Abstract

We propose a novel formulation of minimal case solutions for determining the relative pose of perspective and generalized cameras given a partially known rotation, namely, a known axis of rotation. An axis of rotation may be easily obtained by detecting vertical vanishing points with computer vision techniques, or with the aid of sensor measurements from a smartphone. Given a known axis of rotation, our algorithms solve for the angle of rotation around the known axis along with the unknown translation. We formulate these relative pose problems as Quadratic Eigenvalue Problems which are very simple to construct. We run several experiments on synthetic and real data to compare our methods to the current state-of-the-art algorithms. Our methods provide several advantages over alternatives methods, including efficiency and accuracy, particularly in the presence of image and sensor noise as is often the case for mobile devices.

1. Introduction

The problem of determining a camera’s position and orientation is a major area of focus in computer vision [7, 18, 19]. Robust solutions to pose problems are of particular interest to the SLAM and Structure from Motion communities where accuracy and efficiency are especially important. This paper focuses on solving pose problems when a partial rotation is known. Specifically, we focus on the case when the axis of rotation is known, which reduces the degrees of freedom in the pose problems by two. This leaves us to solve for four degrees of freedom: an unknown angle of rotation about the known rotation axis and the unknown translation. The angle-axis rotation parameterization is equivalent to other parameterizations such as Euler angles (all rotations may be represented by both parameterizations); however, an axis of rotation is more flexible to use because it allows for any type of partial rotation to be used as a prior instead of only pitch and roll rotations.

A partial rotation in the form of a rotation axis can be accurately determined without knowing the full rotation through a variety of methods [3, 5, 9, 12, 14]. Vertical vanishing points can be detected using computer vision techniques [16] and aligned to the “up” direction (e.g., \([0 1 0]^{\top}\)). After aligning the “up” direction, only the rotation angle about the “up” direction and the translation remain unknown. In mobile robotics, imaging from a fixed tripod, or acquiring panoramas from cars (e.g., Google Street View), the cameras are oriented vertically and it is common to assume that motions are planar. In these cases, the “up” direction corresponds to the vertical image axis. Alternatively, modern
smartphone devices are equipped with inertial measurement unit (IMU) sensors capable of providing partial rotation information with high accuracy (smartphone sensors have an operating noise of about 0.5 degrees while noise of high accuracy sensors is around 0.06 degrees [9]). It is not typically possible to recover a full rotation estimation from IMU sensors because the compass sensor is extremely unreliable [23], so only some of the sensor measurements may be used confidently. However, most phones provide a direct measurement of the gravity direction relative to the phone. This measurement can be used to align cameras with the “down” direction (similarly to the alignment of vanishing points to the “up” direction), and the gravity vector can thus be used as the known axis of rotation (c.f. Fig. 2). Gyroscope measurements can be used in a similar manner to recover a known partial rotation. The ease with which partial rotations can be extracted places an importance on efficient and accurate pose algorithms that utilize all sensor measurements, especially in the areas of computer vision, SLAM, and robotics.

Methods that utilize partial knowledge of the rotation require fewer correspondences, resulting in greater efficiency within a RANSAC scheme [2]. Finding an outlier-free sample in RANSAC depends on the number of parameters needed to form a minimal sample and reducing the required number of correspondences effectively increases the convergence rate of RANSAC exponentially. For an inlier probability of 0.5, for instance, reducing the number of required correspondences from 5 to 3 provides a 2–4x speed increase for convergence. Much work has been done to increase the efficiency and quality of RANSAC-like methods [1, 28], and further speed increases can likely be observed with a more intelligent RANSAC scheme. Reducing the number of correspondences required is a key advantage of our approaches.

In this paper we propose two new methods for estimating relative pose when an axis of rotation is known: a 3-point algorithm for determining the relative pose between two calibrated cameras and a 4-point algorithm for determining the relative pose between two generalized cameras, i.e., cameras that produce image rays that do not contain a central origin. We formulate the relative pose problems as Quadratic Eigenvalue Problems (QEPs) that lead to increased efficiency compared to state-of-the-art algorithms. Further, our formulations are simple and produce constraints that solve directly for relative rotation and translation instead of, for instance, essential matrices. Synthetic experiments show that our methods lead to increased robustness to image and IMU noise compared to alternative methods. Additionally, we demonstrate that our algorithms are well-suited for mobile devices such as smartphones by conducting a localization experiment with real data from an iPhone 4. We have provided efficient C++ implementations of all proposed methods as open-source software within the Theia Multiview Geometry Library [26] 1.

2. Related Work

Our work builds on much previous work in the domain of pose estimation aided by sensor measurements. Recent Structure from Motion work focuses on calculating global rotations for each camera first, then solving for translation [22]. The robotics and computer vision communities have made extensive use of non-visual sensors to aid pose estimation and motion tracking [3, 6, 12, 15]. The proliferation of smartphone devices allows for convenient access to cameras equipped with IMUs. The limited computational power of those devices (and other mobile devices, such as micro air vehicles) stresses the importance of using as much sensory information as is available to limit the computational resources needed for visual tracking.

When an axis of rotation is known, the number of correspondences needed to determine the essential matrix relating two cameras is reduced to 3 instead of the standard 5-point algorithm [18]. Our 3-point algorithm is closely related to that of Fraundorfer et al. [3] which creates a simplified essential matrix composed of three angle-axis rotation matrices that correspond to measurements for yaw, pitch, and roll. The simplified essential matrix is used to solve for the unknown yaw angle and translation (given relative pitch and roll from a gyroscope measurement) by considering the determinant and trace constraints, yielding 4 candidate solutions for the essential matrix (each of which has 4 possible rotation and translations). Our method directly solves for up to 4 relative rotations and translations and yields more stable results under the presence of image noise. More complex formulations also solve for the essential matrix given a

1http://cs.ucsb.edu/~emsweeney/theia
partially known rotation [5, 17] but they are slower and require constructing an action matrix that is 4-10 times larger than the matrices used in our method.

Generalized cameras were first introduced by Grossberg and Nayar [4] and have since become a common camera model for multi-camera and panoramic-camera setups [5, 20]. In particular, generalized cameras have become the standard for multi-camera systems on moving vehicles [12]. Generalized camera models can also be used for localization tasks by using nearest neighbor correspondences from multiple posed images (thus providing multiple image rays with different origins) to determine the query image’s camera pose. Generalized cameras can produce highly stable motion estimates because of their potentially large visual coverage. Lee et al. [13] propose a solution to the 4-point relative pose problem for generalized cameras by using Euler angles to rewrite the generalized epipolar constraint [20] in terms of the one unknown Euler angle and the three unknown translation parameters. By using the hidden variable resultant technique, the yaw angle can be determined from the roots of an eight-degree univariate polynomial and the translation can be easily extracted. Lee et al. [13] also propose a linear 8-point algorithm; however, it is quite sensitive to image and IMU noise.

Kukelova et al. [8] describe how to formulate various relative pose problems as Polynomial Eigenvalue Problems (PEPs), which are a generalization of QEPs. This method is later generalized to provide guidelines for transforming other problems into PEPs that can be solved efficiently [10]. These solutions, however, assume no partial information about the relative pose is available and do not apply to generalized camera models.

3. Estimating Relative Pose with a Known Rotation Axis

In this section we formulate our solutions to the relative pose problem with a known rotation axis. If the transformation from the coordinate system of image 1 to the coordinate system of image 2 is given by a rotation $R$ and a translation $t$, then the standard epipolar constraint relating a normalized point $p_1$ in image 1 to the corresponding normalized image point $p_2$ in image 2 is:

$$p_2 \cdot (t \times Rp_1) = 0,$$  \hspace{1cm} (1)

which can be rewritten as

$$-(p_2 \times Rp_1) \cdot t = 0.$$  \hspace{1cm} (2)

Following Stewénius et al. [25], we use quaternions to represent the rotation. Quaternions are a 4-vector $[v_1 \ v_2 \ v_3 \ s]^T$, which can be written as $[v^T \ s]^T$. When an axis of rotation is known, we set $v$ to the known axis (normalized to unit length), and a scaled version of the rotation can be written as:

$$R \sim 2(vv^T + sv) + (s^2 - 1)I,$$  \hspace{1cm} (3)

where $[v \times]$ is the skew-symmetric cross product matrix. Subsequent equations are homogeneous in $R$, so the norm of $R$ does not affect the results of our algorithms. This parameterization of the rotation allows for very simple formulations for the 3 and 4-point relative pose problems.

3.1. A 3-Point algorithm for the essential matrix

By substituting the parameterization of the rotation equation (3) into the epipolar constraint (2), we have

$$-(p_2 \times (2(vv^T + sv) + (s^2 - 1)I)p_1) \cdot t = 0.$$  \hspace{1cm} (4)

By expanding Eq. (4), and collecting coefficient vectors of $s$, the epipolar constraint can be rewritten as

$$0 = ((p_2 \times (p_2 \times v))s^2 + (2p_2 \times v \times p_1)s + 2v \times p_1)(p_2 \times v) - p_2 \times p_1 \cdot t.$$  \hspace{1cm} (5)

Each correspondence gives a constraint in this form. We can stack the minimal three correspondences to obtain a $3 \times 3$ constraint matrix that is a function of $s$:

$$(s^2M + sC + K) \cdot t = 0.$$  \hspace{1cm} (6)

This is a Quadratic Eigenvalue Problem, a well-studied problem in linear algebra [27]. The QEP of Eq. (6) can be solved by first converting it to a Generalized Eigenvalue Problem of the form

$$[C \ K] z = s [-M \ 0 \ 0 ; 0 \ 0 \ -I] z,$$  \hspace{1cm} (7)

where $z = [st^T \ t^T]^T$ is the eigenvector and $s$ is the eigenvalue. This can be converted to a standard eigenvalue problem of the form $Av = \lambda v$ by multiplying the left-hand side of Eq. (7) with the inverse of the right-hand matrix. In our case, the inverse is particularly simple:

$$[-M \ 0 ; 0 \ -I]^{-1} = [-M^{-1} \ 0 ; 0 \ -I].$$

Inversion of a small matrix $(3 \times 3$ in this case, or $4 \times 4$ for the 4-point algorithm described in Section 3.2) is generally very efficient, and we have not found it to be prohibitive in our implementation. The Generalized Eigenvalue Problem of Eq. (7) is thus reduced to a standard eigenvalue problem,

$$[-M^{-1}C \ -M^{-1}K ; I \ 0] z = sz,$$

which can be solved with standard methods. The solution to this produces 6 eigenvalues corresponding to solutions for
3.2. A 4-Point algorithm for the generalized essential matrix

The generalized epipolar constraint between two images maps a set of rays in one coordinate system to a set of rays in a second coordinate system [20]. We represent image rays in Plücker coordinates:

\[ L = \begin{bmatrix} p \\ q \end{bmatrix}, \tag{8} \]

where \( p \) and \( q \) are 3-vectors formed from image ray origins and unit directions. For a thorough review of Plücker coordinates, we refer the reader to [20] or [21]. By using Plücker coordinates, we are able to derive a similar set of equations for the epipolar constraint as the 3-point algorithm. For two corresponding image points in Plücker coordinates \( L_1 \) and \( L_2 \), the generalized epipolar constraint is:

\[ p_2 \cdot Rq_1 + p_1 \cdot R^{-1}q_2 - (p_2 \times Rp_1) \cdot t = 0, \tag{9} \]

which can be written in terms of the dot product of two 4-vectors:

\[ (-p_2 \times Rp_1, p_2 \cdot Rq_1 + p_1 \cdot R^{-1}q_2) \cdot \dot{t} = 0, \tag{10} \]

where \( \dot{t} \) is the 4-vector: \( [t^\top \ 1]^\top \).

In a similar method as the 3-point pose algorithm, \( R \) can be expanded into \( s \) and the known rotation axis \( v \), and the constraint can be written as a quadratic of \( s \). Using the minimal four correspondences, a \( 4 \times 4 \) constraint matrix is created:

\[ (s^2 M + sC + K) \cdot \dot{t} = 0. \tag{11} \]

This QEP can be solved the same way as the 3-point algorithm. Unlike the QEP for the 3-point algorithm, scale can be recovered for the translation by dividing the extracted 4-vector by the homogeneous component \( w \). 8 eigenvalues can be extracted, of which 6 are real, leading to 6 candidate solutions for the rotation and translation. The generalized essential matrix from Pless [20] is then:

\[ G = \begin{bmatrix} [t]_\times R & R \\ R & 0 \end{bmatrix}. \tag{12} \]

3.3. An efficient solution to the QEP

In order to produce non-trivial solutions, the constraint matrices of Eqs. (6) and (11) must be singular and thus the determinant must be zero:

\[ \det(s^2 M + sC + K) = 0. \tag{13} \]

Expanding this constraint for the 3-point algorithm gives a six-degree polynomial in \( s \), which, when created with the coefficients from Eq. (5), is always divisible by \( s^2 + 1 \), leading to a quartic polynomial that can be solved in closed form. For each real root found, the constraint matrix can be created according to Eq. (6) and the corresponding null vector \( t \) can be extracted using standard methods.

For the 4-point algorithm, the determinant from Eq. (11) results in an eight-degree polynomial in \( s \), the unknown rotation angle. Expanding the singularity constraint yields an equation that is divisible by \( s^2 + 1 \), leading to a six-degree polynomial with a maximum of 6 solutions. This solution is similar to the method of Lee et al. [12] that also reduces the problem to an eight-degree polynomial. However, they parameterize the univariate polynomial as a function of the tangent of an unknown Euler angle which is not as robust to image and IMU noise as we will demonstrate in Section 4.

4. Experiments

We conducted several experiments using synthetic and real data to measure the performance of our algorithms. We first tested the numerical stability of our solutions over \( 10^5 \) noise-free trials. In each trial, scenes were generated using 3D points randomly distributed in a 3D cube. The points were then projected by cameras with random feasible orientations and positions to establish correspondences. We measured the angular rotation error and the translation error for our 3 and 4-point algorithms as well as the 3-point algorithm of [3] and the 4-point algorithm of [13]. For all algorithms, the errors were less than machine precision, so the errors cannot be displayed properly here. These findings are consistent with [9] and [8].

In the following sections we test the performance of our algorithms compared to state-of-the-art methods under increasing levels of pixel noise and increasing levels of IMU noise, respectively. An analysis of degenerate scenarios follows. We then demonstrate the practical use of our algorithms on a real dataset composed of image and IMU measurements from an iPhone 4 with ground truth poses available for comparison. Finally, we give a runtime analysis of each of our algorithms. For all experiments, we compare our algorithms to the state-of-the-art 3-point algorithm by Fraundorfer et al. [3] and the 4-point algorithm by Lee et al. [13]. All algorithms were implemented in C++. We use our own implementations of [13] and [3] that were ported from Matlab implementations provided by the authors.
4.1. Robustness to image noise

The high numerical stability of the 3 and 4-point algorithms means that we should expect the performance of our minimal solvers under noise to be similar on average to the state-of-the-art algorithms. Indeed, we found this to be the case. However, the different rotation parameterizations (i.e., Euler angles instead of quaternions) and the different solution methods lead to slightly different distributions of errors. To best demonstrate how the distributions of errors affect the performance of our algorithms, we use each algorithm in a RANSAC [2] scheme. This is more indicative of the performance that can be expected when using the algorithms in real-world applications.

To measure the accuracy and robustness of our methods, we performed an experiment on synthetic data with Gaussian pixel noise ranging from 0 to 5 pixels. We performed 1,000 trials (running RANSAC once per trial) at each level of pixel noise. For each trial, we generate relative pose with a rotation in the range of [-30, 30] degrees rotated on each axis, and a translation in a random direction with a baseline length of 1 unit. We generate 1000 random 3D points with a depth between 2 and 6 units, and reproject them into each camera to establish correspondences that are randomly sampled in RANSAC. For the 4-point algorithms, we use the same randomly feasible multi-camera setup consisting of 4 cameras for each generalized camera.

The results are plotted in Figure 3. For the 3-point algorithms, we measure the translation error as the angular error because translation is only defined up to scale. All of our algorithms have performance that is slightly better than alternative algorithms for all levels of image noise. It should be noted, though, that the errors for all algorithms are small for all levels of image noise. Robustness to image noise indicates our algorithms are suitable for use on mobile devices, which often have low-resolution cameras with large pixel noise.

4.2. Robustness to IMU noise

In real scenarios, estimating the rotation axis is an imperfect process. To measure the robustness of our algorithms to errors in the rotation axis, we simulate noisy IMU measurements corresponding to three gyroscopes and determined an axis of rotation from two of those measurements. We increased the level of IMU noise within the range of expected operating noise of sensors found on common devices today (smartphone IMUs typically have noise of about 0.5 degrees). We use the same camera and point configuration as the image noise experiment and plot the mean errors from RANSAC over 1000 trials at each noise level. All trials were run with 0.5 pixel noise.

As shown in Figure 4, our algorithms perform better than state-of-the-art algorithms for all levels of IMU noise. This is due in part to our chosen representation of the rotation. The known axis of rotation is generally not orthogonal to the noise in the rotation so the optimal solution angle about the rotation axis is able to minimize the effect of the noise on the solution rotation. The performance under IMU noise indicates that our algorithms are well-suited for mobile devices and other common cameras that are equipped with IMU sensors within the common range of operating noise (less than 1 degree).

4.3. Degenerate cases

Our 3-point algorithm is able to handle some cases that are degenerate for the standard 5-point algorithm [18] such as coplanar points. Our 3-point algorithm can also handle colinear points similar to [3] unless the line is parallel to the camera translation. The 5-point algorithm is known to be unstable for forward motion, but our algorithm is able to handle this motion without issue. Additionally, we compute rotation and translation for the 3 and 4-point algorithms directly, and our formulation allows for more stable computation of translation when there is no rotation. While this is not a degenerate configuration, it is one that is often unstable for standard methods.

The 4-point algorithm additionally suffers from two well-known degeneracies for generalized cameras. First, for generalized cameras with very close camera-to-camera distances (leading to near image ray origins), the 4-point algorithm is unstable. The second degenerate case occurs...
when all cameras in a multi-camera setup lie along a line and all points are observed by the same camera in the first and second observations [25]. Then the solution set for the generalized essential matrix consists of all solutions along a particular 1D line in space; however, it is straightforward to detect this scenario [24, 25].

4.4. Real data with an Apple iPhone 4

We demonstrate that our algorithms are suitable for current mobile devices by conducting experiments with real image and sensory data from the Metaio Outdoor Dataset² [11]. This dataset consists of nearly 800 images and corresponding sensor measurements from an iPhone 4 with 6 degree of freedom ground truth poses. Some example images from this dataset are shown in Figure 5. We obtain the gravity vector with measurements taken directly from the iPhone 4, and align that vector to the vector in world coordinates: \([0\ 0\ 1]\). After aligning each image such that the gravity direction corresponds to \([0\ -1\ 0]\)^T, we use this vector as the axis of known rotation for our algorithms. SIFT features were extracted and matched between images to form feature correspondences. For the 3-point algorithm, we chose the image with the highest number of correspondences and used 2D-2D correspondences inside a RANSAC loop to determine the essential matrix. For the 4-point algorithm we treat the database images as a single generalized camera with ray origins and directions corresponding to the ground truth camera and feature positions. We create a second generalized camera from two images that are not in the database, and localize this camera with respect to the database generalized camera in a RANSAC loop. We only consider symmetric feature matches that pass the ratio test.

Table 1 shows the results for 100 images localized. All algorithms were able to successfully localize all images. Our algorithms gave comparable or better accuracy to the state-of-the-art solutions, while performing more efficiently. The speedup of 1.2-2.1× is due to our faster pose solvers as well as the fewer number of candidate solutions.

Table 2: Our methods produce fewer candidate solutions compared to alternative methods, resulting in better performance in RANSAC. We compare the number of total iterations required for RANSAC to obtain an inlier sample with 0.99 confidence when assuming an inlier ratio of 0.5.

4.5. Computational complexity

Our algorithms are solved as QEPs, for which efficient solvers exist. Our quaternion representation avoids costly trigonometric functions unlike other approaches that rely on Euler angles. Over \(10^5\) trials, the 3-point algorithm of [3] had a mean execution time of 21.6 \(\mu s\) compared to 16.7 \(\mu s\) for our QEP method and 12.3 \(\mu s\) for our fast method described in Sec. 3.3. The 4-point algorithm of [13] required 27.5 \(\mu s\) compared to 26.2 \(\mu s\) for our QEP method and 23.7 \(\mu s\) for our fast method.

Table 2: We performed localization using the 3 and 4-point algorithms with image and sensor data from an iPhone 4. The mean position error and median localization timings for 100 images is given. Our algorithms provide comparable or better accuracy compared to state-of-the-art methods while performing up to 2× faster.

Table 1: We performed localization using the 3 and 4-point algorithms with image and sensor data from an iPhone 4. The mean position error and median localization timings for 100 images is given. Our algorithms provide comparable or better accuracy compared to state-of-the-art methods while performing up to 2× faster.

4.4. Real data with an Apple iPhone 4

We demonstrate that our algorithms are suitable for current mobile devices by conducting experiments with real image and sensory data from the Metaio Outdoor Dataset² [11]. This dataset consists of nearly 800 images and corresponding sensor measurements from an iPhone 4 with 6 degree of freedom ground truth poses. Some example images from this dataset are shown in Figure 5. We obtain the gravity vector with measurements taken directly from the iPhone 4, and align that vector to the vector in world coordinates: \([0\ -1\ 0]\)^T. After aligning each image such that the gravity direction corresponds to \([0\ -1\ 0]\)^T, we use this vector as the axis of known rotation for our algorithms. SIFT features were extracted and matched between images to form feature correspondences. For the 3-point algorithm, we chose the image with the highest number of correspondences and used 2D-2D correspondences inside a RANSAC loop to determine the essential matrix. For the 4-point algorithm we treat the database images as a single generalized camera with ray origins and directions corresponding to the ground truth camera and feature positions. We create a second generalized camera from two images that are not in the database, and localize this camera with respect to the database generalized camera in a RANSAC loop. We only consider symmetric feature matches that pass the ratio test.

Table 1 shows the results for 100 images localized. All algorithms were able to successfully localize all images. Our algorithms gave comparable or better accuracy to the state-of-the-art solutions, while performing more efficiently. The speedup of 1.2-2.1× is due to our faster pose solvers as well as the fewer number of candidate solutions.

Table 2: Our methods produce fewer candidate solutions compared to alternative methods, resulting in better performance in RANSAC. We compare the number of total iterations required for RANSAC to obtain an all-inlier sample with 0.99 confidence when assuming an inlier ratio of 0.5.

4.5. Computational complexity

Our algorithms are solved as QEPs, for which efficient solvers exist. Our quaternion representation avoids costly trigonometric functions unlike other approaches that rely on Euler angles. Over \(10^5\) trials, the 3-point algorithm of [3] had a mean execution time of 21.6 \(\mu s\) compared to 16.7 \(\mu s\) for our QEP method and 12.3 \(\mu s\) for our fast method described in Sec. 3.3. The 4-point algorithm of [13] required 27.5 \(\mu s\) compared to 26.2 \(\mu s\) for our QEP method and 23.7 \(\mu s\) for our fast method.

²http://www.metaio.com/research
Additionally, our algorithms produce fewer solutions than the alternative state-of-the-art algorithms. Fraudorfer et al.’s [3] method produces 4 solutions for essential matrices. Each of these essential matrices must be decomposed into 4 relative rotations and translations, yielding 16 possible relative poses in total compared to the 4 relative poses that our 3-point algorithm produces. Similarly, the method of [13] “gives up to 8 real solutions” while our method only produces up to 6 real solutions. The reduced number of possible solutions helps limit the time required for RANSAC operations (c.f. Table 2) which makes the speed gains in practical applications noteworthy.

5. Conclusion

In this paper, we have presented two new algorithms for determining the relative pose of a camera with a known rotation axis: a 3-point algorithm for central cameras, and a 4-point algorithm for generalized cameras. With more devices being equipped with IMUs and gravity sensors, these algorithms are of increasing relevance. By formulating rotations as quaternions, the pose constraints lead to simple and extremely efficient algorithms. We solve the relative pose problems as QEPs that are more efficient than alternative algorithms. Additionally, our QEP methods slightly more robust to image noise and produce high accuracy results in the presence of IMU noise. These algorithms are particularly suited for use in RANSAC schemes, as the fewer parameters leads to fewer samples needed for convergence. Given the accuracy, simplicity, and efficiency of these methods, our methods are extremely well-suited and practical for use on mobile devices.

Acknowledgements

We would like to thank Petri Tanskanen and Gim Hee Lee for providing source code for alternative methods as well as Henrik Stewénius) for his insightful feedback. This work was supported in part by NSF Grant IIS-1219261 and NSF Graduate Research Fellowship Grant DGE-1144085.

References


