Estimation of the Epipole Using Optical Flow at Antipodal Points

John Lim * Nick Barnes

Research School of Information Sciences and Engineering, Australian National University, Australia

NICTA, Canberra Research Laboratory, Australia

Abstract

We present algorithms for estimating the epipole or direction of translation of a moving camera. We use constraints arising from two points that are antipodal on the image sphere in order to decouple rotation from translation. One pair of antipodal points constrains the epipole to lie on a plane, and two such pairs will correspondingly give two planes. The intersection of these two planes is an estimate of the epipole. This means we require image motion measurements at two pairs of antipodal points to obtain an estimate. Two classes of algorithms are possible and we present two simple yet extremely robust algorithms representative of each class. These are shown to have comparable accuracy with the state of the art when tested in simulation under noise and with real image sequences.

Key words: Egomotion estimation, Omnidirectional cameras, Multi-view Geometry

* Corresponding author.
Email addresses: john.lim@rsise.anu.edu.au (John Lim),

1 Introduction

A monocular observer moving in a scene of unknown depth undergoes a rigid motion that is a combination of translational and rotational motions. We focus on estimating these motions using image motion measurements, or optical flow. This is a classical problem and the last few decades of research has produced a vast number of self-motion estimation methods. These include the well-known epipolar geometry formulation of [1,2] which leads to algorithms such as the 8-point [3], 6-point [4] and 5-point algorithms [5,6]; methods involving nonlinear optimization [7,8]; qualitative search methods [9,10] and the recovery of a 'flow fundamental matrix' from optical flow [11]. Other approaches of note include [12], [13], [14], [15] and a great many others, which can be found in reviews such as [16] and [17].

However, most approaches assume the use of a planar image with a limited field-of-view (FOV) such as that found in traditional cameras. Although omnidirectional cameras have become widely available of late, few self-motion estimation algorithms actually exploit the large FOV property explicitly in order to aid or simplify the task. [9] and [18] are examples of some such prior work. However, the method proposed here is significantly more efficient compared to those algorithms, which solve the problem via a search.

This paper expands on our recent work in [19], where we presented a method for estimating the direction of translation or the epipole of the camera based on the geometrical properties of points that are antipodal on the image sphere. The image sphere is simply a more natural representation for the images of

Nick.Barnes@nicta.com.au (Nick Barnes).

omnidirectional cameras. From the optical flow at two antipodal points we were able to constrain the epipole to lie on a great circle (that is, the intersection of a plane through the camera center with the image sphere). The rotational contribution to the measured optical flow was geometrically eliminated, leaving an equation dependent only on the translational component of motion. These constraints were then used to estimate the location of the epipole.

Whilst our method does not directly estimate rotational motion, we will show that once direction of translation has been found, a least squares estimate of rotation that is *robust* to outliers would not be difficult to recover. Note that due to the use of antipodal points, this algorithm is, by its very nature, suited for omnidirectional sensors and large FOV cameras.

A somewhat similar antipodal point constraint was observed in [20]. However, significant differences in the theoretical derivation and in the resulting constraint exist between the method of [20] and our work (Section 2.1). Furthermore, as the authors of [20] noted, the lack of widely available omnidirectional sensors at that time (1994) meant that such a constraint was hardly of any practical use then. As a result, their constraint was merely observed as an equation and was not investigated further. The emergence of omnidirectional cameras as a popular tool in computer vision today warrants an investigation into methods that specifically exploit the large FOV of these sensors.

Our constraint is somewhat simpler to derive and use compared to [20] and it succeeds in certain scene depth configurations where [20] fails. Both the constraint presented here and that of [20] may be thought of as special cases of the linear subspace methods investigated by [12].

The method proposed here is also related to the approach of [21] which utilizes antipodal points as well. However, whereas our approach is based on eliminating the rotational component of flow in order to constrain translation, the method of [21] obtains constraints without eliminating rotation. As a result, [21] obtains constraints on translation and on rotation, whilst the method proposed here constrains translation only (rotation is found via a second step). However, this also means that the constraint on translation in [21] is weaker than the one proposed here.

Section 2 begins with recapitulating the theory presented in [19]. In Section 3, we identify two basic classes of algorithms utilizing the antipodal point constraint - *point-based* algorithms and *line/curve-based* algorithms. We then present two novel algorithms representative of each class. The first is an algorithm using the derived constraint within a Random Sample and Consensus (RANSAC) [25,26] framework. The second algorithm performs Hough-reminiscent voting [22–24] along the great circles that constrain the direction of translation. It runs faster than the first algorithm but is just as robust to noise and outliers.

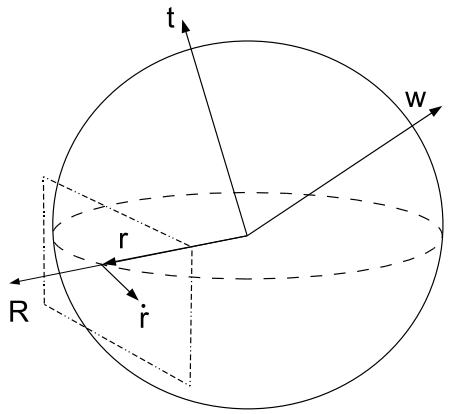
In Section 4, we compare the performance of our two algorithms with what is often considered the state-of-the-art for self-motion estimation in calibrated cameras - the 5-point algorithm [5,6] within a RANSAC framework. Comparisons were done using Matlab simulations under noise, and also using noisy real image sequences that involved independently moving objects in the scene. In Section 5, we show that our method is just as accurate as 5-point with RANSAC, with the added advantages of having improved robustness to outliers, constant run-time with increasing noise and outliers (for the voting algorithm), and a naturally parallelizable framework which makes the method

potentially viable for egomotion estimation at high speeds.

1.1 Background

For an image sphere, the equation relating the rigid motion of the camera with image motion is given in Equation 1 [27]. At an image point $\mathbf{r}$ (refer Figure 1), the optical flow $\dot{\mathbf{r}}$ resulting from the translational motion $\mathbf{t}$ and the rotational motion $\mathbf{w}$ is given by:

$$\dot{\mathbf{r}} = \frac{1}{\mid \mathbf{R}(\mathbf{r}) \mid}((\mathbf{t} \cdot \mathbf{r})\mathbf{r} - \mathbf{t}) - \mathbf{w} \times \mathbf{r} \tag{1}$$



(a)

Fig. 1. For some translational motion, $\mathbf{t}$ and some rotation about the axis $\mathbf{w}$, at point $\mathbf{r}$ on the image sphere, with scene depth $\mathbf{R}$, the optical flow is $\dot{\mathbf{r}}$ and the flow vector lies on the tangent plane to the sphere at that point.

The self-motion estimation task attempts to recover the translational and rotational motions, where it is well-known that the translation can only be recovered up to a scale [2]. A least squares solution is not possible since the scene depth, $\mathbf{R}$, is a function of $\mathbf{r}$ and the system of equations is under-constrained. Without any knowledge of depth, recovering the five unknown motion param-

eters is difficult.

In this work, the constraints described will recover the direction of translation or the epipole. The epipole can be defined as the intersection of the line joining the two camera centres with the image sphere [2]. The second camera center is related to the first camera center via some translation, so the direction of translation and the epipole are equivalent.

Note that the above equation is an approximation that only holds for small baselines and small rotations. This assumption is generally true in image sequence videos, where the motion between frames is small. Section 6 discusses how our algorithm can be implemented to give sufficiently real-time, high frame rate estimates of translation for such videos.

In this paper, we use the term 'optical flow' to refer to any measurement or approximation of image motion. This includes image velocities obtained from feature correspondences such as SIFT [28] or Harris corners, as well as image velocity fields obtained using methods like Lucas-Kanade [29] or Horn-Schunk [30]. For the former, point correspondences are found and matched for two images and the velocity vector transforming one point to the other is calculated. The method presented here works with both classes of image motion measurements.

2 Removing Rotation by Summing Flow at Antipodal Points

On the image sphere, the optical flow, $\dot{\mathbf{r}}_\mathbf{1}$ and $\dot{\mathbf{r}}_\mathbf{2}$, at two antipodal points, $\mathbf{r_1}$ and $\mathbf{r_2}$, can be written as:

$$\dot{\mathbf{r_1}} = \frac{1}{\mid \mathbf{R}(\mathbf{r_1}) \mid}((\mathbf{t} \cdot \mathbf{r_1})\mathbf{r_1} - \mathbf{t}) - \mathbf{w} \times \mathbf{r_1} \tag{2}$$

$$\begin{aligned} \dot{\mathbf{r_2}} &= \frac{1}{\mid \mathbf{R}(\mathbf{r_2}) \mid}((\mathbf{t} \cdot \mathbf{r_2})\mathbf{r_2} - \mathbf{t}) - \mathbf{w} \times \mathbf{r_2} \\ &= \frac{1}{\mid \mathbf{R}(-\mathbf{r_1}) \mid}((\mathbf{t} \cdot \mathbf{r_1})\mathbf{r_1} - \mathbf{t}) + \mathbf{w} \times \mathbf{r_1} \end{aligned} \tag{3}$$

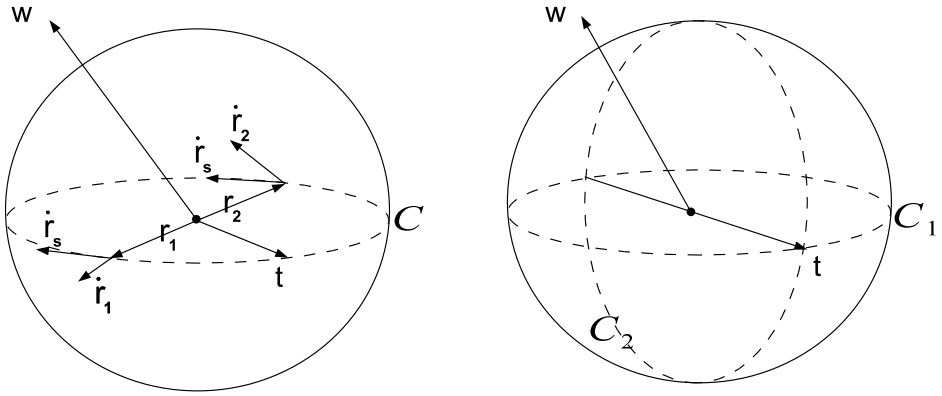
since $\mathbf{r_2} = -\mathbf{r_1}$ if they are antipodal. By summing Equations 2 and 3, we have an expression that arises purely from the translational component of motion. The rotational components cancel out.

$$\begin{aligned} \dot{\mathbf{r}}_\mathbf{s} &= \dot{\mathbf{r_1}} + \dot{\mathbf{r_2}} \\ &= (\frac{1}{\mid \mathbf{R}(\mathbf{r_1}) \mid} + \frac{1}{\mid \mathbf{R}(-\mathbf{r_1}) \mid})((\mathbf{t} \cdot \mathbf{r_1})\mathbf{r_1} - \mathbf{t}) \\ &= K((\mathbf{t} \cdot \mathbf{r_1})\mathbf{r_1} - \mathbf{t}) \end{aligned} \tag{4}$$

From Equation 4, we see that vectors $\mathbf{r_1}$, $\dot{\mathbf{r}}_\mathbf{s}$ and $\mathbf{t}$ are coplanar. The normal of that plane is given by $\mathbf{r_1} \times \dot{\mathbf{r}}_\mathbf{s}$ where $\times$ denotes the vector cross product. The epipole or direction of translation, $\mathbf{t}$, lies on that plane. The intersection of that plane with the image sphere gives a great circle. See Figure 2(a) for an illustration.

By picking another pair of antipodal points (that do not lie on the first great circle) and repeating, we obtain a second such plane or great circle. The intersection of the two planes or great circles gives an estimate of the epipole, $\mathbf{t}$. See Figure 2(b) for an illustration.

Of course, the intersection of the two great circles actually yields two points corresponding to $\mathbf{t}$ and $-\mathbf{t}$. To disambiguate between the two, one may pick one of the two points and calculate the angle between it and the vector $\dot{\mathbf{r}}_\mathbf{s}$. If the angle is larger than $\pi/2$ radians, then that point is $\mathbf{t}$. Otherwise, it is $-\mathbf{t}$.



(a) (b)

Fig. 2. (a) Summing the flow $\dot{\mathbf{r}}_1$ and $\dot{\mathbf{r}}_2$ yields the vector $\dot{\mathbf{r}}_s$, which, together with $\mathbf{r}_1$ (or $\mathbf{r}_2$) , gives rise to a plane on which $\mathbf{t}$ is constrained to lie. The intersection of that plane with the sphere is the great circle C. (b) From two pairs of antipodal points, two great circles C_1 and C_2 are obtained. $\mathbf{t}$ is the intersection of these two circles.

2.1 Comparison with the constraint of Thomas & Simoncelli [20]

The underlying principle here was first observed in [20]. However, major differences differentiate this work from that of [20]. The approach used there defined *angular flow*, which is obtained by taking the cross product of optical flow at a point with the direction of that point. In effect, a dual representation of flow is obtained and [20] shows that if the angular flow at two antipodal points was *subtracted* from each other, the rotational component would vanish.

Both the constraint developed here and the one in [20] probably stem from a similar geometrical property inherent to antipodal points but the methods by which they were derived and the final results are quite different. The constraint in this paper does not require the transformation of optical flow into angular

flow as in [20], a step which incurs an additional cross product.

Furthermore, the method of [20] requires a *subtraction* of angular flow at antipodal points and thus, that method fails when the world points projecting onto the antipodal points on the imaging device are at equal distances from the camera centre (subtracting the angular flow in that case yields zero) - a problem observed by the authors of that paper. An example of this would be the distances to the two opposite walls when the camera is exactly in the middle of a corridor. The method presented in this paper *sums* the optical flow at antipodal points and therefore, does not encounter such instabilities when antipodal scene points are at equal, or close to being at equal distances.

Finally, because omnidirectional and panoramic sensors were not widely available then, the method of [20] was not fully developed into an algorithm and no implementations or experiments were ever conducted. Here two complete algorithms are presented, tested on simulations under increasing noise and outliers, and demonstrated to work on fairly difficult real image sequences.

3 Obtaining robust estimates of t

We now have a method for obtaining some estimate of the direction of $\mathbf{t}$ given the flow at any pair of antipodal points. In this section, we outline methods for doing this *robustly*. Two classes of approaches are possible and we present an algorithm representative of each class in Sections 3.1 and 3.2.

Class 1: Point-Based - Firstly, the flow at two pairs of antipodal points gives a unique (up to a scale) solution for the location of the epipole from the intersection of two great circles. Repeatedly picking 2 pairs from a set of N

antipodal point pairs can give up to ${}^{N}C_2 = \frac{N!}{2!(N-2)!}$ possible solutions, where each candidate solution is a point on the image sphere. We loosely term the class of methods that estimates the best solution from a set of point solutions, *point-based* algorithms.

Section 3.1 presents an algorithm representative of this class that finds the solution point best supported by the optical flow data using the RANSAC framework. The *maximum-inlier* method used in [19] is also a member of this class. Moreover, since the points all lie in a Lie-group (which a sphere is), [31] shows that the mean shift algorithm [32] may also be used for finding the mode of the points. K-means is another possibility for clustering points on a sphere [33]. Other algorithms in this class include robust estimators such as [34] and variants of RANSAC such as [35–38].

Class 2: Line/Curve-Based - Alternatively, since the flow at every pair of antipodes yields a great circle which must intersect $\mathbf{t}$, the epipole could be estimated by simultaneously finding the best intersection of many of these great circles. We will later show this can be reduced to the problem of intersecting straight lines. Once again, many algorithms exist, including linear and convex programming. We observe that the problem is quite similar to that of intersecting many vanishing lines to find the vanishing point [42–44], and inspired by a popular solution in this area, we use a Hough-reminiscent voting approach. There are of course myriad variations to Hough voting (such as [22–24,39–41]) and we present an algorithm representative of the class which is not necessarily the optimal solution, but which nevertheless works very well in the experiments.

Algorithms that are a combination of the two classes are also possible, such

as finding the best supported solution with RANSAC, and then doing linear programming using the inlier flow measurements that have been found.

3.1 Antipodal constraint + RANSAC

Algorithm 1 Estimating the epipole - antipodal+RANSAC algorithm

1: Set $M = \infty$, $count = 0$
2: **while** $M > count$ AND $count < MAX$ **do**
3: Select a pair of antipodes $\mathbf{r_1}$, $\mathbf{r_2}$ with optical flow $\dot{\mathbf{r}}_\mathbf{1}$, $\dot{\mathbf{r}}_\mathbf{2}$.
4: $\dot{\mathbf{r}}_\mathbf{s} = \dot{\mathbf{r}}_\mathbf{1} + \dot{\mathbf{r}}_\mathbf{2}$ and $\mathbf{n_1} = \dot{\mathbf{r}}_\mathbf{s} \times \mathbf{r_1}$
5: Repeat for a different pair of antipodes to obtain $\mathbf{n_2}$ such that $\mathbf{n_1} \neq \mathbf{n_2}$
6: Take cross product $\hat{\mathbf{t}} = \mathbf{n_1} \times \mathbf{n_2}$
7: **for** $i = 1$ to N **do**
8: For the ith antipodal pair, find the plane normal $\mathbf{n_i}$
9: If $\frac{\pi}{2} - cos^{-1}(\mathbf{n_i} \cdot \hat{\mathbf{t}}) < \theta_{thres}$, increment support for $\hat{\mathbf{t}}$
10: **end for**
11: Update M, increment *count*
12: **end while**
13: The best supported $\hat{\mathbf{t}}$ is the solution.

In our previous paper, [19], we presented a consensus style algorithm which calculated a measure of the density of the 'cloud' of candidate solution points at each point, and used points with the highest density to obtain the best solution. The idea is that the more solutions there are that are 'close' to a particular candidate solution, the more likely it was that the candidate was a good solution.

Here, we use a different approach, where the quality of a solution is measured not by the number of nearby candidate solutions, but by the proportion of

data points (i.e. optical flow measurements) that fit the solution. Algorithm 1 summarizes our use of the constraint within the RANSAC sampling framework.

Each pair of antipodal flow vectors yields a plane (or great circle) on which the true solution must lie. Step 9 in Algorithm 1 finds the angle between the candidate solution and such a plane. If that angle is less than some threshold, θ_{thres}, we consider that pair of antipodal flow vectors as data which supports the candidate.

M is the number of iterations of the algorithm and is calculated as:

$$M = \frac{log(1-p)}{log(1-w^s)} \tag{5}$$

where p is the probability of choosing at least one sample of data points free of outliers, w is the inlier probability, and s is the number of data points required to obtain a candidate solution.

N is the number of antipodal flow measurements available. MAX is the upper limit on number of iterations to guarantee termination of the algorithm. If several solutions have the same maximum support, then some mean of their directions can be used instead. Details of the RANSAC framework may be found in [2,25].

The method is quite simple but very robust, as the results in Section 5 will show. However, some tuning of the parameters is required for best results. Furthermore, the processing time increases quite quickly as the probability of

outliers and noise in the data increases [1] . If processing time is critical, then a more efficient method is necessary, which leads us to the voting approach in the next section.

3.2 Antipodal Constraint + Voting

Another approach to finding a robust estimate of translation direction, $\mathbf{t}$, was to simultaneously find the best intersection of all the great circles arising from our method of summing flow at antipodal points. An efficient way of doing this is via a Hough-reminiscent voting method as summarized in Algorithm 2.

Our implementation uses coarse-to-fine voting. First, we pick the flow at M_{coarse} pairs of antipodes and obtain M_{coarse} great circles with the steps detailed in Section 2. Having divided the image sphere into a two-dimensional voting table according to the azimuth-elevation convention, we proceed to vote along each great circle. This is the coarse voting stage and the area on the sphere represented by a voting bin can be fairly large. The area with maximum votes is then found, giving us a coarse estimate of translation, $\mathbf{t_{coarse}}$.

For the fine voting stage, a region of the image sphere in the neighborhood of the coarse estimate is projected onto a plane that is tangent to a point on

[1] For any estimation technique, the difficulty of finding a high quality solution increases with the percentage of outliers. If an accurate estimate is sought with high probability, RANSAC and its variants will show increasing time with large proportions of outliers, as exemplified in this paper by the standard approach (see Section 5). This can be mitigated by trading accuracy for processing time to some extent, but in this paper we seek to compare the ability to find a high quality solution.

Algorithm 2 Estimating the epipole - antipodal+voting algorithm

1: **for** $i = 1$ to M_{coarse} **do**

2: Select a pair of antipodes $\mathbf{r_1}$, $\mathbf{r_2}$ with optical flow $\dot{\mathbf{r_1}}$, $\dot{\mathbf{r_2}}$.

3: $\dot{\mathbf{r_s}} = \dot{\mathbf{r_1}} + \dot{\mathbf{r_2}}$

4: The sphere center, vectors $\dot{\mathbf{r_s}}$ and $\mathbf{r_1}$ (or its antipode, $\mathbf{r_2}$) define a great circle, C_i on the image sphere.

5: The spherical coordinates (θ, ϕ) of points on great circle C_i are calculated and coarse voting is performed.

6: **end for**

7: Find the bin with maximum votes. This is the coarse estimate.

8: Choose a projection plane at or close to coarse estimate.

9: **for** $j = 1$ to M_{fine} **do**

10: Repeat steps 2 to 4 to obtain a great circle C_j.

11: Project C_j onto the projection plane to obtain a straight line L_j.

12: Perform fine voting by voting along the straight line L_j.

13: **end for**

14: Find the bin with maximum votes. This is the fine estimate.

the sphere (refer Figure 3). The point can be at the location of the coarse estimate, $\mathbf{t_{coarse}}$, found earlier or anywhere near it. The center of projection is the sphere center and the mapping will project points on the surface of the sphere onto the tangent plane. Let the sphere centre be the origin and $\mathbf{r_{sph}}$ be a point that lies on the unit image sphere. If the plane is tangent to the sphere at a point $\mathbf{n}$, then $\mathbf{n}$ is also a unit normal of that plane. In that case, the projection of $\mathbf{r_{sph}}$ onto the plane is a point, $\mathbf{r_{pln}}$, that lies on the tangent plane and is given by $\mathbf{r_{pln}} = \mathbf{r_{sph}}/(\mathbf{r_{sph}} \cdot \mathbf{n})$.

This is called a gnomonic projection [45]. This projection will map great circles

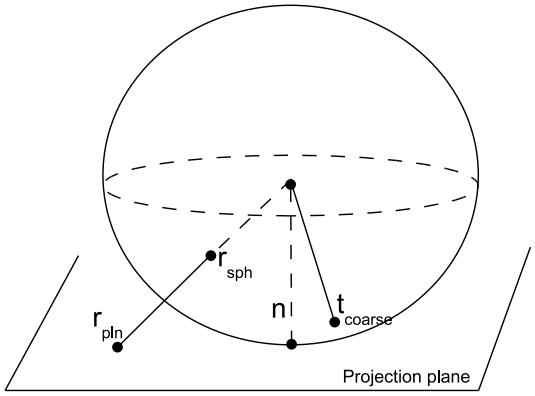


Fig. 3. Projection of a point $\mathbf{r_{sph}}$ which lies on the image sphere, to a point $\mathbf{r_{pln}}$ which lies on the plane that is tangent to the sphere at point $\mathbf{n}$.

on the sphere onto straight lines on the plane. This is an advantageous mapping since very efficient algorithms for voting along straight lines exist, such as the Bresenham line algorithm [46]. This is the motivation for performing the fine voting stage on a projection plane rather than on the image sphere. We vote along M_{fine} straight lines that have been obtained by projecting M_{fine} great circles onto the plane, and the point with maximum votes represents the best estimate of the intersection of all the great circles. If necessary, this can be repeated for progressively finer scales such as with the schemes proposed by [39,41].

One possible variation to this algorithm involves 'blurring' out the votes for the straight lines in a Gaussian fashion. This means the highest votes are cast along the center of the line, with the votes falling off in a Gaussian manner in the direction normal to the line. Furthermore, many optical flow calculation methods return a covariance matrix which gives an uncertainty measure of the accuracy of the calculated flow. Hence, straight lines resulting from flow with high accuracy can be given higher weights during voting compared to lines arising from flow of uncertain accuracy. In our experiments however, we

found that the unmodified version of the algorithm was sufficiently robust and accurate and we did not require any of these extra measures.

4 Experiments - Simulations and Real Videos

Both the algorithms presented here were tested in Matlab simulations and with real image sequences. We compared the performance of our method against the 5-point algorithm within a RANSAC framework [5,6].

We chose 5-point with RANSAC as a comparison since it is well-known, well-studied, widely used and code is widely available. We used the five-point algorithm of [5] (code available from author) and it was implemented within a RANSAC framework using code from [47]. Sampling was adaptive with probability $p = 0.99$ and Sampson distance threshold of 0.01 (see [2] for details).

The error in an estimate of the translation direction was measured as the angle between the recovered motion direction and the true motion (known in simulations and measured in the real videos).

Matlab simulations: We simulated the camera to simultaneously rotate and translate such that an optical flow field was generated. Since the imaging surface is a sphere and flow is available in every direction, we can fix the direction of translation without any loss of generality as the result would be the same in any direction. The axis of rotation however, was varied randomly from trial to trial since the angle made between the direction of translation and axis of rotation does in fact influence the outcome. Baseline was 2 units and the rotation angle 0.2 radians. [2] 500 pairs of random antipodal scene points

[2] If the motion is too small, 5-point may become less stable. By trial and error, we

were uniformly distributed in all directions with depth ranging from 10 to 15 units. Results were averaged over 100 trials.

In the simulations, 5-point used the point matches as input whilst our method took the flow vectors as input. Experiments were conducted for increasing probability of outliers (with no Gaussian noise) and also for an increasing level of Gaussian noise (with no added outliers). Outliers were simulated by randomly replacing matches or flow with errors. To simulate Gaussian noise, the position of a matched point was perturbed from its true value by a noise vector modeled as a 2D Gaussian distribution with standard deviation σ and zero mean. [3]

Real Videos: Real sequences were captured with a Ladybug camera [49] which returns 5 images positioned in a ring. These are mapped onto an image sphere according to a calibration method supplied by Point-Grey Research, the makers of the camera system. The camera translated along the ground with baseline $2cm$ per frame in the direction of the x-axis, which is parallel to the ground plane. It simultaneously rotated about the z-axis, which is perpen-

picked a motion size for which the operating range of both methods overlap. The errors observed were small (Figure 6) so both methods appear to be working stably and properly for the motions used. Also, in the real image experiments for 5-point, we skipped every alternate frame so that the motion was twice as large compared to that used for our method.

[3] Note that this noise model is different from that used previously in [19]. Previously, following [48], we modeled noise as a perturbation in only the *angle* of the flow vector. In this paper the noise model perturbs both angle and magnitude of the flow vector. Thus, for the same σ, the level of noise is roughly higher in this paper compared to in [19].

dicular to the ground plane. The results in Section 5 are for rotation angles of 5° per frame and 2° per frame. The former case involves the camera moving in a static scene whilst the latter involves the camera moving in a scene that contains multiple independently moving objects.

For our algorithms, optic flow was calculated using the iterative Lucas-Kanade method [29] in pyramids using code available from OpenCV [50]. The iterative pyramidal scheme is a multi-scale refinement process that calculated flow quickly and accurately given a list of antipodal points. An example of the typically noisy flow found is shown in Figure 4. Meanwhile, the 5-point with RANSAC method obtained point matches using Scale Invariant Feature Transform (SIFT) feature matching with code from [51].

Let us label the ring of 5 cameras on the Ladybug camera rig as cam0, cam1, ... cam4, thus completing the ring. Figure 4 shows two images taken from cameras on the Ladybug multi-camera rig that were facing different directions. Blue lines pair up antipodes where flow (red vectors) was found stably at both points. The blue lines show the pairing of points in cam2 with their antipodes in cam4.

Note that Figure 4 only shows the antipodal pairs for cam2 and cam4. More antipodes for the cam2 image exist in cam0, and more antipodes for cam4 exist in cam1. Figure 5(a) shows the flow at points in cam2 that had antipodes in cam0 and cam4. Figure 5(b) shows the flow at points in cam4 that had antipodes in cam1 and cam2.

Flow vectors with large error (the flow algorithm returns an uncertainty measure) were thrown away, but a few obvious mismatches remain. However, most of the flow appears reasonably stable. Over the entire view-sphere, for these

experiments, flow was found for about 500 - 700 pairs of antipodes, which is generally more than our algorithms need for a good estimate.

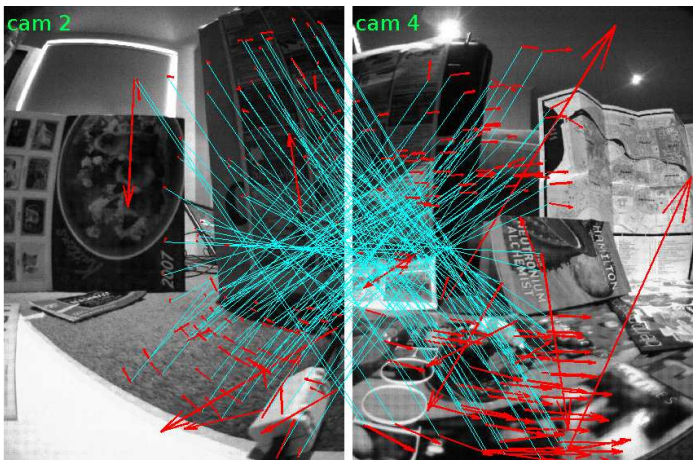
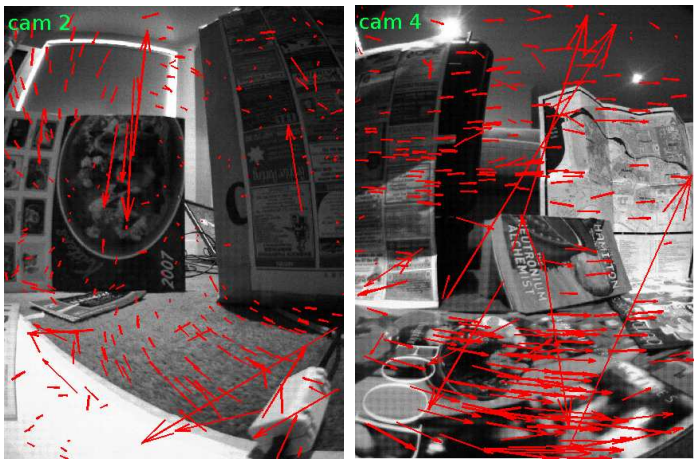


Fig. 4. Blue lines indicate corresponding antipodes in two cameras (cam2 and cam4) facing different directions. Flow vectors shown in red.



(a) (b)

Fig. 5. (a) All the antipodal flow found for the cam2 image using views from cam0 and cam4. (b) All antipodal flow found for the cam4 image using views from cam1 and cam2.

Previously, in [19], we used SIFT matching as an input for the maximum-inlier algorithm presented there. However, SIFT can be slower than Lucas-Kanade flow and is less suitable for real-time, parallel implementations. Therefore, although SIFT tends to be more accurate compared to the noisier, pyramidal

Lucas-Kanade, we chose to use the latter in this paper. In spite of this, Section 5 will show little degradation of the results compared to that reported in [19], which reinforces the point about the robustness of the methods.

In terms of a comparison on the accuracy of 5-point with RANSAC (using SIFT) versus the antipodal point methods (using Lucas-Kanade), this gives a slight disadvantage to the antipodal methods since their inputs are noisier, but as we shall see in the results of Section 5, the performance of both approaches on real images is roughly equivalent.

5 Results

The results summarized in Figure 6 demonstrate that the antipodal+RANSAC and antipodal+voting algorithms, which are based on the sum of flow at antipodal points constraint, work accurately and robustly in simulation and with real image sequences under a variety of circumstances.

Outliers: Outliers are large errors in flow data due to causes such as point mismatches and the presence of objects moving independently in the scene. The RANSAC framework for 5-point is known to be an effective means of separating the inliers and outliers in a data set, so that the estimation is performed only with inliers. However, the simulation result of Figure 6(a) shows that both our algorithms perform better than 5-point with RANSAC when faced with an increasing proportion of random outliers in the data. This is an interesting result and there are several reasons for it, as discussed in the following.

The Hough-like voting algorithm is extremely resistant to outliers. Figure 6(c)

shows the votes cast in a voting table for 60% random outliers in the data. The figure shows the straight lines obtained from projecting great circles onto a plane during the fine-voting stage of the antipodal+voting algorithm detailed in Section 3. Inlier straight lines intersect at a common point whilst the outlier straight lines do not. Most of them fall outside the region of the plane shown in the figure but quite a few outliers can still be seen. From the figure, it is clear that the intersection can easily be found even for this high proportion of outliers.

The antipodal+RANSAC framework worked better because it needed only 2 points for a constraint as opposed to 5 points for the 5-point-RANSAC algorithm. If q is the probability of an inlier, then the probability of obtaining a good constraint is q^2 for the antipodal method and q^5 for 5-point. Therefore, both antipodal methods are always more likely to obtain more good constraints when flow at antipodal points is available. Here, errors in RANSAC arise because some outliers fall within the model distance threshold used (0.01 for 5-point-RANSAC). Reducing distance threshold improves performance; but in general, the antipodal methods tend to outperform 5-point-RANSAC for very large outlier proportions.

Gaussian noise: Figure 6(b) shows the error in estimated translation direction under increasing Gaussian noise. The trend is that all three methods degrade gracefully with increasing noise. For large Gaussian noise, the antipodal point methods tended to perform better than 5-point-RANSAC since outliers would begin to creep into the data at that stage.

Processing time: An interesting property of the antipodal+voting algorithm is its constant processing time. The number of antipodal points considered is

predetermined, and the processing time is quite fast and always constant, regardless of the probability of outliers or amount of noise in the data.

This is in contrast to 5-point-RANSAC which runs in approximately cubic time for our experiments as outliers increase. Theoretically, increasing outliers means that RANSAC will draw more samples from the data according to the formula $M = log(1-p)/log(1-w^s)$ (Equation 5) where $s = 5$ in this case (refer [2] for details). This is illustrated in Figure 6(d). The numbers in 6(d) are obviously implementation dependent but the trend will remain the same.

The antipodal+RANSAC algorithm has similar limitations and since the results show only a small difference in accuracy compared to the voting+antipodal algorithm, we recommend using voting for applications which are time critical.

Real Videos: Figure 6(e) shows the error in the estimated translation direction with respect to measured ground truth. Our algorithms perform with accuracy comparable to 5-point-RANSAC. Figure 6(f) shows the error for a different scene; one containing multiple independently moving objects. The flow arising from these would be outliers whilst the flow arising purely from the the egomotion of the camera are the inliers. Once again, our methods show excellent results. The differences in the average errors of different methods were quite small - around 1° to 2° - which is within the bounds on the accuracy of the ground truth measurements.

In the supplementary material, refer to "(A)frontal - static scene.mpg" for the sequence of Figure 6(e) and "(B)frontal - independently moving objects.mpg" for the sequence of Figure 6(f). Those videos show the view from the frontal camera of the Ladybug camera system, where antipodal estimates, 5-point+RANSAC

estimates and the ground truth are marked as colored crosses.

6 Discussion

Improving the Accuracy of the Voting Method: A consequence of the voting step was the segmentation of antipodal flow data into inliers and outliers. Inliers are the antipodal flows that gave rise to great circles or straight lines (after projection) that intersected at the estimated translation direction.

If a more accurate translation estimate is desired (one not limited by voting bin resolution), a maximum likelihood estimate (assuming Gaussian noise) can be found by finding the intersection of the inlier straight lines by linear least squares. This additional step incurs little extra effort and we note that it is standard practice in motion estimation algorithms to add a linear or non-linear (eg: bundle adjustment) optimization step at the end.

Finding Rotation Robustly: We remind the reader that at this point, we have only recovered the epipole or direction of translation. The rotation was not estimated directly from our method. However, knowing translation, it becomes much easier to then estimate rotation. For instance, taking dot products on both sides of Equation 1 with $\mathbf{t} \times \mathbf{r}$ eliminates dependence on depth:

$$\begin{aligned}(\mathbf{t} \times \mathbf{r}) \cdot \dot{\mathbf{r}} &= (\mathbf{t} \times \mathbf{r}) \cdot (\frac{1}{\mid \mathbf{R}(\mathbf{r}) \mid}((\mathbf{t} \cdot \mathbf{r})\mathbf{r} - \mathbf{t})) - (\mathbf{t} \times \mathbf{r}) \cdot (\mathbf{w} \times \mathbf{r}) \\ (\mathbf{t} \times \mathbf{r}) \cdot \dot{\mathbf{r}} &= (\mathbf{t} \times \mathbf{r}) \cdot (\mathbf{w} \times \mathbf{r})\end{aligned} \tag{6}$$

The estimated translation direction is substituted into Equation 6, giving us

an equation linear in rotation, $\mathbf{w}$. Taking flow at several points, we can build a system of over-constrained linear equations which is solvable by the method of linear least squares. This would be done using only the inlier flow vectors obtained from the antipodal+voting or antipodal+RANSAC algorithms, so the rotation estimate would also be independent of outliers. However, whilst almost all outlier flow vectors were discarded, it is still possible for a few outliers to slip in (called 'leverage points' in statistics). This happens if, by chance, the noise in two outlier antipodal flow vectors cancel when added, giving a great circle that passes through the epipole. Therefore, another layer of outlier rejection may be needed in practice. Fortunately, since previous steps would have reduced the number of unknowns and weeded out almost all outliers, this should converge very quickly.

Speed: The voting approach is potentially a method for estimating motion at high speeds. The flow calculation is typically performed for only a few hundred antipodal points ($\sim$200 to 500) and the mathematical operations used in the voting algorithm are simple - a few dot products, addition and the casting of votes along circles (the coarse-voting stage) or lines (fine-voting). Furthermore, since both the pyramidal Lucas-Kanade flow calculation and the voting step are naturally parallelizable, it is possible to obtain fast implementations using parallel computing architectures such as FPGAs and GPUs. The final step to robustly recover rotation via least squares should incur only a small, additional processing time.

Complex Environments: The voting algorithm's resistance to outliers (Figure 6(a)) and its constant processing time with respect to increasing outliers (Figure 6(d)) would make this method well suited for certain applications and scene environments - for instance, a moving camera in the midst of a crowd

of hundreds of pedestrians. Other examples include estimating vehicle self-motion in busy traffic scenes, or in a windy forest where constantly waving leaves and branches may occupy large sections of the image. Due to the excessively large proportion of outliers from the independently moving objects in the scenes, 5-point-RANSAC would incur a large processing time and may potentially be less accurate, making the antipodal+voting algorithm preferable in such situations.

7 Conclusion

In summary, we have presented a constraint on the epipole arising from the optical flow at two antipodal points. We demonstrated the validity of the constraint and presented two classes of algorithms utilizing it to estimate the location of the epipole. We have shown that this method works robustly and accurately both in simulations and with noisy, real images, giving results comparable to the current state-of-the-art.

Supplementary videos: For the real image sequence of Figure 6(e), see the video "(A)frontal- static scene.mpg" for the frontal camera view of the moving Ladybug camera system. Estimates and ground truth are marked with crosses. For the sequence of Figure 6(f) which is a scene involving multiple independently moving objects, see "(B)frontal- independently moving objects.mpg". The views from the other cameras on the Ladybug that were also used to estimate egomotion are included for the reader's reference.

Acknowledgements: NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Dig-

ital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] H. Longuet-Higgins, A computer algorithm for reconstruction of a scene from two projections, Nature, vol. 293, 1981, pp. 133-135.

[2] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2000.

[3] R. Hartley, In defence of the 8-point algorithm, in: Proceedings of the 5th International Conference on Computer Vision, 1995, pp. 1064-1075.

[4] H. Li, A Simple Solution to the Six-Point Two-View Focal-Length Problem, in: Proceedings of ECCV '06, 2006, pp. 200-213.

[5] H. Li and R. Hartley, Five-Point Motion Estimation Made Easy, in: Proceedings of 18th International Conference of Pattern Recognition, 2006, pp. 630-633. code available at: <http://users.rsise.anu.edu.au/~hongdong/lhd_5ptEssentialfordistribution.zip>

[6] D. Nister, An efficient solution to the five-point relative pose problem, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI '04), 2004, pp. 756-770.

[7] A. R. Bruss and B. K. Horn, Passive navigation, in: Computer Vision, Graphics and Image Processing, vol. 21, 1983, pp. 3-20.

[8] J. W. Roach and J. K. Aggarwal, Determining the movement of objects from a sequence of images, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 2, no. 6, 1980, pp. 55-62.

[9] C. Fermuller and Y. Aloimonos, Qualitative Egomotion, International Journal of Computer Vision, vol. 15, 1995, pp. 7-29.

[10] C. Silva and J. Santos-Victor, Direct Egomotion Estimation, in: Proc. 13th International Conference on Pattern Recognition, 1996.

[11] K. Kanatani, Y. Shimizu, N. Ohta, M.J. Brooks, W. Chojnacki and A. van den Hengel, Fundamental matrix from optical flow: optimal computation and reliability evaluation, Journal of Electronic Imaging, vol. 9, no. 2, April, 2000, pp. 194-202.

[12] A. Jepson and D. Heeger, Subspace methods for recovering rigid motion I: algorithm and implementation, International Journal of Computer Vision, vol. 7, no. 2, 1992, pp. 95-117.

[13] B. Horn and E. Weldon, Direct methods for recovering motion, International Journal on Computer Vision, 1988, pp. 51-76.

[14] S. Negahdaripour and B. Horn, Direct passive navigation, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 1, 1987, pp. 168-176.

[15] J. J. Koenderink and A. J. van Doorn, Invariant Properties of the Motion Parallax Field Due to the Movement of Rigid Bodies Relative to an Observer, Optica Acta, vol. 22, no. 9, 1975, pp.773-791.

[16] T. Y. Tian, C. Tomasi and D. J. Heeger, Comparison of Approaches to Egomotion Computation, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1996, pp. 315-320.

[17] T. Huang and A. Netravali, Motion and structure from feature correspondences: A review, Proceedings of the IEEE, vol. 82, no. 2, February, 1994, pp. 253-268.

[18] R. Nelson and J. Aloimonos, Finding motion parameters from spherical flow fields (or the advantages of having eyes in the back of your head), Biological Cybernetics, vol. 58, 1988, pp. 261-273.

[19] J. Lim and N. Barnes, Estimation of the Epipole using Optical Flow at Antipodal Points, in: OMNIVIS '07, 2007.

[20] I. Thomas and E. Simoncelli, Linear Structure from Motion, Technical Report: IRCS, University of Pennsylvania, 1994.

[21] J. Lim and N. Barnes, Directions of Egomotion from Antipodal Points, in: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR '08), Anchorage, 2008.

[22] P. V. C. Hough, Method and Means for Recognizing Complex Patterns, US Patent 3,069,654, December 18, 1962.

[23] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[24] J. Illingworth and J. V. Kittler, A Survey of the Hough Transform, CVGIP, vol. 44, no. 1, October 1988, pp. 87-116.

[25] M. A. Fischler and R. C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, CACM, vol. 24, no. 6, June 1981, pp. 381-395.

[26] R. Raguram, J. M. Frahm and M. Pollefeys, A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus, In Proc. ECCV 2008, pp. 500-513.

[27] T. Brodsky, C. Fermuller and Y. Aloimonos, Directions of Motion Fields are Hardly Ever Ambiguous, International Journal of Computer Vision, vol. 26, 1998, pp. 5-24.

[28] David G. Lowe, Object recognition from local scale-invariant features, International Conference on Computer Vision, September, 1999, pp. 1150-1157.

[29] B. D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proceedings of Imaging Understanding Workshop, 1981, pp. 121-130.

[30] B. K. P. Horn and B. G. Schunck, Determining optical flows, Artificial Intelligence, vol. 17, 1981, pp. 185-203.

[31] O. Tuzel, R. Subbarao and P. Meer, Simultaneous multiple 3d motion estimation via mode finding on lie groups. In Proc. 10th IEEE Intl. Conf. on Computer Vision, pp. 18-25, 2005.

[32] D. Comaniciu and P. Meer, Mean Shift: A Robust Approach Toward Feature Space Analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 603-619, May 2002.

[33] A. Torii and A. Imiya, The Randomized-Hough-transform-based method for great-circle detection on a sphere, Pattern Recognition Letters, vol. 28, no. 10, pp. 1186-1192, 2007.

[34] H. Chen and P. Meer, Robust regression with projection based M-estimators, Int. Conf. on Computer Vision, pp. 878-885, Oct 2003.

[35] P. H. S. Torr and A. Zisserman, MLESAC: A New Robust Estimator with Application to Estimating Image Geometry, Journal of Computer Vision and Image Understanding, vol. 78, no. 1, pp. 138-156, 2000.

[36] J. Matas and O. Chum, Randomized ransac with $T_{d,d}$ test, Image and Vision Computing, vol. 22, no. 10, pp. 837-842, September 2004.

[37] O. Chum and J. Matas, Optimal Randomized RANSAC, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 8, pp. 1472 - 1482, August 2008.

[38] D. Nister, Preemptive RANSAC for live structure and motion estimation, Machine Vision Applications, vol. 16, no. 5, pp. 321-329, 2005.

[39] M. Atiquzzaman, Multiresolution Hough Transform - An Efficient Method of Detecting Patterns in Images, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 11, pp. 1090-1095, November 1992.

[40] L. Xu, E. Oja and P. Kultanen, A New Curve Detection Method: Randomized Hough Transform, Pattern Recognition Letters, vol. 11, no. 5, pp. 331-338, May 1990.

[41] J. Illingworth and J. V. Kittler, The Adaptive Hough Transform, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 5, pp. 690-698, September 1987.

[42] L. Quan and R. Mohr, Determining Perspective Structures Using Hierarchical Hough Transform, Pattern Recognition Letters, vol. 9, no. 4, 1989, pp. 279-286.

[43] M. J. Magee and J. K. Aggarwal, Determining Vanishing Points From Perspective Images, Computer Vision, Graphics and Image Processing, vol. 26, 1984, pp. 256-267.

[44] J. A. Shufelt, Performance Evaluation and Analysis of Vanishing Point Detection Techniques, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 3, 1999, pp. 282-288.

[45] H. S. M. Coxeter, Introduction to Geometry, 2nd ed. New York: Wiley, 1969, pp. 93 and 289-290.

[46] J. E. Bresenham, Algorithm for computer control of a digital plotter, IBM Systems Journal, vol. 4, no. 1, 1965, pp. 25-30.

[47] P. D. Kovesi, MATLAB and Octave Functions for Computer Vision and Image Processing, School of Computer Science & Software Engineering, The University of Western Australia, available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>

583 [48] J. L. Barron, D. J. Fleet and S. S. Beauchemin, Performance of Optical Flow
584 Techniques, International Journal of Computer Vision, vol. 12, no. 1, 1994, pp.
585 43-77.

586 [49] Point Grey Research, 2007, <http://www.ptgrey.com>.

587 [50] Intel Open Source Computer Vision Library, 2007, available from:
588 <http://sourceforge.net/projects/opencv>.

589 [51] D.G. Lowe, available from: <http://www.cs.ubc.ca/∼lowe/keypoints/>.

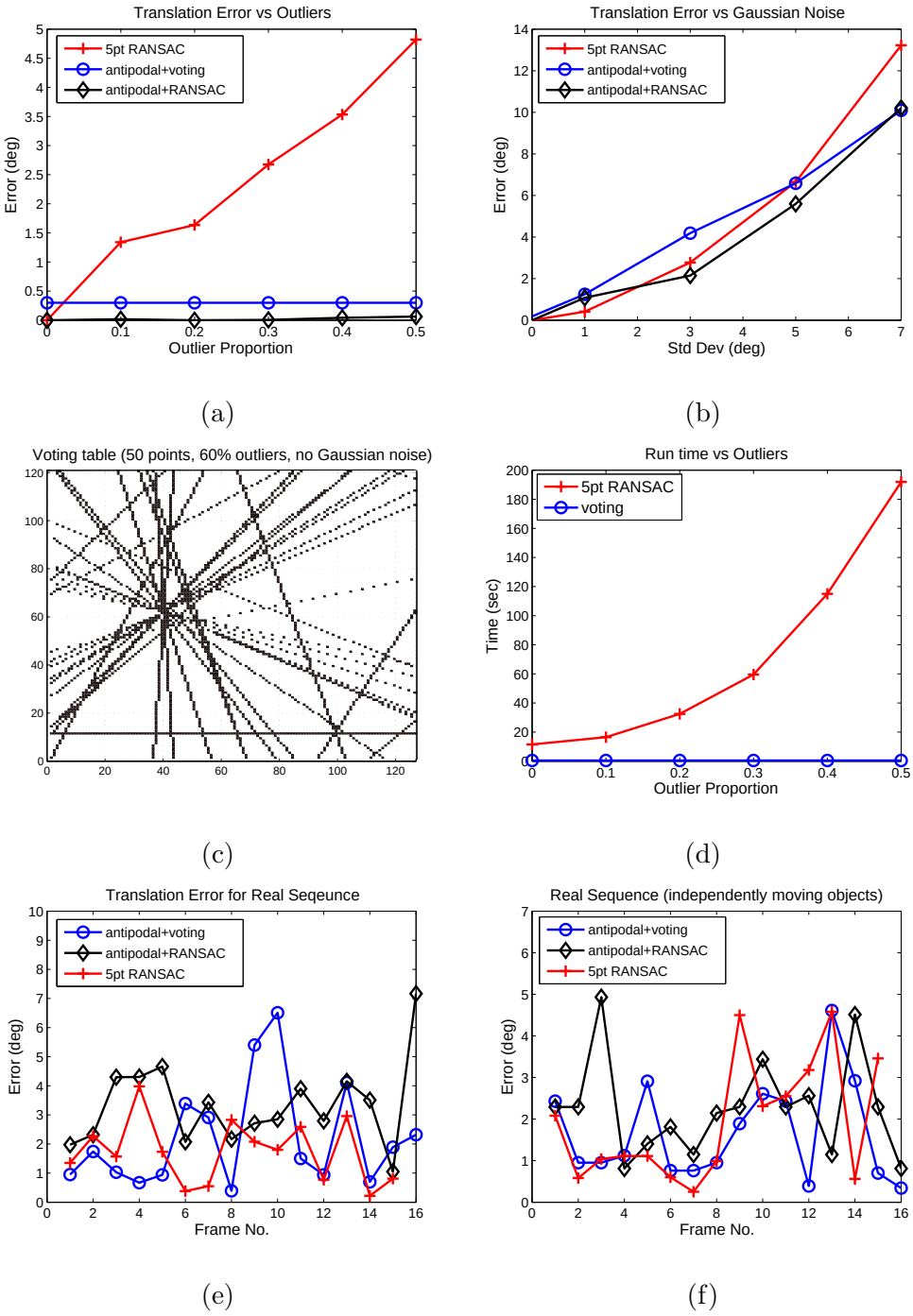


Fig. 6. (a) Error in estimated epipole as proportion of outliers in data increases. (b) Error as Gaussian noise increases. (c) The voted lines for the voting algorithm. Data with 60% outlier probability. (d) Runtime for voting algorithm and 5-point with RANSAC as outlier proportion increases. (e) Error in estimated epipole for real sequence - baseline 2cm/frame, rotation 5°/frame, static scene (f) Error for real sequence - 2cm/frame, 2°/frame, multiple independently moving objects in the scene.