2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

# A 4-Point Algorithm for Relative Pose Estimation of a Calibrated Camera with a Known Relative Rotation Angle

Bo Li, Lionel Heng, Gim Hee Lee and Marc Pollefeys
Computer Vision and Geometry Group, ETH Zürich

*Abstract*— **We propose an algorithm to estimate the relative camera pose using four feature correspondences and one relative rotation angle measurement. The algorithm can be used for relative pose estimation of a rigid body equipped with a camera and a relative rotation angle sensor which can be either an odometer, an IMU or a GPS/INS system. This algorithm exploits the fact that the relative rotation angles of both the camera and relative rotation angle sensor are the same as the camera and sensor are rigidly mounted to a rigid body. Therefore, knowledge of the extrinsic calibration between the camera and sensor is not required. We carry out a quantitative comparison of our algorithm with the well-known 5-point and 1-point algorithms, and show that our algorithm exhibits the highest level of accuracy.**

## I. Introduction

Vehicle platforms equipped with a camera and either an odometer, IMU, or yaw rate sensor have been widely used in the areas of computer vision, robotics, and automatic control. For years, research has focused on using these low-cost sensors to localize the vehicle as well as reconstruct the vehicle's environment. This research is also referred to as visual SLAM in robotics.

One key step of visual SLAM is to estimate the relative camera pose between each frame pair. One commonly used method is feature-based estimation in which a subset of image feature correspondences is selected to estimate the fundamental matrix or essential matrix between two frames. The relative rotation and translation can then be extracted from the matrix. A series of "n-point" (n-correspondences) algorithms has been proposed for this objective. If the camera has unknown intrinsics, the fundamental matrix can be estimated by the 8-point algorithm or 7-point [1] algorithm. If the camera has calibrated intrinsics, either the 6-point [2], [3] algorithm or 5-point [4], [5], [6] algorithm can be used to compute the essential matrix. Based on these algorithms, robust estimation methods such as RANSAC or LMedS are used to generate the best estimate from a set of point correspondences containing both inliers and outliers. The performance of "n-point" algorithms is significantly affected by the quality of the feature correspondences detected from images. It is well-known that an algorithm using fewer point correspondences requires fewer iterations for robust estimation. For the case of a calibrated camera, the minimal solution requires 5 point correspondences to solve for the 5-DoF relative pose. In [4], [5], the 5-point algorithm shows the best estimation performance compared to the 6-point, 7-point, and 8-point algorithms. For visual SLAM and structure-from-motion problems, the 5-point algorithm is the most commonly used algorithm for a calibrated camera.

However, the 5-point algorithm is not always guaranteed to have a stable estimation. On a vehicle with a forward-looking camera, relative pose estimation is more difficult for the 5-point algorithm compared to a side-looking camera. The main difficulties are the lower stability of the algorithm for forward motion compared to sideways motion, and the larger depth of features seen by a front camera which makes the estimation less accurate. Readers can find some discussion about the performance of the 5-point algorithm in [4], [5]. To improve the estimation accuracy, research has focused on exploiting extra information from other sensors or from specific motion models. For example, [7] obtains two rotation angles from the IMU, and uses a 3-point algorithm to estimate the relative pose of a micro aerial vehicle. This method requires that the extrinsic calibration between the camera and the IMU is known. In [8], a novel 1-point algorithm is proposed. The algorithm assumes that the vehicle follows the general Ackermann steering model. The 1-point algorithm can compute stable relative pose estimates very quickly; however, it requires the camera to be located along the rear axis of the vehicle. In [9], another 3-point algorithm is proposed for relative pose estimation. This method uses a generalized camera but only applies to 3-DoF planar motion.

If the vehicle platform has either an odometer or GPS/INS system, and its pose with respect to the camera's frame is known, we can directly obtain the camera's relative pose from the odometer pose $H_o$ as $H^{-1}H_oH$, where $H$ is the transform between the camera and odometer frames. However, in practice, estimating $H$ is not easy. This problem is known as the hand-eye calibration [10], [11]. Moreover, for accurate estimation of relative transforms, hand-eye calibration algorithms require accurate visual odometry estimation. This visual odometry estimation also requires feature-based relative pose estimation algorithms such as the 5, 6, 7, and 8-point algorithms.

Our approach is similar in spirit to [7], [8], [9]; in this paper, we propose an algorithm to improve the relative pose estimation using relative rotation angle measurements. The algorithm uses four feature point correspondences found from an image pair and one rotation angle from any relative rotation sensor such as an odometer, IMU, or GPS/INS. In the algorithm, the camera can be mounted anywhere on the platform; the advantage is that no extrinsic calibration is required. Since the rotation angle sensor readings are very stable and accurate in general, the proposed algorithm significantly improves the accuracy of relative pose estimates

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

compared with existing methods.

The rest of the paper is organized as follows. Section II establishes notations and formulas used in the proposed method. Section III presents the formulation of the 4-point relative pose problem, and section IV presents two algorithms to solve the problem. The performance of the algorithm is studied in section V; we use simulations to compare our results with those from the 5-point algorithm, and quantify the algorithm's improvement over the 5-point algorithm. Furthermore, the algorithm is compared with the 5-point and 1-point algorithms on two real-world datasets obtained with our vehicle platform.

## II. Preliminaries

Image points from the first and second frames are denoted by homogeneous vectors $p_1 = (x_1, y_1, 1)^\top$ and $p_2 = (x_2, y_2, 1)^\top$ respectively. The intrinsic matrix of the camera is denoted as $K$. Since the proposed algorithm requires $K$ to be known, we hereby assume that $p_1$ and $p_2$ are always premultiplied by $K^{-1}$.

Denote $R$ and $t$ as the relative rotation and translation between the first and second frame. The essential matrix corresponding to $R$ and $t$ can be denoted as

$$E = [t]_\times R \tag{1}$$

where $[t]_\times$ denotes the skew symmetric matrix:

$$[t]_\times \equiv \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix} \tag{2}$$

An ideal image point correspondence $(p_1, p_2)$ satisfies the constraint:

$$p_2{}^\top E p_1 = 0 \tag{3}$$

**Rodrigues' rotation formula** Given a 3D unit rotation axis vector $r = (r_x, r_y, r_z)^\top$ and a rotation angle $\theta$, it is easy to find the corresponding rotation matrix using Rodrigues' rotation formula.

$$R(\theta, r) = (\cos\theta)I + (1 - \cos\theta)rr^\top + (\sin\theta)[r]_\times \tag{4}$$

where $I$ is a $3 \times 3$ identity matrix.

**Theorem** The relative rotation angle of the camera and that of the relative rotation sensor are equal.

This is a known fact for rigid motion. Denote the relative motion of the rotation sensor as $R_s$ and $t_s$, and the transform between the camera and the sensor as $R$ and $t$. We know that the camera rotation can be denoted as $R_c = R^{-1}R_sR$, which is independent of the translation. Here, we use the quaternion representation to provide a simple proof to show that the rotation angle of $R_c$ and $R_s$ is the same. Denote $q_s$, $q_c$, and $q$ as the corresponding quaternions respectively. We have

$$\begin{aligned} q_c =& q^{-1}q_s q \\ =& \left(\cos\frac{\theta}{2} - (xi + yj + zk)\sin\frac{\theta}{2}\right) \cdot \\ & \left(\cos\frac{\theta_s}{2} + (x_s i + y_s j + z_s k)\sin\frac{\theta_s}{2}\right) \cdot \\ & \left(\cos\frac{\theta}{2} + (xi + yj + zk)\sin\frac{\theta}{2}\right) \end{aligned} \tag{5}$$

where $\theta$ and $(x, y, z)^\top$ is the rotation angle and rotation axis of a quaternion $q$. Consider the real part of $q_c$, as $q_c^{\text{real}}$. By some simple deduction, we can obtain

$$q_c^{\text{real}} = \cos\frac{\theta_s}{2} \tag{6}$$

Since $q_c^{\text{real}} = \cos\frac{\theta_c}{2}$ by definition, the relative rotation angles $\theta_c$ and $\theta_s$ are equal. This means that the relative rotation angle reading from the sensor can be directly used as the relative rotation angle of the camera without knowing the extrinsics.

## III. Problem Formulation

Substituting (4) into (1), we express the essential matrix as a function of the rotation angle, rotation axis and translation:

$$E(\theta, r, t) = [t]_\times \left(\cos\theta I + (1 - \cos\theta)rr^\top + \sin\theta[r]_\times\right) \tag{7}$$

where $r$ is a 3D unit vector and $t$ is assumed to have unit norm since it is up to scale. With the assumption that we know $\theta$ from the sensor reading, the number of DoFs for the relative camera pose is reduced from 5 to 4. By using 4 image point correspondences, we can solve for $r$ and $t$ for the minimal case.

Thus, we form the equation system for solving for the relative camera pose:

$$p_2^{i\top} E(\theta, r, t) p_1^i = 0 \quad \text{for } i = 1, 2, 3, 4 \tag{8}$$

$$||r||^2 = 1 \tag{9}$$

$$||t||^2 = 1 \tag{10}$$

where $r = (r_x, r_y, r_z)^\top$ and $t = (t_x, t_y, t_z)^\top$ are six unknowns.

## IV. Solution

Solving polynomial systems for minimal problems in computer vision has become a focus of recent research. However, for an equation system with a high degree and many unknowns, it is often difficult to obtain an efficient closed-form solution. The equation system (8) includes 4 cubic polynomials with the highest monomial in the form of $t_\star r_\star r_\star$, where $\star$ denotes any arrangement of '$x$', '$y$', '$z$'. (9) and (10) are two quadratic polynomials. Compared with the minimal problem for both the 5 and 6-point algorithms, the above system has more variables and a higher degree, both of which makes it more difficult to solve. In this section, we propose two different solvers for our equation system. The first one is a closed-form solution based on the Groebner basis. The second one is an efficient numerical solution.

### A. Closed-Form Solver

The Groebner basis provides a useful technique for solving general polynomial systems. In this paper, we use an automatic solver generator [12] to generate the Groebner basis solver for our problem. This generator works by first generating a series of polynomials from the original problem. Their coefficients are denoted in a coefficient matrix. Next, the coefficient matrix is eliminated. An action matrix can then be formed using elements of the elimination result. The eigenvectors of the action matrix consist of solutions to the original problem. Taking the 5-point algorithm for example, this involves a $10\times20$ coefficient matrix and a $10\times10$ action matrix. This is similar to the solver used in [5].

We first simplify our problem by replacing (10) with $t_z = 1$. This is easy to understand since $t$ is only defined up to scale. Thus, we remove one unknown and only have five unknowns $r = (r_x, r_y, r_z)^\top$ and $t = (t_x, t_y, 1)^\top$ and 5 equations (8, 9) to solve. Note that this simplification may cause numerical failure if $t_z$ is extremely small comparing with $t_x$ and $t_y$. In practice, this simplification does not cause a numerical failure because even if the camera is moving on the $xy$ plane, $t_z$ always has some small deviation from zero, which is sufficient for the solver to work stably. Directly solving the system without this simplification can avoid the problem of numerical failure, but makes the solution much more complex.

Inputting the simplified equation systems to the automatic solver generator, we obtain a coefficient matrix of size $270\times290$. The action matrix is of size $20\times20$ which implies the problem has 20 complex roots. We use all the real roots as possible solutions for $r$ and $t$.

This closed-form solver is elegant and easy to use. However, we point out two drawbacks. Firstly, decomposing or eliminating a coefficient matrix of size $270\times290$ can be computationally expensive. Secondly, in some extreme cases, the correct root may not be a real number due to data noise. This can be illustrated by the following small example. Consider the problem: $(x-1)^2 = 0.01$ where the roots are $x = 0.9$ and $x = 1.1$. In the case of noisy data, there is a case where we solve $(x-1)^2 = -0.01$. Then, the root will be $x = 1 \pm 0.1i$, which is not a real number. By taking the real part of the root, we can approximately obtain a real solution. However, for more complicated polynomial systems, this sometimes can lead to a large deviation of the estimated result from the actual result.

### B. Numerical Solver

To avoid the drawbacks of the Groebner basis solver, we hereby propose a numerical solver for our problem using the gradient descent method. This solver can quickly solve the problem and obtain real roots.

We reformulate equation (8) as:

$$f_1^i(r_x,r_y,r_z)t_x + f_2^i(r_x,r_y,r_z)t_y + f_3^i(r_x,r_y,r_z)t_z = 0 \quad \text{for } i = 1,2,3,4 \quad (11)$$

where $f_\star^i$ is a polynomial whose terms include $r_x$, $r_y$ and $r_z$. We can stack $f_\star^i$ as a matrix:

$$F(r_x,r_y,r_z)t \equiv \begin{pmatrix} f_1^1 & f_2^1 & f_3^1 \\ f_1^2 & f_2^2 & f_3^2 \\ f_1^3 & f_2^3 & f_3^3 \\ f_1^4 & f_2^4 & f_3^4 \end{pmatrix} \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = 0 \quad (12)$$

Since we know that $t$ is up to scale, the rank of $F$ must be 2. This means that the determinant of all $3\times3$ submatrices must be 0. This is equivalent to the following equations.

$$\begin{vmatrix} f_1^1 & f_2^1 & f_3^1 \\ f_1^2 & f_2^2 & f_3^2 \\ f_1^3 & f_2^3 & f_3^3 \end{vmatrix} = 0 \quad (13)$$

$$\begin{vmatrix} f_1^2 & f_2^2 & f_3^2 \\ f_1^3 & f_2^3 & f_3^3 \\ f_1^4 & f_2^4 & f_3^4 \end{vmatrix} = 0 \quad (14)$$

Combining (13), (14) and (9), we have a new equation system that has three unknowns: $r_x$, $r_y$, $r_z$. (13) and (14) are of degree 5. Since (10) implies that $(r_x, r_y, r_z)^\top$ is on the surface of a unit sphere, we parameterize $(r_x, r_y, r_z)^\top$ in 2D space and use the gradient descent method to find the roots from a set of initial guesses. This gradient descent-based method is inspired by [13], which uses a similar method for solving the 5-point algorithm. Our initial guesses are drawn from uniform sampling on the surface of a unit sphere. Note that for the minimal case, there is a small chance that the correct root is missed in the gradient descent method as mentioned in [13]. We can minimize the probability that convergence fails by increasing the number of initial guesses. For our implementation, we find that 100 samples are sufficient for convergence to the correct solution for $r$. We found out from doing 10000 simulations that the probability of missing the correct root is less than 0.001. In addition, when the solver is embedded in a robust estimation framework, for example, RANSAC, a failure to converge can entirely be avoided due to the multiple iterations in robust estimation. In our simulation tests, we did not encounter a convergence failure.

After $r_x$, $r_y$, $r_z$ are solved, $F$ is then obtained and $t$ is the null vector of $F$. Note that multiple roots $r$ exist for the equation system, and for each possible solution $(r, t)$, $(r, -t)$ is a possible solution too.

### C. Robust Estimation

Similarly in the cases of the 5, 6, 7, and 8-point algorithms, a robust estimation framework can be used for the 4-point algorithm to find the optimal relative pose from a set of both inlier and outlier point correspondences. Taking the widely-used robust estimation framework RANSAC as an example, in each iteration, 4 points are randomly sampled to generate hypotheses for the relative camera pose estimate. From the above discussion, we know that multiple relative pose hypotheses may exist. By checking the reprojection error such as the Sampson error for the whole point set, candidate solutions are rejected until only two candidate solutions $(r, t)$ and $(r, -t)$ remain. One of these two candidate solutions

is rejected by checking if the reconstructed 3D points have positive depth. This is also called the cheirality check in [4]. In contrast to the 5-point algorithm, the 4-point algorithm requires a lower number of iterations to achieve the same confidence level. Consider a point set with $w = 50\%$ inliers; in each iteration, the probability of selecting 4 inliers is $w^4 = 6.25\%$ while the probability of selecting 5 inliers is $w^5 = 3.125\%$. The number of iterations required for RANSAC is $\frac{\log 1-p}{\log 1-w^n}$ where $n$ is the number of point correspondences and $p$ is the confidence level. To get an estimate with a confidence level of $p = 0.99$, the 4-point algorithm requires 71 iterations while the 5-point algorithm requires 145 iterations.

## V. Experiments

### A. Implementation Details and Timing Issues

We implemented both the Groebner basis solver and numerical solver for our 4-point algorithm. In the Groebner basis solver, we use the sparse QR decomposition implementation from the Eigen[1] library to eliminate the coefficient matrix. In the numerical solver, we use Powell's hybrid method from the GNU General Scientific Library (GSL)[2]. Powell's hybrid method retains the fast convergence of Newton's method but is more reliable. We also implemented a 5-point algorithm based on the solver in [4]. The 5-point solver implementation uses the uni-variable polynomial solver from the OpenCV[3] library. The three implementations are available online [4].

Figure 1 shows the computational times for the two 4-point solvers and 5-point solver. The measured computational time is for the minimal case. The reader can easily see that the closed-form Groebner basis solver is the most computationally expensive. The numerical solver is slightly slower than the 5-point algorithm. Considering that additional constraints make the relative pose estimation problem more complex, the extra computational cost for the numerical solver is small; the numerical solver is still fast enough for real-time use.

We would like to also point out a further optimization for the numerical solver. The computation of the coefficients in (13) and (14) involves a series of extremely large polynomials generated by Maple; computing the coefficients take up approximately 70% of the computational time incurred by the current implementation. We can reduce the computation time by re-arranging the terms in the polynomials.

### B. Performance under Noise

We use simulation data in this section to test the performance of the 4-point and 5-point algorithms. We do not consider the 6, 7 and 8-point algorithms as the 5-point algorithm is known to outperform these algorithms for the case of a calibrated camera. Therefore, in this section, we only compare our algorithm with the 5-point algorithm. Detailed comparisons between the 5, 6, 7 and 8-point algorithms can be found in [4], [5].

[1] https://bitbucket.org/eigen/eigen/
[2] http://www.gnu.org/software/gsl/
[3] http://opencv.org
[4] https://sites.google.com/site/prclibo/four-point

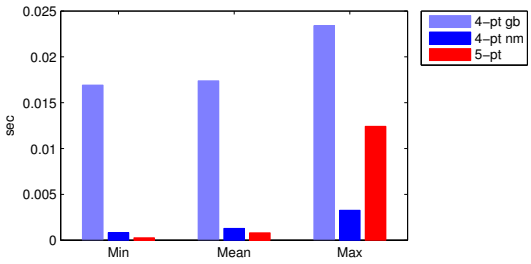


Fig. 1. Computation time for the minimal case for the various relative pose estimation algorithms. 4-pt gb stands for the closed-form Groebner basis solver. 4-pt nm stands for the numerical solver using gradient descent.

TABLE I
Experiment settings for simulation data.

| | |
|---|---|
| Minimal Distance | 10 |
| Depth | 10 |
| Baseline | 1 |
| Image Size | $350 \times 350$ |
| Field of View | $60°$ |
| Error Measurement | Translation deviation angle |
| Error Estimator | Lower quartile (minimal case)<br>Mean (RANSAC case) |
| Tests per Noise Level | 1000 (minimal case)<br>100 (RANSAC case) |

We structure our experiment setup in line with existing research by using similar simulation settings such as image size, field of view, and point distance from [4]. The settings are summarized in table I and figure 2. The two algorithms are tested with forward and sideways motions. The relative pose estimation error is measured by the angle between the ground truth translation and estimated translation vectors. This error measurement is based on the fact that the translation estimation is much more sensitive to noise compared to the rotation estimation; see [14] for details. In general, for both the 4-point and 5-point algorithms, the rotation error is less than $0.1°$. We approximate the image feature noise as a zero-mean Gaussian noise distribution with a varying range of standard deviations. For the minimal case test, four and five point correspondences are generated for the 4-point and 5-point algorithms respectively. For the RANSAC case test, we use 50 point correspondences to run a RANSAC framework to generate the best estimation. 1000 tests were executed for each noise level in the minimal case test. 100 tests were executed for each noise level in the RANSAC test. Figure 3 plots the relative pose error against the standard deviation of the feature noise distribution. In figures 3a and 3b, we plot the lower quartile of the relative pose error at each noise level for the minimal case tests. In figures 3c and 3d, we plot the mean error at each noise level for the RANSAC tests. The test criterion were selected to be consistent with those in [4]. From the plots, we can clearly see that the 4-point algorithm outperforms the 5-point algorithm.

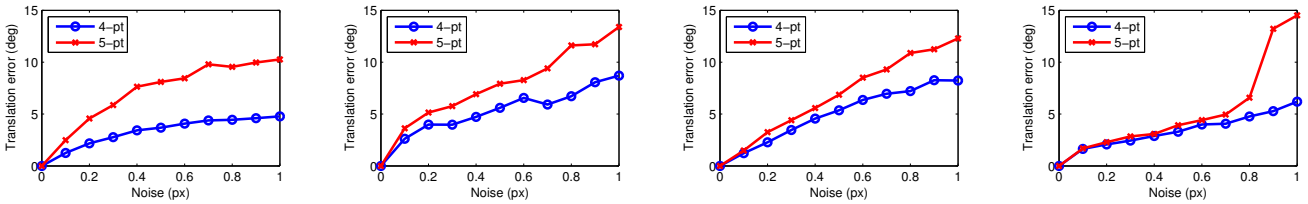


Fig. 3. Translation error in degrees against noise standard deviation in pixels. (a) Minimal cases, forward motion. (b) Minimal cases, sideways motion. (c) 50 points, forward motion. (d) 50 points, sideways motion.

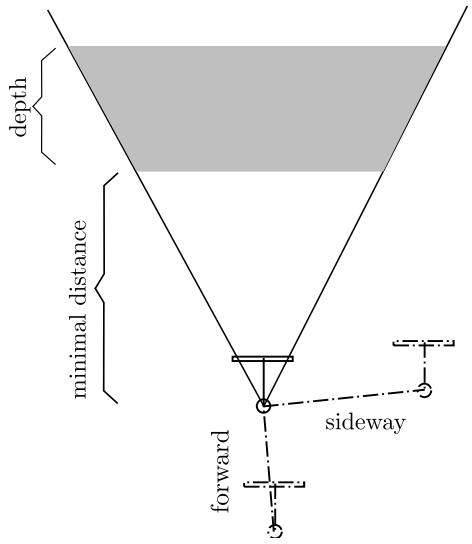


Fig. 2. Experiment settings for simulation data. The two dashed cameras mark the locations to which the camera moves with forward and sideways motion respectively.

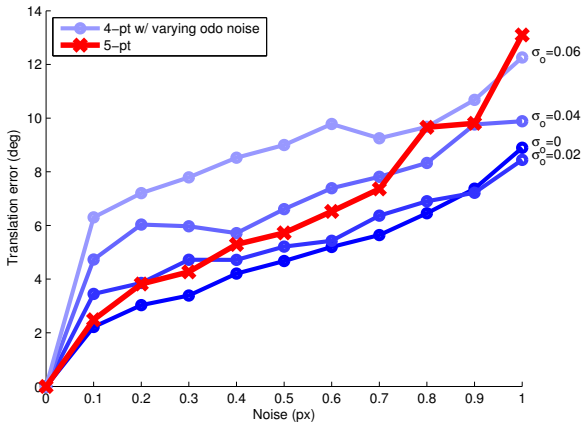


Fig. 4. Translation error in degrees against noise standard deviation in pixels. The standard deviation $\sigma_\theta$ of the relative rotation angle measurement noise ranges between 0 and 0.06. 50 points are used to execute a RANSAC scheme for each test.

We perform another test to show how the noise from relative rotation angle measurements influences the 4-point algorithm. We assume here that the relative rotation angle $\theta$ is measured as $(1+e)\theta$ by the sensor, where $e$ follows a zero-mean Gaussian distribution. In this test, we select the standard deviation $\sigma_\theta$ values to be 0, 0.02, 0.04, and 0.06. For each of the 4 noise levels, we generate a plot of translation error against the noise level. We also use the same settings with the above RANSAC tests except that the translation direction is arbitrary and the relative rotation angle is between $-10^\circ$ and $10^\circ$. The plot is shown in figure 4. We find that with the error $e$ smaller than 0.04, the 4-point algorithm gives a better result than the 5-point algorithm. In practice, the error of the relative rotation angle measurements provided by the rotation sensor is much smaller, and hence, the 4-point algorithm outperforms the 5-point algorithm in general.

### C. Real-World Performance

In this section, we compare the 4-point algorithm with the 5-point and 1-point algorithms which are two well-known algorithms for relative pose estimation for a vehicle platform. Here, we use the numerical solver for the 4-point algorithm as the numerical solver has a significantly shorter computational time than the Groeber basis solver. Our platform is a VW Golf outfitted with a camera, odometry and an iTrace GPS/INS system. The camera is mounted at the front of the vehicle and its intrinsics are calibrated beforehand. In addition, the camera pose with respect to the odometer and GPS/INS is known. We use these extrinsics to generate reference camera trajectories from GPS/INS and odometry data. The visual odometry generated by the 4-point algorithm assumes no knowledge of the extrinsic calibration between the camera and relative rotation angle sensor. In following discussions, we use the words GPS/INS and odometry to refer to the camera trajectories generated from GPS/INS and odometry respectively. The plotted trajectories are shown in figures 6 and 7. The GPS/INS trajectory is shown with other trajectories in each image as ground truth for comparison.

We compare the algorithms on two datasets collected with our platform. The first dataset is a single loop trajectory consisting of 2000 keyframes with a keyframe distance of 0.4 m. The second dataset is taken from multiple loops in a more challenging environment; 2800 keyframes are used with the same keyframe distance of 0.4 m. Figure 5 shows the aerial imagery of the scenes where the datasets are collected.

Fig. 5. Aerial imagery of the scenes used for the real-world experiments. Left: The parking lot where the first dataset with 1400 frames is collected. Right: The parking lot where the second dataset with 3000 frames is collected. The GPS/INS trajectory of the vehicle is plotted in gray.

ORB [15] feature correspondences detected from image data are passed as input into the compared algorithms. We provide the results from the 4-point algorithm using both the heading from the odometer readings and the rotation angle from the INS readings. Since we only compare the relative pose estimation results in this paper, the scale information between each pair of frames is directly obtained from the GPS/INS data for all the compared algorithms. Furthermore, only the relative pose between consecutive frames is estimated, and no bundle adjustment in any form is used.

From the plots, we clearly see that the 4-point algorithm using relative rotation angle measurements from INS data generates a trajectory closest to the ground truth. The 4-point algorithm using relative rotation angle measurements from odometry data generates a similar trajectory but with more drift due to the higher inaccuracy of odometer readings. The trajectories computed by the 5-point and 1-point algorithms have larger drifts. For the 5-point algorithm, if the image feature quality is low, the relative pose accuracy is significantly degraded. For the 1-point algorithm, the trajectory is very smooth due to the Ackermann steering assumption; however, the 1-point algorithm only works as long as the camera is located on the vehicle's rear axis. Our front camera configuration does not adhere to the rear axis requirement, leading to a continuous bias for each frame.

## VI. Conclusions

In this paper, we show that by using relative rotation angle measurements from a relative rotation sensor with unknown extrinsics, the relative camera pose can be estimated from only four image feature correspondences. In both simulated and real experiments, the algorithm shows significant improvement in terms of accuracy over the 5-point and 1-point algorithms. Intuitively, the 4-point algorithm outperforms the 5-point algorithm as the 4-point algorithm is far more likely to find a good initial estimate for the relative pose from RANSAC given the same number of iterations, and the use of relative rotation measurements narrows down the space of possible solutions. Similarly, the 4-point algorithm outperforms the 1-point algorithm; the assumption by the 1-point algorithm that the camera lies along the vehicle's rear axis is violated. There is room for further optimization of the implementation in terms of speed.

One limitation of the 4-point algorithm is that if large instantaneous changes in rotation are observed, we require the rotation sensor to be synchronized with the camera. Such synchronization may be difficult to implement.

Our 4-point algorithm can be used for any platform with a camera and a rotation sensor. For example, our 4-point algorithm can be used on mobile phones for which computer vision and augmented reality applications are increasingly becoming popular. The internal gyroscope sensors provide relative rotation angle measurements which are similar to those from odometry and INS. In addition, the 4-point algorithm is also applicable to robotic systems such as micro aerial vehicles which move in 3D space.

The main advantage of the 4-point algorithm is that no knowledge about the extrinsics is required, and thus, an extrinsic calibration is not needed. This non-requirement can be extremely useful for hand-eye calibration implementations in which rotation angle information can be used to improve the visual odometry estimates, and thus, the resulting hand-eye transform.

## VII. Acknowledgement

The second author was funded by the DSO National Laboratories Postgraduate Scholarship. In addition, this work was supported in parts by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #269916 (V-Charge) and 4DVideo ERC Starting Grant Nr. 210806.

## References

[1] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge Univ Press, 2000, vol. 2.

[2] H. Stewénius, D. Nistér, F. Kahl, and F. Schaffalitzky, "A minimal solution for relative pose with unknown focal length," *Image and Vision Computing*, vol. 26, no. 7, pp. 871–877, 2008.

[3] H. Li, "A simple solution to the six-point two-view focal-length problem," *Computer Vision–ECCV 2006*, pp. 200–213, 2006.

[4] D. Nistér, "An efficient solution to the five-point relative pose problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6, pp. 756–770, 2004.

[5] H. Stewénius, C. Engels, and D. Nistér, "Recent developments on direct relative orientation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 60, no. 4, pp. 284–294, 2006.

[6] H. Li and R. Hartley, "Five-point motion estimation made easy," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1. IEEE, 2006, pp. 630–633.

[7] 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.

[8] D. Scaramuzza, "1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *International journal of computer vision*, vol. 95, no. 1, pp. 74–85, 2011.

[9] G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Motion estimation for self-driving cars with a generalized camera," in *Computer Vision and Pattern Recognition, 2013.*

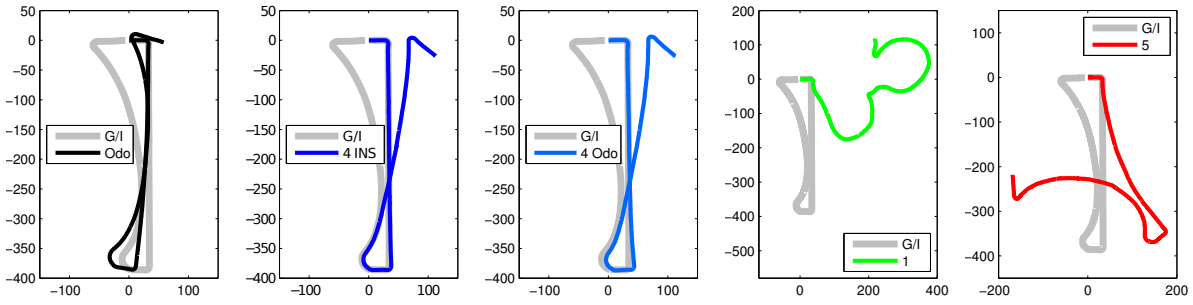


Fig. 6. Visual odometry results on single loop data with 2000 frames. All axis values are expressed in meters. We use the GPS/INS (G/I) trajectory as ground truth as shown in each image for comparison. From left to right: 1) Wheel odometry plot. 2) Visual odometry by 4-point algorithm with numerical solver and using relative rotation angle measurements from the INS readings. 3) Visual odometry by 4-point algorithm with numerical solver and using relative rotation angle measurements from odometry readings. 4) Visual odometry by 1-point algorithm. 5) Visual odometry by 5-point algorithm.

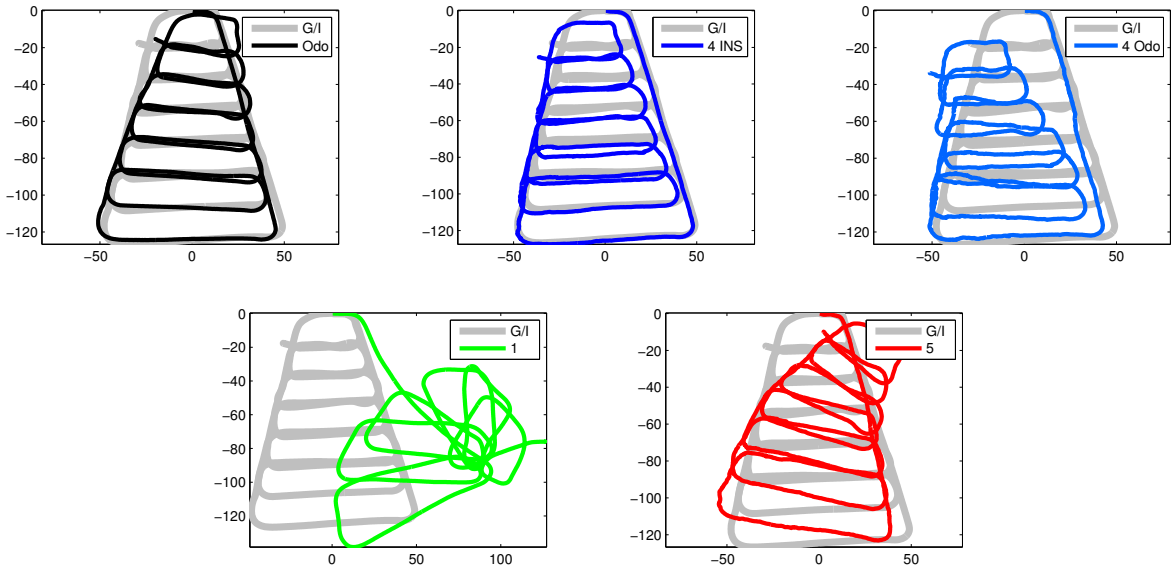


Fig. 7. Visual odometry results on multiple loop data with 2800 frames. All axis values are expressed in meters. We use the GPS/INS (G/I) trajectory as ground truth as shown in each image for comparison. From left to right and top to bottom: 1) Wheel odometry plot. 2) Visual odometry by 4-point algorithm with numerical solver and using relative rotation angle measurements from the INS readings. 3) Visual odometry by 4-point algorithm with numerical solver and using relative rotation angle measurements from odometry readings. 4) Visual odometry by 1-point algorithm. 5) Visual odometry by 5-point algorithm.

[10] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *Robotics and Automation, IEEE Transactions on*, vol. 5, no. 3, pp. 345–358, 1989.

[11] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax= xb," *Robotics and Automation, IEEE Transactions on*, vol. 5, no. 1, pp. 16–29, 1989.

[12] Z. Kukelova, M. Bujnak, and T. Pajdla, "Automatic generator of minimal problem solvers," *Computer Vision–ECCV 2008*, pp. 302–315, 2008.

[13] D. Batra, B. Nabbe, and M. Hebert, "An alternative formulation for five point relative pose problem," in *Motion and Video Computing, 2007. WMVC'07. IEEE Workshop on*. IEEE, 2007, pp. 21–21.

[14] T. Y. Tian, C. Tomasi, and D. J. Heeger, "Comparison of approaches to egomotion computation," in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*. IEEE, 1996, pp. 315–320.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.