Two Efficient Solutions for Visual Odometry Using Directional Correspondence

Oleg Naroditsky, *Student Member, IEEE,* Xun S. Zhou, *Student Member, IEEE,*
Jean Gallier, Stergios I. Roumeliotis, *Member, IEEE* and Kostas Daniilidis, *Senior Member, IEEE*

Abstract—
This paper presents two new, 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

1 Introduction

DATA association has been identified as one of the two main challenges in visual odometry next to observation noise. Cluttered environments with independently moving objects yield many erroneous feature correspondences which have to be detected as outliers. Random Sample Consensus (RANSAC) provides a stable framework for the treatment of outliers in monocular visual odometry [1]. 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. In this paper we propose a new minimal method, the "three-plus-one" 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 thus enable visual odometry using RANSAC with only a four-point minimal solver, as long as the fourth point is sufficiently far away.

The main contribution of this paper is the introduction of two new, efficient algorithms for the three-plus-one problem. After formulating the relative pose problem as a system of four polynomial equations in Section 3, we present a direct, closed-form solution, leading to a quartic polynomial of one variable in Section 4. Our second method based on action matrix method from Byrod *et al.* [3], is presented in Section 5.

The second contribution of this paper threefold: we provide a detailed computational analysis of both solutions in Section 7, highlight the differences in numerical properties in Section 8, and show that the three-plus-one method can be used in place of the widely used five-point method in real-world visual odometry applications in Section 9.1. 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. In Section 9.2, in order to demonstrate the potential of our method for vision-inertial fusion, we present the results of a real experiment where we use the IMU to provide a directional constraint.

2 Related Work

Minimal solvers were first introduced by Nister [4] with his famous five-point algorithm for structure from motion. Since then, minimal solutions have been found for a number of problems in geometric vision, including the solutions to the autocalibration of radial distortion [5], [6], pose with unknown focal length [7], infinitesimal camera motion problem [8] 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 [9], but recently started to include other related methods for finding solutions to algebraic systems in order to improve speed and accuracy [10], [3].

It has been known [2] and it is straightforward to deduce that the knowledge of the directional correspondence reduces the number of rotation unknowns

- *Oleg Naroditsky, Jean Gallier and Kostas Daniilidis are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104. E-mail: {narodits, jean, kostas}@cis.upenn.edu.*
- *Xun Zhou and Stergios Roumeliotis are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. E-mail: {zhou, stergios}@cs.umn.edu.*

to oneThere have been three previous papers dedicated to solving the three-plus-one problem. However, all of them fall short of providing either efficient or closed-form solutions. The formulation from Kalantari *et al.* [11] requires three variables for the translation and uses the tangent half-angle formula to create a polynomial system. They solve the system using the action matrix method, leading to 12 solutions and a singularity at the rotation of 180 degrees.

Our formulation improves on this method by taking advantage of the fact that one of the translation components is non-zero, leading to a smaller system, which has only four solutions and avoids the rotation instability.

In a recent work by Fraundorfer et al. [12], a different formulation is used to obtain a system with only 4 solutions. Their method follows the method used by Nister for the five-point algorithm [4], except the solution subspace is now three-dimensional instead of four-dimensional. This formulation leads to a 4^{th}-order polynomial, and the minimum number of solutions. However, this solution requires null-space extraction and a Gauss-Jordan elimination of a dense 10×6 matrix to obtain the coefficients. While our formulation also leads to 4th order polynomial, the coefficients of our polynomial are computed faster and in closed form.

A related problem was solved in the work of Lobo and Dias [13] who use a general formulation of a given reference direction (vertical in their case) to solve several geometric vision problems by using vanishing points or inertial measurements.

In addition, compared to the three papers above, we provide a more elaborate experimental analysis, including real-world situations and a detailed discussion of the degenerate configurations.

3 Problem Formulation and Notation

Given image point correspondences $\mathbf{q}$ and $\mathbf{q}'$ in two calibrated views, it is known that the "essential matrix" constraint relating them is $\mathbf{q}'^{\top} E \mathbf{q} = 0$, where $E \equiv \hat{\mathbf{t}} S$ where the rotation matrix $S \in \mathbf{SO}(3)$ and $\hat{\mathbf{t}}$ is a 3×3 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 $\mathbf{q}_i \leftrightarrow \mathbf{q}'_i$, $i = 1, .., 3$ 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 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 $\hat{\mathbf{t}}$ and a rotation angle θ 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 R and R' that coincide $\mathbf{d}$ and $\mathbf{d}'$ with $\mathbf{e}_2$, and apply them to the respective image points, yielding $\mathbf{p_i} = R\mathbf{q_i}$ and $\mathbf{p_i}' = R'\mathbf{q_i}'$ for each $i = 1, .., 3$. 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 $\tilde{E}$ relates the points $\mathbf{p}$ and $\mathbf{p}'$ as follows:

$$\mathbf{p}_i'^{T} \tilde{E} \mathbf{p}_i = 0, \tag{1}$$

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

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

where $\tilde{\mathbf{t}} = R'\mathbf{t}$.

Each image point correspondence gives us one such equation of the form (1), for a total of three equations in four unknowns (elements of $\tilde{\mathbf{t}}$ and θ). 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 four equations in five unknowns. In order to reduce the number of unknowns and take care of the scale ambiguity in $\tilde{E}$, we choose the direction of the epipole by assuming that the translation vector $\tilde{\mathbf{t}}$ has the form $[x, y, 1]^{\top}$. This means that for each $\tilde{\mathbf{t}}$ that we recover, $-\tilde{\mathbf{t}}$ will also need to be considered as a possible solution.

Once we substitute for $\tilde{E}$ in equation (1), the resulting system of four polynomial equations has the following form. For $i = 1, .., 3$,

$$a_{i1}xs + a_{i2}xc + a_{i3}ys + a_{i4}yc + a_{i5}x - a_{i2}s + a_{i1}c + a_{i6} = 0 \tag{3}$$

$$s^2 + c^2 - 1 = 0. \tag{4}$$

We will refer to these equations as $F = \{f_i(x, y, s, c), i = 1, ..., 4\}$ in the rest of the paper. The coefficients a_{ij} are expressed in terms of image correspondences as follows:

$$\begin{aligned} &a_{i1} = p'_{iy}p_{ix}, \quad &&a_{i2} = -p'_{iy}, \quad &&a_{i3} = -p'_{ix}p_{ix} - 1, \\ &a_{i4} = p'_{ix} - p_{ix}, \quad &&a_{i5} = p_{iy}, \quad &&a_{i6} = -p'_{ix}p_{iy}, \end{aligned} \tag{5}$$

such that $\mathbf{p}_i = [p_x, p_y, 1]^{\top}$ and $\mathbf{p}'_i = [p'_x, p'_y, 1]^{\top}$. 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 $\tilde{\mathbf{t}}$. When the motion of the camera in the z direction (after the rotation by R and R') is extremely small, the parametrization $\tilde{\mathbf{t}} = [x, y, 1]^{\top}$ is numerically unstable. We can deal with this rare instability by formulating and solving a system for the parametrizations $\tilde{\mathbf{t}} = [x, 1, z]^{\top}$ and $\tilde{\mathbf{t}} = [1, y, z]^{\top}$, which can be easily done

using the methods we describe below, but omitted for the purposes of this presentation. However, when used with RANSAC, the different directional correspondences give rise to different $\tilde{\mathbf{t}}$ (though still the same $\mathbf{t}$), making the probability of z being close to zero very small. Thus the reformulations are unnecessary in practice.

In order to find the pose matrices from solutions to the system F, we first recover the rotation as $R_{\mathbf{e}_2} = \exp(\text{atan2}(s,c)\hat{\mathbf{e}}_2)$, and translation as $\tilde{\mathbf{t}} = \pm[x, y, 1]^\top$. Finally, we reconstruct each pose as follows:

$$P = \left[\begin{array}{c|c} S & \mathbf{t} \end{array}\right] = \left[\begin{array}{c|c} R'^\top R_{\mathbf{e}_2} R & R'^\top \tilde{\mathbf{t}} \end{array}\right]. \quad (6)$$

Point triangulation and chirality checks are used to eliminate some of the 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.

4 Closed-form Solution

We hereafter show that the system F 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 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 below.

1) Solve for x and y as a function of c and s using the first two equations in (3). 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 (3). 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 [14]), 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 (3) as linear functions of c and s as follows:

$$\begin{bmatrix} a_{11}s + a_{12}c + a_{15} & a_{13}s + a_{14}c \\ a_{21}s + a_{22}c + a_{25} & a_{23}s + a_{24}c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{12}s - a_{11}c - a_{16} \\ a_{22}s - a_{21}c - a_{26} \end{bmatrix} \quad (7)$$

and solve the above linear system for x and y:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{d} \begin{bmatrix} a_{23}s + a_{24}c & -(a_{13}s + a_{14}c) \\ -(a_{21}s + a_{22}c + a_{25}) & a_{11}s + a_{12}c + a_{15} \end{bmatrix} \cdot \cdot \begin{bmatrix} a_{12}s - a_{11}c - a_{16} \\ a_{22}s - a_{21}c - a_{26} \end{bmatrix}, \quad (8)$$

where the determinant

$$d = (a_{11}s + a_{12}c + a_{15})(a_{23}s + a_{24}c) - \\ - (a_{21}s + a_{22}c + a_{25})(a_{13}s + a_{14}c). \quad (9)$$

Substituting the expression for x and y into the third equation in (3) and multiplying both sides of the equation by d, yields a cubic equation in s and c:

$$g_1 s^3 + g_2 cs^2 + g_1 sc^2 + g_2 c^3 + g_3 s^2 + g_4 sc + g_5 c^2 + g_6 s + g_7 c = 0. \quad (10)$$

The coefficients g_i for $i = 1, ..., 6$ are derived symbolically and are found in Appendix A, equation (14). 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

$$g_1 s + g_2 c + g_3 s^2 + g_4 sc + g_5 c^2 + g_6 s + g_7 c = 0. \quad (11)$$

In the final step, we employ the Sylvester resultant to eliminate one of the two remaining variables from equations (4) and (11). The resultant of the two polynomials is the determinant of the Sylvester matrix

$$\begin{bmatrix} g_3 & g_4 c + g_1 + g_6 & g_5 c^2 + g_2 c + g_7 c & 0 \\ 0 & g_3 & g_4 c + g_1 + g_6 & g_5 c^2 + g_2 c + g_7 c \\ 1 & 0 & c^2 - 1 & 0 \\ 0 & 1 & 0 & c^2 - 1 \end{bmatrix}, \quad (12)$$

which leads to a 4^{th}-order polynomial equation $\sum_{i=0}^{4} h_i c^i = 0$, with coefficients h_i given in Appendix A, equation (15). This shows that in general, the system has four solutions for c. Back-substituting the solutions of c into equation (11), 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 (11) 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 (8) for a total of four solutions. We will describe how to recover the pose matrix from x, y, s and c in Section 3.

5 Action Matrix Solution

In this section, we will present a method for solving the system F via eigendecomposition of an action matrix. The elimination template was constructed using symbolic mathematics software Maple. We present only the steps required for implementation of the method and refer the reader to [3], [15], [16] for details. Below we list the steps required to construct this action matrix and read off the solutions from its eigenvectors.

1) Construct the 21×25, sparse elimination template matrix C using the coefficients a_{ij}, as described in Appendix B.
2) Perform Gaussian elimination with partial pivoting or LU decomposition of C. Let U be the resulting upper trapezoidal matrix.
3) Extract two sub-matrices $A = U_{19..21,19..21}$ and $B = U_{19..21,22..25}$. Compute $D = A^{-1}B$.

4) Construct the 4×4 action matrix M_c as follows: $M_{c1..3,1..4} = D$ and $M_{c4,1..4} = [0, 1, 0, 0]$. The columns of M_c correspond to monomials $[y, c, s, 1]$.
5) Compute the real eigenvectors of M_c, and divide each by its last element.
6) The values of y, c and s in that order are the first three entries of each of the normalized eigenvectors.
7) For each set of values for y, c and s, compute the value for x by solving the Equation (3), which is linear in x.

These values can be once again used to extract the pose matrices, as described in Section 3.

6 Degenerate Configurations

It was pointed out in [12] that the three-plus-one algorithm is not degenerate for collinear world points, except for the line parallel to the translation direction. There exist two additional degenerate configurations.

The first one occurs when all world points lie on the horopter [17], i.e. their projections are the same in the first and second images. This configuration causes the coefficients a_{i4} from (5) to vanish, removing the terms of the form $a_{i4}yc$ from the system. The resulting polynomial system does not generate a zero-dimensional ideal, and thus has an infinite number of solutions. This fact was verified using Maple.

The second degenerate configuration occurs when the determinant d in (8) is zero. When this occurs, the translation $\tilde{\mathbf{t}}$ cannot be estimated from the point correspondences using the equation (8). We can derive the geometric condition that causes the determinant to vanish as follows. After projecting two generic 3D points $\mathbf{x}_i = [X_i, Y_i, Z_i]^\top$ for $i = 1, 2$ into the camera frames, we get $\mathbf{p}_i = \mathbf{x}_i$ and $\mathbf{p}'_i = R_{\mathbf{e}_2}\mathbf{x}_i + \tilde{\mathbf{t}}$. We compute the corresponding coefficients (5), and substitute them into equation (9). After using the fact that $s^2 + c^2 = 1$, the determinant condition becomes

$$(Z_2Y_1 - Z_1Y_2)(cx - s) + (Z_1X_2 - X_1Z_2)y + \\ + (X_1Y_2 - X_2Y_1)(sx + c) = 0, \quad (13)$$

which we can rewrite as $(-R_{\mathbf{e}_2}^\top \tilde{\mathbf{t}})^\top \hat{\mathbf{x}}_1 \mathbf{x}_2 = 0$. This condition means that the second camera's center of projection, expressed in the first camera's coordinate system, is orthogonal to the vector formed by the cross product of the world points. In other words, the degeneracy occurs when the world points are coplanar with the translation vector. This is a more general case than the three points parallel to the translation direction discussed in [12].

7 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. When we estimate the epipolar geometry using image correspondences only, there are two sets of inliers: the set that can be used as a directional correspondences and a set that can be used as a point correspondences. Both inlier ratios have to be taken into account when computing the RANSAC stopping criterion. If the number of distant points is sufficiently large (such as in outdoor scenes), we can realize a significant performance gain with our method since fewer hypotheses will need to be considered [18] due to smaller model size.

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 real roots of the 4th degree polynomial can be extracted in closed form by computing and solving the depressed quartic and two quadratics for a total of about 40 operations and six square roots. The computation of the remaining variables takes additional 144 operations. The main computational step of the action matrix algorithm is Gaussian elimination (LU decomposition) of a 21×25 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 4×4 matrix is done by solving a quartic equation and eigenvectors are extracted with an inverse power iteration, which costs 88 multiplications.

On the other hand, the main computational steps in the classic five-point algorithm [4] are: extraction of the null space of a dense 5×9 matrix, requiring 280 operations, Gauss-Jordan elimination of a dense 10×20 matrix, requiring about 1300 operations, and real root isolation of a 10th degree polynomial, which can be accomplished as eigenvalue decomposition of a 10×10 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. In real experiments, the performance of the C implementation of the closed-form algorithm outperformed a highly optimized implementation of the five-point method, on average, by a factor of 5 ($2.6\mu s$ compared to $13.0\mu s$ on a 3GHz laptop).

8 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. We study both single and double precision arithmetic implementations for the action-matrix and closed-form algorithms, and look for numerical differences between them, as well as the differences between the five-point method and the more constrained three-plus-one method.

The input data was generated as follows. The pose of the first camera was defined to be the identity pose $[I|0]$,

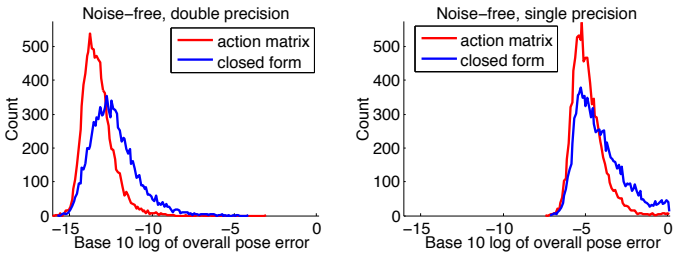


Fig. 1. Distribution of numerical errors in recovered poses for 10^4 random configurations with single and double precision implementations. The error measured is the Frobenius norm of the difference between the true and estimated pose matrices. The median errors for double precision are $3.9 \cdot 10^{-14}$ for the action matrix and $3.1 \cdot 10^{-13}$ for the closed form method. For single precision the errors are $9.3 \cdot 10^{-6}$ and $3.5 \cdot 10^{-5}$, respectively.

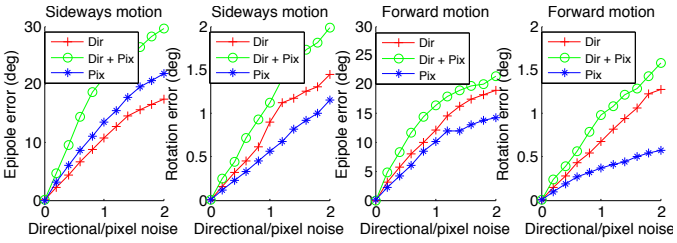


Fig. 2. Median translation and rotation errors for the sideways and forward motion of the baseline camera against noise standard deviation for the double precision implementation of the closed-form method. The directional correspondence noise (Dir) is in degrees, and the image noise (Pix) is in pixels. As with other motion estimation methods, the sideways motion gives significantly worse performance than forward motion on the same data.

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 $\mathbf{t}$ and three Euler angles corresponding to roll, pitch and yaw of the second camera within the limits specified by the experiment. 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.

8.1 Numerical Stability with Noise-free 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. The numerical stability of the closed-form method can be improved by solving the problem three times, using different pairs of points in Equation (7), but the computational cost of computing and then scoring the additional hypothesis makes this impractical, as we saw in Section 7.

8.2 Image and Directional Correspondence Noise

In the rest of this section, we will simulate 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 and directional correspondence 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. Further experiments described in the technical report demonstrate that in these noisy conditions, the performance of the single and double precision action matrix method and the single precision closed-form method are almost identical.

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.

8.3 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 implementations of our closed-form algorithm and the five-point algorithm. Figure 3 demonstrates the effect of the field of view on the algorithms. The rotation estimation is generally better

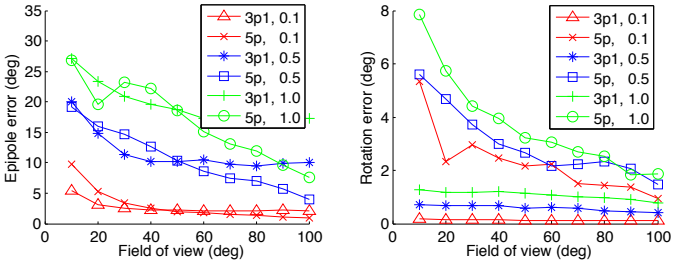


Fig. 3. Median translation and rotation errors for varying fields of view of the baseline camera and random poses. In the legend, the three-plus-one algorithm is labeled "3p1", and the five-point algorithm is "5p". The number after the algorithm name indicates the standard deviation of pixel and directional (for three-plus-one method only) error standard deviations levels in pixels and degrees. The colors also correspond to noise levels: red is 0.1 pixel and 0.1°, red is 0.5 pixel and 0.5°, and green is 1.0 pixel and 1.0°.

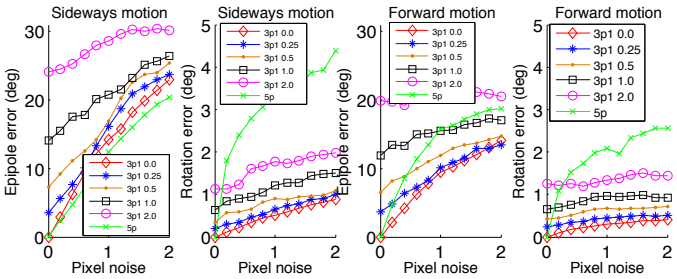


Fig. 4. Comparison of the median errors of the three-plus-one algorithm with the five-point algorithm for the cases of forward and sideways motion for different directional noise levels. In the legend, the three-plus-one algorithm is labeled "3p1", and the five-point algorithm is labeled "5p". The number after algorithm name indicates the standard deviation of the directional noise in degrees. The green sequence with 'x' marker corresponds are the median errors in the five-point algorithm.

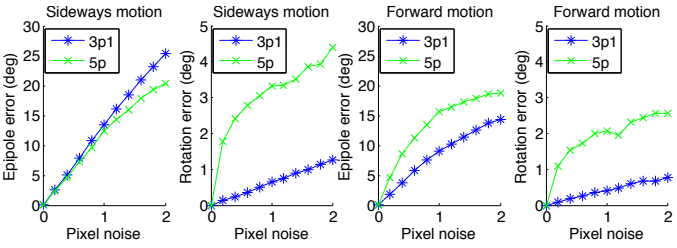


Fig. 5. Comparison of the three-plus-one algorithm with the five-point algorithm in the case when the directional constraints are derived from image points at infinity. The plots show median errors in pose estimation. The green sequences with the 'x' marker show performance of the five-point algorithm and the blue sequences with the '*' marker correspond to the three-plus-one algorithm. The directional correspondences are derived from points at infinity and contaminated with the same pixel noise as the other image points. This graph shows the superior performance of the three-plus-one method in rotation estimation for forward and sideways motion, as well as translation estimation in forward motion.

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 4 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.

In real experiments, we will used our method to compute vision-only relative pose, when points at infinity are present. But first, we compare the performance of the five-point and the three-plus-one methods in this scenario in simulation. The directional correspondence in this case is generated as a projection of a point at infinity, contaminated with noise and made unit-length. The directional correspondence noise can now be measured in pixels, putting the two methods on equal footing. The results are shown in Figure 5. 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.

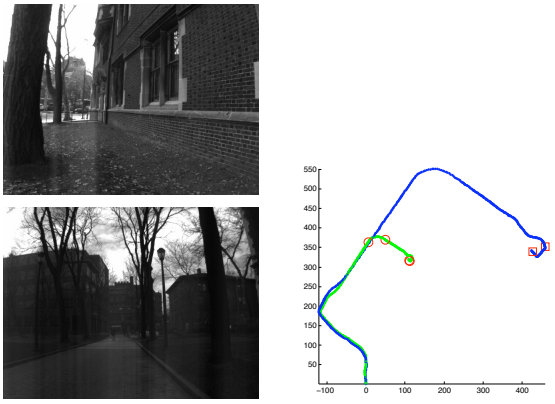
9 Experiments with Real Data

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, 640×480 pixel, black and white camera with a 50° field of view lens to record an 825-frame, outdoor video sequence for comparison with the five-point algorithm (see sample images in Figure 6 (a)). The second data set was recorded with a similar camera from a mobile robot platform and included high accuracy ground truth (see Figure 7).

For both data sets we used the monocular scheme as described in [1]. The experiments consisted of using the correspondences to estimate camera motion with the standard five-point algorithm and the new three-plus-one 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. The directional correspondence was simply a unitized image point correspondence.

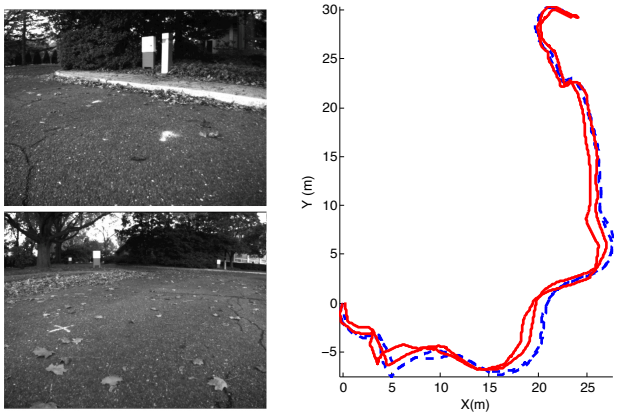
9.1 Structure from Motion Results

In Figure 6 (b) 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



(a) (b)

Fig. 6. (a) Sample images from our hand-held data set. (b) 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. Overall, the poses recovered with each method are very similar up to scale. 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.



(a) (b)

Fig. 7. (a) Sample images from the 2582-frame mobile robot data set. (b) Trajectories obtained using visual odometry with the proposed hypothesis generator and the ground truth collected using a Topcon tracking total station. The three-plus-one visual odometry (solid red) was manually scaled and aligned with the ground truth (dashed blue). The results demonstrate that the algorithm performs correctly in outdoor scenes.

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.

To further demonstrate the real-world performance, we collected a 2582-frame video from a mobile robot, where the position of the robot was tracked with a Topcon tracking total station. The resulting trajectory is plotted in Figure 7.

9.2 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. The results are presented in Figure 8. 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.

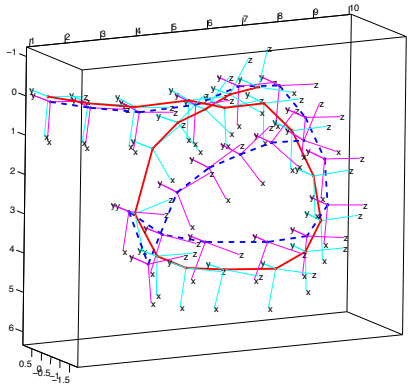


Fig. 8. Camera trajectories from a short segment of an indoor dataset where the reference direction was provided by the IMU. The red solid lines and dashed blue lines connect the centers of projection determined with our method, the five-point method, respectively. The coordinate axes attached to each point show the rig's relative orientation in space. The motion was approximately a loop, produced by hand, while exercising all six degrees of freedom, as seen by the orientation of the coordinate axis.

10 Conclusions

We presented two efficient algorithms to determine relative pose from three image point correspondences and a directional correspondence. From our analysis and experiments we learned that

- The more constrained three-plus-one method does a better job of estimating rotations than the five-point method.
- Both the closed-form and action-matrix implementations are faster than the five-point method, making them attractive for real-time applications.
- The action matrix method yields a solution with better numerical performance than the simpler, closed-form algorithm, but the differences are not signifi-

cant in realistic settings.
- Since the action matrix method can perform better in single precision implementation, it should be considered for processors with 32-bit floating-point arithmetic where extra precision is required.
- When used with RANSAC, smaller minimal data set can lead to improved robustness.
- 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 visual odometry for outdoor scenes with or without aid from an IMU.

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 and speed of visual navigation systems.

ACKNOWLEDGMENTS

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

[1] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," *CVPR 2004*, vol. 1, pp. I–652 – I–659 Vol.1, Jan 2004.
[2] T. Vieville, E. Clergue, and P. Facao, "Computation of ego-motion and structure from visual and inertialsensors using the vertical cue," in *Computer Vision, 1993. Proceedings., Fourth International Conference on*, 1993, pp. 591–598.
[3] M. Byrod, K. Josephson, and K. Åström, "Fast and stable polynomial equation solving and its application to computer vision," *International Journal of Computer Vision*, Jan 2009.
[4] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE PAMI*, Jan 2004.
[5] Z. Kukelova and T. Pajdla, "A minimal solution to the autocalibration of radial distortion," *CVPR 2007*.
[6] M. Byrod, Z. Kukelova, K. Josephson, and T. Pajdla, "Fast and robust numerical solutions to minimal problems for cameras with radial distortion," *CVPR 2008*, Jan 2008.
[7] M. Bujnak, Z. Kukelova, and T. Pajdla, "A general solution to the p4p problem for camera with unknown focal length," *CVPR 2008*, Jan 2008.
[8] H. Stewenius, C. Engels, and D. Nister, "An efficient minimal solution for infinitesimal camera motion," *CVPR*, Jan 2007.
[9] H. Stewenius, "Gröbner basis methods for minimal problems in computer vision," *maths.lth.se*, Jan 2005.
[10] M. Byrod, K. Josephson, and K. Astrom, "Improving numerical accuracy of grobner basis polynomial equation solver," *ICCV 2007*, 2007.
[11] M. Kalantari, A. Hashemi, F. Jung, and J. Guedon, "A new solution to the relative orientation problem using only 3 points and the vertical direction," *arXiv*, vol. cs.CV, May 2009.
[12] F. Fraundorfer, P. Tanskanen, and M. Pollefeys, "A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles," *Computer Vision–ECCV 2010*, pp. 269–282, 2010.
[13] J. Lobo and J. Dias, "Vision and inertial sensor cooperation using gravity as a vertical reference," *IEEE PAMI*, vol. 25, no. 12, pp. 1597–1608, 2003.
[14] D. Cox, J. Little, and D. O'Shea, "Ideals, varieties, and algorithms," *Springer*, Jan 1997.
[15] Z. Kukelova, M. Bujnak, and T. Pajdla, "Automatic generator of minimal problem solvers," *ECCV*, Jan 2008.
[16] D. Cox, J. Little, and D. O'Shea, "Using algebraic geometry," *Springer*, Jan 2005.
[17] M. Armstrong, A. Zisserman, and R. Hartley, "Self-calibration from image triplets," *ECCV'96*, pp. 1–16, 1996.
[18] M. Fischler and R. Bolles, "Random sample consensus," *Communications of the ACM*, Jan 1981.

APPENDIX A
CLOSED-FORM COEFFICIENTS

In this appendix we list the coefficients for the closed-form solution presented in Section 4. The coefficients a_{ij} are found in equation (5). The coefficients g_i in the polynomial (11) are as follows:

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

where a_{ij} come from (5). The coefficients of the quartic polynomial in c are

$$\begin{aligned}
h_0 &= -g_1^2 - 2g_1g_6 - g_6^2 + g_3^2 \\
h_1 &= 2g_3g_2 - 2g_4g_6 + 2g_3g_7 - 2g_4g_1 \\
h_2 &= -g_4^2 + g_1^2 + g_6^2 + g_2^2 + g_7^2 - 2g_3^2 + 2g_1g_6 + 2g_2g_7 + 2g_3g_5 \\
h_3 &= 2g_4g_1 + 2g_4g_6 + 2g_5g_2 + 2g_5g_7 - 2g_3g_2 - 2g_3g_7 \\
h_4 &= g_4^2 + g_5^2 + g_3^2 - 2g_3g_5.
\end{aligned} \tag{15}$$

The quartic equation built from the coefficients h_i yields the solution for c.

APPENDIX B
ELIMINATION TEMPLATE MATRIX

In this appendix we give the structure of the 21×25 coefficient matrix C used in Section 5. The coefficients a_{ij} are found in equation (5). From each of the five vectors listed in (16) we create three rows of C when we substitute the coefficients a_{ij} for $i = 1, 2, 3$. These rows come from the coefficients of equation (3).

The last six rows $C_{16..21,1..25}$ come from the coefficients of equation (4) and are shown in equation (17).

$$
\begin{bmatrix}
a_{i4} & a_{i3} & 0 & 0 & 0 & 0 & a_{i5} & 0 & a_{i1} & 0 & 0 & -a_{i2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{i6} & 0 & 0 & 0 & 0 \\
0 & a_{i4} & a_{i1} & a_{i3} & 0 & 0 & 0 & 0 & 0 & a_{i5} & 0 & a_{i1} & -a_{i2} & 0 & 0 & 0 & a_{i6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{i2} & a_{i4} & a_{i1} & a_{i3} & 0 & 0 & 0 & 0 & 0 & a_{i5} & 0 & 0 & 0 & 0 & 0 & a_{i1} & -a_{i2} & 0 & a_{i6} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & a_{i2} & a_{i4} & 0 & a_{i1} & a_{i3} & 0 & 0 & 0 & a_{i5} & 0 & -a_{i2} & 0 & 0 & 0 & a_{i1} & 0 & 0 & a_{i6} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{i2} & a_{i1} & a_{i3} & 0 & a_{i5} & a_{i4} & 0 & 0 & 0 & a_{i1} & -a_{i2} & a_{i6}
\end{bmatrix} \tag{16}
$$

$$
\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1
\end{bmatrix} \tag{17}
$$