Solving geoinformatics parametric polynomial systems using the improved Dixon resultant

Robert H. Lewis, Béla Paláncz, Joseph Awange

Fordham University, New York, NY 10458, USA

Budapest University of Technology, Hungary

Curtin University, Perth, Australia

Abstract

Improvements in computational and observational technologies in geoinformatics, e.g., the use of laser scanners that produce huge point cloud data sets, or the proliferation of global navigation satellite systems (GNSS) and unmanned aircraft vehicles (UAVs), have brought with them the challenges of handling and processing this "*big data*". These call for improvement or development of better processing algorithms. One way to do that is integration of symbolically presolved sub-algorithms to speed up computations.

Using examples of interest from real geoinformatic problems, we will discuss the Dixon-EDF resultant as an improved resultant method for the symbolic solution of parametric polynomial systems. We will briefly describe the method itself, then discuss geoinformatics problems arising in minimum distance mapping (MDM), parameter transformations, and pose estimation essential for resection. Dixon-EDF is then compared to older notions of "Dixon resultant", and to several respected implementations of Gröbner bases algorithms on several systems. The improved algorithm, Dixon-EDF, is found to be greatly superior, usually by orders of magnitude, in both CPU usage and RAM usage. It can solve geoinformatics problems on which the other methods fail, making symbolic solution of parametric systems feasible for many problems.

Keywords: parametric polynomial system, resultant, Dixon, determinant, symbolic computing, Gröbner basis, big data.

1 Introduction

In geoinformatics, a brach of science that deals with the collection and analysis of spatial data, solving polynomial systems of equations is a recurring task. In geodesy, for example, most problems utilize polynomials as mathematical tools for finding unknown parameters such as positions of stations. The global navigation satellite system (GNSS) for instance requires solution of a system of polynomial equations to give not only the correct position of points on the surface of the Earth and in space, see, e.g., [1],[3],[5],[32], but also to transform coordinates from the GNSS systems to local systems and vice versa (e.g., [4],[7]). For instance, local coordinates may be known, but measurements are given in the GNSS system whose coordinates are in a global system (e.g., World Geodetic System 1984 - WGS 84 - for the US based global positioning system GPS). In such a case, coordinates in both local and global systems would be used to obtain the transformation parameters, which are later used to transform measured coordinates from one system to another. Equations relating coordinates in the systems and the required transformation parameters are usually of polynomial type (e.g., [7],[8]). Photogrammetry and remote sensing also encounter polynomials in solving the camera or sensor orientation problem in what is known as the *pose estimation problem*, see e.g., [21], [29], while in computer vision and medical sciences, polynomials play a key role in determining image orientations among others [22], [26], [28], [34], [38]. Polynomial equations, therefore, arise frequently not only in geoinformatics, but in all of applied mathematics (e.g., Lewis [9], [25], [27]) and are vital tools in day to day operations.

In the last decade, geoinformatics has experienced a revolution in technology. Observation tools have improved and significantly increased in use. For example, in geodesy, whereas previously the US-based GPS dominated positioning from space, the world is now inundated with other positioning systems, such as the improved Russian based GLONAS, the Chinese Beidou, and the European Galileo (see, details of these systems in [1]). Photogrammetry and remote sensing have seen the rapid emergence of unmanned aircraft vehicles (UAVs) and laser scanning, which offer improved products compared to the traditional aerial photogrammetry (e.g., [17]). Computer vision is being revolutionized by improvements in computer and information technology (IT), see e.g., [12]. These revolutions have brought with them not only the challenge of handling huge data sets, in what is currently known as *big*

data, but also that of processing the data. Processing data calls for efficient algorithms that would not only be fast (in terms of optimizing the computation time) but also be robust (capable of managing outliers in data, see e.g., [2]). Between the huge observations and the required solutions, polynomial equations often stand as the required mathematical models.

Geoinformatics approaches for dealing with polynomial equations have largely been through numerical iterations due to the difficulty in solving polynomial systems [19]. Often, the equations are linearized using Taylor series expansion and least squares [5]. Other numerical methods may also be used, e.g., [11], [36]. In the last decade, however, *closed form* or *symbolic* solutions have been proposed by [4], [33], [20] among others as alternatives to iterative numerical approaches. Symbolic methods have the advantage of not requiring initial values, which are often not known, and can lead to lack of convergence of iterative procedures in some cases, e.g., [7]. Among symbolic methods, the improved Dixon approach proposed by Lewis et al [20], [25] has strong advantages [25]. In light of the challenges brought about by the big data era, and in order to keep improving algorithms for geoinformatics tasks, this contribution extends the Dixon resultant method in several ways, including *early detection of factors*, EDF.

The advantages of the Dixon-EDF approach in comparison to other methods, e.g. [32], for solving geoinformatic problems of polynomial nature include working with reduced dimension matrices to compute the solutions, and also the fact that the solutions often appear early in the computational process without the need for the program to run to the end. For example, in the case of the determined GNSS problem, we have four unknowns and sixteen parameters. The homotopy approach of [31] requires numerical integration, which has round off errors. To avoid these round off errors, high precision integration with numerical inverse should be used. Employing parallel computations to try to circumvent the problem using homotopy results in considerably long computation time. However, using the proposed parametric solution (i,e., Dixon-EDF) avoids this problem. Dixon-EDF thus could provide faster determined solutions that could be used by the Gauss-Jacobi combinatorial approach of [6] to solve the GNSS overdetermined problem.

To understand how the method works, the concept of systems of polynomial equations and how to solve them is expounded below.

2 Symbolic versus Numeric Solution

In this work we emphasize *symbolic solution* over *numeric solution*. Basically, a numeric (or numerical) method is one that could be done with a simple hand-held calculator, using basic arithmetic, square roots, trigonometry functions, logarithms, and exponentials. Depending on the task, one may have to press the calculator buttons thousands (or even millions) of times, but theoretically a person with a calculator and some paper could implement a numerical method. When finished, the paper would be full of arithmetic.

A symbolic method involves algebra. It is a method that if a person implemented, would involve algebraic or higher rational thought. A person implementing a symbolic method will rarely need to reach for a calculator. When finished, there may be some numbers, but the paper would be full of variables like x, y, z.

Quadratic equations are a familiar topic from high school. An engineering application may need to solve the equation $f(x) = x^2 + 3x - 2 = 0$. With a hand-held calculator, one could simply do "intelligent guessing." Let's guess, say, $x = 1$. Plug it into $f(x)$, find that $f(1)$ is positive. Therefore, 1 is too big. Now try $x = 0$; that's too small. Go back and forth; stop when satisfied with the accuracy. It doesn't take long to get $x = 0.56155$, which might well be considered accurate enough. Furthermore, it is easy to write a computer program to implement this idea. That's a numeric method.

But there is another solution, which the numeric method missed, namely -3.56155. Even worse, if one were to continue this method on many problems, one would soon notice that some equations don't seem to have solutions, such as $x^2 - 2x + 4 = 0$. A great deal of effort could be expended in arithmetic until finally giving up and finding no solution.

The problem is cured by learning algebra and the symbolic method that yields the quadratic formula. Given $ax^2 + bx + c = 0$ the solution is

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Note that this formula solves *all* quadratic equations at once because it uses the symbolic names a, b, c (later termed *parameters*). It is now immediately obvious why some problems have no solution: it happens precisely when $b^2 - 4ac < 0$. *Symbolic methods lead to insight.*

In the previous example, $x^2 + 3x - 2 = 0$, we see that the two roots are exactly $(-3 \pm \sqrt{17})/2$. In this expression there is no approximation whatever. Should a decimal answer correct to, say, 16 digits be desired for the engineering problem, that would be trivially obtained on any modern computer.

There is more. Not only does the symbolic method concisely represent all solutions, it invites the question, can we define a new kind of number in which the negative under the square root may be allowed? The symbolic solution leads to a new concept, that of complex numbers!

Symbolic methods may be hard to develop, and they may be difficult for a computer to implement, but they are always desirable. Fortunately, we are not forced into a strict either/or dichotomy. There are symbolic-numeric hybrids using the strengths of both ideas.

Most researchers instinctively desire symbolic methods, postponing numeric calculations as long possible. Of course, eventually most solutions will involve numerical results, just as in our hypothetical example of the quadratic equation.

3 Polynomial Systems

"Solve a system of polynomial equations" means different things to different people. Everyone will agree that we take a collection of multivariate polynomials, set each to 0, and search for the common roots.

For us in this paper, we have a ground ring K of numerical coefficients, variables $x_1, x_2, \ldots, x_n$, and parameters $a_1, a_2, \ldots, a_m$, so mathematically we are working in a *ring* of polynomials $K[a_1, \ldots, a_m, x_1, x_2, \ldots, x_n]$ [13]. K is primarily Q, the field of rational numbers. It is often a useful tool to work over a finite field K, such as Z/p for p "large", say 40000 to 2^{31}. We are not interested in cryptography, in which $K = Z/2$. We want our "numbers" to be exactly represented in the computer; we don't want to work with "reals" or "floats" that have round-off problems (until late in the process when data is plugged in).

To emphasize, it is important to understand the difference between *variables, parameters,* and *numerical coefficients*. For example, recall that the equation of a standard parabola with vertex at the origin and focus p is $4py = x^2$. In this paper, all our equations have right hand side $= 0$, so we would write it as $4py - x^2 = 0$, and then

we often don't bother to write the 0. The two variables are x and y, the parameter is p, and the numerical coefficients are 4 and -1. The (numerical) coefficients are in the ring of integers, Z. We could have written it as $py - \frac{1}{4}x^2$. Then the coefficients are in Q. A polynomial equation with coefficients in Q can always be converted to one in Z by multiplying through by the least common multiple of the denominators, so the distinction between Z and Q is not important.

As stated above, we are not interested in purely numerical solution. We pursue symbolic solution as long as possible. We typically have n equations in n variables $x_1, x_2, \ldots, x_n$ and some parameters. Usually $3 \leq n \leq 15$. There are always parameters. Ideally, the system is neither under- nor over-determined. This is not a restriction since underdetermined systems occurring frequently in geodesy can be easily transformed into determined systems by considering them as optimization problems, or by transforming them into combinatorial sets of deterministic systems (see section 7).

In this work, "solve the system" means to eliminate all but one of the variables. We are then left with one equation in one variable and the parameters – the *resultant*. The theory of resultants goes back to Bezout around 1760. Other important names in the subject are Cayley, Sylvester, Macaulay, and Dixon [13], [37], [14]. If desired, numerical values for the parameters can then be substituted into the resultant, and the variable obtained numerically. If desired, to get numerical values for all the variables we could run this method in parallel independently on different processors and then test all combinations of values for each variable. In most cases, this last step is very fast. But often the resultant is really the desired solution already.

The Bezout-Dixon method produces a matrix whose determinant is a multiple of the resultant. Dixon-EDF [25] is a way to compute the resultant without finding the entire determinant. Often the determinant is too large to compute, but it has many factors and so the resultant is much smaller than the determinant. The other factors are called *spurious*. Often the resultant occurs with multiplicity. We detect these polynomial factors "early," hence EDF = Early Detection of Factors. The output of the algorithm is a list of polynomials whose product is the determinant. Interesting problems tend to have many factors. There is no guarantee that this will always work better than a standard determinant method. However, on many real problems from interesting applications, it does very well [26], [27], [28].

Gröbner Bases are well known [13], [37]. Briefly, just as a basis is a good, efficient set of vectors to generate a set of vectors forming a *vector space*, a Gröbner basis is a good, efficient set of polynomials to generate a set of polynomials that form an *ideal*. The concept has a wide range of many applications, only one of which is solving systems of polynomial equations.

Let us emphasize a key difference between resultant solutions and Gröbner bases solutions. The resultant of a system of n polynomials in n variables is a single polynomial in one variable (and, usually, parameters). A Gröbner basis used this way yields a number of polynomials in triangular form. That is, the first polynomial contains only one variable (and the parameters), the second has two variables, etc. It seems reasonable that the Gröbner basis is more "complete", and may well take longer to compute. However, often one or two resultants are not only sufficient, they are exactly what is desired – the other variables are mere artifacts. In any event, one may compute the resultants for different desired variables in parallel on different machines. This is a huge advantage.

Over the last fifteen years we have noticed again and again that when engineers, scientists, and most mathematicians want to solve a polynomial system, they want a symbolic solution. They try Gröbner bases, usually in either Maple, Mathematica, or Magma. Many times, the program crashes or the user gives up after many hours. Almost always these systems would be enormously easier to solve with Dixon-EDF. We do not know of any examples of the type of problem described here where Gröbner bases are better than Dixon-EDF.

There is a further advantage to Dixon-EDF. We are computing the determinant of a matrix (or factors thereof). There are many ways to do that. That is a very well-studied field. Basically Dixon-EDF is a modified and adaptive row and column reduction. But one can easily examine the state of the computation and interrupt it part way to switch to another method. As we will see below, it is often useful to switch to the Gentleman-Johnson idea of expansion by minors with storage of minors [18].

Computations in this paper were run on an Intel Imac at 2.3 ghz with 16 gigabytes of RAM, and on faster Linux servers with 130 gig. Dixon-EDF was run in Fermat [23]. Some Fermat code for Dixon-EDF is at [24]. Some of the commands used in Mathematica, Magma, and Maple are in an appendix. Maple has some built-in

triangularize and Gröbner bases routines, as well as the FGb package of Faugere [15]. FGb is usually superior and the others were not often used here. The Maple FGb commands were explained to us by J.C. Faugere [16]. The way to use Magma was specified by Magma programmer Allan Steel [35]. We also use Mathematica for some examples. Mathematica is the only well known large system that has any Dixon resultant utility. It is a package one must download and install. It implements some of the KSY idea (see next section) but not Dixon-EDF.

4 Brief Explanation of Dixon-EDF

This has been published in, e.g., [25], [27].

Given n equations in n variables $x_1, x_2, \ldots$ the Bezout-Dixon method is a two step process.

- Using $n-1$ dummy variables $b_1, b_2, \ldots$ create an $n \times n$ matrix M, set $p =$ its determinant $Det[M]$ divided by $q = (x_1 - b_1)(x_2 - b_2)\ldots(x_{n-1} - b_{n-1})$. p is called the Dixon polynomial. ($Det[M]$ is always divisible by q.)

- Extract coefficients of p relative to monomials of $x_1, x_2, \ldots, x_{n-1}$ and monomials of $b_1, b_2, \ldots$ and put them into a second matrix M_2. The basic idea is that the coefficient relative to ${x_i}^r {b_j}^s$ goes into $M_2[k, m]$ where k depends on ${x_i}^r$ and m depends on ${b_j}^s$. Each entry $M_2[k, m]$ is a polynomial in x_n and the parameters.

When $n = 2$, M_2 is square, and it is easy to predict the dimensions of M_2. The resultant is $Det[M_2]$, the determinant of M_2. When $n = 3$, Dixon [14] proved that in a certain ideal situation, M_2 is square and the resultant is $Det[M_2]$. However that ideal situation never arises in real problems. When $n > 2$ often $Det[M_2]$ is identically 0 and therefore worthless. Even worse, especially for $n > 3$, usually M_2 is not a square matrix, and then the entire concept seems to break down. There the idea languished for many years. In 1994 Kapur, Saxena and Yang [20] (KSY) revived it by showing that the resultant is a factor of ANY maximal minor of M_2. Their proof was incomplete and was finished by Buse, Elkadi and Mourrain [10].

Continuing then,

- Extract a maximal minor from M_2. This is actually quite easy to do. Reduce the computations modulo a large prime, say $p > 40000$. Set M_3 equal to M_2 with x_n and all the parameters replaced with random primes less than p.

- Row (or column) normalize M_3 keeping a record of all row and column swaps made. This is very easy and fast.

- The previous step ends with a maximal minor. Use the same recorded list of rows and columns to extract a maximal minor of M_2. We call this final matrix M_4.

- M_4 contains polynomials in the parameters and the one remaining variable. The resultant is a factor of $Det[M_4]$ – probably a small factor.

Remark One: The last few steps are a probabilistic algorithm. There is an extremely minute chance the chosen prime p or the substitution values will be unlucky in that the rank of M_3 is less than the rank of M_2. With a little bit of care, this never happens in practice.

Remark Two: This will not work if the system is under-determinded (Buse et. al.). The method will proceed and an M_4 chosen, but the final answer will probably be bogus.

Except for the probabilistic part and the minors extraction, the algorithm so far could be called Dixon-KSY, Dixon as modified by Kapur, Saxena, and Yang. Unfortunately, the method is still often infeasible because $Det[M_4]$ can be a gigantic polynomial that might not fit into the RAM of the machine, even if it could be computed.

To continue,

- We wish to avoid simply computing the determinant of M_4. Instead we begin to column normalize the matrix M_4.

- To avoid creating large messy denominators (rational functions) we pull out denominators from each row as soon as they arise. Also we factor out gcds (greatest common divisor) whenever possible from the numerators in each row and column.

- We keep track of all denominators and gcds so discovered, and we divide out common factors in the denominator list and the numerator list. In the end, the denominator list must be all 1s. The product of the numerator list is $Det[M_4]$.
- This can work efficiently because $Det[M_4]$ usually has many factors. This is a bad way to compute the determinant of a random matrix. But random matrices are seldom of interest.

There are subtleties and variations that can have significant impact on execution time: trying to select the "best" maximal minor by some heuristic, such as total number of terms; the order of the variables; the pivot strategy during the matrix normalization; running the entire algorithm first over Z/p; not running all the way to completion, as the answer may already be in the numerator list.

Here is a simple example. Given initially

$$M_0 = \begin{bmatrix} 9 & 2 \\ 4 & 4 \end{bmatrix} \quad \text{numerators:} \qquad \text{denominators:}$$

We factor a 2 out of the second column, then a 2 from the second row. Thus:

$$M_0 = \begin{bmatrix} 9 & 1 \\ 2 & 1 \end{bmatrix} \quad \text{numerators: 2, 2} \qquad \text{denominators:}$$

Note that $9 \times 4 - 2 \times 4 = 2 \times 2 \times (9 \times 1 - 2 \times 1)$. We change the second row by subtracting 2/9 of the first:

$$M_0 = \begin{bmatrix} 9 & 1 \\ 0 & 7/9 \end{bmatrix} \quad \text{numerators: 2, 2} \qquad \text{denominators:}$$

We pull out the denominator 9 from the second row, and factor out 9 from the first column:

$$M_0 = \begin{bmatrix} 1 & 1 \\ 0 & 7 \end{bmatrix} \quad \text{numerators: 2, 2, 9} \qquad \text{denominators: 9}$$

Note that $9 \times 7/9 - 1 \times 0 = (2 \times 2 \times 9)/9 \times (1 \times 7 - 1 \times 0)$.

We "clean up" by dividing out the common factor of 9 from the numerator and denominator lists; any 1 that occurs may be erased and the list compacted. Since the

first column is cannonically simple, we are finished with one step of the algorithm, and have produced a one-smaller M_1 for the next step.

$$M_1 = \begin{bmatrix} 7 \end{bmatrix} \quad \text{numerators: } 2,\ 2 \qquad \text{denominators: } 1$$

The algorithm terminates by pulling out the 7:

numerators: 2, 2, 7 denominators: 1

At the end: three numerators, one denominator (= 1). As expected (since the original matrix contained all integers) the denominator list is trivial. The product of all the entries in the numerator list is the determinant, but we never needed to deal with any number larger than 9.

In the following sections some geodetic problems are solved with the method to illustrate its power and efficiency. The ground ring $K = Q$ in all cases. Actually, all numerical coefficients are in fact integers.

5 Minimum Distance Mapping

The revolution brought about by positioning using global navigation satellite systems (GNSS) in geodesy has necessitated a fresh look at the mathematics underpinning it. This is because GNSS measures on the topographical surface, while the measurements are referred to a mathematical figure approximating the earth, the ellipsoid. The projection of points from the topographical surface to their equivalent on the reference ellipsoid thus remains a fundamental task in geodesy (see e.g. Awange and Palancz 2016 [7]). For planets similar to the Earth, Awange and Palancz (2016) point to the biaxial ellipsoid, i.e., "ellipsoid of revolution" as the best approximation. In section 5.1, we show how a topographical point can be mapped to a standard ellipsoid, while in section 5.2 we illustrate the mapping to any ellipsoid or 3D conic.

5.1 Mapping of topographical point to standard ellipsoid

Given a standard position ellipsoid $x^2/a^2 + y^2/b^2 + z^2/c^2 - 1 = 0$ and a point $U = (u, v, w)$, compute the point $X = (x, y, z)$ on the ellipsoid closest to the point U. The ellipsoid equation has three variables x, y, z and three parameters a, b, c.

We can easily derive equations using partial derivatives to find the minimum distance (alternatively, one can use Lagrange multipliers). Think of z as a function

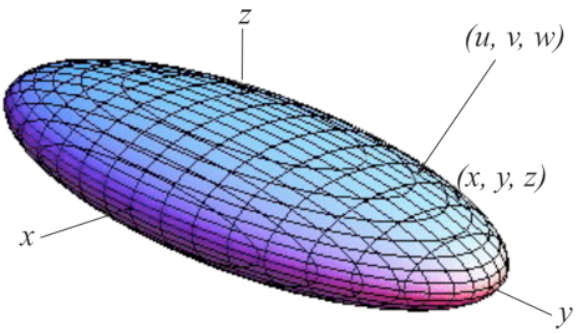

Figure 1: Given u, v, w find x, y, z for a standard ellipsoid.

of x and y. Let D be the square of the distance between the points U and X. Set the partial derivatives $\partial D/\partial x, \partial D/\partial y$ equal to 0. We must add two more variables to stand for $\partial z/\partial x, \partial z/\partial y$; we use dzx, dzy. There are six parameters a, b, c, u, v, w. The five equations (set each expression to 0) are

$x^2/a^2 + y^2/b^2 + z^2/c^2 - 1$

$x/a^2 + z/c^2\ dzx$

$y/b^2 + z/c^2\ dzy$

$(x - u) + (z - w)dzx$

$(y - v) + (z - w)dzy$

The new variables dzx, dzy are artifacts: we don't care about them. We want to know just x, y, z. One advantage of resultants is that you can't tell a Gröbner basis algorithm not to bother with some of the variables.

This is a fairly easy problem for several methods. The resultant has 66 terms and is degree 6 in x. With Dixon: 0.038 seconds, 22 meg RAM, with Magma: 1 second, 100 meg. Similar results were obtained with Maple and Mathematica. But we can say more: the coefficient of x^6 is

$$b^2c^2 - 2abc^2 + a^2c^2 - 2ab^2c + 4a^2bc - 2a^3c + a^2b^2 - 2a^3b + a^4.$$

This factors as $(a - c)^2(a - b)^2$, so we learn that if $b = a$ or $c = a$ there is a simpler solution. In fact, if $c = a$ the resultant drops to degree 4. As is often the case, a symbolic method leads to insight!

5.2 Mapping of a point to any 3D conic

Here (Figure 2) is the image for a general ellipsoid arbitrarily oriented in space, but we could have any 3D conic.

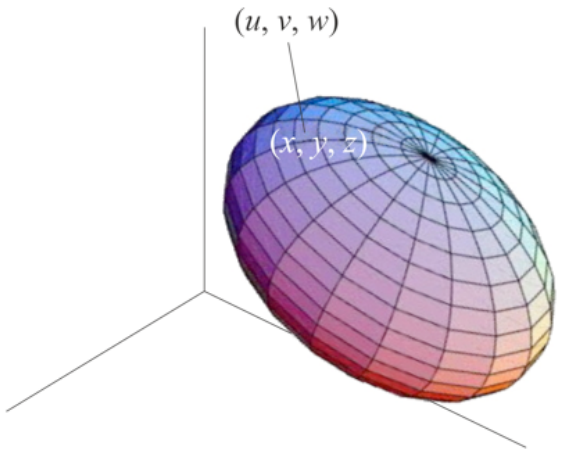

Figure 2: Given u, v, w find x, y, z for an arbitrary ellipsoid.

The equation for an arbitrary 3D conic is

$$ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j = 0.$$

Given point (u, v, w), compute point (x, y, z) with shortest distance. This is much more difficult than the standard position ellipsoid in the previous section. We have again three variables x, y, z, but now 13 parameters $a, b, c, \ldots, u, v, w$. There are two good ways to set up the equations for this problem. First, it is well known that the gradient $grad(x, y, z)$ is normal to the surface. It is obvious that the line of minimal length is normal to the surface at (x, y, z) and passes through (u, v, w). Therefore, there is a real number A such that A times $grad(x, y, z)$ is exactly the line segment connecting (x, y, z) to (u, v, w). This leads to four equations in the variables x, y, z, A.

Secondly, the minimization problem can be set up as a classical Lagrange multipliers problem. This adds one variable λ and gives four equations. It turns out this yields the same set of equations as before, so we will use it.

This problem is much harder than the previous: solving for x with Dixon-EDF takes 12 seconds, 270 meg RAM. The answer has 38984 terms, degree 6. However,

with Magma the program was killed after 24 hours using 24 gig RAM. With Maple FGb, success was obtained after 5.8 hours and 52 gig RAM. Mathematica was killed after seven hours and 1.5 gig RAM.

But we can say more. The coefficient of x^6 has two factors, one is

$$af^2 - def + be^2 + cd^2 - 4abc$$

If this were 0, the resultant simplifies.

To finish the problem, one would similarly compute the resultants for y and z. Or, by symmetry, one could note that those resultants are the same as the one for x with a simple permutation of parameters. In any event, take the three resultants one by one, substitute in numerical values for the parameters, use standard one-variable solvers, and produce up to six real number possibilities for each variable. Test up to 6^3 combinations for the minimal distance.

This is an interesting problem that is worth further discussion. The four equations are

$cz^2 + fyz + exz + iz + by^2 + dxy + hy + ax^2 + gx + j,$

$ez\lambda + dy\lambda + 2ax\lambda + g\lambda + 2x - 2u,$

$fz\lambda + 2by\lambda + dx\lambda + h\lambda + 2y - 2v,$

$2cz\lambda + fy\lambda + ex\lambda + i\lambda + 2z - 2w$

λ occurs more often than any other variable and with exponent 1 only. Based on experience with many problems, this indicates that it will be easier to solve for λ than the others. Indeed, Dixon-EDF solves for λ in 0.05 seconds and 1 meg RAM. It produces two factors, of degrees 3 and 6, and 15 and 717 terms. The degree 3 factor is spurious, but to be fair, that is not obvious at this point, so we will continue to include it. Mathematica's Gröbner basis function takes 25.5 seconds to compute the 717 term answer, and at least 10 meg RAM. The Dixon-KSY resultant package in Mathematica takes about 1.2 seconds and at least 6 meg RAM, but produces a single polynomial of 17430 terms, which is the correct answer (717 terms) times a large spurious factor.

To continue with the two factors of degree 3 and 6, by plugging in numerical values for the parameters and using a standard one-variable solver, one could obtain up to nine possible real values for λ.

Next, most unusually, note that the last three equations are linear in x, y, z.

Therefore, one could proceed by plugging in the numerical values for the parameters and the various values of λ from the previous paragraph, then solving the three variable linear system very quickly with standard software. That would yield up to nine solutions, and it would be trivial to check which is closest to (u, v, w). This is an example of a *symbolic-numeric hybrid* method.

6 Datum Transformation Problems

A transformation of coordinates motivated by geodesy and photogrammetry was discussed by Awange et. al. [4], [7]. The transformation involves seven "parameters" (variables in our terminology here) named $a, b, c, X_0, Y_0, Z_0, s_1$. Three points in one coordinate system given by $(X_i, Y_i, Z_i), i = 1, 2, 3$ are being transformed into coordinates $(a_i, b_i, c_i), i = 1, 2, 3$ in the other system by a skew-symmetric rotation matrix, using a, b, c, a scaling specified by s_1, and a translation given by X_0, Y_0, Z_0. The seven equations (set each to 0) are

$$s_1X_1 - s_1cY_1 + s_1bZ_1 + X_0 - a_1 - cb_1 + bc_1,$$
$$s_1cX_1 + s_1Y_1 - s_1aZ_1 + Y_0 + ca_1 - b_1 - ac_1,$$
$$-s_1bX_1 + s_1aY_1 + s_1Z_1 + Z_0 - ba_1 + ab_1 - c_1,$$
$$s_1X_2 - s_1cY_2 + s_1bZ_2 + X_0 - a_2 - cb_2 + bc_2,$$
$$s_1cX_2 + s_1Y_2 - s_1aZ_2 + Y_0 + ca_2 - b_2 - ac_2,$$
$$-s_1bX_2 + s_1aY_2 + s_1Z_2 + Z_0 - ba_2 + ab_2 - c_2,$$
$$-s_1bX_3 + s_1aY_3 + s_1Z_3 + Z_0 - ba_3 + ab_3 - c_3$$

All the point coordinates are parameters (in our terminology). The paper [4] discusses a clever way to substitute and reduce the system to four variables and four equations. That system was then solved for s_1 with Gröbner bases.

Suppose we do not see the clever substitution. Let's try to solve the system of seven equations. With Dixon-EDF, solving for s_1, it takes 0.016 seconds and 1.6 meg RAM. The resultant factors into two quadratic polynomials in s_1 of 24 and 18 terms.

This is a surprisingly fast solution time, and the answer is surprisingly small for a system of seven equations, even bearing in mind that all variables appear with exponent 1. Based on our experience with many problems, this is a strong hint that the equations can be algebraically simplified – which is indeed true as we mentioned above.

Trying to solve for s_1, Magma was killed after 610 minutes and 4 gig of RAM. Maple crashed after 510 minutes and 38 gig RAM.

Solving for a different variable is harder. Notice that s_1 occurs three times in each equation, while a occurs twice in only five equations. Again based on experience with many problems, this indicates that it will be harder to solve for a than s_1. To solve for a takes 2.8 seconds and 120 meg RAM. The answer is of degree 2 in a but has 24004 terms.

A better, more general transformation of coordinates $(X_i, Y_i, Z_i), i = 1, 2, 3$ to $(a_i, b_i, c_i), i = 1, 2, 3$ uses a different scaling factor in each dimension, so that there are three such factors s_1, s_2, s_3 [33]. There are then nine equations and variables. With Dixon-EDF, the matrix M_4 is 7×7. Solving for s_1 takes 20.6 seconds and 536 meg RAM. The resultant factors into one quadratic polynomial in s_1 and two linear ones, of 132, 12, and 12 terms. The linear pieces probably represent singular cases of no real significance.

This is an example where the answer of 132 terms appears "early" by step 3 (out of 7), after only 10 seconds and 232 meg RAM. It is also a case in which spending ten seconds to look for the "best" maximal minor is fruitful. The matrix M_2 is 31×8. It has many maximal minors of size 7×7, and some not only take longer to work with EDF, but the process finishes with spurious factors that play no role in the real solution.

Trying to solve for s_1, Magma crashed after 173 minutes and 104 gig of RAM. Maple was killed after 300 minutes and 26 gig RAM.

7 Helmert Transforms

The 2D Helmert transformation [7] with parameters s and Ω is

$$\begin{bmatrix} X \\ Y \end{bmatrix} = s \begin{bmatrix} \cos(\Omega) & -\sin(\Omega) \\ \sin(\Omega) & \cos(\Omega) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

or more simply substituting α and β

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

(Note that if α and β are known we can easily deduce s and Ω.)

Let us suppose that we have measurements for three corresponding data pairs $(x_i, y_i) \to (X_i, Y_i)$, $i = a, b, c$, respectively. We require the least squares estimates of the transformation parameters α and β, and simultaneously the adjustment of the coordinates x_a, x_b, x_c, X_a, X_b, and X_c. Applying the two equations of the transformation for each of the three point-pairs with the adjusted values, we get six equations (set each to 0):

$$\begin{gathered}
\alpha(x_a + dx_a) - \beta y_a - (X_a + dX_a), \\
\beta(x_a + dx_a) + \alpha y_a - Y_a, \\
\alpha(x_b + dx_b) - \beta y_b - (X_b + dX_b), \\
\beta(x_b + dx_b) + \alpha y_b - Y_b, \\
\alpha(x_c + dx_c) - \beta y_c - (X_c + dX_c), \\
\beta(x_c + dx_c) + \alpha y_c - Y_c
\end{gathered}$$

In these six equations there are eight unknowns, the adjustments dx_a, dx_b, dx_c, dX_a, dX_b, dX_c and the two parameters α, β. This underdetermined system can be transformed into a constrained minimization problem formulated with Lagrange multipliers $\lambda_1, \ldots, \lambda_6$. The goal is to minimize $dx_a^2 + dx_b^2 + dx_c^2 + dX_a^2 + dX_b^2 + dX_c^2$. The variables $\lambda_1, \ldots, \lambda_6$ are artifacts.

The condition for the existence of the optimum yields the following fourteen polynomial equations (set each to 0):

$$\begin{gathered}
2dx_a + \alpha\lambda_1 + \beta\lambda_2, \quad 2dx_b + \alpha\lambda_3 + \beta\lambda_4, \\
2dx_c + \alpha\lambda_5 + \beta\lambda_6, \quad 2dX_a - \lambda_1, \quad 2dX_b - \lambda_3, \\
2dX_c - \lambda_5, \quad (dx_a + x_a)\lambda_1 + y_a\lambda_2 + (dx_b + x_b)\lambda_3 + y_b\lambda_4 + (dx_c + x_c)\lambda_5 + y_c\lambda_6, \\
-y_a\lambda_1 + (dx_a + x_a)\lambda_2 - y_b\lambda_3 + (dx_b + x_b)\lambda_4 - y_c\lambda_5 + (dx_c + x_c)\lambda_6, \\
-dX_a - X_a + (dx_a + x_a)\alpha - y_a\beta, \quad -Y_a + y_a\alpha + (dx_a + x_a)\beta, \\
-dX_b - X_b + (dx_b + x_b)\alpha - y_b\beta, \quad -Y_b + y_b\alpha + (dx_b + x_b)\beta, \\
-dX_c - X_c + (dx_c + x_c)\alpha - y_c\beta, \quad -Y_c + y_c\alpha + (dx_c + x_c)\beta
\end{gathered}$$

Some of the equations are very small and therefore some variables could be solved in terms of others. However, Dixon-EDF has no trouble solving this system of fourteen equations symbolically. Solving for α produces a resultant of 57707 terms, takes 32 seconds and 387 megabytes; the answer for β has 34843 terms, takes 11 seconds and 226 megabytes. α and β occur in the equations more often than the other variables, so it is not surprising that solving for, say dx_a, is much harder. The resultant has 202620 terms, takes 18 minutes and 2.7 gigabytes.

Two frequently used ideas sped up the last computation. First, a minute was spent looking for better maximal minors. With different maximal minors, we have observed running times three times longer. Secondly, the EDF computation was run modulo the prime 44449, a randomly chosen "large" prime. This speeds up all numerical computations. Without it, running over the ring of integers Z, intermediate numerical coefficients routinely grow to 20 digits. The disadvantage of running modulo a prime is that the answer may not be correct over Z, which of course is what the user wants. For example, a certain correct numerical coefficient might be 44452 or 88901, which both equal 3 modulo 44449. However, in this case, and in many others, the answer modulo the prime is in fact correct. It is easy to see this because all numerical coefficients are less than 8000 in absolute value.

Solving for β was tried with Gröbner bases in Magma and Maple. Maple was killed after 18 hours and 61 gigabytes of RAM. Magma was killed after 24 hours and 14 gigabytes of RAM.

8 Pose Estimation

Pose estimation in photogrammetric or laser scanning concerns the determination of the position and orientation of the aerial camera (or laser scanner) during photography (i.e., photogrammetric resection, Awange and Palancz 2016 [7]). We illustrate in this section how Dixon-EDF can be used to rapidly solve this problem.

Suppose we have a quadrilateral $ABCE$; it does not have to be planar. The distances between each pair of vertices are known. The object moves. We observe it from point P, noting the angles spanned by each pair of vertices. The classic four point pose problem is to deduce the distances X_1, X_2, X_3, X_4.

There are four variables X_1, X_2, X_3, X_4. The parameters are AB, BC, CE, AE, AC, BE and cosines p, q, r, s, t, u (see below). An overdetermined system results from applying the law of cosines to each triangle having vertex P (see Figure 4).

Using three equations including the diagonals AC or BE gives an easy system of equations, solvable by many means. For example, one could use the equations

$X_1^2 + X_2^2 - X_1 X_2 r - |AB|^2 = 0$
$X_2^2 + X_4^2 - X_2 X_4 u - |BE|^2 = 0$
$X_1^2 + X_4^2 - X_1 X_4 s - |AE|^2 = 0$

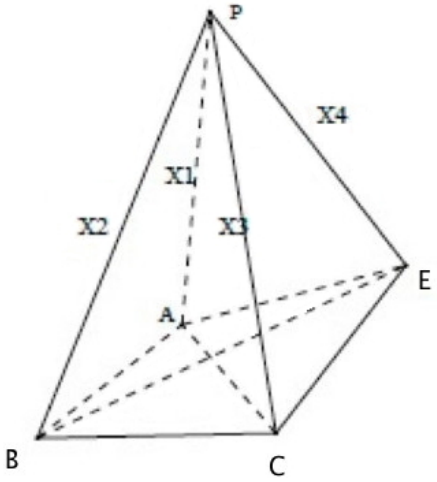

Figure 3: Viewing four points on an object

$$
\begin{aligned}
&X_1^2 + X_2^2 - X_1\, X_2\, r - |AB|^2 \\
&X_1^2 + X_3^2 - X_1\, X_3\, q - |AC|^2 \\
&X_2^2 + X_3^2 - X_2\, X_3\, p - |BC|^2 \\
&X_1^2 + X_4^2 - X_1\, X_4\, s - |AE|^2 \\
&X_4^2 + X_3^2 - X_4\, X_3\, t - |CE|^2 \\
&X_2^2 + X_4^2 - X_2\, X_4\, u - |BE|^2
\end{aligned}
$$

Figure 4: Equations from law of cosines, angles p, q, r, s, t, u

and solve for X_1, X_2 or X_4.

But suppose the object could be flexible! Then we have to use only the outside edges; diagonal distances might change. We now need four equations and four variables.

Maple and Magma both fail on this. FGb was killed after 25 hours, exhausting 62 gig of RAM. Magma crashed after 308 minutes, 8.2 gig.

Dixon-EDF finishes in 36 secs, 275 meg RAM. However, the resultant can be computed in less than one second by the variation of EDF in which we run EDF to a certain point (in this case after the third row) and then use Gentleman-Johnson [18] to compute the determinant of the remaining 9×9 matrix. This is a good example of the enormous flexibility of Dixon-EDF, in which the user is in control throughout the process. The resultant for X_1 has 24068 terms.

The analogous problem with a five-sided figure, using only the outside edges, is also solvable by Dixon by a two step process. We now have variables X_1, X_2, X_3, X_4, X_5 and five equations. Use four of them to eliminate all variables but X_1, X_2. That takes 144 seconds. Then take that resultant (47295 terms) and the remaining fifth equation and eliminate X_2. That takes 4 hours. The final answer has 37291784 terms.

9 Conclusions

This study sought to improve the traditional Dixon resultant applicable to geoinformatics by introducing the new Dixon-EDF approach. Compared to earlier versions of "Dixon resultant," Dixon-EDF has two key advantages:

- It does not matter that the Dixon matrix, here called M_2, could be non-square or have determinant identically 0. We easily extract a maximal minor, M_4.
- It does not matter that $Det[M_4]$ could be an enormous polynomial of billions of terms. We usually do not compute the entire determinant, we compute a list of factors.
- Furthermore, the resultant may appear "early" in the list of factors, and the process can be stopped.

We found great success in applying Dixon-EDF to polynomial systems arising in important applications. Dixon-EDF succeeds on many other systems [9], [26], [27], [30], [31]. Maple failed repeatedly with several implementations of Gröbner bases, as does Magma. Some of the systems above were also tried in Mathematica and failed. However, experienced users can sometimes find solutions with some systems with Gröbner bases, choosing properly the elimination order of variables.

In summary,

- Dixon-EDF is a powerful tool for symbolic solution of systems of multivariate equations.
- Dixon-EDF succeeds where other methods fail. It is usually orders of magnitude more effective, at least on systems with parameters.

- The mathematics involved in Dixon-EDF is at the undergraduate level. It is much easier to understand than Gröbner bases.
- As we saw in the datum transformation problem (section 6), if a system of equations admits useful algebraic simplifications, Dixon-EDF can indicate that by completing very quickly.
- Dixon-EDF challenges the user's creativity. There are many variations and options.

10 Appendix

The Maple-FGb commands for the pose example:

```
Maple 2015 (X86 64 LINUX)
Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2015
> with(FGb):
p := 0;
v1 := [ x2, x3, x4 ];
v2 := [ x1, b1,b2,b3,b4,c12,c23,c34,c41 ];
sys := [x1^2 + x2^2 - c12*x1*x2 - b1, x2^2 + x3^2 - c23*x2*x3 - b2,
           x3^2 + x4^2 - c34*x3*x4 - b3, x4^2 + x1^2 - c41*x4*x1 - b4];
> ll1:=fgb_gbasis_elim(sys, p,v1,v2,{"step"=8,"verb"=3,"index"=40000000});
```

Magma commands for the pose example:

```
Magma V2.21-8   Thu Dec 10 2015 13:26:28 on ace-math01 [Seed = 2343837211]
Type ? for help.  Type <Ctrl>-D to quit.
 Q:=RationalField();
 F<b1,b2,b3,b4,c12,c23,c34,c41> := FunctionField(Q,8);
 R<x1,x2,x3,x4> := PolynomialRing(F,4, "elim", [2,3,4]);
 I := Ideal ([x1^2 + x2^2 - c12*x1*x2 - b1, x2^2 + x3^2 - c23*x2*x3 - b2,
       x3^2 + x4^2 - c34*x3*x4 - b3, x4^2 + x1^2 - c41*x4*x4*x1 - b4]);
 time G := GroebnerBasis(I);
```

Mathematica command for the general 3D conic problem, solving for x (the e_i are the four equations from above):

```
AbsoluteTiming[solx = GroebnerBasis[{e1, e2, e3, e4}, {x, y, z, λ}, {y, z, λ},
                  MonomialOrder → EliminationOrder];]
```

References

[1] J. Awange, GNSS environmental sensing. Springer International Publishers, 2018.

[2] Awange JL, Palancz B, Lewis R, Lovas T, Heck B, and Fukuda Y (2016) An algebraic solution of maximum likelihood function in case of Gaussian mixture distribution. Australian Journal of Earth Sciences 63(2): 193-203, doi: 10.1080/08120099.2016.1143876.

[3] J. Awange, Environmental monitoring using GNSS. Springer-Verlag. New York, 2012.

[4] J. Awange, Y. Fukuda, E. Grafarend, Exact solution of the nonlinear 7 parameter datum transformation by Groebner basis. Bollettino di Geodesia e Scienze Affini (1) 2004.

[5] J. Awange, E. Grafarend, Algebraic solution of GPS pseudo-ranging equations. GPS Solutions 5(4): 20-32, 2002.

[6] J. Awange, E. Grafarend, Nonlinear Adjustment of GPS Observations of Type Pseudo-Ranges. GPS Solutions 5(4): 80-93, 2002.

[7] J. Awange, B. Paláncz, Geospatial Algebraic Computations, Theory and Applications. Springer-Verlag. New York, 2016.

[8] J. Awange, B. Palancz, B., R.H. Lewis, L. Vlgyesi, Mathematical Geosciences -Hybrid Symbolic-Numeric Methods-. Springer International Publishers, 2018.

[9] Bozoki, S., Lewis, R. H. Solving the least squares method problem in the AHP for 3×3 and 4×4 matrices. Central European Journal of Operations Research, 13(3), pp.255-270 (2005).

[10] L. Buse, M. Elkadi, and B. Mourrain, Generalized resultants over unirational algebraic varieties. J. Symbolic Comp. **29** (2000), p. 515-526.

[11] D. Bates, J. Hauenstein, A. Sommese, C. Wampler. Bertini: Software for Numerical Algebraic Geometry. https://bertini.nd.edu

[12] N. Buch, S.A. Velastin, J. Orwell, A Review of Computer Vision Techniques for the Analysis of Urban Traffic. IEEE Transactions on Intelligent Transportation Systems, vol 12, Issue: 3,(2011), p. 920 - 939, doi: 10.1109/TITS.2011.2119372.

[13] D. Cox, J. Little, D. O'Shea. Using Algebraic Geometry. Graduate Texts in Mathematics, 185. Springer-Verlag. New York, 1998.

[14] A. L. Dixon, The eliminant of three quantics in two independent variables. Proc. London Math. Soc. 6 (1908) 468 – 478.

[15] J.-C. Faugere, A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra (Elsevier Science) 139 (1999) 61 – 88.

[16] J.-C. Faugere, personal communication, July 8, 2014.

[17] P.K. Freeman, R.S. Freeland, Agricultural UAVs in the U.S.: potential, policy, and hype. Remote Sensing Applications: Society and Environment vol 2 (20150, p. 35-43, doi: 10.1016/j.rsase.2015.10.002

[18] W. Gentleman and S. Johnson, The evaluation of determinants by expansion by minors and the general problem of substitution. Mathematics of Computation, 28, (126) 1974, 543 – 548.

[19] B. Grenet, P. Koiran, N. Portier. (2010) The Multivariate Resultant is NP-hard in Any Characteristic. In: MFCS 2010. Lecture Notes in Computer Science, vol 6281. Springer, Berlin, Heidelberg.

[20] D. Kapur, T. Saxena, and L. Yang, Algebraic and geometric reasoning using Dixon resultants. In: *Proc. of the International Symposium on Symbolic and Algebraic Computation.* A.C.M. Press (1994).

[21] Z Kukelova, M Bujnak, T Pajdla, Polynomial Eigenvalue Solutions to the 5-pt and 6-pt Relative Pose Problems. - BMVC, 2008 - cmp.felk.cvut.cz// kukelova/webthesis/publications/Kukelova-etal-BMVC-2008.pdf [Accessed 12/1/2018]

[22] V. Lepetit, F. Moreno-Noguer, P. Fua, EPnP: An Accurate O(n) Solution to the PnP Problem. International Journal of Computer Vision, (2009) 81: 155. https://doi.org/10.1007/s11263-008-0152-6

[23] R. H. Lewis, Computer algebra system *Fermat*. http://home.bway.net/lewis/

[24] R. H. Lewis, Fermat code for Dixon-EDF, http://home.bway.net/lewis/dixon

[25] R. H. Lewis, Heuristics to accelerate the Dixon resultant, Mathematics and Computers in Simulation 77, Issue 4 (2008) 400 – 407.

[26] R. Lewis and S. Bridgett, Conic tangency equations arising from Apollonius problems in biochemistry, Mathematics and Computers in Simulation 61(2) (2003) 101 – 114.

[27] R. H. Lewis and E. A. Coutsias, Flexibility of Bricard's Linkages and Other Structures via Resultants and Computer Algebra. Mathematics and Computers in Simulation. November 2014. http://arxiv.org/abs/1408.6247

[28] R. H. Lewis and Peter Stiller, Solving the Recognition Problem for Six Lines Using the Dixon Resultant, Mathematics and Computers in Simulation **49** (1999) p. 203 – 219.

[29] D. Nister, An efficient solution to the five-point relative pose problem. IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004), Issue: 6, p. 756-770, doi: 10.1109/TPAMI.2004.17

[30] B. Paláncz, R. H. Lewis, P. Zaletnyik, and J. Awange, Computational study of the 3D affine transformation part I. 3-point problem. March 2008. online at http://library.wolfram. com/infocenter/MathSource/7090/

[31] B. Paláncz, J. Awange, P. Zaletnyik, and R. H. Lewis, Linear homotopy solution of nonlinear systems of equations in geodesy, Journal of Geodesy. September 2009. http://www.springerlink.com/content/78qh80606j224341/

[32] B. Paláncz, P. Zaletnyik, J. Awange, and E. Grafarend, Dixon resultants solution of systems of geodetic polynomial equations, Journal of Geodesy. September 2008, vol. 82, p. 505-511.

[33] B. Paláncz, P. Zaletnyik, A symbolic solution of a 3D affine transformation. Mathematica Journal, vol. 13, p. 1 - 15.

[34] S. Periaswamy, H. Farid, Medical image registration with partial data. Medical Image Analysis, vol 10, Issue 3 (2006, p. 452-464, doi: 10.1016/j.media.2005.03.006.

[35] A. Steel, personal communications. September 2 – 7, 2015.

[36] A. J. Sommese, C. W. Wampler. The Numerical Solution of Systems of Polynomials: Arising in Engineering And Science. World Scientific. London, 2005.

[37] B. Sturmfels, Solving systems of polynomial equations. CBMS Regional Conference Series in Mathematics 97, American Mathematical Society, Providence, 2003.

[38] J. Ventura, C. Arth, G. Reitmayr, D. Schmalstieg, A Minimal Solution to the Generalized Pose-and-Scale Problem, Computer Vision and Pattern Recognition (CVPR) 2014 IEEE Conference on, pp. 422-429, 2014, ISSN 1063-6919.