lexicographical one*
- *Set $k = 0$ $M_0 = 1$ and for $k > 0$ compute L_k, C_k and M_k*
- *While ($rank(M_k) = max(nbrows(M_k), nbcols(M_k))$) do $k = k+1$*
- *Construct $Supp = \{$the monomials $L_k[i]\}$ whose normal form belongs to the linear dependence relation in M_k*
- *Compute $pts = \{\zeta_1 \ldots \zeta_{\#Supp}$, points whose lie on the surface $S\}$*
- *Compute the pseudo-Vandermonde matrix $V = (L_k[j](\zeta_i))$*
- *Find a vector v in its kernel.*
- *check that $\Sigma_{i=1..\#Supp} m_i v_i \ \in \mathbb{C}[x,y,z]$ is in $\tilde{\mathcal{I}}$ if not choose another p and restart.*

OUTPUT: *the implicit equation*

The described algorithm is not the implemented one. We do not need to compute the rank of the matrix naively at each step. The matrix becomes quickly huge. In fact, we made a trigonalization on the matrix by row operations and report operations on L_k (swap on columns are allowed but must be reported on C_k). If the last row of the matrix is formed by zeroes at the end of the trigonalization, the last entry of L_k is the implicit equation modulo p. Else, at step $k+1$, we add the news the monomials at the end of L_{k+1} and C_{k+1}. This computational manipulation is a great improvement in practice.

4 Experiments

The algorithms are implemented in the computer algebra system MAGMA. We benchmark our algorithms with different kinds of problems. The computation of the Grœbner basis is rarely the high cost step of our approach because we use a modular arithmetic and a suitable order. The order conversion can be really time consuming since the size of the matrix is $\mathcal{O}\left(D^3\right)$, where D is the degree of the implicit equation. We use a modular Gaussian elimination, this give a $\mathcal{O}\left(D^6\right)$ complexity in the worst case, i.e. when the matrix is not sparse. The same problem occur with the interpolation process. If the Vandermonde matrix is not sparse then the complexity of this step is $\mathcal{O}\left(D^9\right)$. Practically our algorithm can treat problems where the degree of the implicit equation is 20 or less, even if the system is not sparse. Bigger system can be treated if the constructed matrices are sparse. We first describe some parametric systems in the following table. In a second table we detail the parameters of the resolution. Computations were done on a Pentium III with 500 Mo of sdram memory.

System's references	Parametric representations
System 1	$h1 := s^3 + t^2$ $h2 := t^4$ $h3 := s^9 + t^1 1$ $h4 := 1$
System 2	$h1 := s^3 + t^2$ $h2 := t^4$ $h3 := s^9 + t^1 1$ $h4 := 1$
System 3	$h1 := s^3 + t^2$ $h2 := t^4$ $h3 := s^9 + t^1 1$ $h4 := s^3 + t^4$
System 4	$h1 := -3 - 3 * s^3 * t$ $h2 := 6 * s * t - 4 * s^2$ $h3 := -6 + 9 * s$ $h4 := t$
System 5	$h1 := s * t * (s + 2 * t^4 - 1)$ $h2 := s * (s - t - 2) * (s * t^2 + 1)$ $h3 := s - t - 2$ $h4 := s * t * (s - t - 2)$
System 6	$h1 := s * (s + t)$ $h2 := (s + t^3) * (t^2 + t * s - 3)$ $h3 := t^2 + s * t - 3$ $h4 := (s + t) * (t^2 + t * s - 3)$
System 7	$h1 := (s^2 + t^2 + t^3) * (t^2 + 1)$ $h2 := (t + s^3) * (s^2) * (t^2 + 1)$ $h3 := s^3$ $h4 := s^2 * (t^2 + 1)$
System 8	$h1 := (2 * s * t + 2 * t^2 - 3 * s^2) * (t - 3 * s) * t * (t - 2 * s)$ $h2 := -s * (2 * t + 1) * s * (t + 1) * t * (t - 2 * s)$ $h3 := (2 + s * t + 3 * t^2) * s * (t + 1) * (t - 3 * s)$ $h4 := s * (t + 1) * (t - 3 * s) * t * (t - 2 * s)$
System 9	$h1 := s^3 + t^3 + 5 * s * t^2 + 2 * s * t - 1$ $h2 := 2 * s^2 + 2 * t^3 + 3 * s * t - 2$ $h3 := 5 * s^3 + 5 * t^2 + 3 * s * t^2 + 2 * s^2 * t + 5 * s * t - 5$ $h4 := s^4 + t^4 - 1$
System 10	$h1 := 32 * s^2 * t - 56 * s * t + 24 * t + 16 * s^3 - 120 * s^2 +$ $128 * s - 24$ $h2 := 16 * t^3 - 16 * t^2 + 16 * s * t^2 + 48 * s^2 * t$ $-456 * s * t + 392 * t + 8 * s^2 + 384 * s - 392$ $h3 := 64 * t^3 - 48 * t^2 + 48 * s * t^2 + 32 * s^2 * t - 1328 * s * t$ $+1232 * t + 16 * s^3 + 192 * s^2 + 1040 * s - 1248$ $h4 := 16 * t^3 - 240 * s * t + 224 * t - 240 +$ $224 * s + 16 * s^3$
System 11	$h1 := 32 * s^2 * t - 56 * s * t + 24 * t + 16 * s^3 - 120 * s^2 + 128 * s - 24$ $h2 := 16 * t^3 - 16 * t^2 + 16 * s * t^2 + 48 * s^2 * t -$ $456 * s * t + 392 * t + 8 * s^2 + 384 * s - 392$ $h3 := 64 * t^3 - 48 * t^2 + 48 * s * t^2 + 32 * s^2 * t - 1328 * s * t +$ $1232 * t + 16 * s^3 + 192 * s^2 + 1040 * s - 1248$ $h4 := -6 + 4 * s - 6 * s * t + 2 * s^2 + 6 * t$

System 12	$h1 := 32*s^2*t - 56*s*t + 24*t + 16*s^3 - 120*s^2 + 128*s - 24$ $h2 := 16*t^3 - 16*t^2 + 16*s*t^2 + 48*s^2*t -$ $456*s*t + 392*t + 8*s^2 + 384*s - 392$ $h3 := 64*t^3 - 48*t^2 + 48*s*t^2 + 32*s^2*t - 1328*s*t +$ $1232*t + 16*s^3 + 192*s^2 + 1040*s - 1248$ $h4 := -6 + 8*s - 2*s^2$
System 13	$h1 := 32*s^2*t - 56*s*t + 24*t + 16*s^3 - 120*s^2 + 128*s - 24$ $h2 := 16*t^3 - 16*t^2 + 16*s*t^2 + 48*s^2*t -$ $456*s*t + 392*t + 8*s^2 + 384*s - 392$ $h3 := 64*t^3 - 48*t^2 + 48*s*t^2 + 32*s^2*t - 1328*s*t +$ $1232*t + 16*s^3 + 192*s^2 + 1040*s - 1248$ $h4 := 2 - 2*s + 2*t$
System 14	$h1 := 32*s^2*t - 56*s*t + 24*t + 16*s^3 - 120*s^2 + 128*s - 24$ $h2 := 16*t^3 - 16*t^2 + 16*s*t^2 + 48*s^2*t -$ $456*s*t + 392*t + 8*s^2 + 384*s - 392$ $h3 := 64*t^3 - 48*t^2 + 48*s*t^2 + 32*s^2*t - 1328*s*t +$ $1232*t + 16*s^3 + 192*s^2 + 1040*s - 1248$ $h4 := -t$
System 15	$h1 := 684288*s*t^2 - 684288*t^2 - 1419264*s^2*t +$ $1495296*s*t - 76032*t + 684288*s^3 - 608256*s^2 -$ $836352*s + 760320$ $h2 := 684288*t^3 - 1119744*s*t^2 + 1119744*t^2 +$ $983808*s^2*t - 8711424*s*t + 7043328*t + 1043712*s^2 +$ $7803648*s - 8847360$ $h3 := 2052864*t^3 + 1430784*s*t^2 -$ $1430784*t^2 - 6983424*s^2*t - 15667200*s*t + 20597760*t +$ $4790016*s^3 - 1126656*s^2 + 17556480*s - 21219840$ $h4 := -4790016*t^3 + 13996800*s*t^2 -$ $13996800*t^2 - 19660032*s^2*t + 74437632*s*t - 49987584*t + 6158592*s^3 -$ $12780288*s^2 - 62152704*s + 68774400$
System 16	$h1 := 14004 - 7632*t - 14004*s + 7716*s*t - 84*s^2*t^2$ $h2 := 1692 - 972*t - 1692*s + 1056*s*t - 84*s*t^2$ $h3 := 4284 - 3024*t - 4284*s + 3360*s*t - 84*s*t^2 - 252*s^2*t$ $h4 := 51 - 57*s + 6*s^2 + 6*t$

System's references	Degree	Memory (Mo)	Times (seconds)	Comments
System 1	6	< 1	0.070	
System 2	12	2	2.450	
System 3	5	< 1	0.080	
System 4	5	< 1	0.060	
System 5	11	4	3.600	
System 6	7	< 1	0.180	
System 7	19	45	352.6	
System 8	10	< 1	5.270	
System 9	13	158	283.0	
System 10	3	< 1	0.290	> 6 base points
System 11	4	< 1	0.100	> 4 base points
System 12	5	< 1	0.310	> 3 base points
System 13	7	< 1	1.490	> 2 base points
System 14	8	9	3.140	1 base point
System 15	1	< 1	0.010	> 8 base points
System 16	5	< 1	0.260	> 2 base points

The systems number 11,12,13 and 14 illustrate the impact of base points on the degree of the implicit equation and so to the complexity of our algorithm.

5 Conclusions and perspectives

We propose a new and efficient algorithm for the implicitization problem. Our approach based on Grœbner basis improves the previous one on several points. The examples treated show that our approach allows to treat problems even if the degree of the implicit equation becomes large. Several improvements of the method are possible :

- We can use a formal Newton operator to lift the modulo p implicit equation to the exact one (see [28])
- We can improve the use of the matrices sparsity in the base conversion process
- We can use a low level language as C or C++ to improve the implementation.

The implicitization problem is a typical algebraic problem that occurs on GCAD. Extension of our algorithm to other problems is direct (offset, ...). The need of more efficient algebraic method in GCAD is a major goal of modern computer algebra. But a comparative study of all the proposed methods is necessary and is still undone.

References

[1] C. Alonso, J. Gutiérrez, and T. Recio. An implicitization algorithm with fewer variables. *Aided Geom. Design*, 12:251–258, 1995.

[2] Franck Aries and Rachid Senoussi. An implicitization algorithm for rational surfaces with no base points. *Journal of Symbolic Computation*, pages 357–365, 2001.

[3] Buchberger B. Algebraic methods for non-linear computational geometry. *Symposium on Computational Geometry*, pages 81–82, 1988.

[4] B. Buchberger. *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal (German)*. PhD thesis, Math. Inst, Univ. of Innsbruck, Austria, 1965. (also in Aequationes Math. 4/3, 1970).

[5] L. Busé. Residual resultant over the projective plane and the implicitization probl em. *proceedings ISSAC2001*, pages 48–55, 2001.

[6] Laurent Busé, David Cox, and Carlos D'Andréa. Implicitization of surfaces in $\mathbb{P}^3$ in the presence of base points. *Preprint*, 2002.

[7] John F. Canny and Dinesh Manocha. Algorithm for implicitizing rational parametric surfaces. *Computer Aided Geometric Design*, 9:25–50, 1992.

[8] John F. Canny and Dinesh Manocha. Implicit representation of rational parametric surfaces. *Journal of Symbolic Computation*, 13:485–510, 1992.

[9] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer Verlag, New York, 1992.

[10] David Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and algorithms*. Springer, 1996.

[11] David A. Cox, Ronald Goldman, and Zhang Ming. On the validity of implicitization by moving quadrics for rationnal surfaces with no base points. *J. Symbolic Computation*, 29, 2000.

[12] Carlos D'Andrea. Resultants and moving surfaces. *Journal of Symbolic Computation*, 31:585–602, 2001.

[13] T. Dokken. Approximate implicitization. *Math. Methods in CAGD; Oslo*, 2001.

[14] David Eisenbud. *Commutative Agebra with a View Toward Algebraic Geometry*. Number 150 in Graduate Text in Mathematics. Springer, 1991.

[15] M. Elkadi and B. Mourrain. *Géométrie algébrique effective en dimension 0 : de la théorie aux applications*. Soumis, 2002.

[16] Mohamed Elkadi, Bernard Mourrain, and Olivier Ruatta. A new algorithm for rational surfaces implicitization. Preprint, 2002.

[17] J.C. Faugère. *Résolution des systèmes d'équations polynômiales*. PhD thesis, Université de Paris VI, 1994.

[18] J.C. Faugère. A new efficient algorithm for computing Gröbner Basis (F4). *J. of Pure and Applied Algebra*, 139:61–88, 1999.

[19] G. Fix, C-P Hsu, and T. Luo. Implicitization of rational parametric surfaces. *J. Symbolic Comput.*, 21:329–336, 1996.

[20] X. S. Gao and S. C. Chou. Implicitization of rational parametric equations. *J. Symbolic Comput.*, 14:459–470, 1992.

[21] L. Gonzalez-Vega. Implicitization of parametric curves and surfaces by using multidimensional newton formulae. *J. Symbolic Comput.*, 23, 1997.

[22] L. Gonzalez-Vega and Trujillo G. Implicitization of parametric curves and surfaces by using symetric functions. *ISSAC'95, ACM Press*, pages 180–186, 1995.

[23] Robin Hartshorne. *Algebraic Geometry.* Number 52 in Graduate texts in mathematics. Springer-Verlag, 1987.

[24] Christoph M. Hoffmann. *Geometric and solid modeling : an introduction.* Morgan Kaufmann publishers, Inc., 1989.

[25] J.-P. Jouanolou. Résultant anisotrope: Compléments et applications. *The electronic journal of combinatorics*, 3(2), 1996.

[26] M. Kalkbrener. Implicitization of rational parametric curves and surfaces. applied algebra, algebraic algorithms and error-correcting codes. *Lecture Notes in Comput. Sci.*, 508:249–259, 1990.

[27] M. Kalkbrener. Implicitization by groebner basis conversion. *Euromath Bulletin*, 2:197–204, 1996.

[28] Gregoire Lecerf. *Une altérnative aux méthodes de réécritures pour la résolution des systèmes algèbriques.* PhD thesis, École Polytechnique, 2001.

[29] Sandra Liccardi and Teo Mora. Implicitization of hypersurfaces and curves by the primbasissatz and basis conversion. *proceding ISSAC 94*, pages 191–196, 1994.

[30] Olivier Ruatta. *Dualité algebrique, structures et applications.* PhD thesis, Université de la Méditerranée, 2002.

[31] T. Sederberg, Anderson D., and R. Goldman. Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics, and Image Processing*, pages 72–84, 1984.

[32] T.W. Sederberg and F. Chen. Implicitization using moving curves and surfaces. *Proceedings of SIGGRAPH*, pages 301–308, 1995.