An Efficient Minimal Solution for Infinitesimal Camera Motion

Henrik Stewénius
http://www.vis.uky.edu/~stewe

Chris Engels
http://www.vis.uky.edu/~engels

David Nistér
http://www.vis.uky.edu/~dnister

Abstract

Given five motion vectors observed in a calibrated camera, what is the rotational and translational velocity of the camera? This problem is the infinitesimal motion analogue to the five-point relative orientation problem, which has previously been solved through the derivation of a tenth-degree polynomial and extraction of its roots.

Here, we present the first efficient solution to the infinitesimal version of the problem. The solution is faster than its finite counterpart. In our experiments, we investigate over which range of motions and scene distances the infinitesimal approximation is valid and show that the infinitesimal approximation works well in applications such as camera tracking.

1. Introduction

This paper presents an efficient solution to the following problem: given five motion vectors observed in a calibrated camera, what is the rotational and translational velocity of the camera? This is the infinitesimal motion analogue to the finite motion five-point relative orientation problem.

The infinitesimal problem is known, just as its finite counterpart, to have ten solutions. However, while the finite problem has previously been solved efficiently through the derivation of a tenth-degree polynomial and extraction of its roots, no such solution was previously known for the infinitesimal problem.

We present what is to our knowledge the first such solution to the infinitesimal problem and show that the solution is more efficient than the finite counterpart.

Just like the finite motion algorithm, the infinitesimal motion approximation can be used on image correspondences extracted with any algorithm of choice. The small motion approximation made here is a geometric approximation of small camera translation and rotation. Thus, the approximation is very different from the one made in optical flow, where image patches are assumed to undergo changes that can be Taylor-approximated. Here, it is assumed that the geometric effects of camera motion on the image points can be Taylor-approximated. As a result, the approximation is valid for far larger motions than optical flow is. One can in fact use finite motion correspondences in the differential five-point algorithm, or use motion vectors extracted with optical flow to generate correspondences for the usual five-point algorithm. The method for correspondence extraction

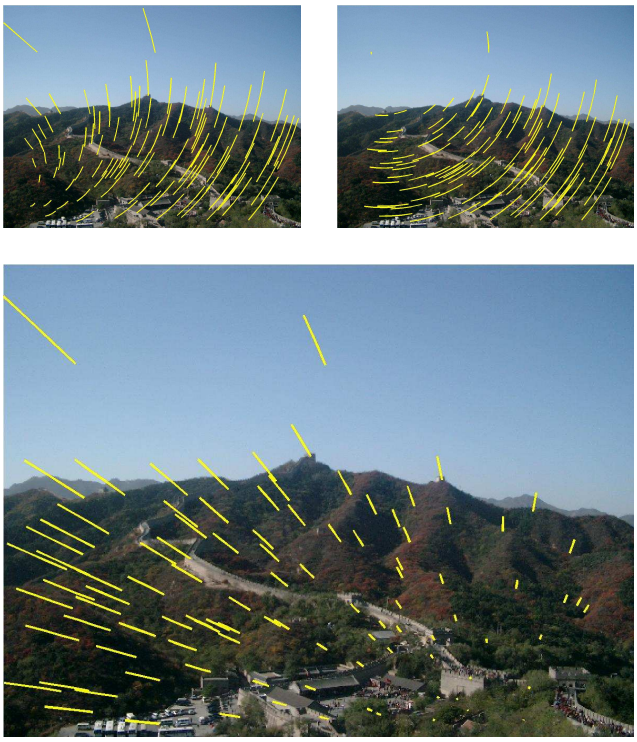

Figure 1. This paper considers the differential five-point relative orientation problem. One way to understand the constraint used in relative orientation is the following. Given the observed motion field (top left), a hypothesized rotation (top right) is subtracted from the motion field. For the correct rotation the remaining motion field (bottom) must converge on a point.

and the method for extraction of camera motion given point correspondences are decoupled.

In the experiments we show that the infinitesimal approximation works well for applications such as camera tracking. We also investigate over which range of motions and scene distances the approximation is valid.

2. Prior Art

It is known that the differential five-point problem has ten solutions.A proof of this can be found in [8]. However, this is to our knowledge the first solution that corresponds directly to the intrinsic difficulty of the problem, in the sense that we derive a tenth degree polynomial with roots corresponding to the ten solutions. Moreover, it is the first time an algorithm has been used to solve this problem in a practical context and to investigate the performance under noise.

The presented solver is similar to the finite five-point solver [9]. The finite motion case has been extensively studied. Kruppa showed that the finite problem has at most 22 solutions [7]. Later, this number was reduced to 20 and it was shown that the roots could be computed in pairs by solving for the essential matrix resulting in 10 solutions [3, 6].

The infinitesimal case has been relatively less studied with some important exceptions [1, 11]. Our solver represents an alternative to the finite five-point solver in applications where small motion is expected.

3. Solving the Problem

A geometrically very nice way to think of both the finite and differential versions of the relative orientation problem is the following: the motion of the image points is a composition of a rotational and translational component, see Figure 1. The rotational component is given directly by the camera rotation or rotational velocity, while the translational component causes the image points to move towards or away from a single focus of expansion $\mathbf{e}$ (or epipole in the finite motion case). The amount of translation is different for each image point, and depends on the depth of the image point, but the direction of translation is always on the line towards $\mathbf{e}$. If we ignore whether the point is in front of or behind the camera, so that the direction of image motion is immaterial, this is in fact the only constraint, since any point on the line towards $\mathbf{e}$ can be obtained with some depth for the point, where the depth is also unknown. To obtain equations from which the problem can be solved, one can therefore observe the following:

Observation *A rotation is part of a valid solution for the camera motion if and only if it is the case that when we subtract the motion induced by this rotation from the observed motion field, the remaining motion field converges to or diverges from a common point* $\mathbf{e}$.

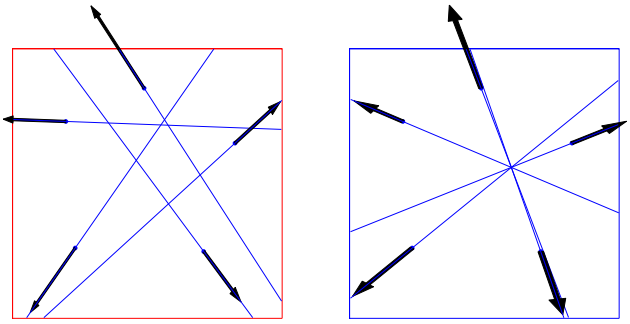

Figure 2. The lines defined by five observed point derivatives do not in general come together at a single point (left). The constraint on the rotation is that when we subtract the hypothesized rotational component from the motion, the residual velocity vectors should define lines that come together in a single point (right). This provides exactly the three scalar constraints on the rotation required to solve the differential five-point problem. The finite five-point problem can be thought of in a similar way.

This constraint is illustrated in Figure 2.

In the case of five observed image points, a hypothesis for the rotation can thus be used to de-rotate the five points or the five velocity vectors, yielding five lines that have to come together at a single point. Since two distinct lines always intersect at exactly one point (at least in projective image space), and each of the additional three lines have to satisfy a single scalar equation in order to go through that point as well, we have exactly the three scalar constraints on the rotation that are necessary to obtain a finite number of solutions.

Let the homogeneous coordinates of an observed image point be denoted by $\mathbf{u}$. Further, let the rotational velocity be $\mathbf{r}$, inducing the image motion $[\mathbf{r}]_\times \mathbf{u}$ at that point, where the motion is regarded as a tangent vector on the sphere. Let also the observed velocity vector of a point be denoted by $\mathbf{v}$, again regarded as a tangent vector on the sphere.

The de-rotated velocity vector is then

$$\mathbf{m} = \mathbf{v} - [\mathbf{r}]_\times \mathbf{u} = \mathbf{v} + [\mathbf{u}]_\times \mathbf{r} = \begin{bmatrix} [\mathbf{u}]_\times & \mathbf{v} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} \qquad (1)$$

This velocity vector is on the line defined by the focus of expansion $\mathbf{e}$ and the point $\mathbf{u}$ if and only if

$$\mathbf{e}^\top [\mathbf{u}]_\times \mathbf{m} = \mathbf{e}^\top [\mathbf{u}]_\times \begin{bmatrix} [\mathbf{u}]_\times & \mathbf{v} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = 0. \qquad (2)$$

Note that $\mathbf{e}$ and $\mathbf{r}$ define the motion of the camera, and are unknown, while $\mathbf{u}$ and $\mathbf{v}$ are observations (and different for each point). Our goal is now to eliminate $\mathbf{r}$ from the equations given by five point observations. It is also possible to instead eliminate $\mathbf{e}$ and thereby obtain equations similar to

the ones occurring for the finite five-point problem [9], but for the differential case we obtain a more efficient solver by eliminating $\mathbf{r}$. For this purpose, define

$$\mathbf{a}_i(\mathbf{e}) = \mathbf{e}^\top [\mathbf{u}_i]_\times \begin{bmatrix} [\mathbf{u}_i]_\times & \mathbf{v}_i \end{bmatrix}, \tag{3}$$

where i is the point number. Note that for each point, $\mathbf{a}_i(\mathbf{e})$ is a 1×4 row vector where each entry is a linear homogeneous function of $\mathbf{e}$. We can now rewrite Equation (2) as

$$\mathbf{a}_i(\mathbf{e}) \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = 0 \tag{4}$$

and by stacking $\mathbf{a}_1(\mathbf{e}), \ldots, \mathbf{a}_5(\mathbf{e})$ into the 5×4 matrix

$$\mathbf{A}(\mathbf{e}) = \begin{bmatrix} \mathbf{a}_1(\mathbf{e}) \\ \vdots \\ \mathbf{a}_5(\mathbf{e}) \end{bmatrix} \tag{5}$$

we obtain the equation system

$$\mathbf{A}(\mathbf{e}) \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = 0. \tag{6}$$

Since $\mathbf{A}(\mathbf{e})$ has a nullvector, it must be of at most rank three. Hence, all the five 4×4 sub-determinants of $\mathbf{A}(\mathbf{e})$ must be zero. Each of these sub-determinants is obtained by removing a row from $\mathbf{A}(\mathbf{e})$ and taking the determinant of what is left. The requirement that the five sub-determinants vanish can be written

$$\underbrace{\begin{bmatrix} [0] & [1] & [2] & [3] & [4] \\ [0] & [1] & [2] & [3] & [4] \\ [0] & [1] & [2] & [3] & [4] \\ [0] & [1] & [2] & [3] & [4] \\ [0] & [1] & [2] & [3] & [4] \end{bmatrix}}_{\mathbf{C}(e_2, e_3)} \begin{bmatrix} e_1^4 \\ e_1^3 \\ e_1^2 \\ e_1 \\ 1 \end{bmatrix} = 0, \tag{7}$$

where $[n]$ denotes an n-th degree homogeneous polynomial in e_2 and e_3. Since $\mathbf{C}$ has a non-trivial nullspace, its determinant

$$p(e_2, e_3) = \det \mathbf{C}(e_2, e_3) \tag{8}$$

must vanish. Note that $p(e_2, e_3)$ is a tenth degree homogeneous polynomial in e_2 and e_3. We set $e_3 = 1$, and solve for the up to ten roots e_2 of p. We can then solve linearly for e_1 using Equation (7) and for $\mathbf{r}$ using Equation (6).

3.1. Alternate Solver Using a Gröbner Basis

A somewhat slower but potentially more stable path is given by computing a Gröbner basis [2] with the monomials ordered in Graded Reverse Lexicographic order (GrevLex). This method is slower since it requires solving an eigen-problem instead of just iterating on the real solutions using a Sturm-sequence based solver. When using a root solver based on a companion matrix this way of computing the solutions is actually just as fast as when computing a univariate polynomial.

As mentioned above, Equation (6) implies that all 4×4 sub-determinants of $\mathbf{A}$ vanish. These equations can be written

$$BX = 0, \tag{9}$$

where B is a 5×15 matrix of scalars and X is a vector of monomials of degree 4 in (e_1, e_2, e_3). We order X in GrevLex order. By performing Gauss-Jordan elimination on B we get a Gröbner basis for the ideal defined by BX:

e_1^4	$e_1^3e_2$	$e_1^2e_2^2$	$e_1e_2^3$	e_2^4	e_1^3	$e_1^2e_2$	$e_1e_2^2$	e_2^3	e_1^2	e_1e_2	e_2^2	e_1	e_2	1
1					•	•	•	•	•	•	•	•	•	•
	1				•	•	•	•	•	•	•	•	•	•
		1			•	•	•	•	•	•	•	•	•	•
			1		•	•	•	•	•	•	•	•	•	•
				1	•	•	•	•	•	•	•	•	•	•

Given the Gröbner basis we extract the action matrix N_{e_1} for multiplication by e_1 in the quotient ideal. The left eigenvectors of N_{e_1} encode the solutions for (e_1, e_2, e_3). We then solve for $\mathbf{r}$ in Equation (6). See Appendix A for a Matlab implementation of these operations.

4. Efficiency Considerations

In this section we present detailed efficiency optimizations. The solver derived in Section 3 is more efficient than the discrete five-point solver since the matrices that have to be eliminated are smaller. However, there are several improvements that can be made. The goal of this section is to point out the details that make an implementation efficient. We start with Equation (6)

$$\mathbf{A}(\mathbf{e}) \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = 0. \tag{10}$$

Each row of $\mathbf{A}$ is represented by 12 scalar numbers, of which only 9 are distinct. We can exploit this to save operations and time when manipulating $\mathbf{A}$. By applying row operations $\mathbf{A}$ can be reduced to

$$\mathbf{A}_{red} = \begin{bmatrix} e_1 & 0 & \bullet e_3 & \bullet e_1 + \bullet\, e_2 + \bullet\, e_3 \\ e_2 & e_1 & \bullet e_3 & \bullet e_1 + \bullet\, e_2 + \bullet\, e_3 \\ e_3 & 0 & e_1 + \bullet\, e_3 & \bullet e_1 + \bullet\, e_2 + \bullet\, e_3 \\ 0 & e_2 & e_2 + \bullet\, e_3 & \bullet e_1 + \bullet\, e_2 + \bullet\, e_3 \\ 0 & e_3 & \bullet e_3 & \bullet e_1 + \bullet\, e_2 + \bullet\, e_3 \end{bmatrix}, \tag{11}$$

where $\bullet$ represents multiplication by a scalar. That is, each row of $\mathbf{A}_{red}$ is defined by four scalar numbers.

Since $\mathbf{A}_{red} \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = 0$, we have that all 4×4 sub-matrices of $\mathbf{A}_{red}$ must have vanishing determinants. Since $\mathbf{A}_{red}$ has a very special structure, the computation of these sub-determinants can be implemented efficiently.

By applying row operations to Equation (7), this system is reduced to

$$\begin{bmatrix} 1 & [1] & [2] & [3] & [4] \\ 0 & [1] & [2] & [3] & [4] \\ 0 & [1] & [2] & [3] & [4] \\ 0 & [1] & [2] & [3] & [4] \\ 0 & [1] & [2] & [3] & [4] \end{bmatrix} \begin{bmatrix} e_1^4 \\ e_1^3 \\ e_1^2 \\ e_1 \\ 1 \end{bmatrix} = 0 \tag{12}$$

$$\Rightarrow$$

$$\underbrace{\begin{bmatrix} [1] & [2] & [3] & [4] \\ [1] & [2] & [3] & [4] \\ [1] & [2] & [3] & [4] \\ [1] & [2] & [3] & [4] \end{bmatrix}}_{\mathbf{C}_{red}(e_2,e_3)} \begin{bmatrix} e_1^3 \\ e_1^2 \\ e_1 \\ 1 \end{bmatrix} = 0 \Rightarrow$$

$$p(e_2, e_3) = \det \mathbf{C}_{red}(e_2, e_3) = 0. \tag{13}$$

From $p(e_2, e_3)$ we compute e_2 and e_3 using a Sturm-sequence based root-solver. e_1 is then computed by taking the nullspace of $\mathbf{C}_{red}(e_2, e_3)$. $\begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix}$ can then be computed as a nullspace of $\mathbf{A}_{red}(e_1, e_2, e_3)$.

The nullspaces needed are all of small matrices and can be computed using Cramer's rule. We have implemented our root-solver along the lines given in [9].

Computing Symbolic Determinants

The sub-determinants of $\mathbf{A}_{red}$ and the determinant of $\mathbf{C}_{red}$ have in common that the matrix entries are polynomials and that the determinants that we need to compute are of size 4×4.

In order to derive these determinants we first compute all sub-determinants for the two first columns and then, all sub-determinants in the last two columns. The determinant is computed as a sum of products of these sub-determinants.

5. Experiments

We first examine the stability and performance of the differential five-point algorithm under various motion and noise conditions and compare our results with those of other solvers. In Section 5.2, we give an example of using our algorithm within a structure from motion system using real video data. Finally, we give a performance comparison between optimized versions of our algorithm and the finite five-point solver.

5.1. Stability of Motion Estimates

To test the stability of the algorithm, we generated a synthetic scene with definable camera motion and observation error. 500 world points were created in space and projected into the cameras; observation error was subsequently added to the image points in the form of Gaussian noise. Unless otherwise stated, the noise has a standard deviation equivalent to one pixel in a 640×480 image, and the baseline is half the depth to the closest points in the scene. Motion derivatives are simply defined by the inter-camera point motion. Five points generated the motion solutions, and a sixth point was used to find the real solution with minimal Sampson distance [5]. We also show results of the discrete five-point [9] and eight-point [5] solvers for comparison. Error measurements in the figures represent the median of 1000 random input point configurations.

From Figures 3 and 4, we see that the differential five-point algorithm is resistant to error caused by noise in cases of forward and sideways motion. Our solver also performs well in cases where the camera is translating with little or no rotation, as seen in Figures 5 and 6. Note that for very small baselines–especially in the case of forward motion–noise has a greater influence than image point motion on the camera motion estimate. If we look at the error as a function of translation direction, as in Figure 7, the eight-point algorithm has a clear forward bias, whereas the other solvers are less dependent on this direction. This bias explains the disparity of errors seen in the small baseline range of Figures 5 and 6 for the eight-point solver.

Figures 8 and 9 were created from a scene with sideways camera motion and increasing rotation. Compared to the finite motion approaches, the error in the differential algorithm quickly becomes apparent in cases of strong rotation. By using a first order derivative, we implicitly assume linear motion of feature points in the image. If this assumption does not hold, we can expect increasing error based on the nonlinearity of the motion. Since the finite motion methods make no such assumption, the errors in those methods remain flat. However, for video, optical flow or any narrow baseline situation, rotation between consecutive frames would rarely be sufficient for the linear assumption to fail. In the context of wide baseline or classical photogrammetric applications, the differential approach is less valid.

5.2. Application to Structure from Motion System

In this section, we give an example of using the differential five-point algorithm in a real world application for structure from motion. The system operates on an image sequence in a manner similar to the monocular scheme given in [10]. We use a simplified approach, looping over the following:

- Track feature points over several frames in the sequence. Estimate relative poses between three frames using the differential five-point algorithm within a RANSAC framework [4]. Perform iterative refinement over these frames.

- Triangulate world points from tracked features.

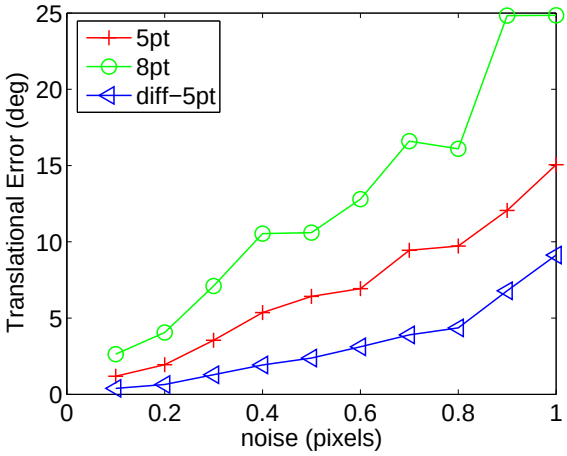

Figure 3. Translational error relative to increasing noise from a camera with sideways motion.

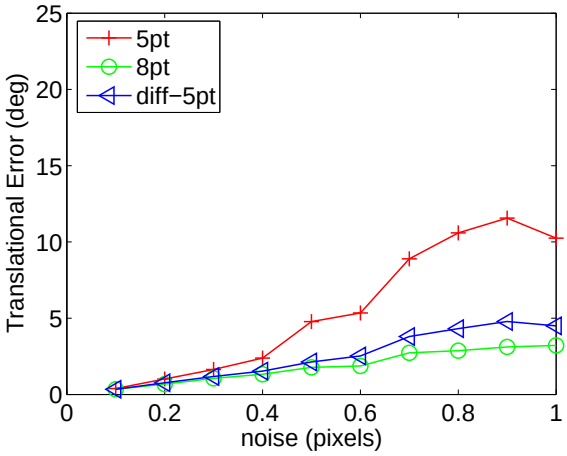

Figure 4. Translational error relative to increasing noise from a camera with forward motion.

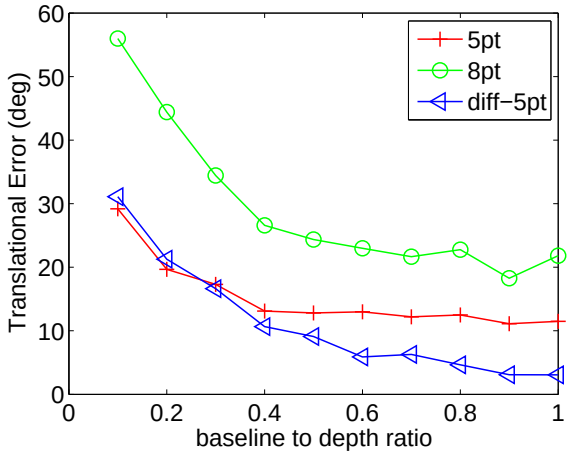

Figure 5. Translational error relative to increasing sideways translation.

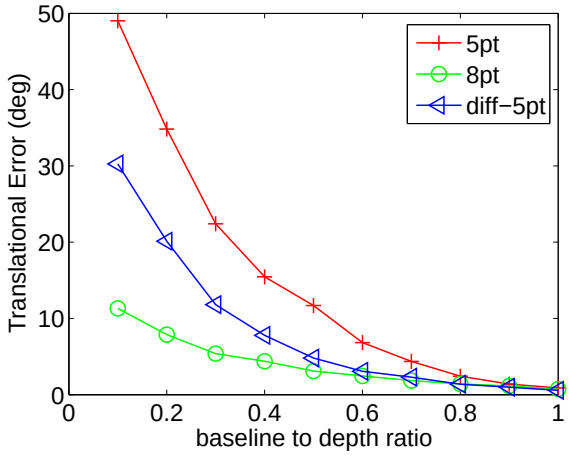

Figure 6. Translational error relative to increasing forward translation.

- Except for the first iteration, transform the relative pose of new frames to the global coordinate system. Use RANSAC to estimate the scale difference between the coordinate systems.

- (*optional*) Perform bundle adjustment over recent frames.

Results of a 200 frame sequence taken from a vehicle-mounted camera are shown in Figure 10. No outlier rejection was performed; the system relies instead on the robustness of the RANSAC process to remain unaffected. The outliers are visible by their large reprojection errors, represented as red lines in the image frames. The true motion is approximately straight, although both the differential and finite motion algorithms drift slightly over time. This problem is readily fixed by adding the bundle adjustment step.

Figure 11 presents a more challenging turntable sequence. Because rotation is a factor, we expect that the finite motion method will perform better than our method. The differential method does misestimate the motion in a small number of cases, viewable as kinks in the camera path, but these are not enough to cause bundle adjustment to fail.

5.3. Speed

We have implemented both the differential five-point and the finite motion five-point methods using Sturm sequence iterations to bracket the roots, followed by bisection to refine the brackets for the individual roots.

Timing can be divided into two parts. The first part is the

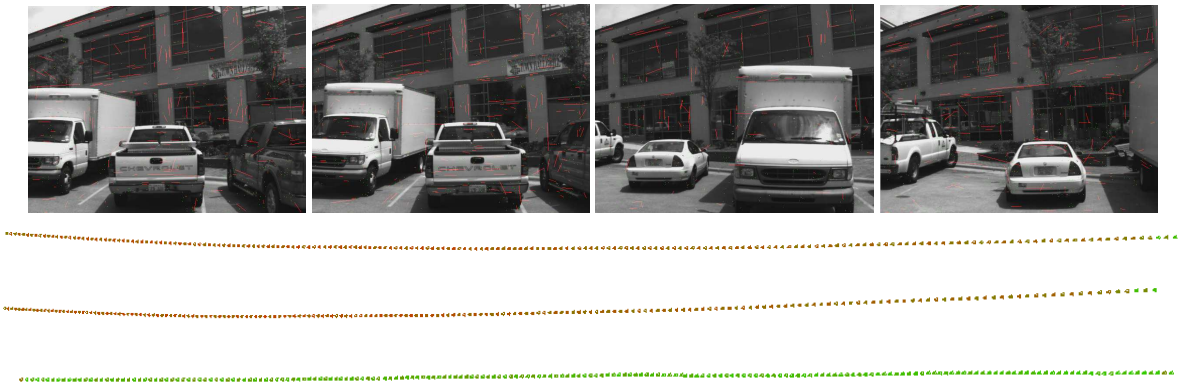

Figure 10. Reconstructed vehicle-mounted camera sequence using the differential five-point algorithm. From top to bottom: Frames from original sequence, with red lines representing reprojection error; Differential five-point; Finite motion five-point; Bundle adjusted differential five-point.

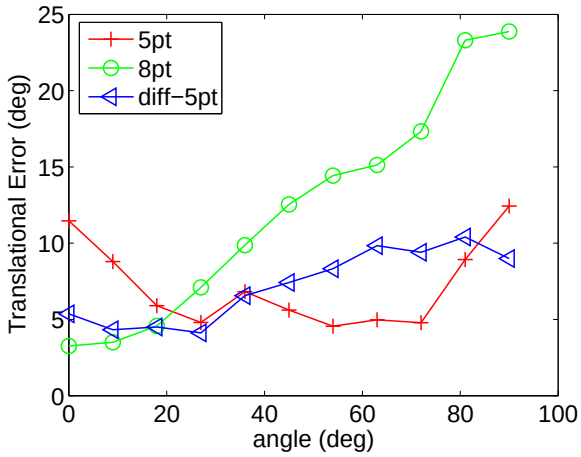

Figure 7. Translational error relative to angle of translation from forward direction.

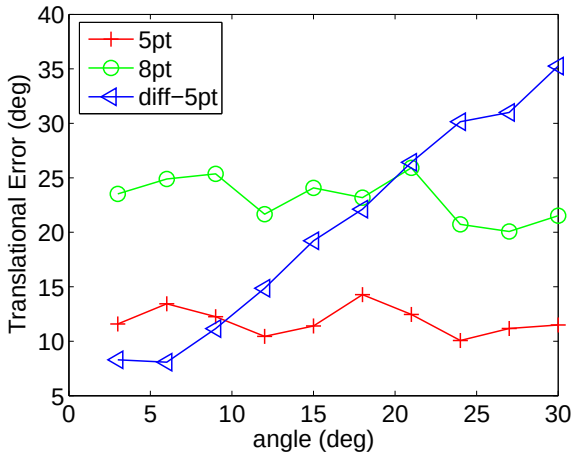

Figure 8. Translational error relative to increasing rotation about the vertical axis.

cost of running the non-iterative parts of the solvers; this cost is on the order of $1.6\mu s$ for the differential five-point solver and $7.5\mu s$ for the finite motion five-point. The second part is the iterative solver used to actually find the roots of the respective polynomials. Since the number of roots are equal this is mainly a question of how many bisections are permitted. For the settings used in the experiments the needed time is $4\mu s$ giving a total of $5.8\mu s$, which is twice as fast as the finite motion version.

The reason why the differential formulation is so much faster than the finite motion formulation is that the solver in the finite motion case has to perform Gauss-Jordan elimination on a 10×20 matrix, which is quite costly.

6. Summary and Conclusions

An efficient algorithm for solving the differential five-point problem was presented. We have tested the method both in a RANSAC-based framework for structure from motion and in synthetic tests against the finite five- and eight-point methods. Within reasonable limits on the image-to-image rotation the differential five-point solver has a performance on par with the finite five-point and beats the eight-point algorithm.

Since this type of solver is normally used in a hypothesize-and-test framework, it is important to point out that the differential five-point solver can be made significantly faster than the finite five-point solver.

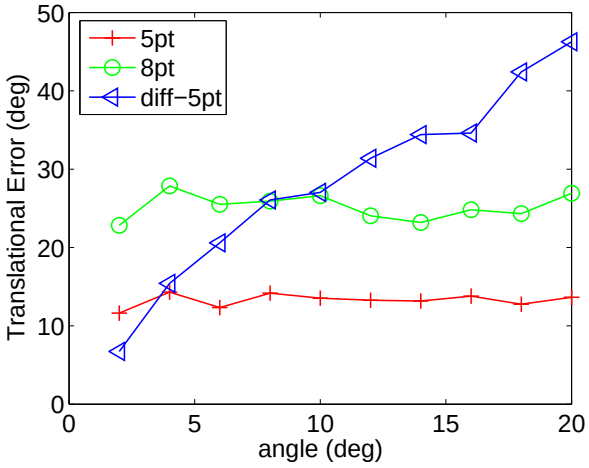

Figure 9. Translational error relative to increasing rotation about the forward axis. The large error here arises because the implicit assumption of linear motion in the differential five-point solver becomes invalid after sufficiently large rotation.

Appendix A: Code for Computing a Gröbner Basis

The easiest way to explain the simplicity of the Gröbner basis solver presented in Section 3.1 is the following Matlab code:

```
function e_sols=get_e_from_B(B)
% B is 5x15
% t is 3 times 10,
% one solution per column

%Compute Grobner Basis
  M = inv(B(:,1:5))*B(:,6:15);

%Extract transposed Action Matrix
  N = zeros(10);
  N(1:4,:) = -M(2:5,:);
  N([15 26 37 58 69 90])=1;


%Solve eigenproblem
  [V,D] = eig(N);

%Extract solutions
  e_sols = V(8:10,:);
```

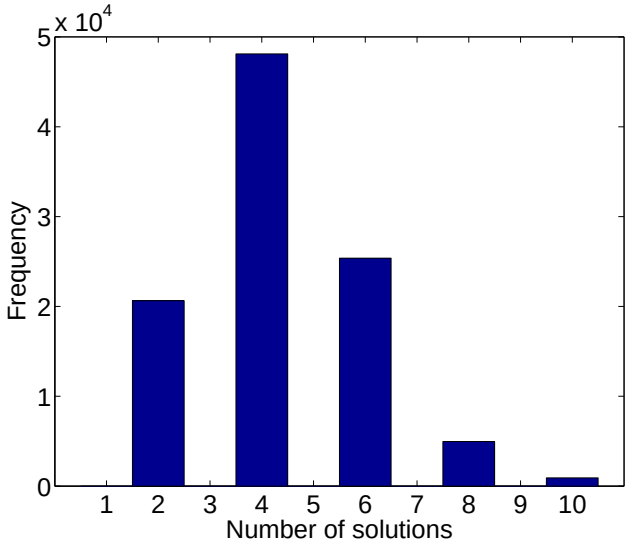

Figure 12. For random data there are on average 4.3483 real roots 12

References

[1] K. Åström and A. Heyden. Multilinear constraints in the infinitesimal-time case. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1996. 2

[2] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer Verlag, 1997. 3

[3] O. Faugeras and S. Maybanks. Motion from point matches: multiplicity of solutions. *International Journal of Computer Vision (IJCV)*, 1990. 2

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications-of-the-ACM*, 24(6):381–95, 1981. 4

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 4

[6] B. Horn. Relative orientation revisited. *Journal of the Optical Society of America, A*, 8:1630–1638, 1991. 2

[7] E. Kruppa. Zur Ermittlung eines Objektes Zwei Perspektiven mit innerer Orientierung. *Sitz-Ber. Akad. Wiss., Wien, math. naturw. Kl. Abt*, IIa(122):1939–1948, 1905. 2

[8] S. Maybank. *Theory of Reconstruction from Image Motion*. Springer Verlag, 1993. 2

[9] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–770, June 2004. 2, 3, 4

[10] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, Jan. 2006. 4

[11] B. Triggs. Differential matching constraints. In *IEEE International Conference on Computer Vision (ICCV)*, volume 01, page 370, 1999. 2

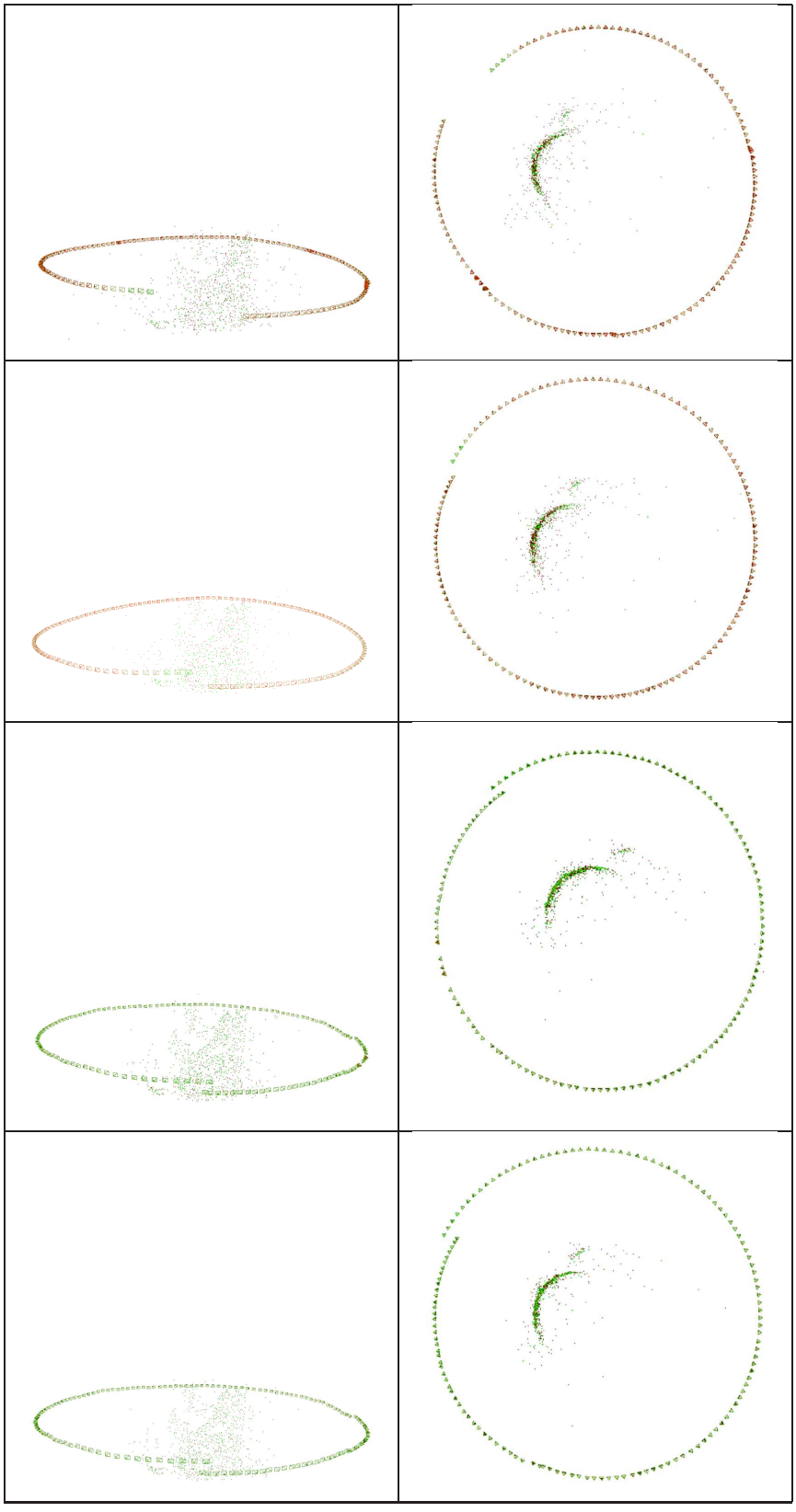

Figure 11. Reconstructions from 140 frames of a turntable sequence; Left: Differential Five-point; Right: Finite motion Five-point; Top: Unbundled reconstructions; Bottom: With bundle adjustment. The point cloud represents triangulated feature points visible from the last camera. Green indicates lower reprojection error.