1-1-2011

Structure From Motion With Directional Correspondence for Visual Odometry

Oleg Naroditsky  
*University of Pennsylvania*, narodits@seas.upenn.edu

Xun S. Zhou  
*University of Minnesota*, zhou@cs.umn.edu

Jean H. Gallier  
*University of Pennsylvania*, jean@cis.upenn.edu

Stergios I. Roumeliotis  
*University of Minnesota*, stergios@cs.umn.edu

Kostas Daniilidis  
*University of Pennsylvania*, kostas@cis.upenn.edu

Follow this and additional works at: [http://repository.upenn.edu/cis_reports](http://repository.upenn.edu/cis_reports)

Recommended Citation  


This paper is posted at ScholarlyCommons. [http://repository.upenn.edu/cis_reports/960](http://repository.upenn.edu/cis_reports/960)  
For more information, please contact repository@pobox.upenn.edu.
Structure From Motion With Directional Correspondence for Visual Odometry

Abstract
This report presents two efficient solutions to the two-view, relative pose problem from three image point correspondences and one common reference direction. This three-plus-one problem can be used either as a substitute for the classic five-point algorithm using a vanishing point for the reference direction, or to make use of an inertial measurement unit commonly available on robots and mobile devices, where the gravity vector becomes the reference direction. We provide a simple closed-form solution and a solution based on techniques from algebraic geometry and investigate numerical and computational advantages of each approach. In a set of real experiments, we demonstrate the power of our approach by comparing it to the five-point method in a hypothesize-and-test visual odometry setting.

Comments

This technical report is available at ScholarlyCommons: http://repository.upenn.edu/cis_reports/960
Structure from Motion with Directional Correspondence for Visual Odometry  
GRASP Laboratory Technical Report MS-CIS-11-15

Oleg Naroditsky, Xun S. Zhou, Jean Gallier, Stergios I. Roumeliotis, and Kostas Daniilidis

Abstract

This report presents two efficient solutions to the two-view, relative pose problem from three image point correspondences and one common reference direction. This three-plus-one problem can be used either as a substitute for the classic five-point algorithm using a vanishing point for the reference direction, or to make use of an inertial measurement unit commonly available on robots and mobile devices, where the gravity vector becomes the reference direction. We provide a simple closed-form solution and a solution based on techniques from algebraic geometry and investigate numerical and computational advantages of each approach. In a set of real experiments, we demonstrate the power of our approach by comparing it to the five-point method in a hypothesize-and-test visual odometry setting.

Index Terms

computer vision, structure from motion, visual odometry, minimal problems, Groebner basis

I. INTRODUCTION

Data association has been identified as one of the two main challenges in visual odometry next to observation noise (see special issue to the workshop [1]). Cluttered environments with independently moving objects yield many erroneous feature correspondences which have to be detected as outliers. It has been shown [2] that Random Sample Consensus (RANSAC) provides a stable framework for the treatment of outliers in monocular visual odometry. For RANSAC it is highly desirable to have a hypothesis generator that uses the minimal number of data points to generate a finite set of solutions, since this minimizes the probability of choosing an outlier as part of the data. For example, in minimal cases, absolute pose estimation requires three correspondences between the world and image points, and relative pose requires five image to image correspondences. In this paper we propose a new minimal method for computing relative pose for monocular visual odometry that uses three image correspondences and a common direction in the two camera coordinate frames, which we call a "directional correspondence". We call this the "three-plus-one" method. The main motivation for using the three-plus-one method is to enable visual odometry using RANSAC with a four-point minimal solver (instead of the traditional five), as long as the fourth point is at infinity (and thus provides a directional correspondence). However, with the advent of robots and mobile devices with inertial measurement units (IMU), the three-plus-one algorithm becomes an attractive way to solve the visual odometry problem where the directional correspondence is provided by an IMU (e.g. the gravity vector).

The main goal of this paper is to introduce two efficient algorithms for the three-plus-one problem. It has been known [3] and it is straightforward to deduce that the knowledge of the directional correspondence reduces the number of rotation unknowns to one, yielding a system of three quadratic equations in three unknowns. In Section III, we show how to formulate the full relative pose problem as a system of four polynomial equations. We then present two methods for solving this system. The first method is a direct, closed-form solution, leading to a quartic polynomial of one variable, and is found in Section IV. Before introducing the second method, we give a brief introduction to the algebraic geometry techniques in Section V. The method based on these techniques, and specifically on the “action matrix” method from...
Byrod et al [4], is presented in Section VI. In this case, eigenvalue decomposition of a $4 \times 4$ matrix recovers three of the four variables simultaneously.

Our second goal is to investigate the properties of both solutions, exploring performance under noise and adverse imaging conditions. In Section VII, we demonstrate that both methods solve the problem correctly and highlight the differences in numerical properties and computational requirements. Since we envision that this algorithm may be implemented on mobile devices and low-power CPUs that frequently lack the hardware to process double precision (64-bit) floating point numbers, we investigate the numerical properties of both solutions in single and double precision arithmetic, and show that except for specific imaging conditions, both algorithms are stable in both implementations.

Our third goal is to demonstrate that the three-plus-one method can be used in place of the five-point method in visual odometry applications. To that end, we compare the accuracy and the computational requirements of the five-point algorithm with the closed-form algorithm for the three-plus-one problem using simulated data, and show it to be superior in estimating camera rotation in noisy conditions without sacrificing translation estimation accuracy.

The final goal of this paper is to demonstrate that the three-plus-one method is a reliable alternative to the five-point method in a real-world visual odometry application, if an ample supply of distant points is present (such as in outdoor environments). In Section VIII-A, we use our method in a RANSAC framework to compute monocular visual odometry on outdoor video data. When used with RANSAC, our visual odometry does not require any knowledge about which points are at infinity, because we simply let RANSAC choose the inlier hypothesis from all available image correspondences. We will show that since we only require four correspondences, our method leads to more robust visual odometry than the five-point method. Moreover, in cases where only few nearby point correspondences can be found, having it as another hypothesis generator reduces the probability of failure of a five-point-based navigation algorithm.

In Section VIII-B, in order to demonstrate the potential of our method for vision-inertial fusion, we present the results of a real experiment, comparing the three-plus-one and the five-point method when an IMU is available.

II. RELATED WORK

Our work places itself at the intersection of minimal solvers for geometric vision problems with approaches using motion constraints.

Minimal solvers were first introduced by Nister [5] with his famous five-point algorithm for structure from motion. Since then, minimal solutions have been found for a number of problems in geometry. Among them are the solutions to the autocalibration of radial distortion [6], [7], pose with unknown focal length [8], infinitesimal camera motion problem [9] and others. The trend in this field has been to use algebraic geometry techniques to analyze problem structure and construct solvers. This body of work was initially based on Gröbner bases techniques [10], but recently started to include other related methods for finding solutions to algebraic systems in order to improve speed and accuracy [11], [4]. These techniques have been applied non-minimal problems as well, such as three-view triangulation [12], [13].

There have been two previous papers dedicated to solving the three-plus-one problem. However, both fall short of providing an efficient, closed-form solution. The solution in Kalantari et al. [14] uses a Gröbner basis method, but due to suboptimal formulation, ends up with 12 solutions. A related problem was solved in the work of Lobo and Dias [15] who use a general formulation of a given reference direction (vertical in their case) to solve several geometric vision problems by using vanishing points and/or inertial measurements.

Structure from motion has benefited from attitude measurements. When all 3 DOF of rotation are known, either from vanishing points [16], [17] or inertial measurements, the problem can be reduced to a tractable estimation of the focus of expansion. Vieville [3] demonstrated the first approach where only the gravity vector is used to simplify structure from motion. The relation between the use of gravity vector and the lack of knowledge in correspondences has been studied in [18], [19]. When multiple frames are used,
inertial measurements have been naturally integrated along visual features as measurements in nonlinear Kalman filtering [20], [21], [22], [23], [24]. Partial information like altitude has been used [25] to eliminate the unknown scale of monocular vision. Diel et al. [26] presented how a new epipolar constraint based on inertial information can be added in the visual odometry estimation process. In augmented reality, Azuma [27] and You et al. [28] proposed hybrid inertial-vision trackers where vision-based algorithms refine orientation estimates provided by an inertial sensor. Burschka and Hager developed a vision approach to SLAM [29], which circumvents the problems caused from drift in the inertial measurements by using a vision algorithm to estimate directly the relative pose of the camera between frames.

III. PROBLEM FORMULATION AND NOTATION

We now introduce notation for the basic geometric objects we will use to formulate the problem. Image points are represented by homogeneous 3-vectors \( \mathbf{q} = (x, y, 1)^\top \). Scene (world) points are represented by homogeneous 4-vectors \( \mathbf{Q} = (X, Y, Z, 1)^\top \). Given image point correspondences \( \mathbf{q} \) and \( \mathbf{q}' \) in two calibrated views, it is known that the “essential matrix” constraint relating them is \( \mathbf{q}'^\top \mathbf{E} \mathbf{q} = 0 \), where \( \mathbf{E} \equiv \mathbf{t} \mathbf{S} \) where the rotation matrix \( \mathbf{S} \in \text{SO}(3) \) and \( \mathbf{t} \) is a \( 3 \times 3 \) skew-symmetric matrix corresponding to the translation vector \( \mathbf{t} \), which is known only up to scale. The essential matrix thus has five parameters.

We will now define and formulate the three-plus-one problem. We are given three image correspondences \( \mathbf{q}_i \leftrightarrow \mathbf{q}_i' \), \( i = 1, \ldots, 3 \) from calibrated cameras, and a single directional correspondence in the form of two unit vectors \( \mathbf{d} \leftrightarrow \mathbf{d}' \). Our goal is to find the essential matrix \( \mathbf{E} \) which relates the two cameras, and thus find the rigid transformation between them up to a scale factor. We will first show that this problem is equivalent to finding the translation vector \( \mathbf{t} \) and a rotation angle \( \theta \) around an arbitrary rotation axis.

Let us choose the arbitrary rotation axis to be \( \mathbf{e}_2 = [0, 1, 0]^\top \). We can now compute the rotation matrices \( \mathbf{R} \) and \( \mathbf{R}' \) that coincide \( \mathbf{d} \) and \( \mathbf{d}' \) with \( \mathbf{e}_2 \), and apply them to the respective image points, yielding \( \mathbf{p}_i = \mathbf{R} \mathbf{q}_i \) and \( \mathbf{p}_i' = \mathbf{R}' \mathbf{q}_i' \) for each \( i = 1, \ldots, 3 \). This operation aligns the directional correspondence in the two views with \( \mathbf{e}_2 \). Once the axis is chosen, we only need to estimate the rotation angle around it and the translation vector in order to reconstruct the essential matrix.

After taking the directional constraint into account, from the initial five parameters in the essential matrix, we now only have to estimate three. This three-parameter essential matrix \( \hat{\mathbf{E}} \) relates the points \( \mathbf{p} \) and \( \mathbf{p}' \) as follows:

\[
\mathbf{p}_i^\top \hat{\mathbf{E}} \mathbf{p}_i = 0, \tag{1}
\]

Since the rotation is known to be around \( \mathbf{e}_2 \), we can use the axis-angle parameterization of a rotation matrix to parametrize \( \hat{\mathbf{E}} \) as follows:

\[
\hat{\mathbf{E}} = \hat{\mathbf{t}}(I + \sin \theta \hat{\mathbf{e}}_2 + (1 - \cos \theta) \hat{\mathbf{e}}_2^2),
\]

where \( \hat{\mathbf{t}} = \mathbf{R}' \mathbf{t} \).

Each image point correspondence gives us one such equation of the form (1), for a total of three equations in four unknowns (elements of \( \mathbf{t} \) and \( \theta \)). To create a polynomial system, we set \( s = \sin \theta \) and \( c = \cos \theta \), and add the trigonometric constraint \( s^2 + c^2 - 1 = 0 \), for a total of five equations in four unknowns. In order to reduce the number of unknowns and take care of the scale ambiguity in \( \hat{\mathbf{E}} \), we choose the direction of the epipole by assuming that the translation vector \( \mathbf{t} \) has the form \( [x, y, 1]^\top \). This means that for each \( \mathbf{t} \) that we recover, \( -\mathbf{t} \) will also need to be considered as a possible solution.

Once we substitute for \( \hat{\mathbf{E}} \) in equation (1), the resulting system of polynomial equations has the following form:

\[
a_{i1}x + a_{i2}x + a_{i3}y + a_{i4}yc + a_{i5}x - a_{i6} + a_{i6} = 0 \tag{2}
\]

for \( i = 1, \ldots, 3 \), and the equation

\[
s^2 + c^2 - 1 = 0. \tag{3}
\]
We will refer to these equations as \( F = \{ f_i(x, y, s, c), i = 1, \ldots, 4 \} \) in the rest of the paper. The coefficients \( a_{ij} \) are expressed in terms of image correspondences as follows:

\[
\begin{align*}
    a_{11} &= p_{iy}'p_{ix} \\
    a_{i2} &= -p_{iy}' \\
    a_{i3} &= -p_{ix}'p_{ix} - 1 \\
    a_{i4} &= p_{ix}' - p_{ix} \\
    a_{i5} &= p_{iy} \\
    a_{i6} &= -p_{ix}'p_{iy},
\end{align*}
\]

(4)

where \( p_{ix} \) and \( p_{iy} \) (\( p_{ix}' \) and \( p_{iy}' \)) are the first and second components of the rotated image points \( p_i \) (\( p_i' \)). In the next section we will analyze and solve this system in closed form and show that it has up to four solutions. The total number of possible pose matrices arising from our formulation is therefore at most 8, when we take into account the fact that we have to consider the sign ambiguity in \( \tilde{t} \). When the motion of the camera in the \( z \) direction (after the rotation by \( R \) and \( R' \)) is extremely small, the parametrization \( \tilde{t} = [x, y, 1]^\top \) is numerically unstable. We deal with this rare instability by formulating and solving a system for the parametrizations \( \tilde{t} = [x, 1, z]^\top \) and \( \tilde{t} = [1, y, z]^\top \), which can be easily done using the methods we describe below, but omitted for the purposes of this presentation.

IV. CLOSED-FORM SOLUTION

We hereafter show that the system has four solutions, and that it can be solved analytically by elimination and back-substitution. Specifically, we first present an elimination procedure to obtain a \( 4^{th} \)-order univariate polynomial in \( c \), which can be solved in closed-form. Subsequently, we determine the remaining three variables through by back-substitution, where each solution of \( c \) returns exactly one solution for the other three variables. Therefore, we have a total of 4 solutions for the relative rotation matrix and translation vector.

The main steps of the elimination procedure are listed as follows.

1) Solve for \( x \) and \( y \) as a function of \( c \) and \( s \) using the first two equations in (2). The variables \( x \) and \( y \) can be expressed as quadratic functions of \( c \) and \( s \).

2) Substitute \( x \) and \( y \) in the third equation in (2). This yields again a cubic polynomial in \( c \) and \( s \), which is reduced into a quadratic by exploiting the relationship between its coefficients and the trigonometric constraint.

3) Finally, using the Sylvester resultant (see Chapter 3, §5 in [30] ), we can eliminate one of the remaining two unknowns, say \( s \), and obtain a \( 4^{th} \)-order polynomial in \( c \).

Now, we describe the details of our approach. Rewrite the first two equations in (2) as linear functions of \( c \) and \( s \) as follows:

\[
\begin{bmatrix}
    a_{11}s + a_{12}c + a_{15} \\
    a_{21}s + a_{22}c + a_{25}
\end{bmatrix}
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
= \begin{bmatrix}
    a_{12}s - a_{11}c - a_{16} \\
    a_{22}s - a_{21}c - a_{26}
\end{bmatrix},
\]

(5)

and solve the above linear system for \( x \) and \( y \):

\[
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
= \frac{1}{d} \begin{bmatrix}
    a_{23}s + a_{24}c \\
    -(a_{21}s + a_{22}c + a_{25})
\end{bmatrix}
\begin{bmatrix}
    a_{12}s - a_{11}c - a_{16} \\
    a_{22}s - a_{21}c - a_{26}
\end{bmatrix},
\]

(6)

where the determinant \( d = (a_{11}s + a_{12}c + a_{15})(a_{23}s + a_{24}c) - (a_{21}s + a_{22}c + a_{25})(a_{13}s + a_{14}c) \). Substituting the expression for \( x \) and \( y \) into the third equation in (2) and multiplying both sides of the equation by \( d \), yields a cubic equation in \( s \) and \( c \):

\[
g_1s^3 + g_2cs^2 + g_1sc^2 + g_2c^3 + g_3s^2 + g_4sc + g_5c^2 + g_6s + g_7c = 0.
\]
The coefficients \( g_i \) for \( i = 1, \ldots, 6 \) are derived symbolically and are found in Appendix A, equation (12). By using the fact that \( s^2 + c^2 = 1 \), and exploiting the relation between the coefficients of the first four terms, we can reduce this equation to the following quadratic

\[
g_1s + g_2c + g_3s^2 + g_4sc + g_5c^2 + g_6s + g_7c = 0. \tag{7}
\]

In the final step, we employ the Sylvester resultant to eliminate one of the two remaining variables from equations (3) and (7). The resultant of the two polynomials is the determinant of the Sylvester matrix

\[
\begin{bmatrix}
g_3 & g_4c + g_1 + g_6 & g_5c^2 + g_2c + g_7c & 0 \\
0 & g_3 & g_4c + g_1 + g_6 & g_5c^2 + g_2c + g_7c \\
1 & 0 & g_3 & g_4c + g_1 + g_6 \\
0 & 1 & 0 & g_3
\end{bmatrix},
\tag{8}
\]

which leads to a 4th-order polynomial equation

\[
\sum_{i=0}^{4} h_i c^i = 0, \tag{9}
\]

with coefficients \( h_i \) given in Appendix A, equation (13). This shows that in general, the system has four solutions for \( c \). Back-substituting the solutions of \( c \) into equation (7), we compute the corresponding solutions for \( s \). Note that each solution for \( c \) corresponds to one solution for \( s \) because we can reduce the order of equation (7) to linear in \( s \), once \( c \) is known, by replacing the quadratic terms \( s^2 \) with \( 1 - c^2 \). After \( s \) and \( c \) are determined, we compute the corresponding solutions for \( x \) and \( y \) using (6) for a total of four solutions. We will describe how to recover the pose matrix from \( x, y, s \) and \( c \) in Section VI-D.

In the next section, we will solve the polynomial system in (2) and (3) using algebraic geometry techniques.

V. ALGEBRAIC GEOMETRY BACKGROUND

In this section, we review the algebraic geometry concepts that have been applied to solving geometry problems in computer vision. The definitive introduction to these concepts can be found in textbooks on algebraic geometry [31], [30].

One of the textbook ways [31] of solving algebraic systems is via the so-called action matrix. We will give a brief overview of this method. Consider a system of polynomial equations in \( m \) variables \( f_1(x) = \ldots = f_n(x) = 0 \), where \( x = (x_1, \ldots, x_m) \), and coefficients from a field \( K \). A geometric description of the solution set to a polynomial system is given by an affine variety \( V \). In the case where there are finitely many solutions, the variety \( V \) is zero-dimensional which includes a finite number of points in \( K^m \) when \( K \) is an algebraically closed field. The ring of all polynomials in \( x \) is denoted by \( K[x] \).

The polynomials \( f_i \) are the generators of the polynomial ideal \( I = \{\sum_{i=1}^{m} h_i f_i : h_1, \ldots, h_m \in K[x]\} \). In other words, an ideal generated by \( f_i \) is a set that includes the generators, and is also closed under addition and multiplication by other polynomials in \( K[x] \). It is easy to show that the polynomials in the ideal vanish on the same variety as the generating set. The problem of solving the system now becomes a problem of finding a subset of equations in the ideal with properties that make them easy to solve.

The methods used to solve polynomial systems in computer vision rely heavily on the properties of a set of equivalence classes for polynomial division (the remainders) of members of \( K[x] \) by members of the ideal \( I \). This set of equivalence classes is called the quotient ring, and is denoted as \( K[x]/I \). If the variety is zero-dimensional (i.e., the system has finitely many solutions), the quotient ring is a vector space whose dimension equals the number of solutions. On this vector space, we can define linear maps, which are often represented in matrix forms and called action matrices.

The action matrix is the key for solving systems of polynomial equations. A univariate polynomial can be solved using eigenvalue decomposition of a companion matrix. The action matrix is a multivariate equivalent of the companion matrix. The idea is to find a linear operator \( T_p \) for some \( p \in K[x] \) that
represents the multiplication by \( p \) in the vector space defined by \( K[x]/I \), i.e., \( T_p : f(x) \rightarrow p(x)f(x) \). If we select a basis for this vector space, we can represent \( T_p \) as a matrix \( m_p \) with entries in \( K \). It was shown in [31] that \( \lambda \) is an eigenvalue of this matrix if and only if \( \lambda \) is a value of the function \( p \) evaluated on the variety \( V \) of the ideal. This means that if we set \( p = x_k \), we can find the value of \( x_k \) which satisfies the initial system of equations. We can also determine the solutions through eigenvectors. It is known that the eigenvectors of the action matrix represent the scaled solutions to the same problem. We can also determine the scale, because the monomial 1 is always in the basis for zero-dimensional varieties.

Finding the dimensionality and basis for this vector space is the first step in recovering solutions. The dimensionality immediately tells us the number of solutions, while the basis is important in the action matrix computation. One way of obtaining these two quantities is through division of polynomials in \( K[x] \) by the Gröbner basis, which is a special basis for the ideal, division by which cancels out all the possible leading terms of the polynomials in the ideal. A good introduction to Gröbner base is found in [30].

Computing a Gröbner basis using finite precision arithmetic is known to be a numerically unstable process. However, algorithms developed by Traverso in [32] allow us to analyze the ideal generated by our system using coefficients from a prime field \( K = \mathbb{Z}/r \) (integers modulo \( r \)), where \( r >> 7 \) is a prime number [33]. Since this field is finite, the computation with polynomials with coefficients in \( \mathbb{Z}/r \) (including Gröbner basis) is exact. The algorithms ensure that if a stable Gröbner basis is found in this field under repeated trials with random coefficients, the monomials will remain the same when we change the field to \( \mathbb{Q} \) with some probability. In our case, we only need the Gröbner basis for one system, \( F \), and it is easy to check when we have it. Once a Gröbner basis \( G \) is found, a linear basis for the quotient space can be formed by the monomials in the remainder after division by \( G \).

In order to use efficient linear algebra techniques to manipulate the system (2)-(3), we rewrite it as follows:

\[
CX = 0,
\]

where \( C \) is a matrix whose columns contain coefficients of the monomials, and \( X \) is the vector of monomials corresponding to the columns of \( C \). It should be noted that the ideal is closed under row operations on \( C \).

In solving our problem, we will follow the method outlined in [4], which allows us to build the action matrix without constructing a Gröbner basis (we will still analyze the system and extract its Gröbner basis in the finite field for the purposes listed above). We briefly describe their method here. The key idea is to determine the so-called solving basis \( B \) (in our case, we use the monomial basis for the quotient ring), and the required monomials \( R = x_kB \setminus B \). Specifically, our objective is to find the minimum number of monomials needed to construct the action matrix, and then re-arrange the matrix such that those monomials, along with the basis monomials, occupy the last columns of the matrix. Using algebraic geometry software we can find a candidate linear basis \( B \) for the quotient space. For the action matrix corresponding to multiplication by \( x_k \), the set of monomials that need to be expressed in terms of \( B \) is the set \( R = x_kB \setminus B \). The rest of the monomials in the system are called \( E \). The polynomial system with coefficients \( C \) can then be expressed as follows:

\[
CX = \begin{bmatrix} C_E & C_R & C_B \end{bmatrix} \begin{bmatrix} X_E \\ X_R \\ X_B \end{bmatrix} = 0.
\]

The only requirement on this coefficient matrix, after this matrix is put into the row-echelon form, is that its \(|R| \times |R|\) submatrix corresponding to the monomials in \( R \) and the last \(|R|\) equations has full rank. This submatrix is called \( C_{R2} \) in [4]. When we discuss our solution, we will illustrate how to use this matrix to extract the action matrix. The complete details are found in [4].

The initial set of equations \( F \) (see (2)-(3)) is unlikely to have a coefficient matrix \( C \) that meets the above requirement. This is where we will draw on the ideal members to expand the original system with additional equations, until the requirement on the action matrix construction is satisfied. The technique
to generate ideal members efficiently proposed in [33] involves multiplying the original polynomials by monomials starting with the lowest orders. This operation, when applied to an equation (a row of \( C \)), will result in the coefficients from that row to be shifted to the left in the matrix to take their places in columns corresponding to their new monomials. We will continue adding polynomials (checking for linear dependence and unneeded ones), until \( C \) is large enough to produce a full rank \( C_{R2} \). We call the resulting set of polynomials (which are monomial multiples of the original system) an \textit{elimination template}, and and matrix \( C \) an \textit{elimination matrix}. This part of the process can be done with coefficients drawn randomly from \( \mathbb{Z}/r \).

VI. \textbf{ACTION MATRIX SOLUTION}

In this section, we will follow our implementation of the action-matrix method due to [4], [33]. The elimination template should be computed using a symbolic mathematics software, such as Maple or Macaulay2. We found Maple to be a more convenient choice.

We have already showed that the system has four solutions, but this can also be verified using Maple’s Groebner package. Let \( J \) be the ideal generated by \( F \), where coefficients \( a_{ij} \) were chosen at random from \( \mathbb{Z}/30029 \). We then computed the GrevLex-order Gröbner basis for \( J \). Since this ideal is zero-dimensional, and the vector space spanned by the polynomials of the quotient ring was four-dimensional, there are in general four solutions to the system in the field of real numbers.

In the next two subsections, we describe the details that are specific to the three-plus-one problem, and thus the set of polynomials formed by (2) and (3). The choices of variable order and action monomial (\( c \) in our case) that we made below are not arbitrary. Other choices can produce much larger elimination templates or may be less favorable numerically. The entire process of elimination template generation was repeated for several variable orderings and action monomials to ensure stability and small size of the template. This process paralleled the automated method proposed in [33]. (We did not use that method directly due to its use of full Gröbner basis and a requirement for reduced row-echelon form, instead of row-echelon form, for the coefficient matrix which causes numerical instability.)

A. \textbf{Finding the Bases}

We used the Gröbner basis of \( J \) in a finite field (such as the one computed to verify the number of solutions) to determine the solving basis monomials. We chose to order the variables \((x, y, c, s)\). After computing the GrevLex order Gröbner basis with that order, we chose the solving basis \( B \) to be \([y, c, s, 1]\), the same as the quotient ring basis monomials. The set of required monomials \( R \), which is the set of monomials that need to be expressed in terms of \( B \) for the action matrix \( m_c \), is thus \([yc, c^2, cs]\).

B. \textbf{Constructing the Elimination Template}

Once we know the solving basis and the required monomials, we must extend the initial set of four polynomials with other polynomials from the ideal \( J \) such that the rank condition on \( C_{R2} \), after elimination, is satisfied. We multiplied the four original polynomials by the monomials in \((x, y, c, s)\) of degrees 1 and 2 and added them to the system, put the resulting coefficient matrix in row-echelon form and check the rank of \( C_{R2} \). We then eliminated the redundant polynomials from the template.

The result of this process is elimination template, which consists of a set of 21 monomial multipliers and corresponding polynomials from the initial system:

\[
\{ f_1, f_2, f_3, f_4, f_1s, f_2s, f_3s, f_4s, f_1c, f_2c, f_3c, f_4y, f_1x, f_1s^2, f_2s^2, f_3s^2, f_1cs, f_2cs, f_3cs, f_4ys, f_4xs \} \quad (10)
\]

as well as, a vector of 25 monomials:

\[
[ yc^2s \hspace{0.2cm} ys^2c \hspace{0.2cm} xs^3 \hspace{0.2cm} ys^3 \hspace{0.2cm} xc^2 \hspace{0.2cm} yc^2 \hspace{0.2cm} xcs \hspace{0.2cm} ycs \hspace{0.2cm} c^2s \hspace{0.2cm} xs^2 \hspace{0.2cm} yxs \hspace{0.2cm} s^2c \hspace{0.2cm} s^3 \hspace{0.2cm} xcs \hspace{0.2cm} xs \hspace{0.2cm} ys \hspace{0.2cm} s^2x \hspace{0.2cm} xyc \hspace{0.2cm} c^2s \hspace{0.2cm} cs \hspace{0.2cm} y \hspace{0.2cm} c \hspace{0.2cm} s \hspace{0.2cm} 1 ]. \quad (11)
\]
The coefficients from the equations in (10) will form the entries in the $21 \times 25$ elimination matrix with columns corresponding to the monomials in the vector (11). The exact arrangement of coefficients is given in Appendix B.

C. Reduction and Action Matrix Extraction

With the coefficient matrix at hand, we leave Maple and $\mathbb{Z}/r$. The template will remain the same across all instances of the problem. We construct the $21 \times 25$ matrix from the coefficients $a_{ij}$ (see (4)) for the particular instance of the problem, and perform Gaussian elimination with partial pivoting or LU decomposition. The elimination can be stopped 3 rows early for added efficiency. We then extract the $3 \times 3$ matrix $C_{R_2}$ representing the monomials in $R$ (columns 19, 20 and 21) in the last three rows of the upper triangular matrix. We invert this matrix and multiply it with the matrix $C_B$ representing the monomials in $B$ (columns 22 through 25) in the last three rows. The rows of the resulting $3 \times 4$ matrix $C_{R_2}^{-1}C_B$ become the first three columns of the $4 \times 4$ action matrix $m_c$. The last column has a 1 in the third position, indicating that $c$ (a required monomial $1 \cdot c$) is already expressed in the basis as a vector $[0, 1, 0, 0]^T$. The solutions are extracted as the real eigenvectors of this action matrix which can be computed in closed form. Since the value of a constant polynomial evaluated at any point is also constant, we set the scale of our solutions by dividing each element of the eigenvector by the last element, which corresponds to the monomial 1.

We have recovered up to four sets of values for $y, c$ and $s$, and must now find the corresponding values for $x$ by solving one of the equations from (2) for each set of values. These equations are linear in $x$.

D. Back Substitution and Pose Recovery

We will now describe how to find the pose matrices from solutions to the system. We recover the rotation as

$$R_{e_2} = \exp(\text{atan2}(s, c)\hat{e}_2),$$

and translation as

$$\tilde{t} = \pm[x, y, 1]^T.$$

Finally, we reconstruct each pose as follows:

$$P = \begin{bmatrix} S & t \end{bmatrix} = \begin{bmatrix} R^T R_{e_2} R & R^T \tilde{t} \end{bmatrix}.$$

There are up to 8 such pose matrices for each instance of the problem. Point triangulation and chirality checks are used to eliminate false solutions. Since this solution method is designed to be used in robust estimation frameworks (such as RANSAC), any remaining false hypotheses can be eliminated by triangulating an additional point and choosing the $P$ with the minimum reprojection error.

VII. Simulation Results

In this section we establish the expected performance level of our algorithms in noise-free and noisy conditions, comparing them first to each other and then to the five-point relative pose estimation algorithm. This is accomplished with simulated data. We study both single and double precision arithmetic implementations for the action-matrix and closed-form algorithms, and look for numerical differences between them. In the comparison with the five-point method, we expect that the more constrained three-plus-one method will give better accuracy in noisy conditions.

In each figure where single and double precision versions of the three-plus-one algorithm are compared, the legend is as follows: C and A refer to the “closed form” and “action matrix” methods, respectively, and 32f and 64f refer to the floating point precision, single and double, respectively.

The input data was generated as follows. The pose of the first camera was defined to be the identity pose $[I|0]$, and the reference direction was generated as a random unit vector. The pose of the second camera
was generated uniformly at random as a unit translation vector $t$ and three Euler angles corresponding to roll, pitch and yaw of the second camera within the limits specified by the experiment. The Euler angles were converted to a rotation matrix $R$, which together with $t$ formed the camera pose $[R|t]$. Sets of five three dimensional world points were generated within a spatial volume defined by the parameters of the experiment, so as to be visible by both cameras. The world points were then projected into the image planes of the two cameras (with identical intrinsic calibration defined by the experiment) to form image correspondences, and contaminated with Gaussian noise with standard deviation in pixels defined by the experiment. The second camera’s reference direction was then computed, and the directional correspondence vectors were contaminated by Gaussian rotational noise with standard deviation in degrees defined by the experiment. The sets of image and directional correspondences were then used to compute pose with the three-plus-one and the five-point algorithms. Each method produces a set of pose hypotheses for each input set. The error reported for each input set is the minimum error for all hypotheses. All comparisons between algorithms were run on identical input data.

A. Perfect Data

First, we establish the correctness and numerical stability of our algorithms. In these experiments, the pose was allowed to vary over the entire range of rotations, and the translation and directional correspondence vectors were generated uniformly at random and normalized to length 1. Figure 1 shows errors in pose recovery on perfect, simulated data. The noise metric is the minimum Frobenius norm of the differences between the true pose matrix and each computed pose matrix (up to eight per instance). The error distribution shows that both algorithms perform as expected, with the action matrix method exhibiting better numerical stability. We will discuss this difference in the next section.

B. Image Noise

In subsequent simulated results we examine the performance for “standard” imaging conditions, which we define as a 640x480 camera with a 60° FOV, where structure points are found between 10 and 40
baselines away, where one baseline is the distance between camera centers. We will first consider only pixel noise, and deal with directional correspondence noise later. Figure 2 compares performance for forward and sideways motion of the camera under different pixel noise conditions. It comes as no surprise that forward motion is generally better numerically, and that the rotation estimate (1DOF) is significantly better than the estimate of the epipole. The plots also conclusively demonstrate numerical stability of both single and double precision implementations under "standard" imaging conditions. These experiments show that once realistic noise is added, the numerical precision of either algorithm is sufficient for implementation on single precision processors.

The seemingly large errors in epipole estimation are due to the fact that the scene points are located far from the cameras, but we believe it to be a realistic (if difficult) configuration.

C. Directional Correspondence Noise

In this section, we investigate the performance impact of errors in directional correspondences. The directional noise was simulated as a rotation of the direction vector around a random axis with an angle magnitude drawn from a normal distribution. The standard deviation of the noise is plotted on the x-axis. The effect of directional noise only for a range of errors between 1 and 2 degrees can be seen in Figure 3. Performance under both pixel and directional noise is presented in Figure 4.
Fig. 4. Median translation and rotation errors for varying levels of noise in both directional and image correspondences for the "standard" camera. The noise standard deviation varies from 0° to 2°, and from 0 to 2 pixels for image correspondences.

Fig. 5. Distribution of errors in epipole orientations in degrees for 10⁴ trials under forward motion with pixel error standard deviation of 0.3 and field of view of 10°. The median errors for double precision are 1.1° for both methods. For single precision the errors are 8.5° for the closed form and 1.9° for the action matrix method.

D. Numerical Stability

With noise-free data, the closed-form, single precision algorithm has noticeably worse performance than the action matrix algorithm (see Figure 1), however, there is no noticeable difference when the noise is added for "standard" camera, as we saw in the previous sections. Figure 5 demonstrates that under more difficult imaging conditions of 10° field of view with 0.3 pixel noise, the median errors in the epipole direction are the same in double precision, but in single precision the error is 4.5 times greater for the closed-form solution. This demonstrates that under some conditions, it is advantageous to use the action-matrix solution because of its superior numeric properties.

E. Comparison with the Five-point Method

We also compare the three-plus-one method to the classic five-point method. While they are not equivalent (since the five-point method does not require a specific point to be at infinity), they can
be used interchangeably in some real situations, as described in the next section.

Since both closed-form and action-matrix-based algorithms exhibit similar performance, we only compare the double precision, closed-form solution to a double precision implementation of the five-point algorithm. Figure 6 demonstrates the effect of the field of view on the algorithms. The graph demonstrates that the rotation estimation is generally better with the three-plus-one algorithm, while translation error does not decrease as quickly with the field of view in the three-plus-one case as in the five-point case. In Figure 7 we plot errors for several levels of directional noise, while varying the pixels noise. It is clear from the graphs that the three plus one algorithm is better at estimating rotations than the five point algorithm, even under significant error in the directional correspondence, but the five-point method is better at estimating sideways translation, even in the cases of small error in the directional correspondence.

As we mentioned in the introduction, our method allows us to do purely vision-based relative pose, if points at infinity are present. We will now compare the performance of the five-point and three-plus-one methods for the case where the directional correspondences come from image points, i.e. vanishing points or other points at infinity. In this case, the directional correspondence noise can be measured in pixels, thus putting the two methods on equal footing. The test data were generated differently for this experiment. The first three correspondences were projected into the image from a range of 10 to 40 baselines, as before. An additional point at infinity was randomly generated within the field of view of the camera and projected into the images. This correspondence was unitized and contaminated with pixel noise of the same standard deviation as the first three image points, giving us a directional constraint from a point at infinity. This experiment assumes that a real camera would have a sufficiently wide depth of field to keep nearby and distant features in focus simultaneously, which is expected when its field of view is wide, or its aperture is sufficiently small. The results are shown in Figure 8. From this graph we conclude that our method outperforms the five-point method, while using only four image points, in estimating rotation in forward and sideways motion, and translation in forward motion. Our method does a slightly worse job estimating translation in the sideways motion.

F. Computational Considerations

When using RANSAC, we can estimate the probability of success in getting an outlier-free hypothesis based on the number of elements in the minimal data set. We can observe that with four points instead
of five, we can realize a significant advantage since fewer hypotheses need to be considered [34].

Since the hypothesis generator will run hundreds of times per frame in RANSAC-based visual odometry schemes, it is important to compare the computational requirements for the five point algorithm with the proposed methods. Computing the coefficients $a_{ij}$ requires 9 multiplications. The closed-form solution requires 125 multiplications before arriving at the quartic polynomial. The roots of the 4th degree polynomial can be extracted in closed form. The back substitution for the remaining variables takes additional 92 operations. The main computational step of the action matrix algorithm is a Gaussian elimination (LU decomposition) of a $21 \times 25$ matrix. While theoretically taking $O(2n^3/3)$, or about 9000 operations, the elimination of our sparse matrix only requires about 500 multiplications. The eigenvalue decomposition of a $4 \times 4$ matrix and extraction of roots of a 4th degree polynomial can also be done in closed form, after which we can extract all the variables after 80 operations. On the other hand, the main
computational steps in the classic five-point algorithm [5] are: extraction of the null space of a dense $5 \times 9$ matrix, requiring 280 operations, Gauss-Jordan elimination of a dense $10 \times 20$ matrix, requiring about 1300 operations, and real root isolation of a 10th degree polynomial, which can be accomplished as eigenvalue decomposition of a $10 \times 10$ sparse companion matrix or as an iterative root isolation and refinement process. From these observations we can conclude that both the closed-form and the action-matrix forms of the three-plus-one algorithm are significantly more efficient than the five-point algorithm.

VIII. EXPERIMENTAL RESULTS

In the introduction we specified as one of the main goals of this work the demonstration of monocular, RANSAC-based visual odometry with a four-correspondence hypothesis generator. We used our C++, double-precision implementation of the action-matrix-based method to test the algorithm in this context. We used a hand-held, $640 \times 480$ pixel, black and white camera with a $50^\circ$ field of view lens to record an 825-frame, outdoor video sequence. The sequences included motion exercising all degrees of freedom, and was a realistic representation of a robot localization task (see sample images in Figure 9).

Harris corners and correlation matching was used to obtain image correspondences. The matches were used to estimate camera motion following the monocular scheme similar to the one described in [2]. The experiments consisted of using the correspondences to estimate camera motion with the standard five point algorithm and the new three point plus direction algorithm. We computed 200 hypotheses for each image pair. The correspondences themselves, the number of hypotheses and the other system parameters remained the same, and only the pose hypothesis generator was changed between experiments. In the structure from motion experiment without an IMU, the directional correspondence was simply a unitized image point correspondence. Since most outdoor scenes have no shortage of faraway feature matches, RANSAC had no trouble choosing the right hypothesis with our method.

We will briefly describe the steps of the monocular visual odometry scheme:

1) Track features between consecutive frames. Estimate relative poses between two frames using preemptive RANSAC [35] with hypotheses generated either by the five point algorithm or the three-plus-one algorithm.

2) Use iterative refinement to polish the estimated pose with respect to all the inlier points.

3) Triangulate the feature matches in the two frames into 3D points. If this is not the first pair of frames, estimate the common scale between the current and last pose estimate using a 1-point preemptive RANSAC procedure.

4) Set the scale for the current pose estimate and attach it to the last estimate.

5) Repeat from step 1.

This procedure produces a robust, monocular visual navigation solution. If the features cannot be tracked for some reason or the preemptive RANSAC fails to produce a correct hypothesis, the scale estimate will fail, and the pose will jump.

A. Structure from Motion Results

In Figure 10 we stitched together the poses and highlighted the places where breaks in the path occurred. Since we know that we have enough points to track, the failures are due to RANSAC-based pose estimation or RANSAC-based scale estimation, and is a result of a failure to choose an inlier subset. It is interesting to note that the failures happened in different places with different algorithms due to randomness of sampling. We expect more robust results (fewer breaks) from the three-plus-one method, and we found it to be the case due to the limited number of hypothesis.

In order to further quantify the real-world performance, we also computed relative pose for each consecutive pair of frames in the data set with each algorithm. Figure 11 shows the histogram of relative errors between the three-plus-one and five-point algorithm. For each algorithm, the pose was computed as follows: corner features were matched between two consecutive frames, and RANSAC-based relative pose estimation, followed by iterative refinement was performed on all matches. The error between poses is, as
before, the logarithm of the Frobenius norm of the difference between the pose matrices. The algorithms give very similar results, except in a handful of frames, where pose was computed incorrectly due to a failure to select an inlier point set in one or the other algorithm.

B. Structure from Motion Results with a Camera and an IMU

We investigated using our algorithm to combine visual and inertial data by introducing the gravity vector in the camera coordinate system as the directional correspondence. For this data collection, the camera was rigidly mounted on a rig with a Microstrain 3DM-GX1 IMU, and data was synchronously acquired from both devices. We collected an indoor data set and used the visual odometry setup described above to compare the five-point method with our three-points-plus-gravity method. There were no visible points at infinity in this data set, and the reference direction was set to the gravity vector of the IMU in the camera coordinate system. The camera and IMU rig was moved by hand in all six degrees of freedom. The results are presented in Figure 12. In this data set, RANSAC with the five-point hypothesis generator generally performed similarly to our method, but failed to accurately recover relative pose for one of the frames, resulting in a jump near the bottom left of the trajectory, and failure to close the loop.

IX. Conclusion

We presented two efficient algorithms to determine relative pose from three image point correspondences and a directional correspondence. We show that the algorithm based on algebraic geometry yields better numerical performance than the simpler, closed-form algorithm, but the differences are not significant in
Fig. 10. Estimated camera trajectories for the outdoor data set using our three-plus-one method (blue) and the five-point algorithm (green). The red squares and circles indicate places where scale was lost, and trajectory was manually stitched together. Given the same input, our method jumped twice, while the five-point method jumped four times. The scale was not reset after stitching, so each piece of the trajectory has a different scale. Since the translation is up to scale, the translation units are set arbitrarily. The total length of the track in the real world was about 430m, of which we were able to travel about 230m before the first break under challenging imaging conditions. Sample frames are shown in Figure 9.

most realistic settings. However, we also demonstrated that in certain difficult imaging configurations, the action matrix method can perform better in single precision implementation, and is therefore recommended for processors with 32-bit floating-point arithmetic where extra accuracy is required.

In our comparison with the five-point method, we showed that the more constrained three-plus-one method does a better job of estimating rotations than the five-point method, as expected. We also showed that both the closed-form and action-matrix implementations are faster than the five-point method, making it even more attractive for real-time applications.

Another attribute of our algorithm is its non-degeneracy for colinear world points in general, however, we have identified and confirmed experimentally three degenerate configurations: when all world points lie on the horopter [36], and when the three world points are lie on a line parallel to the direction of motion or the reference direction.

We also demonstrated that the three-plus-one algorithm can provide accurate and robust results in real-world settings when used with RANSAC and bundle adjustment, and can be used to perform complete six degree of freedom visual odometry for outdoor scenes with or without aid from an IMU. We demonstrated that in those settings, our method exhibits better robustness then the five-point method when used with
where coefficients navigation systems, while improving speed. It is that it can be used as a complement to the five-point algorithm to increase the reliability of visual RANSAC due to having a smaller minimal data set. We believe that the real power of this algorithm is that it can be used as a complement to the five-point algorithm to increase the reliability of visual navigation systems, while improving speed.

APPENDIX A
CLOSED-FORM COEFFICIENTS

In this appendix we list the coefficients for the closed-form solution presented in Section IV. The coefficients \( a_{ij} \) are found in equation (4). The coefficients \( g_i \) in the polynomial (7) are as follows:

\[
\begin{align*}
g_1 &= -a_{11}a_{23}a_{32} + a_{13}a_{21}a_{32} - a_{12}a_{21}a_{33} + a_{11}a_{22}a_{33} + a_{23}a_{12}a_{31} - a_{13}a_{22}a_{31} \\
g_2 &= -a_{24}a_{11}a_{32} + a_{14}a_{21}a_{32} - a_{12}a_{21}a_{34} + a_{11}a_{22}a_{34} + a_{24}a_{12}a_{31} - a_{14}a_{22}a_{31} \\
g_3 &= -a_{23}a_{16}a_{31} + a_{13}a_{26}a_{31} + a_{23}a_{12}a_{35} - a_{13}a_{22}a_{35} - a_{11}a_{26}a_{33} + a_{15}a_{22}a_{33} \\
&\quad+ a_{21}a_{16}a_{33} - a_{25}a_{12}a_{32} - a_{15}a_{23}a_{32} + a_{13}a_{25}a_{32} + a_{11}a_{23}a_{36} - a_{13}a_{21}a_{36} \\
g_4 &= -a_{23}a_{16}a_{32} - a_{24}a_{16}a_{31} + a_{13}a_{26}a_{32} + a_{14}a_{26}a_{31} - a_{11}a_{23}a_{35} + a_{12}a_{24}a_{35} \\
&\quad+ a_{13}a_{21}a_{35} - a_{14}a_{22}a_{35} - a_{11}a_{26}a_{34} - a_{12}a_{26}a_{33} + a_{15}a_{22}a_{34} - a_{15}a_{21}a_{33} \\
&\quad+ a_{21}a_{16}a_{34} + a_{22}a_{16}a_{33} - a_{25}a_{12}a_{34} + a_{25}a_{11}a_{33} + a_{15}a_{23}a_{31} - a_{15}a_{24}a_{32} \\
&\quad- a_{13}a_{25}a_{31} + a_{14}a_{25}a_{32} + a_{24}a_{11}a_{36} + a_{23}a_{12}a_{36} - a_{13}a_{22}a_{36} - a_{14}a_{21}a_{36} \\
g_5 &= -a_{24}a_{16}a_{32} + a_{14}a_{26}a_{32} - a_{24}a_{11}a_{35} + a_{14}a_{21}a_{35} - a_{12}a_{26}a_{34} - a_{15}a_{21}a_{34} \\
&\quad+ a_{22}a_{16}a_{34} + a_{25}a_{11}a_{34} + a_{15}a_{24}a_{31} - a_{14}a_{25}a_{31} + a_{12}a_{24}a_{36} - a_{14}a_{22}a_{36} \\
g_6 &= -a_{23}a_{16}a_{35} + a_{13}a_{26}a_{35} - a_{15}a_{26}a_{33} + a_{25}a_{16}a_{33} + a_{15}a_{22}a_{36} - a_{13}a_{25}a_{36} \\
g_7 &= -a_{24}a_{16}a_{35} + a_{14}a_{26}a_{35} - a_{15}a_{26}a_{34} + a_{25}a_{16}a_{34} + a_{15}a_{24}a_{36} - a_{14}a_{25}a_{36},
\end{align*}
\]

where \( a_{ij} \) come from (4). The coefficients of the quartic polynomial in \( c \) are

\[
\begin{align*}
h_0 &= -g_1^2 - 2g_1g_6 - g_6^2 + g_3^2 \\
h_1 &= 2g_3g_2 - 2g_4g_6 + 2g_3g_7 - 2g_4g_1 \\
h_2 &= -g_4^2 + g_1^2 + g_6^2 + g_2^2 + g_7^2 - 2g_3^2 + 2g_1g_6 + 2g_2g_7 + 2g_3g_5 \\
h_3 &= 2g_4g_1 + 2g_4g_6 + 2g_5g_2 + 2g_5g_7 - 2g_3g_2 - 2g_3g_7 \\
h_4 &= g_1^2 + g_5^2 + g_3^2 - 2g_3g_5.
\end{align*}
\]
The quartic equation built from the coefficients $h_i$ yields the solution for $c$.

**APPENDIX B
Elimination Template Matrix**

In this appendix we list the coefficients of the $21 \times 25$ elimination matrix described in Section VI-B. The coefficients $a_{ij}$ are found in equation (4). The first fifteen rows are arranged in five sets of three rows that are the coefficients of equations in (2) after multiplication by variables listed in equation (10). Each row below is repeated for each $i = 1, 2, 3$:

\[
\begin{bmatrix}
    a_{i4} & a_{i3} & 0 & 0 & 0 & 0 & a_{i5} & 0 & a_{i1} & 0 & 0 & -a_{i2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{i6} & 0 & 0 & 0 & 0 \\
    0 & a_{i4} & a_{i1} & 0 & 0 & 0 & a_{i3} & 0 & 0 & 0 & 0 & a_{i5} & 0 & a_{i1} & -a_{i2} & 0 & 0 & 0 & a_{i6} & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
The last six rows of the matrix come from the coefficients of equation (3) after multiplication:

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & a_2 & a_4 & a_1 & a_3 & 0 & 0 & 0 & 0 & a_5 & 0 & 0 & 0 & 0 & a_1 & -a_2 & 0 & a_6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & a_2 & a_4 & 0 & a_1 & a_3 & 0 & 0 & a_5 & 0 & -a_2 & 0 & 0 & 0 & a_1 & 0 & 0 & a_6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_2 & a_1 & a_3 & 0 & a_5 & a_4 & 0 & 0 & a_1 & -a_2 & a_6
\end{bmatrix}
\]

This matrix can now be used for the reduction and action matrix extraction, as described in Section VI-C.

ACKNOWLEDGMENT

The authors are grateful for support through the following grants: NSF-IIS-0713260, NSF-IIP-0742304, NSF-IIP-0835714 and ARL MAST-CTA W911NF-08-2-0004.

REFERENCES