

Fast Computation of the Bezout and Dixon Resultant Matrices

Eng-Wee Chionh

chionhew@comp.nus.edu.sg

School of Computing

National University of Singapore

Lower Kent Ridge Road, Singapore 119260

Ming Zhang

mzhang@cs.rice.edu

Ronald N. Goldman

rng@cs.rice.edu

Department of Computer Science

Rice University

Houston, Texas 77005

Abstract

Efficient algorithms are derived for computing the entries of the Bezout resultant matrix for two univariate polynomials of degree n and for calculating the entries of the Dixon-Cayley resultant matrix for three bivariate polynomials of bidegree (m, n) . Standard methods based on explicit formulas require $\mathcal{O}(n^3)$ additions and multiplications to compute all the entries of the Bezout resultant matrix. Here we present a new recursive algorithm for computing these entries that uses only $\mathcal{O}(n^2)$ additions and multiplications. The improvement is even more dramatic in the bivariate setting. Established techniques based on explicit formulas require $\mathcal{O}(m^4n^4)$ additions and multiplications to calculate all the entries of the Dixon-Cayley resultant matrix. In contrast, our recursive algorithm for computing these entries uses only $\mathcal{O}(m^2n^3)$ additions and multiplications.

Keywords: Algebraic Geometry; Computer Graphics; Geometric Modeling; Robotics; Elimination Theory; Resultant

1 Introduction

Resultants are a classical algebraic tool for determining whether or not a system of n polynomials in $n - 1$ variables have a common root without explicitly solving for the roots. By

systematically eliminating the variables, one can obtain a single polynomial expression in the coefficients of the original polynomial equations whose vanishing signals the existence of a common root. Elimination theory was once a classical branch of algebraic geometry, but because the computations involved were often unwieldy, by the middle of this century it had fallen out of fashion in favor of more abstract, less constructive techniques.

Computers revived elimination theory by enabling complex computations that were previously beyond the reach of mathematicians. Resultants are now applied systematically to provide constructive solutions to problems in computer graphics [5], robotics [6], geometric modeling [4, 10] and algorithmic algebraic geometry [2]. Groebner bases are another more modern manifestation of this contemporary renewal of elimination theory. Resultants and Groebner bases are currently standard tools in contemporary computer algebra systems such as MAPLE and MATHEMATICA.

Resultants are often represented as the determinant of a matrix whose entries are polynomials in the coefficients of the original polynomial equations. Since these matrices may be very large, especially in the multivariate setting, research has focused on efficient computation of these determinants [7]. Remarkably, however, little attention has been paid to efficient computation of the entries of these resultant matrices. The purpose of this paper is to address this oversight.

In elimination theory, there are two standard formulations for the resultant of two univariate polynomials of degree n : the *Sylvester resultant* and the *Bezout resultant*. The Sylvester resultant is the determinant of a matrix of order $2n$; the Bezout resultant is the determinant of a matrix of order n . Thus the Bezout determinant is generally faster to compute. But whereas the non-zero entries of the Sylvester resultant are just the coefficients of the original two polynomials, the entries of the Bezout resultant are much more complicated expressions in these coefficients. Standard techniques based on explicit formulas require $\mathcal{O}(n^3)$ additions and $\mathcal{O}(n^3)$ multiplications to compute the entries of the Bezout matrix. Here we shall present a new recursive algorithm for computing these entries that requires only $\mathcal{O}(n^2)$ additions and $\mathcal{O}(n^2)$ multiplications.

For three bivariate polynomials of bidegree (m, n) , Dixon presents several different formulations for the resultant [3]. We focus here on two of these representations that are analogous to the univariate resultants of Sylvester and Bezout. We shall call the first approach the *Sylvester resultant* because it can be generated by Sylvester's dialytic method. The second technique we shall call the *Cayley resultant* because it can be constructed by a stratagem similar to Cayley's device for generating the Bezout resultant. In the literature, this Cayley resultant is often called the *Dixon resultant*.

The non-zero entries of the bivariate Sylvester resultant matrix are again just the coefficients of the original three polynomials, but the Sylvester matrix is order $6mn$ which is huge even for bivariate polynomials of moderate bidegree. The Cayley resultant matrix is of order $2mn$, which is much more manageable, but the entries of the Cayley resultant are also much harder to compute. Indeed standard methods based on explicit formulas require $\mathcal{O}(m^4n^4)$ multiplications and $\mathcal{O}(m^4n^4)$ additions to compute these entries. Here we shall develop a new recursive algorithm for calculating the entries of the Cayley resultant matrix that requires only $\mathcal{O}(m^2n^3)$ additions and $\mathcal{O}(m^2n^3)$ multiplications.

To develop these fast algorithms, we have organized the remainder of this paper in the following fashion. In Section 2 we introduce an innovative technique for dividing by $x - y$ a bivariate polynomial $f(x, y)$ that vanishes when $x = y$. We shall have occasion to apply this method several times throughout this paper, since Cayley's device for generating both univariate and bivariate resultants requires exactly this type of division. In Section 3 we apply this division technique to transform the Sylvester resultant matrix into the Bezout resultant matrix and we then use this transformation to develop a fast recursive algorithm for computing the entries of the Bezout resultant. As a bonus, along the way we shall provide an elementary constructive proof that the polynomials represented by the columns of the Bezout resultant lie in the ideal generated by the original two polynomials.

Section 4 is devoted to deriving similar results for the Cayley resultant of three bivariate polynomials of bidegree (m, n) . Again to develop a fast algorithm for computing the entries of the Cayley resultant, we apply our division technique to transform the Sylvester resultant matrix into the Cayley resultant matrix. We take advantage too of the natural block structures that appear in the Sylvester resultant and its accompanying transformation matrix, and we make use as well of the fast algorithm we developed in Section 3 for computing the entries of the Bezout matrix. As in the univariate setting, we also provide an elementary constructive proof that the polynomials represented by the columns of the Cayley resultant lie in the ideal generated by the original three bivariate polynomials.

2 Exact Division by Truncated Formal Power Series

We begin with a device for dividing a bivariate polynomial $f(x, y)$ that vanishes when $x = y$ by the expression $x - y$. This technique will play a central role in our subsequent analysis.

Consider the quotient $\sum_{i=0}^n a_i(y)x^i/(x - y)$, where $a_i(y)$, $0 \leq i \leq n$, are polynomials in y . By treating $1/(x - y)$ as the formal power series $\sum_{u=1}^{\infty} x^{-u}y^{u-1}$, we can write the quotient as the product

$$\sum_{i=0}^n a_i(y)x^i \sum_{u=1}^{\infty} x^{-u}y^{u-1} = \sum_{i=0}^n a_i(y)x^i \sum_{u=1}^i x^{-u}y^{u-1} + \sum_{i=0}^n a_i(y)x^i \sum_{u=i+1}^{\infty} x^{-u}y^{u-1},$$

where the vacuous sum $\sum_{u=1}^0 \cdots$ is taken to be zero. Observe that the first sum on the right hand side is a polynomial in x, y , but the second sum involves only negative powers of x . Thus the quotient is a polynomial if and only if the second sum vanishes. Consequently,

$$\frac{\sum_{i=0}^n a_i(y)x^i}{x - y} = \sum_{i=1}^n a_i(y) \sum_{k=0}^{i-1} y^{i-1-k} x^k, \quad (1)$$

if and only if the quotient is a polynomial — that is, if and only if $\sum_{i=0}^n a_i(y)x^i \equiv 0$ when $x = y$.

An alternative to the above expression can be derived from synthetic division. Collecting the coefficients of x^i , we have

$$\frac{\sum_{i=0}^n \underline{a}_i(y) x^i}{x-y} = \sum_{i=0}^{n-1} \left(\sum_{k=i+1}^n a_k(y) y^{k-i-1} \right) x^i,$$

if $\sum_{i=0}^n \underline{a}_i(y) x^i$ is divisible by $(x-y)$. Though synthetic division and multiplication by truncated power series both give the same quotient, the one given by power series is more suitable for the kind of derivations we want to perform later on because it expands by collecting on the coefficients $a_i(y)$ while synthetic division collects by the powers x^i .

For example, the quotient $(7x^4 - 7y^3x + 9yx^2 - 9y^3)/(x-y)$ is a polynomial, since the numerator is zero when $x=y$. Therefore, by the method of truncated formal power series,

$$\begin{aligned} \frac{7x^4 - 7y^3x + 9yx^2 - 9y^3}{x-y} &= (-9y^3 - 7y^3x + 9yx^2 + 7x^4)(x^{-1} + yx^{-2} + y^2x^{-3} + \dots) \\ &= 9yx^2(x^{-1} + yx^{-2}) - 7y^3x(x^{-1}) \\ &\quad + 7x^4(x^{-1} + yx^{-2} + y^2x^{-3} + y^3x^{-4}) \\ &= 7x(x^2 + xy + y^2) + 9y(x+y). \end{aligned}$$

3 Fast Computation of the Bezout Resultant Matrix

The *Sylvester resultant* matrix $Syl(f, g)$ and the *Bezout resultant* matrix $Bez(f, g)$ for two univariate polynomials

$$f(t) = \sum_{i=0}^n a_i t^i, \quad g(t) = \sum_{i=0}^n b_i t^i$$

are well-known in elimination theory [9]. In this section, we briefly review the constructions of $Syl(f, g)$ and $Bez(f, g)$, show that $Syl(f, g)$ can be transformed to $Bez(f, g)$ by matrix multiplication, and develop an algorithm to compute $Bez(f, g)$ rapidly by exploiting this transformation.

3.1 The Sylvester and Bezout Resultants

The Sylvester resultant for $f(t)$ and $g(t)$ can be constructed using Sylvester's dialytic method. Consider the system of $2n$ polynomials $\{t^\tau f, t^\tau g \mid \tau = 0, \dots, n-1\}$. In matrix form this system can be written as

$$\left[\begin{array}{cccc} (f \ g) & \cdots & t^{n-1}(f \ g) \end{array} \right] = \left[\begin{array}{cccc} 1 & \cdots & t^{2n-1} \end{array} \right] Syl(f, g). \quad (2)$$

The coefficient matrix $Syl(f, g)$ is a square matrix of order $2n$, and the Sylvester resultant for f and g is the determinant $|Syl(f, g)|$.

To write the entries of $Syl(f, g)$ explicitly, let $L_i = [a_i \ b_i]$. Then by construction,

$$Syl(f, g) = \begin{bmatrix} a_0 & b_0 & & & & \\ \vdots & \vdots & \ddots & & & \\ a_{n-1} & b_{n-1} & & a_0 & b_0 & \\ a_n & b_n & & a_1 & b_1 & \\ & & \ddots & \vdots & \vdots & \\ & & & a_n & b_n & \end{bmatrix} = \begin{bmatrix} L_0 & & & & & \\ \vdots & \ddots & & & & \\ L_{n-1} & & \ddots & & L_0 & \\ L_n & & & \ddots & & L_1 \\ & & & & \ddots & \vdots \\ & & & & & L_n \end{bmatrix}. \quad (3)$$

One way to construct the Bezout resultant is to compute the coefficient matrix of the polynomials produced by the Cayley expression

$$\Delta(t, \beta) = \frac{\begin{vmatrix} f(t) & g(t) \\ f(\beta) & g(\beta) \end{vmatrix}}{\beta - t} = \frac{\sum_{i=0}^n (f(t)b_i - g(t)a_i)\beta^i}{\beta - t} \quad (4)$$

for two univariate polynomials. Since the numerator vanishes when $\beta = t$, the quotient $\Delta(t, \beta)$ is actually a polynomial

$$\Delta(t, \beta) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} B_{u,v} t^u \beta^v$$

of degree $n - 1$ in t and degree $n - 1$ in β . In matrix notation

$$\Delta(t, \beta) = \begin{bmatrix} 1 & \cdots & t^{n-1} \end{bmatrix} Bez(f, g) \begin{bmatrix} 1 \\ \vdots \\ \beta^{n-1} \end{bmatrix}. \quad (5)$$

The coefficient matrix $Bez(f, g)$ is a square matrix of order n . The Bezout resultant is the determinant $|Bez(f, g)|$.

While by (3) the non-zero entries of $Syl(f, g)$ are just the coefficients of f and g , the entries $\{B_{u,v}\}$ of $Bez(f, g)$ are more complicated polynomial expressions in these coefficients. In the next two sections we shall devise an efficient technique for computing the entries of $Bez(f, g)$.

3.2 Transforming $Syl(f, g)$ into $Bez(f, g)$

The Cayley expression (4) can be written in polynomial form by the technique of truncated formal power series:

$$\Delta(t, \beta) = \sum_{i=1}^n \sum_{v=0}^{i-1} (f(t)b_i - g(t)a_i) t^{i-1-v} \beta^v \quad (6)$$

Since the coefficient of $t^u\beta^v$ is $fb_{u+v+1} - ga_{u+v+1}$ if $u + v \leq n - 1$, and zero otherwise,

$$\begin{aligned}\Delta(t, \beta) &= \begin{bmatrix} (f \ g) & \cdots & t^{n-1}(f \ g) \end{bmatrix} \begin{bmatrix} R_1 & \cdots & R_n \\ \vdots & \ddots & \\ R_n & & \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ \beta^{n-1} \end{bmatrix} \\ &= [1 \ \cdots \ t^{2n-1}] \cdot Syl(f, g) \cdot \begin{bmatrix} R_1 & \cdots & R_n \\ \vdots & \ddots & \\ R_n & & \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \vdots \\ \beta^{n-1} \end{bmatrix}, \end{aligned} \quad (7)$$

where $R_i = [b_i \ -a_i]^T$. Rewrite Equation (5) as

$$\Delta(t, \beta) = [1 \ \cdots \ t^{n-1} \ t^n \ \cdots \ t^{2n-1}] \cdot \begin{bmatrix} Bez(f, g) \\ 0_{n \times n} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \vdots \\ \beta^{n-1} \end{bmatrix}. \quad (8)$$

Then by Equations (7) and (8), we conclude that

$$\begin{bmatrix} Bez(f, g) \\ 0_{n \times n} \end{bmatrix} = Syl(f, g) \cdot \begin{bmatrix} R_1 & \cdots & R_n \\ \vdots & \ddots & \\ R_n & & \end{bmatrix}. \quad (9)$$

As an aside, notice that this equation implies that the polynomials represented by the columns of $Bez(f, g)$ lie in the ideal generated by f, g . Now considering the first n rows on both sides of Equation (9) and denoting the entries of $Bez(f, g)$ by $\{B_{i,j}\}$, we have

$$\begin{bmatrix} B_{0,0} & \cdots & B_{0,n-1} \\ \vdots & \vdots & \vdots \\ B_{n-1,0} & \cdots & B_{n-1,n-1} \end{bmatrix} = \begin{bmatrix} L_0 & & \\ \vdots & \ddots & \\ L_{n-1} & \cdots & L_0 \end{bmatrix} \begin{bmatrix} R_1 & \cdots & R_n \\ \vdots & \ddots & \\ R_n & & \end{bmatrix}. \quad (10)$$

3.3 Fast Computation of the Entries of $Bez(f, g)$

Equation (10) gives

$$B_{i,j} = \sum_{k=0}^{\min(i, n-1-j)} L_{i-k} R_{j+1+k}, \quad 0 \leq i, j \leq n-1. \quad (11)$$

Since $L_{i-k} R_{j+1+k} + L_{i-l} R_{j+1+l} = 0$ if and only if $k + l = i - j - 1$, the sum from $k = 0$ to $k = \min(0, i - j - 1)$ in Equation (11) is zero. Thus Equation (11) can be improved to

$$B_{i,j} = \sum_{k=\max(0, i-j)}^{\min(i, n-1-j)} L_{i-k} R_{j+1+k}, \quad 0 \leq i, j \leq n-1. \quad (12)$$

Equation (12) also appears in [4]. The symmetry of $Bez(f, g)$ can easily be established from Equation (12).

Along the diagonals $i + j = k$, $i \leq j$, $0 \leq k \leq 2n - 2$, Equation (12) can be written recursively as:

$$B_{i,j} = B_{i-1,j+1} + L_i R_{j+1}. \quad (13)$$

Exploiting this recurrence, we can construct $Bez(f, g)$ in three stages:

1. Initialization

Set

$$Bez(f, g)_{init} = \begin{bmatrix} L_0 R_1 & \cdots & L_0 R_n \\ & \ddots & \vdots \\ & & L_{n-1} R_n \end{bmatrix},$$

that is,

$$(B_{i,j})_{init} = L_i R_{j+1} = a_i b_{j+1} - b_i a_{j+1}, \quad 0 \leq i \leq j \leq n - 1.$$

2. Recursion

Add along the diagonals $i + j = k$ from top to bottom, left to right:

for i from 1 to $n - 2$

for j from i to $n - 2$

$$B_{i,j} \leftarrow B_{i,j} + B_{i-1,j+1}$$

$$Bez(f, g)_{up_diagonal} = \begin{bmatrix} * & * & * & \cdots & * & * & * \\ & \swarrow & \swarrow & \cdots & \swarrow & \swarrow & * \\ & & \swarrow & \cdots & \swarrow & \swarrow & * \\ & & & \ddots & \vdots & \vdots & \vdots \\ & & & & \swarrow & \swarrow & * \\ & & & & & \swarrow & * \\ & & & & & & * \end{bmatrix}_{n \times n}. \quad (14)$$

3. Completion

Since $Bez(f, g)$ is symmetric, set

$$B_{i,j} = B_{j,i}, \quad i > j.$$

For example, let $n = 4$ and abbreviate $L_i R_j$ by $i.j$. The 4 matrices after the initialization and the end of each outer loop are

$$\begin{array}{l} \text{init.} \\ \longrightarrow \end{array} \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ & 1.2 & 1.3 & 1.4 \\ & & 2.3 & 2.4 \\ & & & 3.4 \end{bmatrix} \begin{array}{l} \text{recursion} \\ \longrightarrow \end{array} \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ & 1.2 + 0.3 & 1.3 + 0.4 & 1.4 \\ & & 2.3 & 2.4 \\ & & & 3.4 \end{bmatrix} \begin{array}{l} \text{recursion} \\ \longrightarrow \end{array} \\ \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ & 1.2 + 0.3 & 1.3 + 0.4 & 1.4 \\ & & 2.3 + 1.4 & 2.4 \\ & & & 3.4 \end{bmatrix} \begin{array}{l} \text{symmetry} \\ \longrightarrow \end{array} \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 1.2 + 0.3 & 1.3 + 0.4 & 1.4 \\ 0.3 & 1.3 + 0.4 & 2.3 + 1.4 & 2.4 \\ 0.4 & 1.4 & 2.4 & 3.4 \end{bmatrix}.$$

3.4 Computational Complexity

Since $Bez(f, g)$ is symmetric, we need to compute only the entries $B_{i,j}$ where $i \leq j$. Using the algorithm developed in Section 3.3, we initialize and march southwest updating these $(n^2 + n)/2$ entries as we go:

$$\begin{array}{ccc} \left[\begin{array}{ccc} L_0 R_1 & \cdots & L_0 R_n \\ & \ddots & \vdots \\ & & L_{n-1} R_n \end{array} \right] & & \left[\begin{array}{ccccc} * & * & \cdots & * & * \\ & \swarrow & \cdots & \swarrow & * \\ & & \ddots & \vdots & \vdots \\ & & & \swarrow & * \\ & & & & * \end{array} \right]. \\ \text{Initialization} & & \text{Marching} \end{array}$$

During initialization, each of these entries requires two multiplications and one addition. Thus there are $(n^2 + n)$ multiplications and $(n^2 + n)/2$ additions in this initialization. As we march southwest, each entry above the diagonal $i = j$ — except for the entries in the first row or the last column — needs one addition. Thus there are $(n^2 - n)/2$ more additions. Therefore, to compute $Bez(f, g)$ by the new algorithm, we need to perform $(n^2 + n)$ multiplications and n^2 additions.

On the other hand, to compute $Bez(f, g)$ in the standard way, i.e. computing each entry separately by the entry formula (free of cancellation)

$$B_{i,j} = \sum_{k=\max(0, i-j)}^{\min(i, n-1-j)} (a_{i-k} b_{j+1+k} - b_{i-k} a_{j+1+k}),$$

we must compute the entries $\{B_{i,j}\}$ for all $i \leq j$:

$$B_{i,j} = \begin{cases} \sum_{k=0}^i (a_{i-k} b_{j+1+k} - b_{i-k} a_{j+1+k}), & i + j \leq n - 1; \\ \sum_{k=0}^{n-1-j} (a_{i-k} b_{j+1+k} - b_{i-k} a_{j+1+k}), & i + j > n - 1. \end{cases}$$

Each summand requires two multiplications and one addition. When n is odd, the total number of multiplications to compute all the $B_{i,j}$, $i \leq j$, is

$$\begin{aligned} & 2 \cdot \sum_{i=0}^{(n-1)/2} (n - 2i) \cdot (i + 1) + 2 \cdot \sum_{j=(n+1)/2}^{n-1} (n - j) \cdot (2j - n + 1) \\ &= \frac{2n^3 + 9n^2 + 10n + 3}{12}, \end{aligned}$$

and the total number of additions is

$$\begin{aligned} & \sum_{i=0}^{(n-1)/2} (n - 2i) \cdot (2i + 1) + \sum_{j=(n+1)/2}^{n-1} (2n - 2j - 1) \cdot (2j - n + 1) \\ &= \frac{2n^3 + 3n^2 + 4n + 3}{12}. \end{aligned}$$

Similar results hold when n is even. Therefore, the standard method needs $\mathcal{O}(n^3)$ multiplications and additions to compute all the entries of $Bez(f, g)$, while our new technique requires only $\mathcal{O}(n^2)$ additions and multiplications:

Table 1: Computing the Bezout resultant matrix

	Standard method	New algorithm
# of mul.	$\mathcal{O}(n^3)$	$n^2 + n$
# of add.	$\mathcal{O}(n^3)$	n^2

4 Fast Computation of the Dixon Resultant Matrix

Consider three bivariate polynomials of bidegree (m, n) :

$$f(s, t) = \sum_{i=0}^m \sum_{j=0}^n a_{i,j} s^i t^j, \quad g(s, t) = \sum_{i=0}^m \sum_{j=0}^n b_{i,j} s^i t^j, \quad h(s, t) = \sum_{i=0}^m \sum_{j=0}^n c_{i,j} s^i t^j.$$

[3] outlines several methods for constructing a resultant for f, g, h , and [11] discusses various additional representations for such a resultant. The two most commonly used representations are the *Sylvester resultant* and the *Cayley resultant*. The Sylvester resultant matrix $Syl(f, g, h)$ can be constructed by the dialytic method, and the Cayley resultant matrix $Cay(f, g, h)$ can be generated from an extension of the Cayley expression for the Bezout resultant.

The two resultant matrices $Syl(f, g, h)$ and $Cay(f, g, h)$ are generalizations of the Sylvester resultant matrix and the Bezout resultant matrix for two univariate polynomials. The non-zero entries of $Syl(f, g, h)$ are very simple. In fact, as in the univariate setting, these entries are just the coefficients of $f(s, t), g(s, t), h(s, t)$. However, $Syl(f, g, h)$ is huge — $6mn \times 6mn$. To compute the resultant of f, g, h , people generally use $Cay(f, g, h)$, which is $2mn \times 2mn$, one ninth the size of $Syl(f, g, h)$. Thus $Cay(f, g, h)$ is often called the *Dixon resultant*. Nonetheless, the entries of $Cay(f, g, h)$ are very complicated expressions in the coefficients of f, g, h [1, 10].

In this section, we first review the construction of $Syl(f, g, h)$ and $Cay(f, g, h)$. Then we introduce natural block structures on these two resultant matrices. We shall use these block structures together with the method of truncated formal power series to derive a transformation matrix from $Syl(f, g, h)$ to $Cay(f, g, h)$. Finally combining the block structures and the transformation matrix, we present an efficient algorithm to compute the entries of $Cay(f, g, h)$.

4.1 The Sylvester and Cayley Resultants

The Sylvester resultant for $f(s, t), g(s, t), h(s, t)$ is constructed using Sylvester's dialytic method. Consider the $6mn$ polynomials $\{s^\sigma t^\tau f, s^\sigma t^\tau g, s^\sigma t^\tau h \mid \sigma = 0, \dots, 2m - 1; \tau =$

$0, \dots, n-1\}$. This system of polynomials can be written in matrix notation as

$$\begin{aligned} & [f \ g \ h \ \dots \ t^{n-1}(f \ g \ h) \ \dots \ s^{2m-1}(f \ g \ h) \ \dots \ s^{2m-1}t^{n-1}(f \ g \ h)] \\ &= [1 \ \dots \ t^{2n-1} \ \dots \ s^{3m-1} \ \dots \ s^{3m-1}t^{2n-1}] \cdot \text{Syl}(f, g, h), \end{aligned} \quad (15)$$

where the rows of $\text{Syl}(f, g, h)$ are indexed lexicographically with $s > t$. That is, the monomials are ordered as $1, \dots, t^{2n-1}, \dots, s^\sigma, \dots, s^\sigma t^{2n-1}, \dots, s^{3m-1}, \dots, s^{3m-1}t^{2n-1}$. The coefficient matrix $\text{Syl}(f, g, h)$ is a square matrix of order $6mn$, and the Sylvester resultant for f, g, h is simply $|\text{Syl}(f, g, h)|$.

The Cayley resultant for f, g, h can be derived from the Cayley expression

$$\Delta(s, t, \alpha, \beta) = \frac{\begin{vmatrix} f(s, t) & g(s, t) & h(s, t) \\ f(\alpha, t) & g(\alpha, t) & h(\alpha, t) \\ f(\alpha, \beta) & g(\alpha, \beta) & h(\alpha, \beta) \end{vmatrix}}{(\alpha - s)(\beta - t)}. \quad (16)$$

Since the numerator vanishes when $\alpha = s$ or $\beta = t$, the numerator is divisible by $(\alpha - s)(\beta - t)$. Hence $\Delta(s, t, \alpha, \beta)$ is a polynomial in s, t, α, β , so

$$\Delta(s, t, \alpha, \beta) = \sum_{u=0}^{2m-1} \sum_{v=0}^{n-1} \sum_{\sigma=0}^{m-1} \sum_{\tau=0}^{2n-1} d_{\sigma, \tau, u, v} s^\sigma t^\tau \alpha^u \beta^v,$$

for some constant coefficients $d_{\sigma, \tau, u, v}$. In matrix form,

$$\Delta(s, t, \alpha, \beta) = \begin{bmatrix} 1 \\ \vdots \\ t^{2n-1} \\ \vdots \\ s^{m-1} \\ \vdots \\ s^{m-1}t^{2n-1} \end{bmatrix}^T \cdot \text{Cay}(f, g, h) \cdot \begin{bmatrix} 1 \\ \vdots \\ \beta^{n-1} \\ \vdots \\ \alpha^{2m-1} \\ \vdots \\ \alpha^{2m-1}\beta^{n-1} \end{bmatrix}. \quad (17)$$

The rows and columns of $\text{Cay}(f, g, h)$ are indexed lexicographically by $s^\sigma t^\tau, \alpha^u \beta^v$ with $s > t, \alpha > \beta$ respectively. The coefficient matrix $\text{Cay}(f, g, h)$ is again a square matrix but of order $2mn$, and the Cayley resultant for f, g, h is $|\text{Cay}(f, g, h)|$.

4.2 The Block Structures of $\text{Syl}(f, g, h)$ and $\text{Cay}(f, g, h)$

We can impose a natural block structure on the entries of $\text{Syl}(f, g, h)$. Let

$$f_i(t) = \sum_{j=0}^n a_{i,j} t^j, \quad g_i(t) = \sum_{j=0}^n b_{i,j} t^j, \quad h_i(t) = \sum_{j=0}^n c_{i,j} t^j,$$

and let S_i be the $2n \times 3n$ coefficient matrix for the polynomials $(t^v f_i, t^v g_i, t^v h_i)$, $v = 0, \dots, n-1$:

$$S_i = \begin{bmatrix} a_{i,0} & b_{i,0} & c_{i,0} & & & & \\ \vdots & \vdots & \vdots & \ddots & & & \\ a_{i,n-1} & b_{i,n-1} & c_{i,n-1} & \ddots & a_{i,0} & b_{i,0} & c_{i,0} \\ a_{i,n} & b_{i,n} & c_{i,n} & \ddots & a_{i,1} & b_{i,1} & c_{i,1} \\ & & & \ddots & \vdots & \vdots & \vdots \\ & & & & a_{i,n} & b_{i,n} & c_{i,n} \end{bmatrix}. \quad (18)$$

Here the rows are indexed by the monomials $1, \dots, t^{2n-1}$, and the columns are indexed by the polynomials $t^0(f_i \ g_i \ h_i), \dots, t^{n-1}(f_i \ g_i \ h_i)$.

Note that the matrix S_i is Sylvester-like in the sense that if we drop the f_i -columns from S_i , then we get the univariate Sylvester matrix of g_i and h_i ; dropping the g_i -columns yields the univariate Sylvester matrix of h_i and f_i ; dropping the h_i -columns yields the univariate Sylvester matrix of f_i and g_i . It follows from Equations (15) and (18) that

$$Syl(f, g, h) = \begin{bmatrix} S_0 & & & & & & \\ \vdots & \ddots & & & & & \\ S_{m-1} & \ddots & S_0 & & & & \\ S_m & \ddots & S_1 & S_0 & & & \\ & \ddots & \vdots & \vdots & \ddots & & \\ & & S_m & S_{m-1} & \ddots & S_0 & \\ & & & S_m & \ddots & S_1 & \\ & & & & \ddots & \vdots & \\ & & & & & & S_m \end{bmatrix}. \quad (19)$$

As for the block structure of $Cay(f, g, h)$, we simply write

$$Cay(f, g, h) = \begin{bmatrix} C_{0,0} & \cdots & C_{0,2m-1} \\ \vdots & \vdots & \vdots \\ C_{m-1,0} & \cdots & C_{m-1,2m-1} \end{bmatrix}, \quad (20)$$

where each block $C_{i,j}$ is of size $2n \times n$. The reason why we impose this particular block structure on the entries of $Cay(f, g, h)$ will become clear shortly.

4.3 Transforming $Syl(f, g, h)$ into $Cay(f, g, h)$

Since $\Delta(s, t, \alpha, \beta)$ is a polynomial in s, α , by the technique of truncated formal power series, Equation (16) can be written as

$$\begin{aligned} & \Delta(s, t, \alpha, \beta) \\ &= \left[f(s, t) \left(\sum_{i=0}^m g_i(t) \alpha^i \cdot \sum_{j=0}^m h_j(\beta) \alpha^j - \sum_{i=0}^m g_i(\beta) \alpha^i \cdot \sum_{j=0}^m h_j(t) \alpha^j \right) \cdot \frac{1}{\beta - t} \right] \end{aligned}$$

$$\begin{aligned}
& +g(s, t) \left(\sum_{i=0}^m h_i(t)\alpha^i \cdot \sum_{j=0}^m f_j(\beta)\alpha^j - \sum_{i=0}^m h_i(\beta)\alpha^i \cdot \sum_{j=0}^m f_j(t)\alpha^j \right) \cdot \frac{1}{\beta - t} \\
& +h(s, t) \left(\sum_{i=0}^m f_i(t)\alpha^i \cdot \sum_{j=0}^m g_j(\beta)\alpha^j - \sum_{i=0}^m f_i(\beta)\alpha^i \cdot \sum_{j=0}^m g_j(t)\alpha^j \right) \cdot \frac{1}{\beta - t} \Big] \cdot \frac{1}{\alpha - s} \\
= & \sum_{i=0}^m \sum_{j=0}^m \left(\sum_{u=0}^{i+j-1} s^{i+j-1-u} \alpha^u \cdot f(s, t) \cdot \frac{g_i(t)h_j(\beta) - g_i(\beta)h_j(t)}{\beta - t} \right. \\
& + \sum_{u=0}^{i+j-1} s^{i+j-1-u} \alpha^u \cdot g(s, t) \cdot \frac{h_i(t)f_j(\beta) - h_i(\beta)f_j(t)}{\beta - t} \\
& \left. + \sum_{u=0}^{i+j-1} s^{i+j-1-u} \alpha^u \cdot h(s, t) \cdot \frac{f_i(t)g_j(\beta) - f_i(\beta)g_j(t)}{\beta - t} \right). \tag{21}
\end{aligned}$$

The coefficients of $\alpha^u, \dots, \alpha^u \beta^{n-1}$ in $\Delta(s, t, \alpha, \beta)$ are polynomials in s and t . Denote these polynomials by $p_{u,0}, \dots, p_{u,n-1}$, $u = 0, \dots, 2m - 1$. Then

$$\Delta(s, t, \alpha, \beta) = \sum_{u=0}^{2m-1} \sum_{j=0}^{n-1} p_{u,j} \alpha^u \beta^j. \tag{22}$$

It follows from Equation (21) that the polynomials $p_{u,0}, \dots, p_{u,n-1}$ can be written as linear combinations of $s^i t^j f, s^i t^j g, s^i t^j h$, $0 \leq i \leq 2m - 1$, $0 \leq j \leq n - 1$. Therefore, there exists a $6mn \times 2mn$ matrix F such that

$$\begin{aligned}
& [(f \ g \ h) \ \dots \ s^{2m-1} t^{n-1} (f \ g \ h)]_{1 \times 6mn} \cdot F \\
= & [p_{0,0} \ \dots \ p_{0,n-1} \ \dots \ p_{2m-1,0} \ \dots \ p_{2m-1,n-1}]_{1 \times 2mn} \tag{23}
\end{aligned}$$

That is,

$$\begin{aligned}
& [1 \ \dots \ t^{2n-1} \ \dots \ s^{3m-1} \ \dots \ s^{3m-1} t^{2n-1}] \cdot Syl(f, g, h) \cdot F \\
= & [1 \ \dots \ t^{2n-1} \ \dots \ s^{m-1} \ \dots \ s^{m-1} t^{2n-1}] \cdot Cay(f, g, h). \tag{24}
\end{aligned}$$

Appending $4mn$ rows of zeros to the bottom of the right hand side of Equation (24), we have

$$\begin{aligned}
& [1 \ \dots \ t^{2n-1} \ \dots \ s^{3m-1} \ \dots \ s^{3m-1} t^{2n-1}] \cdot Syl(f, g, h) \cdot F \\
= & [1 \ \dots \ t^{2n-1} \ \dots \ s^{3m-1} \ \dots \ s^{3m-1} t^{2n-1}] \cdot \begin{bmatrix} Cay(f, g, h) \\ 0_{4mn \times 2mn} \end{bmatrix}.
\end{aligned}$$

Therefore,

$$Syl(f, g, h) \cdot F = \begin{bmatrix} Cay(f, g, h) \\ 0_{4mn \times 2mn} \end{bmatrix}. \tag{25}$$

Notice that Equation (25) implies that the bivariate polynomials $\{p_{u,j}\}$ represented by the columns of $Cay(f, g, h)$ lie in the ideal generated by f, g, h .

Next we introduce a block structure on the entries of F compatible with the block structures on $Syl(f, g, h)$ and $Cay(f, g, h)$. The matrix F is of size $6mn \times 2mn$. We write F as

$$F = \begin{bmatrix} F_{0,0} & \cdots & F_{0,2m-1} \\ \vdots & \vdots & \vdots \\ F_{2m-1,0} & \cdots & F_{2m-1,2m-1} \end{bmatrix},$$

where each block $F_{i,j}$ is of size $3n \times n$. By Equation (23), we have, in particular, that

$$[f \ g \ h \ \cdots \ s^{2m-1}t^{n-1}(f \ g \ h)]_{1 \times 6mn} \cdot \begin{bmatrix} F_{0,u} \\ \vdots \\ F_{2m-1,u} \end{bmatrix}_{6mn \times n} = [p_{u,0} \ \cdots \ p_{u,n-1}]_{1 \times n}.$$

Therefore, by Equations (21) and (22), the entries of $F_{\sigma,u}$, $0 \leq \sigma \leq 2m-1$, are generated from the coefficients of

$$\begin{aligned} & \sum_{i+j=\sigma+u+1} \frac{g_i(t)h_j(\beta) - g_i(\beta)h_j(t)}{\beta - t}, \\ & \sum_{i+j=\sigma+u+1} \frac{h_i(t)f_j(\beta) - h_i(\beta)f_j(t)}{\beta - t}, \\ & \sum_{i+j=\sigma+u+1} \frac{f_i(t)g_j(\beta) - f_i(\beta)g_j(t)}{\beta - t}. \end{aligned} \quad (26)$$

Each term in these three sums is a Cayley expression for two univariate polynomials: $\{g_i, h_j\}$, $\{h_i, f_j\}$, and $\{f_i, g_j\}$; hence each term generates a Bezout matrix when written in matrix form [8, 4]. Therefore, each block $F_{\sigma,u}$ contains three summations of Bezout matrices, where the three summations interleave row by row. That is,

$$F_{\sigma,u} = \text{interleave row by row the three matrices} \\ \sum_{i+j=\sigma+u+1} Bez(g_i, h_j), \quad \sum_{i+j=\sigma+u+1} Bez(h_i, f_j), \quad \sum_{i+j=\sigma+u+1} Bez(f_i, g_j). \quad (27)$$

In particular, if $\sigma + u = 2m - 1$, then $i = j = m$ in expression (26), so $F_{\sigma,u}$ is the matrix obtained by interleaving the rows of the three Bezout matrices: $Bez(g_m, h_m)$, $Bez(h_m, f_m)$, $Bez(f_m, g_m)$.

By expression (26), the blocks $F_{\sigma,u}$ have the two following properties

- $F_{\sigma,u} = F_{\sigma',u'}$, if $\sigma + u = \sigma' + u'$;
- $F_{\sigma,u} = 0_{3n \times n}$, if $\sigma + u > 2m - 1$.

Therefore,

$$F_{\sigma,u} = \begin{cases} F_{0,\sigma+u}, & \sigma + u \leq 2m - 1; \\ 0, & \text{otherwise.} \end{cases}$$

Notice the similarity of Equation (32) for $C_{i,j}$ to Equation (12) for $B_{i,j}$. Again, we observe that

$$C_{i,j} = C_{i-1,j+1} + S_i \cdot F_j, \quad 1 \leq i \leq m-1, \quad 0 \leq j \leq 2m-2,$$

which leads to the following fast algorithm for computing the entries of $Cay(f, g, h)$:

1. Initialization:

$$Cay(f, g, h)_{init} = \begin{bmatrix} S_0 \cdot F_0 & \cdots & S_0 \cdot F_{2m-1} \\ \vdots & \vdots & \vdots \\ S_{m-1} \cdot F_0 & \cdots & S_{m-1} \cdot F_{2m-1} \end{bmatrix}.$$

That is,

$$(C_{i,j})_{init} = S_i \cdot F_j, \quad 0 \leq i \leq m-1, \quad 0 \leq j \leq 2m-1.$$

2. Recursion:

for i from 1 to $m-1$
for j from 0 to $2m-2$
 $C_{i,j} \leftarrow C_{i,j} + C_{i-1,j+1}$

$$Cay(f, g, h) = \begin{bmatrix} S_0 \cdot F_0 & \cdots & S_0 \cdot F_{m-1} & S_0 \cdot F_m & \cdots & S_0 \cdot F_{2m-1} \\ \swarrow & \ddots & \swarrow & \swarrow & \ddots & S_1 \cdot F_{2m-1} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \swarrow & \ddots & \swarrow & \swarrow & \ddots & S_{m-2} \cdot F_{2m-1} \\ \swarrow & \ddots & \swarrow & \swarrow & \ddots & S_{m-1} \cdot F_{2m-1} \end{bmatrix}.$$

Notice that this method for computing the entries of $Cay(f, g, h)$ eliminates lots of redundant calculations: after the initialization, we need only update the blocks along the southwest diagonals by one (block) addition per block. This approach is much faster than the standard method of calculating each entry independently [1, 10], especially when we compute the Bezoutians F_k efficiently [see Section 3.4]. In the next section, we will present a more detailed analysis of the computational complexity of this algorithm.

4.5 Computational Complexity

To compute the entries of $Cay(f, g, h)$ by the method developed in Section 4.4, we need the following three steps:

1. Preprocessing: compute F_u , $0 \leq u \leq 2m-1$.
2. Initialization: calculate $(C_{i,u})_{init} = S_i \cdot F_u$, $0 \leq i \leq m-1$, $0 \leq u \leq 2m-1$.
3. Recursion: march along the southwest diagonals, and update the entries of $C_{i,u}$.

Step 1: Each F_u contains the following three summations of Bezout matrices:

$$\sum_{i+j=u+1} \text{Bez}(g_i, h_j), \quad \sum_{i+j=u+1} \text{Bez}(h_i, f_j), \quad \sum_{i+j=u+1} \text{Bez}(f_i, g_j).$$

We can use the technique developed in Section 3 to efficiently compute the sum

$$\sum_{i+j=u+1} \text{Bez}(g_i, h_j).$$

First we initialize each Bezout matrix $\text{Bez}(g_i, h_j)_{init}$; then we add these initializations together; and last we march along the southwest diagonals updating the entries as we go. Altogether, we need

$$\underbrace{(n^2 + n) \cdot \min(u + 2, m + 1)}_{\text{initialization}} \quad \text{multiplications,}$$

and

$$\underbrace{\frac{n^2 + n}{2} \cdot \min(u + 2, m + 1)}_{\text{initialization}} + \underbrace{\frac{n^2 + n}{2} \cdot \min(u + 1, m)}_{\text{summation}} + \underbrace{\frac{n^2 - n}{2}}_{\text{marching}} \quad \text{additions.}$$

Similarly, we can compute $\sum_{i+j=u+1} \text{Bez}(h_i, f_j)$ and $\sum_{i+j=u+1} \text{Bez}(f_i, g_j)$ efficiently. Therefore, in step 1,

$$\begin{aligned} \# \text{ of multiplications} &= \sum_{u=0}^{2m-1} 3 \cdot (n^2 + n) \cdot \min(u + 2, m + 1) \\ &= 3(n^2 + n) \left[\sum_{u=0}^{m-1} (u + 2) + \sum_{u=m}^{2m-1} (m + 1) \right] \\ &= 3(n^2 + n) \frac{3m^2 + 5m}{2} = \frac{3}{2}(n^2 + n)(3m^2 + 5m), \end{aligned}$$

and

$$\begin{aligned} \# \text{ of additions} &= \sum_{u=0}^{2m-1} 3 \cdot \left[\frac{n^2 + n}{2} \min(u + 2, m + 1) + \frac{n^2 + n}{2} \min(u + 1, m) + \frac{n^2 - n}{2} \right] \\ &= 3 \cdot \frac{n^2 + n}{2} \cdot \frac{3m^2 + 5m}{2} + 3 \cdot \frac{n^2 + n}{2} \cdot \frac{3m^2 + m}{2} + 3m(n^2 - n) \\ &= \frac{9}{2}(n^2 + n)(m^2 + m) + 3m(n^2 - n). \end{aligned}$$

Step 2: By Equations (18) and (27),

$$S_i \cdot F_u = \begin{bmatrix} a_{i,0} & & & & \\ & \ddots & & & \\ a_{i,n-1} & & \ddots & & a_{i,0} \\ & & & \ddots & a_{i,1} \\ a_{i,n} & & & & \vdots \\ & & & & & \ddots \\ & & & & & & a_{i,n} \end{bmatrix} \cdot \sum_{i+j=u+1} \text{Bez}(g_i, h_j) +$$

$$\begin{bmatrix} b_{i,0} & & & & \\ \vdots & \ddots & & & \\ b_{i,n-1} & & \ddots & & b_{i,0} \\ b_{i,n} & & & \ddots & b_{i,1} \\ & & & & \vdots \\ & & & & b_{i,n} \end{bmatrix} \cdot \sum_{i+j=u+1} Bez(h_i, f_j) + \\
\begin{bmatrix} c_{i,0} & & & & \\ \vdots & \ddots & & & \\ c_{i,n-1} & & \ddots & & c_{i,0} \\ c_{i,n} & & & \ddots & c_{i,1} \\ & & & & \vdots \\ & & & & c_{i,n} \end{bmatrix} \cdot \sum_{i+j=u+1} Bez(f_i, g_j).$$

Each one of these three matrix multiplications requires

$$2(n + 2n + \cdots + n^2) = n^3 + n^2 \quad \text{multiplications,}$$

and

$$2[n \cdot 0 + n \cdot 1 + \cdots + n(n-1)] = n^3 - n^2 \quad \text{additions.}$$

Adding these three $2n \times n$ matrices together requires $4n^2$ more additions. Therefore, since there are $2m^2$ products $S_i \cdot F_u$, step 2 requires $6m^2 \cdot n^2(n+1)$ multiplications and $2m^2 \cdot (3n^3 + n^2)$ additions.

Step 3: Since each Cayley block $C_{i,j}$ is of size $2n \times n$, marching along the southwest diagonals requires only $(m-1)(2m-1) \cdot 2n^2 = 2n^2(2m^2 - 3m + 1)$ additions and no multiplication.

Altogether, we perform

$$\frac{3}{2}(3m^2 + 5m)(n^2 + n) + 6m^2(n^3 + n^2) \quad \text{multiplications,}$$

and

$$\frac{9}{2}(m^2 + m)(n^2 + n) + 3m(n^2 - n) + 2m^2(3n^3 + n^2) + 2n^2(2m^2 - 3m + 1) \quad \text{additions.}$$

Notice that the main bottleneck is step 2, which requires $\mathcal{O}(m^2n^3)$ multiplications and additions.

On the other hand, the standard way [1, 10] to compute the entries of $Cay(f, g, h)$ needs to compute

$$\frac{1}{6}m(m+1)^2(m+2) \cdot n(n+1)^2(n+2)$$

3×3 determinants, each of which has 6 terms and hence requires 12 multiplications and 5 additions. So if we compute all these 3×3 intermediate determinants just once and store them, we need to perform at least

$$2m(m+1)^2(m+2) \cdot n(n+1)^2(n+2) \quad \text{multiplications}$$

and

$$\frac{5}{6}m(m+1)^2(m+2) \cdot n(n+1)^2(n+2) \quad \text{additions.}$$

A summary of the complexity of computing the entries of the Cayley resultant matrices using the standard method vs. the new fast algorithm is given in Table 2:

Table 2: Computing the Cayley resultant matrix

	Standard method	New algorithm
# of mul.	$\mathcal{O}(m^4n^4)$	$\mathcal{O}(m^2n^3)$
# of add.	$\mathcal{O}(m^4n^4)$	$\mathcal{O}(m^2n^3)$

Parallel computation can be used to speed up the new algorithm even further. For example, we can compute each of the blocks F_u , $0 \leq u \leq 2m - 1$, and perform the initialization of the blocks $(C_{i,u})_{init} = S_i \cdot F_u$ in parallel. Moreover, the recursions along different diagonals are independent, so these steps can also be done in parallel.

Finally notice that the time complexity $\mathcal{O}(m^2n^3)$ for computing the entries of the matrix $Cay(f, g, h)$ is not symmetric in m and n . The reason is that the Sylvester resultant matrix $Syl(f, g, h)$ and the Cayley resultant matrix $Cay(f, g, h)$ are not symmetric in s, t — hence not symmetric in m, n . If we reverse the roles of s, t during the construction of $Syl(f, g, h)$ and $Cay(f, g, h)$, and impose appropriate block structures on these matrices, the time complexity for computing the entries of $Cay(f, g, h)$ is $\mathcal{O}(m^3n^2)$. (The difference between the two Cayley resultant matrices $Cay(f, g, h)$ is that one is the transpose of the other.) Therefore, we can actually compute the entries of $Cay(f, g, h)$ in time $\min(\mathcal{O}(m^2n^3), \mathcal{O}(m^3n^2))$.

Acknowledgments

Eng-Wee Chionh is supported by the National University of Singapore for research at Rice University. He greatly appreciates the hospitality and facilities generously provided by Rice. Ming Zhang and Ron Goldman are partially supported by NSF grant CCR-9712345.

References

- [1] E. W. Chionh. Concise Parallel Dixon Resultant. *Computer Aided Geometric Design*, 14(6) (1997), 561–570.
- [2] D. Cox, J. Little, D. O’Shea. *Using Algebraic Geometry*, Springer-Verlag New York, Inc., 1998.
- [3] A. L. Dixon. The Eliminant of Three Quantics in Two Independent Variables. *Proc. London Math. Soc.*, 6(1908), 49–69, 473–492,

- [4] R. N. Goldman, T. Sederberg, D. Anderson. Vector Elimination: A Technique for the Implicitization, Inversion, and Intersection of Planar Parametric Rational Polynomial Curves. *Computer Aided Geometric Design*, 1(1984), 327-356.
- [5] J. T. Kajiya. Ray Tracing Parametric Patches. *Proceedings of SIGGRAPH*, 1982, pp. 245-254.
- [6] D. Manocha. *Algebraic and Numeric Techniques for Modeling and Robotics*. Ph.D. Thesis, Computer Science Division, University of California, Berkeley, 1992.
- [7] D. Manocha and J. F. Canny. Multipolynomial Resultant Algorithms. *Journal of Symbolic Computation*, 15(2) (1993), 99-122.
- [8] Y. De Montaudouin, W. Tiller. The Cayley Method in Computer Aided Geometric Design. *Computer Aided Geometric Design*, 1 (1984), 309-326.
- [9] G. Salmon. *Lessons Introductory to the Modern Higher Algebra*, G. E. Stechert & Co., New York, 1924.
- [10] T. Sederberg. *Implicit and Parametric Curves and Surfaces*. Ph.D. Thesis, Purdue University, 1983.
- [11] M. Zhang, E. W. Chionh, R. N. Goldman. Hybrid Dixon Resultants, *The Mathematics of Surfaces VIII*, Edited by R. Cripps, Information Geometers Ltd., Winchester, UK, 1998, pp. 193-212.