An Iterative Pose Estimation Algorithm Based on Epipolar Geometry With Application to Multi-Target Tracking

Jacob H. White and Randal W. Beard, *Fellow, IEEE*

***Abstract*—This paper introduces a new algorithm for estimating the relative pose of a moving camera using consecutive frames of a video sequence. State-of-the-art algorithms for calculating the relative pose between two images use matching features to estimate the essential matrix. The essential matrix is then decomposed into the relative rotation and normalized translation between frames. To be robust to noise and feature match outliers, these methods generate a large number of essential matrix hypotheses from randomly selected minimal subsets of feature pairs, and then score these hypotheses on all feature pairs. Alternatively, the algorithm introduced in this paper calculates relative pose hypotheses by directly optimizing the rotation and normalized translation between frames, rather than calculating the essential matrix and then performing the decomposition. The resulting algorithm improves computation time by an order of magnitude. If an inertial measurement unit (IMU) is available, it is used to seed the optimizer, and in addition, we reuse the best hypothesis at each iteration to seed the optimizer thereby reducing the number of relative pose hypotheses that must be generated and scored. These advantages greatly speed up performance and enable the algorithm to run in real-time on low cost embedded hardware. We show application of our algorithm to visual multi-target tracking (MTT) in the presence of parallax and demonstrate its real-time performance on a 640 × 480 video sequence captured on a UAV. Video results are available at https://youtu.be/HhK-p2hXNnU.**

***Index Terms*—Aerial robotics, epipolar geometry, multi-target tracking, pose estimation, unmanned aircraft systems, vision-based flight.**

I. Introduction

ESTIMATING camera motion from a video sequence has many applications in robotics including target tracking, visual odometry, and 3D scene reconstruction. These applications often require on-board processing of the video sequence in real-time and thereby impose size, weight, and power (SWAP) constraints on the computing platform.

Manuscript received April 17, 2020; accepted May 10, 2020. This work was funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award Numbers IIP-1161036 and CNS-1650547, along with significant contributions from C-UAS industry members. Recommended by Associate Editor Chenglin Liu. *(Corresponding author: Randal W. Beard.)*

Citation: J. H. White and R. W. Beard, "An iterative pose estimation algorithm based on epipolar geometry with application to multi-target tracking," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 4, pp. 942–953, Jul. 2020.

The authors are with the Brigham Young University, Provo, Utah 84058 USA (e-mail: snowflakeobsidian3.14@gmail.com; beard@byu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2020.1003222

One method to estimate motion from a video sequence is to calculate the essential matrix between consecutive frames. The essential matrix relates the homogeneous image coordinates between frames using the epipolar constraint. After the essential matrix has been determined, it can be decomposed into a rotation and a normalized translation to determine the relative motion of the camera between frames. In order to be robust to noise and feature mismatches, the essential matrix is typically estimated by generating a large number of hypotheses from five-point minimum subsets of matching features, and selecting the best hypothesis using either random sample consensus (RANSAC) [1] or least median of squares (LMedS) [2]. When using RANSAC, the hypotheses are scored by counting the number of inlier points from the entire set. When using LMedS, the hypotheses are scored by calculating the median error.

State of the art methods calculate essential matrix hypotheses directly from each five-point minimum subset. One of the best known methods is Nister's algorithm [3]. Nister showed that for five matching points, there are potentially ten essential matrices that satisfy the constraints, each corresponding to a real root of a tenth-order polynomial generated from the data. There are a many open-source implementation of Nister's five-point algorithm including OpenCV's findEssentialMat function [4]. However, constructing, solving, and extracting the essential matrix from this tenth-order polynomial is complex and can be computationally expensive. Furthermore, since each minimum subset produces up to ten hypotheses, it can be time consuming to score them.

As an alternative to directly calculating essential matrix solutions, some authors [5]–[9] propose solving for the essential matrix using non-linear optimization algorithms such as Gauss-Newton (GN) and Levenberg-Marquardt (LM). Since the essential matrix has nine entries but only five degrees of freedom, the optimization is performed on the five dimensional essential matrix manifold. There are a number of ways to define the essential matrix manifold. Some authors define the manifold using a rotation and translation unit vector, which are elements of $SO(3)$ and S^2, respectively [5], [6]. Others define the manifold using two elements of $SO(3)$ [8], [9].

A third method of optimizing on a manifold is described in [7]. This approach called LM Basis calculates the four essential matrix basis vectors in the nullspace of the essential matrix equation using SVD. The four coefficients to these matrices are solved for on the S^3 manifold. In contrast to the previously described methods which operate on five-dimensional manifolds, this method uses a three-dimensional manifold.

During each iteration of the optimization algorithm, the optimizer step is solved for in terms of the three or five degrees of freedom along the manifold. The computational requirements of the resulting scheme are significantly less than Nister's five point algorithm. However, one weakness of optimization-based solvers is that they only find one of the ten possible essential matrices at a time. Finding all solutions requires additional optimization runs with different initialization points. The optimization method is also sensitive to initial conditions, which can cause the optimizer to fail to produce a valid solution. For example, GN may diverge if the initial guess is too far from the true solution. LM can be used to prevent increases in the cost function, but still may fail to converge. Because of the need to run the optimizer multiple times from different initial conditions, these existing optimization-based solvers might not necessarily be faster than the direct essential matrix solvers if the same level of accuracy is desired. However, not all of the ten possible solutions are needed in order achieve comparable accuracy to direct essential matrix solvers if the best solution can be found the first time.

After the essential matrix is found, it must then be decomposed into a rotation and normalized translation. Given an essential matrix, there are four possible rotation-translation pairs [10]. The correct rotation-translation pair is typically determined using the Cheirality check that ensures that matching features are in front of both cameras. However, the Cheirality check is sensitive to noise in the image and frequently returns the wrong decomposition.

The main contribution in this paper is a novel optimization-based algorithm that directly solves for the relative pose using the epipolar constraint in the cost function. If an inertial measurement unit (IMU) is available, then it is used to seed the optimization algorithm at the next time step. When an IMU is not available, since the rotation and translation between consecutive video frames is similar to nearby frames, we use the relative pose estimate from the previous time step to initialize the optimization at the current time step. At each iteration, we use the current best hypothesis to seed the LM algorithm.

We show that this approach significantly reduces the number of hypotheses that must be generated and scored to estimate the pose, thus allowing real-time execution of the algorithm on a Jetson TX2 processor.

The remainder of the paper is organized as follows. The problem is formally stated in Section II. The new pose estimation algorithm is developed in Section III. Application of the algorithm to target tracking in the presence of parallax is described in Section IV. Simulation and flight results on a quadrotor UAV are presented in Section V, and conclusions are given in section VI.

II. Problem Description

The problem geometry is shown in Fig. 1 where a UAV captures images at time instants k_1 and k_2. The camera frame at time k_i is denoted $\mathcal{F}^{k_i}$. Between time instances, the UAV rotates by $R_{k_1}^{k_2} \in SO(3)$, where $R_{k_1}^{k_2}$ transforms coordinates in $\mathcal{F}^{k_1}$ into coordinates in $\mathcal{F}^{k_2}$, and is translated by $\xi_{k_1/k_2}^{k_2} = \xi_{k_1/I}^{k_2} - \xi_{k_2/I}^{k_2} \in \mathbb{R}^3$, which denotes the position of $\mathcal{F}^{k_1}$ relative to $\mathcal{F}^{k_2}$ expressed in $\mathcal{F}^{k_2}$, and where $\xi_{k/I}$ is the position of frame $\mathcal{F}^k$ in the inertial frame. Superscripts on vectors are used to denote the coordinate frame in which the vector is expressed.

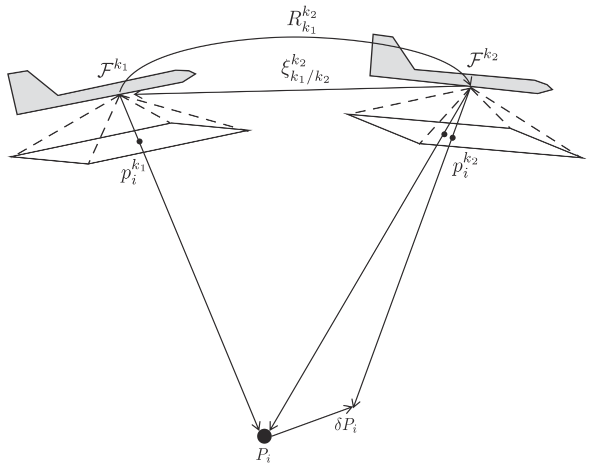

Fig. 1. The geometry for the derivation of the essential matrix.

Suppose that the UAV observes the point P_i in both frames, and let $P_i^{k_1} = \begin{pmatrix} X_i^{k_1} & Y_i^{k_1} & Z_i^{k_1} \end{pmatrix}^T$ and $P_i^{k_2} = \begin{pmatrix} X_i^{k_2} & Y_i^{k_2} & Z_i^{k_2} \end{pmatrix}^T$ be the 3D position of point i with respect to camera frames $\mathcal{F}^{k_1}$ and $\mathcal{F}^{k_2}$, then from the geometry in Fig. 1 we have

$$P_i^{k_2} = R_{k_1}^{k_2} P_i^{k_1} + \xi_{k_1/k_2}^{k_2}. \tag{1}$$

Left multiplying each side of the equation by $\left(\xi_{k_1/k_2}^{k_2}\right)_\times$, where

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_\times \triangleq \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix} \tag{2}$$

gives

$$\left(\xi_{k_1/k_2}^{k_2}\right)_\times P_i^{k_2} = \left(\xi_{k_1/k_2}^{k_2}\right)_\times R_{k_1}^{k_2} P_i^{k_1} + \left(\xi_{k_1/k_2}^{k_2}\right)_\times \xi_{k_1/k_2}^{k_2}. \tag{3}$$

The last term on the right is zero, thus

$$\left(\xi_{k_1/k_2}^{k_2}\right)_\times P_i^{k_2} = \left(\xi_{k_1/k_2}^{k_2}\right)_\times R_{k_1}^{k_2} P_i^{k_1}. \tag{4}$$

Left-multiply each side of (4) by $\left(P_i^{k_2}\right)^T$ give

$$\left(P_i^{k_2}\right)^T \left(\xi_{k_1/k_2}^{k_2}\right)_\times P_i^{k_2} = \left(P_i^{k_2}\right)^T \left(\xi_{k_1/k_2}^{k_2}\right)_\times R_{k_1}^{k_2} P_i^{k_1}. \tag{5}$$

However, the left side of (5) is always zero because the cross product $\left(\xi_{k_1/k_2}^{k_2}\right)_\times P_i^{k_2}$ gives a vector perpendicular to $P_i^{k_2}$, and therefore

$$\left(P_i^{k_2}\right)^T \left(\xi_{k_1/k_2}^{k_2}\right)_\times R_{k_1}^{k_2} P_i^{k_1} = 0. \tag{6}$$

Let

$$p_i \triangleq g(P_i) \triangleq \frac{P_i}{e_3^T P_i}$$

where $e_3 = (0,0,1)^T$ is the normalized homogeneous image coordinates of P_i, and $p_i^{k_1}$ and $p_i^{k_2}$ are the normalized homogeneous image coordinates of point i in camera frame $\mathcal{F}^{k_1}$ and $\mathcal{F}^{k_2}$, respectively. Since the right side of (6) is zero, any scalar multiple of $P_i^{k_1}$ and $P_i^{k_2}$ will also satisfy the equation, which gives the so-called epipolar constraint.

$$\left(p_i^{k_2}\right)^T \left(\xi_{k_1/k_2}^{k_2}\right)_\times R_{k_1}^{k_2} p_i^{k_1} = 0$$

that relates homogeneous coordinates of matching features in two images. Since the epipolar constraint holds for any scaling of $\xi_{k_1/k_2}^{k_2}$, we define $q = \frac{\xi}{\|\xi\|}$ to get

$$\left(p_i^{k_2}\right)^T \left(q_{k_1/k_2}^{k_2}\right)_\times R_{k_1}^{k_2} p_i^{k_1} = 0. \tag{7}$$

The essential matrix is defined as

$$E(R_{k_1}^{k_2}, q_{k_1/k_2}^{k_2}) \triangleq \left(q_{k_1/k_2}^{k_2}\right)_\times R_{k_1}^{k_2}. \tag{8}$$

Since $q_{k_1/k_2}^{k_2} \in S^2$ and $R_{k_1}^{k_2} \in SO(3)$, there are five degrees of freedom associated with E.

The key idea behind our pose estimation algorithm is to optimize the rotation and translation on the manifold $SO(3) \times S^2$ subject to the constraint of (7). The essential matrix is only constructed when it is needed to update the rotation and translation. In contrast, many approaches in the literature solve for the essential matrix directly, which requires decomposing the essential matrix later to obtain the desired rotation and translation.

In the remainder of the paper we drop the frame notation and simply write $R \in SO(3)$ and $q \in S^2$ for the rotation and normalized translation, except where needed for clarity.

III. Iterative Pose Estimation Algorithm

The new pose estimation algorithm proposed in this paper is shown schematically in Fig. 2. At each time step, features are detected in one frame using, for example, OpenCV's goodfeaturesToTrack [11] and then matched to the next frame using, for example, OpenCV's LK optical flow [12]. The corresponding feature pairs are input to the pose estimation algorithm.

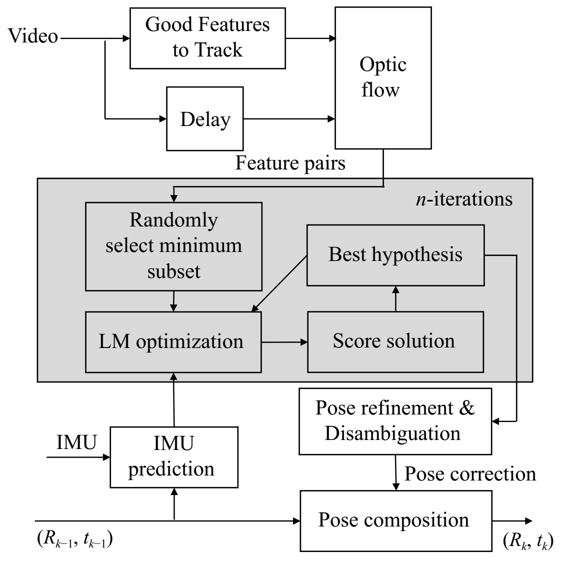

Fig. 2. Block diagram of the proposed recursive pose estimation algorithm.

As shown in Fig. 2, given the set of matching features between images, at each iteration of the optimization loop, a set of N-matching features are randomly selected and the LM algorithm is used to minimize the Sampson error, as explain in Section III-A. The solution of the LM Optimization algorithm is then scored using either LMedS or an inlier count similar to the RANSAC algorithm, as described in Section III-B. If IMU measurements are available, they are used to predict the relative pose between frames, as explained in Section III-C, and are used to seed the LM optimization algorithm. In addition, the algorithm retains the current best hypothesis and uses the best hypothesis to seed the next LM optimization, as also explained in Section III-C. After n-iterations the best hypothesis is then refined using all inlier data, and the rotation and translation are disambiguated, as explained in Section III-D. Finally, the relative pose between frames is composed with the estimated pose at the prior time step to produce the new estimated pose, as explained in Section III-E.

A. LM Optimization

Since the LM optimization block is the core element of the pose estimation algorithm, we explain it first in this section. The LM algorithm is a standard technique for solving nonlinear least squares problems of the form

$$\beta^* = \arg\min r(z,\beta)^T r(z,\beta)$$

where $z \in Z$ is a known data vector, $\beta \in B$ is a parameter vector, and $r : Z \times B \to \mathbb{R}^N$ is a residual vector to be minimized. The optimization problem is solved by iteratively incrementing β by a small δ in a direction that decreases the squared residual

$$S(\beta) = r(x,\beta)^T r(z,\beta).$$

Using the approximation

$$r(z,\beta+\delta) \approx r(z,\beta) + J(z,\beta)\delta$$

where

$$J(z,\beta) \triangleq \frac{\partial r}{\partial \beta}(z,\beta)$$

is the Jacobian of r with respect to the parameters β, we get that

$$S(\beta+\delta) \approx [r(z,\beta) + J(z,\beta)\delta]^T [r(z,\beta) + J(z,\beta)\delta].$$

Taking the partial of $S(\beta+\delta)$ with respect to δ and setting equal to zero gives

$$J^T r = -J^T J\delta. \tag{9}$$

Solving for δ from this equation would result in the GN optimization method. In contrast, gradient descent would replace the right hand side of (9) by $\lambda\delta$, where λ is a scalar. The LM algorithm is a hybrid between GN and gradient descent, requiring δ to satisfy

$$J^T r = -(J^T J + \lambda I)\delta. \tag{10}$$

The general LM optimization algorithm is therefore given by the iteration

$$\beta_{\ell+1} = \beta_\ell + \delta_\ell \tag{11}$$

where

$$\delta_\ell = -\left[J(z,\beta_\ell)^T J(z,\beta_\ell) + \lambda_\ell I\right]^{-1} J(z,\beta_\ell)^T r(z,\beta_\ell)$$

where λ_ℓ is selected at each iteration to ensure that the residual $r(z,\beta)$ decreases.

The remainder of this section explains how the LM algorithm is adapted to the problem of finding relative pose between two camera frames. Let $\mathcal{M}_i^k = (p_i^{k-1}, p_i^k)$ be defined as the ith matching feature pair at time k, and supposing that there are M_k matching features at time k, let $\mathcal{I}_k = \{1,2,\ldots,M_k\}$ denote the index set. Then $\mathcal{M}_{\mathcal{I}_k}$ will denote the set of all matching feature pairs at time k. If $\mathcal{J} \subset \mathcal{I}_k$, then $\mathcal{M}_{\mathcal{J}} \subset \mathcal{M}_{\mathcal{I}_k}$ denotes a subset of matching feature pairs.

The residual function will be defined in terms of the well-known Sampson error, which scales the epipolar error by the reprojection error in each image [13]. If E^k is the essential matrix at time k, then the associated residual based on the Sampson's error is defined by

$$r_i(\mathcal{M}_i^k, E_k) = \frac{p_i^{kT} E^k p_i^{k-1}}{\sqrt{\left\|\Pi_{e_3} E^k p_i^{k-1}\right\|^2 + \left\|\Pi_{e_3} E^{kT} p_i^k\right\|^2}} \tag{12}$$

where $\Pi_x = I - xx^T$ and $e_3 = (0,0,1)^T$. The residual associated with the set $\mathcal{M}_{\mathcal{J}}$ is a vector of length $|\mathcal{J}|$ constructed by stacking the residual for each matching pair in $\mathcal{M}_{\mathcal{J}}$, and will be denoted as $r(\mathcal{M}_{\mathcal{J}}, E_k)$. Recall from (8) that the essential matrix can be written as

$$E(\tilde{R}, \tilde{q}) = \tilde{q}_\times \tilde{R}$$

where $\tilde{R} \in SO(3)$ and $\tilde{q} \in S^2$, and the "tilde" indicates that we are seeking a correction over one time step. Therefore, there are five degrees of freedom in E. To make the math more transparent, we over-parameterize $\tilde{q} \in S^2$ by using an element of $SO(3)$. Accordingly, given $\tilde{q} \in S^2$ we define the matrix $\tilde{Q}$ such that the last column of $\tilde{Q}^T$ is $\tilde{q}$, the first column of $\tilde{Q}^T$ is selected as any unit vector that is orthogonal to $\tilde{q}$ and the second column of $\tilde{Q}^T$ is selected so that $\tilde{Q}$ is orthogonal. Then $\tilde{q} = \tilde{Q}^T e_3$, and

$$E(\tilde{R}, \tilde{Q}) = [\tilde{Q}^T e_3]_\times \tilde{R} = \tilde{Q}^T e_{3\times} \tilde{Q} \tilde{R}.$$

Therefore, the residual will be denoted as $r(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q})$, where a subset of matching pairs $\mathcal{M}_{\mathcal{J}}$ plays the role of the data vector z in the previous discussion, and $\tilde{R}$ and $\tilde{Q}$ play the role of β.

To form the Jacobian $J(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q})$, we need to take the partial derivative of the residual $r(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q})$ with respect to the three degrees of freedom in $\tilde{R}$ and the two degrees of freedom in $\tilde{Q}$. Toward that end, and to simplify notation, we define the boxplus operator [14] as

$$\boxplus : SO(3) \times \mathbb{R}^3 \to SO(3)$$
$$S \boxplus \delta = \exp(\delta_\times) S$$

where

$$\exp(\delta_\times) = I + \sin(\|\delta\|) \frac{\delta_\times}{\|\delta\|} + (1 - \cos(\|\delta\|)) \frac{\delta_\times^2}{\|\delta\|}.$$

If $\delta = \theta \hat{\omega}$ where $\hat{\omega}$ is unit length, then

$$\frac{\partial(\tilde{R} \boxplus \theta\hat{\omega})}{\partial\theta} = \frac{\partial}{\partial\theta} \exp(\theta\hat{\omega}_\times)\tilde{R}\Big|_{\theta=0} \tag{13}$$

$$= \hat{\omega}_\times \exp(\theta\hat{\omega}_\times)\tilde{R}\Big|_{\theta=0} \tag{14}$$

$$= \hat{\omega}_\times R. \tag{15}$$

Let $\delta_R = (\delta_{R,1}, \delta_{R,2}, \delta_{R,3})^T$ represent the (local) three degrees of freedom in $\tilde{R}$, then the partial derivatives of $\tilde{R} \in SO(3)$ along each of the three degrees of freedom are

$$\frac{\partial \tilde{R} \boxplus \delta_R}{\partial \delta_{R,j}} = \left(e_j\right)_\times \tilde{R}, \qquad j = 1,2,3$$

where $e_1 = (1,0,0)^T$, $e_2 = (0,1,0)^T$, and $e_3 = (0,0,1)^T$.

The normalized translation vector is represented as $\tilde{q} = \tilde{Q}^T e_3 \in S^2$ where $\tilde{Q} \in SO(3)$ only has two degrees of freedom. Let $\delta_Q = (\delta_{Q,1}, \delta_{Q,2})^T$ represent the (local) two degrees of freedom in $\tilde{Q}$, and let

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

then the partial derivatives of $\tilde{Q}$ along each of the two degrees of freedom are

$$\frac{\partial \tilde{Q} \boxplus B\delta_Q}{\partial \delta_{Q,j}} = [e_j]_\times \tilde{Q}, \qquad j = 1,2.$$

Defining $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)^T = (\delta_R^T, \delta_Q^T)^T$, then the Jacobian $J(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q}) \in \mathbb{R}^{|\mathcal{J}| \times 5}$ is given by

$$J(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q}) = \begin{pmatrix} \frac{\partial r_1}{\partial \delta_1} & \frac{\partial r_1}{\partial \delta_2} & \frac{\partial r_1}{\partial \delta_3} & \frac{\partial r_1}{\partial \delta_4} & \frac{\partial r_1}{\partial \delta_5} \\ \frac{\partial r_2}{\partial \delta_1} & \frac{\partial r_2}{\partial \delta_2} & \frac{\partial r_2}{\partial \delta_3} & \frac{\partial r_2}{\partial \delta_4} & \frac{\partial r_2}{\partial \delta_5} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_1} & \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_2} & \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_3} & \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_4} & \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_5} \end{pmatrix} \tag{16}$$

where

$$\frac{\partial r_i}{\partial \delta_j} = \frac{\sqrt{s_i}\left(p_i^k\right)^T \frac{\partial E}{\partial \delta_j} p_i^{k-1} - \left(p_i^k\right)^T E p_i^{k-1} \frac{1}{\sqrt{s_i}} \frac{\partial s_i}{\partial \delta_j}}{s_i} \tag{17}$$

with

$$s_i = \left\|\Pi_{e_3} E p_i^{k-1}\right\|^2 + \left\|\Pi_{e_3} E^T p_i^k\right\|^2 \tag{18}$$

where

$$\frac{\partial s_i}{\partial \delta_j} = p_i^{k-1T} \frac{\partial E^T}{\partial \delta_j} \Pi_{e_3} E p_i^{k-1} + p_i^{k-1T} E^T \Pi_{e_3} \frac{\partial E}{\partial \delta_j} p_i^{k-1}$$
$$+ p_i^{kT} \frac{\partial E}{\partial \delta_j} \Pi_{e_3} E^T p_i^k + p_i^{kT} E \Pi_{e_3} \frac{\partial E^T}{\partial \delta_j} p_i^k$$

and where $\frac{\partial E}{\partial \delta_j}$ are given by

$$\frac{\partial E}{\partial \delta_1}(\tilde{R},\tilde{Q}) = \tilde{Q}^T e_{3\times}\tilde{Q}e_{1\times}\tilde{R}$$
$$\frac{\partial E}{\partial \delta_2}(\tilde{R},\tilde{Q}) = \tilde{Q}^T e_{3\times}\tilde{Q}e_{2\times}\tilde{R}$$
$$\frac{\partial E}{\partial \delta_3}(\tilde{R},\tilde{Q}) = \tilde{Q}^T e_{3\times}\tilde{Q}e_{3\times}\tilde{R}$$
$$\frac{\partial E}{\partial \delta_4}(\tilde{R},\tilde{Q}) = \tilde{Q}^T e_{2\times}\tilde{Q}\tilde{R}$$
$$\frac{\partial E}{\partial \delta_5}(\tilde{R},\tilde{Q}) = -\tilde{Q}^T e_{1\times}\tilde{Q}\tilde{R}.$$

In applying the LM algorithm, we begin with $\lambda = 10^{-4}$ and then adaptively modify λ as follows. If the residual error grows after an LM step, then the update is rejected and λ is doubled. On the other hand, if the error is decreased, then we accept the step and divide λ by two. This ensures that the residual monotonically decreases, while allowing the LM algorithm to converge faster if the residual function is well-behaved. The LM optimization algorithm for $\tilde{R}$ and $\tilde{q}$ is summarized in Algorithm 1.

Algorithm 1 LM Optimization for Pose Refinement

```
1: procedure LM OPTIMIZATION $\mathcal{M}_{\mathcal{J}}, \tilde{R}_0, \tilde{q}_0$
2:   Select $\tilde{Q}_0 \in SO(3)$ such that $\tilde{q}_0 = \tilde{Q}_0^T e_3$.
3:   $\ell \leftarrow 0, \lambda \leftarrow 10^{-4}$.
4:   repeat
5:   Compute Jacobian $J_\ell = J(\mathcal{M}_{\mathcal{J}}, \tilde{R}_\ell, \tilde{Q}_\ell)$
6:   from (16).
7:     repeat
8:       Compute $\delta_\ell \in \mathbb{R}^5$ as
9:         $\delta_\ell \leftarrow \left(J_\ell^T J_\ell + \lambda I\right)^{-1} J_\ell^T r_\ell$
10:        Update $\tilde{R}$ as $\tilde{R}_{\ell+1} \leftarrow \tilde{R}_\ell \boxplus \delta_{\ell,1:3}$
11:        Update $\tilde{Q}$ as $\tilde{Q}_{\ell+1} \leftarrow \tilde{Q}_\ell \boxplus \delta_{\ell,4:5}$
12:        $\lambda \leftarrow 2\lambda$
13:     until $\|r_\ell\| < \|r_{\ell+1}\|$
14:     $\ell \leftarrow \ell + 1, \lambda \leftarrow \lambda/2$
15:   until $\|r_{\ell+1}\| < \epsilon$
16: end procedure
```

B. Outlier Rejection

The optic flow algorithm shown in Fig. 2 produces M_k feature pairs, but it is well known that the process is not robust, and that some of the feature pairs will not actually correspond to the same physical location in the environment. Therefore, the LM optimization algorithm given in Algorithm 1 must be made robust to false feature matches. The most common methods reported in the literature are RANSAC [1] and LMedS [2]. In this paper we will compare the efficacy of these two methods.

First note that since the essential matrix manifold $SO(3)\times S^2$ contains five degrees of freedom, and each feature correspondence gives one constraint, that only $N = 5$ feature points are needed to generate a pose using the LM optimization in Algorithm 1, assuming that the feature pairs are true correspondences, and that the 3D points are not co-linear [10]. To make the process robust to outliers, the basic idea is to solve the LM optimization many times on randomly selected subsets of $|\mathcal{J}| = 5$ feature pairs, to score each pose hypothesis using all of the feature pairs, and then to select the pose with the best score.

Recall that the set of matching feature pairs at time k is given by $\mathcal{M}_{\mathcal{I}_k}$. The basic idea of the RANSAC and LMedS algorithms is to randomly select a minimum subset $\mathcal{M}_{\mathcal{J}} \subset \mathcal{M}_{\mathcal{I}_k}$ where $|\mathcal{J}| = 5 << |\mathcal{I}_k|$. The output of Algorithm 1 then forms a model hypothesis associated with $\mathcal{M}_{\mathcal{J}}$. The model hypothesis is then scored against all feature matches in $\mathcal{M}_{\mathcal{I}_k}$. This process is repeated n times, always retaining the model hypothesis with the best score.

In this paper we consider the following two scoring functions:

$$S_{\text{RANSAC}}(\mathcal{M}_{\mathcal{I}_k}, \tilde{R}, \tilde{Q}) \triangleq \sum_{\mathcal{M}_i^k \in \mathcal{M}_{\mathcal{I}_k}} \mathbb{I}\left(\left\|r(\mathcal{M}_i^k, \tilde{R}, \tilde{Q})\right\| > \tau\right) \tag{19}$$

$$S_{\text{LMedS}}(\mathcal{M}_{\mathcal{I}_k}, \tilde{R}, \tilde{Q}) \triangleq \underset{\mathcal{M}_i^k \in \mathcal{M}_{\mathcal{I}_k}}{\text{median}} \left\{r^2(\mathcal{M}_i^k, \tilde{R}, \tilde{Q})\right\} \tag{20}$$

where $\mathbb{I}(\texttt{boolean})$ is the indicator function (1 when `boolean` is true, and 0 when `boolean` is false), and where τ is the RANSAC threshold that must be selected. Note that for RANSAC, the score is the total number of outliers with a residual greater than τ, whereas for LMedS, the score is the square of the median residual over all matching features.

To determine the number of iterations required for RANSAC or LMeds, following [1] we assume that the event that a matching feature pair is an outlier is a binomial distribution with:

$$\epsilon = \text{Probability that } \mathcal{M}_i^k \text{ is a false matching pair.}$$

If N is the number of matching points used to generate a model hypothesis, then the probability that all point correspondences used to generate the model are inliers is

$$p_m = (1-\epsilon)^N. \tag{21}$$

If n model hypotheses are generated, then the probability that at least one of models is generated using only inliers is

$$p = 1 - \left(1 - (1-\epsilon)^N\right)^n. \tag{22}$$

Solving for the number of hypotheses needed to achieve a desired confidence level p, gives

$$n(p) = \frac{\log(1-p)}{\log\left(1-(1-\epsilon)^N\right)}. \tag{23}$$

For example, achieving a 99% confidence ratio when the outlier ratio is 50% and when minimal subsets of $N = 5$ matching features are used to create model hypotheses requires $n = 145$ RANSAC or LMedS iterations.

C. IMU Predication and LM Seeding

In this section, we address the issue of initializing the LM optimization given in Algorithm 1. We are particularly interested in the robotic situation where an IMU, synchronized to the camera, is available to resolve the scale ambiguity. We will also discuss the case where IMU measurements are not available. The discussion in this section follows in some respects, the development in [15].

For the sake of clarity, in this section we will again explicitly specify the different coordinate frames. Let $R_k^I \in SO(3)$ be the rotation from the camera frame at time k, $\mathcal{F}^k$ to the inertial frame $\mathcal{F}^I$, and let $\xi_{k/I}^k$ and $v_{k/I}^k$ be the position and velocity of the camera at time k relative to the inertial frame, expressed in the camera frame $\mathcal{F}^k$, let $a_{k/I}^k$ be the measured specific acceleration of the camera expressed in $\mathcal{F}_k$, and $\omega_{k/I}^k$ be the measured angular velocity of the camera relative to the inertial frame, as expressed in the camera frame $\mathcal{F}^k$, where we have assumed that the IMU biases are known and have been removed from the measurements. Then the kinematics for the camera are given by

$$\begin{aligned} \dot{R}_k^I &= R_k^I \left(\omega_{k/I}^k\right)_\times \\ \dot{v}_{k/I}^k &= R_k^{IT} g^I + a_{k/I}^k \\ \dot{\xi}_{k/I}^k &= v_{k/I}^k \end{aligned} \tag{24}$$

where g^I is the gravity vector in the inertial frame. Let T_s be the sample period of the IMU, and assume that there are m IMU samples between camera frames. We will use the notation $\kappa_0, \kappa_1, \kappa_2, \ldots, \kappa_m$ to denote the intermediate sample from $k-1$ to k, implying that the time instance κ_0 corresponds to the time at image frame $k-1$, and κ_m corresponds to the time at image frame k. Then, integrating over one sample of the IMU and assuming that the measurements are constant over the sample period, we get

$$\begin{aligned} R_{\kappa_{i+1}}^I &= R_{\kappa_i}^I \exp\left((\omega_{\kappa_i/I}^{\kappa_i})_\times T_s\right) \\ v_{\kappa_{i+1}/I}^{\kappa_{i+1}} &= R_{\kappa_i}^{\kappa_{i+1}} v_{\kappa_i/I}^{\kappa_i} + R_{\kappa_{i+1}}^{IT} g^I T_s + R_{\kappa_i}^{\kappa_{i+1}} a_{\kappa_i/I}^{\kappa_i} T_s \\ \xi_{\kappa_{i+1}/I}^{\kappa_{i+1}} &= R_{\kappa_i}^{\kappa_{i+1}} \xi_{\kappa_i/I}^{\kappa_i} + v_{\kappa_{i+1}/I}^{\kappa_{i+1}} T_s \end{aligned}$$

where

$$R_{\kappa_i}^{\kappa_{i+1}} = R_{\kappa_{i+1}}^{IT} R_{\kappa_i}^I = \exp\left((\omega_{\kappa_i/I}^{\kappa_i})_\times T_s\right)^T.$$

The predicted pose after m IMU samples is therefore

$$\tilde{R} = R_{\kappa_0}^{\kappa_m} = R_{\kappa_m}^{IT} R_{\kappa_0}^I \tag{25}$$

$$\tilde{q} = \frac{R_{\kappa_0}^{\kappa_m} \xi_{\kappa_0/I}^{\kappa_0} - \xi_{\kappa_m/I}^{\kappa_m}}{\left\| R_{\kappa_0}^{\kappa_m} \xi_{k_0/I}^{k_0} - \xi_{k_m/I}^{k_m} \right\|}. \tag{26}$$

The predicted relative pose $(\tilde{R}, \tilde{q})$ is used to initialize Algorithm 1.

When an IMU is not available, Algorithm 1 can be seeded with the identity rotation $\tilde{R} = I$ and a randomly selected translation unit vector $\tilde{q}$. This method we will call the "random initialization" method. Another alternative is to initialize Algorithm 1 with the best hypothesis from the previous time step. We will denote this method as the "prior" method. A third alternative, that can also be used with or without the IMU, is to initialize Algorithm 1 with the best RANSAC/LMedS hypothesis that has been found so far from previous LM iterations. Since each hypothesis depends on the previous best hypothesis, when the first hypothesis is selected randomly, we will denote this method as the "random recursive" method. When the first hypothesis is the best hypothesis from the IMU, or the prior time step in the case of no IMU, we call it the "prior recursive" method. In Section V we will show results using each of these initialization techniques.

D. Pose Refinement and Disambiguation

Even the best hypothesis from RANSAC and LMedS usually has some error due to noise on the feature matches in the minimum subset used to create the hypothesis. The estimate can be improved by using least-squares optimization over all inlier feature matches. An advantage of our method over traditional five point or eight point algorithms is that the size of the feature set $\mathcal{M}_\mathcal{J}$ in Algorithm 1 is not limited to a fixed number of points. Therefore, the relative pose can be refined by instantiating Algorithm 1 on the entire set of inlier feature matches.

To determine inlier points, we use the robust inlier detection method described in [16] by setting

$$\hat{\sigma} = 1.4826 \left[1 + \frac{5}{|\mathcal{I}_k| - N}\right] \sqrt{\text{median}\{r(\mathcal{M}_i^k, \tilde{R}, \tilde{q})^2\}}$$

where $N = 5$ is the number of feature matches used to create the model and $|\mathcal{I}_k|$ is the total number of feature matches at time k. A feature match $\mathcal{M}_i^k$ is determined to be an inlier if the Sampson residual $r(\mathcal{M}_i^k, \tilde{R}, \tilde{q})^2 < 2.5\hat{\sigma}$.

It is well known that if the epipolar constraint (7) is satisfied for $(\tilde{R}, \tilde{q})$, that it is also satisfied for $(\tilde{R}', \tilde{q})$, $(\tilde{R}, -\tilde{q})$, and $(\tilde{R}', -\tilde{q})$, where $\tilde{R}'$ is a 180 degree rotation about $\tilde{q}$, i.e.,

$$\tilde{R}' = \left(I + 2\tilde{q}_\times^2\right)\tilde{R}.$$

Algorithm 1 will find one of these four solutions, but the result may not correspond to the correct pose, particularly in the case when the IMU is not present and the initial translation is selected randomly. This is a well-known problem with 5-point and 8-point solvers of the essential matrix. The correct pose is typically determined using the Cheirality check, which involves triangulating each feature match to determine its 3D position, and then selecting the relative pose that puts all 3D points in front of the camera. However, the Cheirality check often gives spurious results. As an alternative, since the two possible rotations $\tilde{R}$ and $\tilde{R}'$ are always 180 degrees apart, we can pick the rotation with the smallest angle and use the Cheirality check to find the correct translation. Since

$$\text{tr}(\tilde{R}) = 1 + 2\cos\theta$$

where θ is the eigen-angle, and since we are expecting small camera deviations between time samples, we select $\tilde{R}$ or $\tilde{R}'$ based on which one has the largest trace. The translation $\tilde{q}$ or $-\tilde{q}$ is then selected based on the Cheirality check.

E. Pose Composition

The global pose at time $k-1$ is given by $(R_{k-1}^I, \xi_{k-1/I}^I)$. The current estimated incremental pose is $(\tilde{R}, \tilde{q}) = (R_{k-1}^k, q_{k-1/k}^k)$. We use the IMU prediction to scale q which gives

$$\xi^k_{k-1/k} = \left\| R^{\kappa_m}_{\kappa_0} \xi^{\kappa_0}_{\kappa_0/I} - \xi^{\kappa_m}_{\kappa_m/I} \right\| \tilde{q}. \tag{27}$$

Accordingly, the pose at time k is

$$R^I_k = R^I_{k-1} R^{kT}_{k-1} \tag{28}$$

$$\xi^I_{k/I} = \xi^I_{k-1/I} - R^I_{k-1} R^{kT}_{k-1} \xi^k_{k-1/k}. \tag{29}$$

F. Algorithm Summary

A summary of the proposed iterative pose estimation scheme is give in Algorithm 2 below.

Algorithm 2 Iterative Pose Estimation

```
 1: procedure ITERATIVE POSE ESTIMATION
 2:   Input: $(I_{k-1}, I_k)$, $(a^{\kappa_i}_{\kappa_i/I}, \omega^{\kappa_i}_{\kappa_i/I})^m_{i=1}$, $(R^I_{k-1}, \xi^I_{k-1/I})$
 3:   From images $I_{k-1}$ and $I_k$ determine set
 4:       of matching features $\mathcal{M}_{\mathcal{I}_k}$.
 5:   Initialize relative pose $(\tilde{R}_0, \tilde{q}_0)$ using either:
 6:       (1) Initialization: $\tilde{R}_0 = I$, $\tilde{q}_0 =$ random.
 7:       (2) Prior: best hypothesis from previous step.
 8:       (3) IMU: initialize using (25) and (26).
 9:   Determine $n(p)$ from (23).
10:   for $i = 1$ to $n$ do
11:       Randomly generate $\mathcal{J}_i \subset \mathcal{I}_k$ where $|\mathcal{J}_i| = 5$,
12:           with corresponding features $\mathcal{M}_{\mathcal{J}_i}$.
13:       Let $(\tilde{R}_i, \tilde{q}_i) \leftarrow$ LM Optimization $(\mathcal{M}_{\mathcal{J}_i}, \tilde{R}_0, \tilde{q}_0)$
14:           using Algorithm 1.
15:       Score $(\tilde{R}_i, \tilde{q}_i)$ using (19) or (20).
16:       Let $(\tilde{R}^*, \tilde{q}^*) \leftarrow \arg\max_i S(\mathcal{M}_{\mathcal{I}_k}, \tilde{R}_i, \tilde{q}_i)$ be the
17:           pose correction with the current lowest score.
18:   end for
19:   Using $(\mathcal{I}_k, \tilde{R}^*, \tilde{q}^*)$, determine inliers and refine the
20:       pose with those inliers, following Section III-D.
21:   Update the pose according to (27)–(29).
22: end procedure
```

IV. Motion Detection and Tracking in the Presence of Parallax

One application of relative pose estimation is motion detection and tracking in the presence of parallax. Motion detection is a valuable source of information in target tracking applications. It can be used to track objects without any prior knowledge about their appearance, in contrast to many trackers that are designed to track specific classes of objects.

There are many successful image-based background subtraction techniques in the literature that work on stationary cameras. In order for image differencing techniques to work on a moving camera, the images must first be aligned using a homography. While this works well for planar scenes, if there is parallax, artifacts will appear in the difference image. If the parallax is small enough in comparison to the movement of objects in the scene, the effects of parallax can be reduced using simple morphological operations and gradient suppression [17].

In the presence of strong parallax, however, a better motion model that accounts for depth variation must be used. There are several methods in the literature that use a geometric model to describe the motion of tracked points in the scene over time. For example, [18] uses orthographic projections to segment moving objects, which works well if the camera is far from the scene or has a narrow field of view. Another approach maintains an affinity matrix and uses principal component analysis to segment moving objects [19]. Another approach uses multiple-frame geometric constraints [20], [21]. However, all of these methods can be computationally prohibitive. In contrast, the technique proposed in this paper exploits the two-frame epipolar constraint and is therefore computationally simple, enabling real-time performance.

A. Motion Detection Algorithm

Given two consecutive frames, with point correspondences detected in each frame, the objective is to determine which points are from stationary objects and which are from moving objects. In this section we assume that the relative pose between cameras $(\tilde{R}, \tilde{q})$ has been calculated using Algorithm 2. The goal is to design a detector $\phi\left(\mathcal{M}^k_i, \tilde{R}, \tilde{q}\right)$ which returns 1 if the feature pair $\mathcal{M}^k_i$ is sourced from a moving object and 0 if it is sourced from a stationary background object. The output of the motion detector is used as an input to a tracking algorithm that produces target tracks as described in Section IV-B.

The essential matrix relates points in one image to the other image with the epipolar constraint. In other words, the essential matrix maps a point in one image to a line in the other image. The location where the point in the other image appears along this line depends on the feature's depth to the camera. As the camera translates, points that are closer to the camera will appear to move more than the points that are far away. This effect is known as parallax.

There are two degrees of freedom for the apparent motion of each point in the image plane. One of these degrees of freedom can be explained by the epipolar constraint if the real-world point is stationary. However, motion along this degree of freedom can also be explained by object motion in the world frame. Hence, the source of any movement along this degree of freedom is ambiguous without additional information. The second degree of freedom for apparent motion of points in the image plane is perpendicular to the epipolar constraint. Thus, the only possible source of motion along this degree of freedom is movement in the real-world frame.

Let $g : \mathbb{R}^3 \to \mathbb{P}^2$ defined by $g(P) = P/e_3^T P$ be the perspective projection operator. With reference to Fig. 1 let $P_i \in \mathbb{R}^3$ be a feature in the world that is imaged by the camera at time k_1 to produce feature $p^{k_1}_i = g(P^{k_1}_i)$. At time time k_2, the feature has moved in the world to position $P_i + \delta P$, and the camera, which has moved by $(\tilde{R}, \tilde{\xi})$ images the feature to produce

$$p^{k_2}_i = g(P^{k_2}_i) = g(\tilde{R} P^{k_1}_i + \tilde{\xi} + \delta P).$$

To streamline the notation, we will drop the i subscript in the following discussion. The epipolar line in frame $\mathcal{F}^{k_2}$ corresponding to the point p^{k_1} is given by

$$\boldsymbol{\ell} = Ep^{k_1} = \tilde{q}_\times \tilde{R} p^{k_1} = \frac{\tilde{\xi}_\times \tilde{R} P^{k_1}}{\|\tilde{\xi}\| e_3^T P^{k_1}}$$

where $e_3^T P^{k_1}$ is the distance that P^{k_1} is from the camera in frame $\mathcal{F}^{k_1}$ and is therefore positive. Note also that the perpendicular to the epipolar line in $\mathcal{F}^{k_2}$ is given by

$$e_3 \times \boldsymbol{\ell} = e_3 \times Ep^{k_1} = \frac{e_{3\times}\tilde{\xi}_\times \tilde{R} P^{k_1}}{\|\tilde{\xi}\| e_3^T P^{k_1}}$$

since e_3 in the camera frame is directed along the optical axis. Our proposed motion detection algorithm is based on the following theorem.

Theorem 1: Let P be a point in the world frame and δP its displacement between times k_1 and k_2. Suppose that the camera moves by $(\tilde{R}, \tilde{\xi})$ between k_1 and k_2 and that $\tilde{\xi} \neq 0$, and let $p^{k_1} = g(P^{k_1})$ and $p^{k_2} = g(P^{k_2}) = g(\tilde{R}P^{k_1} + \tilde{\xi} + \delta P)$ be the image projection of the point P in frame $\mathcal{F}^{k_1}$ and $P + \delta P$ in $\mathcal{F}^{k_2}$, respectively. Let $\boldsymbol{\ell} = \tilde{q}_\times \tilde{R} p^{k_1}$ be the epipolar line in $\mathcal{F}^{k_2}$ corresponding to point p^{k_1} in $\mathcal{F}^{k_1}$, where $\tilde{q} = \tilde{\xi}/\|\xi\|$, and define

$$v_\parallel = (p^{k_2} - g(\tilde{R}p^{k_1}))^T (e_3 \times \boldsymbol{\ell})$$
$$v_\perp = (p^{k_2} - g(\tilde{R}p^{k_1}))^T \boldsymbol{\ell}$$

which can be interpreted as the rotation corrected displacement of the projected position of P in $\mathcal{F}^{k_2}$ projected along the perpendicular to the epipolar line, and the epipolar line, respectively. Then $v_\perp \neq 0$ implies that $\delta P \neq 0$, and $v_\parallel < 0$ implies that $\delta P \neq 0$.

Proof: Note that

$$\begin{aligned} &p^{k_2} - g(\tilde{R}p^{k_1}) \\ &= g(\tilde{R}P^{k_1} + \tilde{\xi} + \delta P) - g(\tilde{R}p^{k_1}) \\ &= \frac{\tilde{R}P^{k_1} + \tilde{\xi} + \delta P}{e_3^T \tilde{R}P^{k_1} + e_3^T\tilde{\xi} + e_3^T \delta P} - \frac{\tilde{R}P^{k_1}}{e_3^T \tilde{R}P^{k_1}} \\ &= \frac{(e_3^T\tilde{R}P^{k_1})\tilde{\xi} - (e_3^T\tilde{\xi})\tilde{R}P^{k_1} + (e_3^T\tilde{R}P^{k_1})\delta P - (e_3^T\delta P)\tilde{R}P^{k_1}}{(e_3^T\tilde{R}P^{k_1} + e_3^T\tilde{\xi} + e_3^T\delta P)e_3^T\tilde{R}P^{k_1}}. \end{aligned} \quad (30)$$

Since $\tilde{\xi}^T\tilde{\xi}_\times\tilde{R}P^{k_1} = (\tilde{R}P^{k_1})^T\tilde{\xi}_\times\tilde{R}P^{k_1} = 0$ we have that

$$v_\perp = \frac{(e_3^T\tilde{R}P^{k_1})\delta P^T \tilde{\xi}_\times \tilde{R}P^{k_1}}{(e_3^T P^{k_2})(e_3^T\tilde{R}P^{k_1})(e_3^T P^{k_1})\|\tilde{\xi}\|}.$$

Clearly $v_\perp \neq 0$ implies that $\delta P \neq 0$.

For $v_\parallel$, note that (30) can be written as

$$p^{k_2} - g(\tilde{R}p^{k_1}) = \frac{e_{3\times}\tilde{\xi}_\times\tilde{R}P^{k_1} + e_{3\times}\delta P\tilde{R}P^{k_1}}{(e_3^T P^{k_2})(e_3^T\tilde{R}P^{k_1})}.$$

Therefore

$$v_\parallel = \frac{\|e_{3\times}\tilde{\xi}_\times\tilde{R}P^{k_1}\|^2 + (e_{3\times}\delta P\tilde{R}P^{k_1})^T(e_{3\times}\tilde{\xi}_\times RP^{k_1})}{(e_3^T P^{k_2})(e_3^T\tilde{R}P^{k_1})(e_3^T P^{k_1})\|\tilde{\xi}\|^2}.$$

Since the denominator and the first term in the numerator are always positive, the only way for $v_\parallel$ to be negative is for the second term in the numerator to be negative and to dominate the first term, which requires that $\delta P \neq 0$.

Therefore, a decision function for whether a point is moving is given by

$$\phi\left(P_i^k, \tilde{R}, \tilde{q}\right) = \begin{cases} 1 & \text{if } |v_\perp| > \tau \text{ or } v_\parallel < -\tau \\ 0 & \text{otherwise} \end{cases} \quad (31)$$

where τ is a positive threshold. Due to small errors in calculating the relative pose and inaccuracies with camera calibration, a threshold of one pixel is the tightest constraint that can be used.

B. Recursive-RANSAC Tracker

Moving points found using (31) are the input to the recursive-RANSAC (R-RANSAC) multiple target tracking algorithm described in [22], [23]. In essence, the R-RANSAC algorithm fits motion model trajectories to sequences of points. For this paper we use a constant velocity motion model. Tracks are initialized when a sufficient number of new data indicate a moving object, and existing tracks are propagated forward using a Kalman Filter. Probabilistic Data Association [24] is used to account for measurement association uncertainty. Each track is given an inlier score based on the percentage of time steps in which the track is detected. R-RANSAC also has a track management system that merges similar tracks and removes tracks that have an inlier score lower than the minimum threshold. For more details see [22], [23].

V. Results

The performance of the pose estimation algorithm will be demonstrated in simulation and with real flight tests. In the simulation study outlined in Section V-A we know the true pose and so we will be able to assess the accuracy of pose estimation. In the flight test outlined in Section V-B, we apply pose estimation to motion detection and tracking as described in Section IV.

A. Pose Estimation - Synthetic Video

Algorithm 2 was tested on a synthetic video sequence of a UAV inside a city generated using the BYU Holodeck simulator [25]. The two-minute video sequence (3600 frames) includes aggressive rotational and translational motions. A screenshot of the video sequence is shown in Fig. 3. The purpose for using the simulator is to compare estimated pose to true pose.

We will consider three error metrics in evaluating Algorithm 2. The rotational error give by

$$e_R = \left\|\log\left(R_{\text{true}}R_{\text{est}}^{-1}\right)\right\| = \frac{\text{tr}\left(R_{\text{true}}R_{\text{est}}^{-1}\right) - 1}{2}$$

is the smallest rotation angle between the estimated and true rotation. The translation error given by

$$e_q = \cos^{-1}\left(q_{\text{true}}^T q_{\text{est}}\right) \quad (32)$$

is the angle between the estimated and true normalized translation. The rotation and translation error metrics do not penalize pose disambiguation errors since they return the same metric for all four possible rotation-translation pairs. Therefore, the third metric is the pose disambiguation metric defined as the percentage of time that Algorithm 2 selects the correct rotation and translation. Both the rotational and translational error are measured in radians. The LMedS

Fig. 3. Screenshot of the holodeck video sequence.

Sampson error is also computed. Unless otherwise noted, all error metrics are averaged over the entire video sequence of 3599 frame pairs.

The error over time for the OpenCV Nister/LMedS polynomial solver [4], LM Basis [7], and Algorithm 2 are shown in Fig. 4. All three algorithms give low error for the UAV trajectory. Notice how the rotation error seems to be proportional to the total rotation, while the translation error becomes very large as the true translation approaches zero.

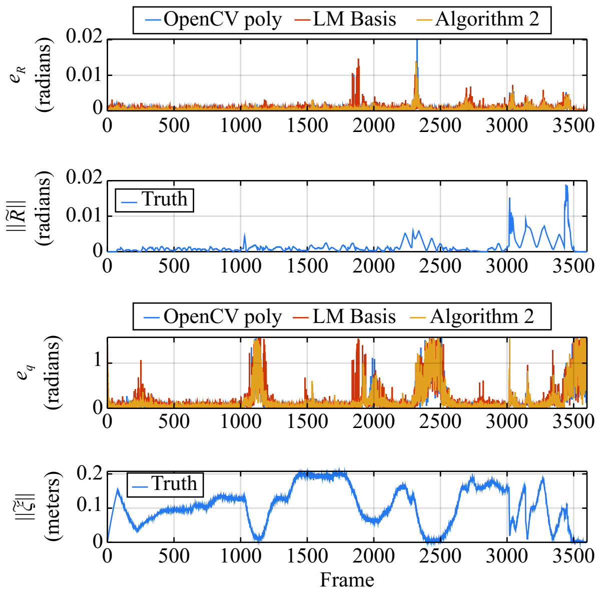

Fig. 4. Incremental rotation and translation error over entire video sequence. True incremental translation and rotation are also shown.

In order to provide a fair comparison of initialization schemes for Algorithm 2 with the OpenCV five-point polynomial solver, the IMU was not used in this section for initialization. Alternatively we compare random initialization, where both $\tilde{R}_0$ and $\tilde{q}_0$ are selected randomly, random recursive initialization, where the first LM optimization at each time step is initialized randomly, but subsequent LM optimizations at that time step use the best pose hypothesis prior, where $(\tilde{R}_0, \tilde{q}_0)$ is the best pose from the previous time step, and prior recursive, where the first LM optimization at each time step is the best pose from the previous time step, but subsequent LM optimizations at that time step use the best pose hypothesis. All results use LMedS for outlier rejection. The LM optimization is repeated 100 times with five matching features at each iteration. The mean error across the entire video sequence is plotted in Fig. 5.

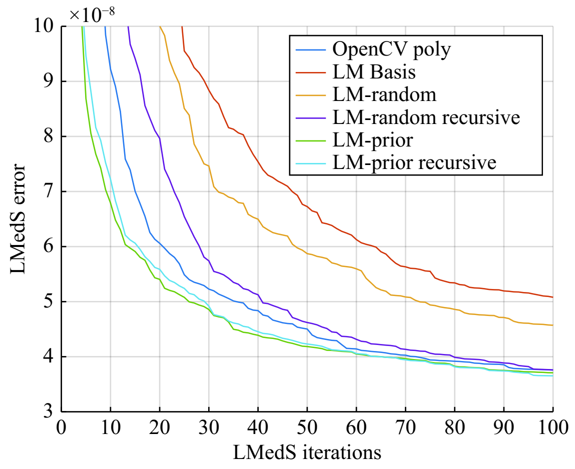

Fig. 5. Comparison of LM seeding methods.

This result shows the importance of initializing the optimizer with a prior. The random initialization method performs the worst out of all four methods, while initializing the optimizer with a prior from the previous time step or the best LMedS hypothesis so far from the current time step significantly reduces the error. IMU prediction will further improve these results. After 100 iterations, the LMedS error for the initialization methods that use prior information is comparable to the OpenCV five-point polynomial solver, despite the fact that only one hypothesis is generated per subset instead of an average of about four hypotheses. LM Basis also generates hypotheses from random seeds, resulting in higher error than methods that initialize from a prior. Note that while LM Basis generates up to 10 hypotheses per subset, it removes duplicates and hypotheses that do not appear to converge, resulting in an average of only 1.82 hypotheses per subset. Dropping these hypotheses early may increase the error based on iteration number, but results in a lower error when compared against time.

Fig. 6 shows the error of Algorithm 2 compared to the OpenCV 5-point algorithm, but with the x-axis changed to be time instead of number of iterations. When under a time constraint, Algorithm 2 significantly outperforms the OpenCV solver [4] and LM Basis [7].

For outlier rejection, RANSAC and LMedS were also compared. For RANSAC the algorithm was tested with 19 different thresholds. For LMedS, the algorithm was run once, because there is no threshold parameter to tune. For each run

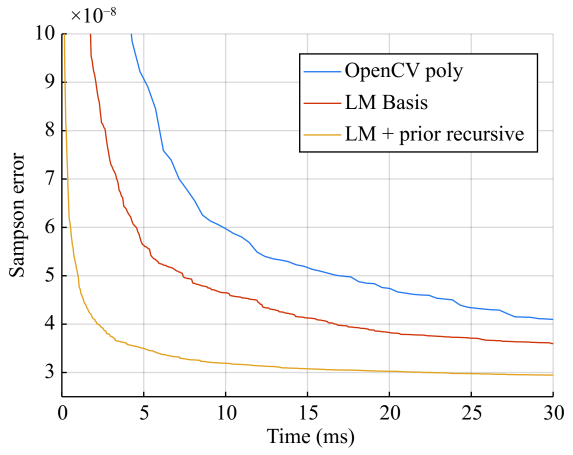

Fig. 6. Sampson error over time.

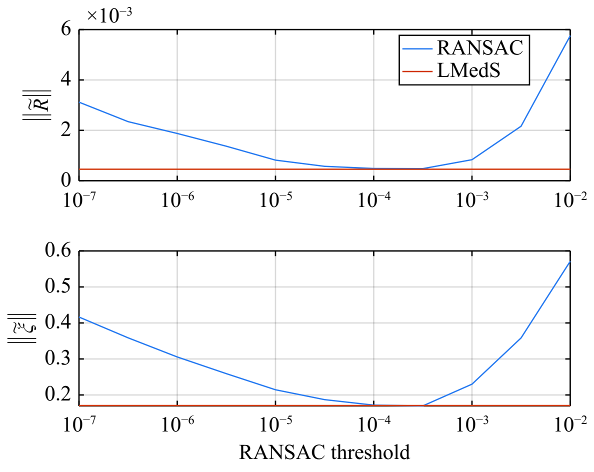

Fig. 7. Average rotation and translation errors for RANSAC and LMedS.

the average truth rotation and translation error over the entire video sequence were calculated. As shown in Fig. 7, LMedS performs well without requiring a threshold. However, in order for RANSAC to perform as well as LMedS, the threshold must be tuned to within an order of magnitude of the optimal threshold.

Table I shows the results of the rotation and translation disambiguation algorithms. The first row within each group shows a baseline comparison, where no method was used for pose disambiguation. The baseline method gives poor results. However, it is worth nothing that Algorithm 2 returns the correct rotation, even without any form of pose disambiguation. This is likely because it is seeded at the first frame with the identity rotation, and in every frame thereafter the best hypothesis from the previous iteration is used as the seed to the optimizer. The second row in each group shows the results when the Cheirality check was used to determine the best of the four possible rotation/translation pairs. The translation direction is often correct but the rotation is correct only about half of the time. The third row in each group shows the results of using the matrix trace to determine which rotation is correct and the Cheirality check to determine the correct translation direction. This third pose disambiguation method consistently outperforms the other methods.

The best hypothesis was refined using LM to optimize the Sampson cost over all inliers (Section III-D). The refined relative pose is only kept if the new relative pose successfully reduces the LMedS error. Table II compares the average error with and without refinement. Refining the best relative pose hypothesis significantly reduces all three error metrics. LM Basis also refines the best hypothesis and successfully reduces the translational and rotational error. However, even with refinement, LM Basis has a higher error than Algorithm 2.

Table III compares the computation time for relative pose estimation between the OpenCV implementation [4], LM

TABLE I
Pose Disambiguation Comparison

Solver	Pose disambiguation method	Rotation correct	Translation correct	Both correct
OpenCV	none	50.2%	14.7%	6.8%
OpenCV	Cheirality	54.0%	93.2%	52.3%
OpenCV	trace + Cheirality	100.0%	96.5%	96.5%
LM Basis	other + Cheirality	100.0%	95.8%	95.8%
Algorithm 2	none	100.0%	57.2%	57.2%
Algorithm 2	Cheirality	40.9%	92.1%	40.8%
Algorithm 2	trace + Cheirality	100.0%	96.5%	96.5%

TABLE II
Relative Pose Refinement

Relative pose solver	Refine (success rate)	Rot err (radians)	Trans err (radians)	LMedS err (Sampson)
OpenCV poly	-	4.843E–04	1.742E–01	3.769E–08
LM Basis	No	7.076E–04	2.059E–01	4.762E–08
LM Basis	Yes	4.159E–04	1.708E–01	5.059E–08
Algorithm 2	No	4.640E–04	1.699E–01	3.653E–08
Algorithm 2	Yes (54.5%)	3.850E–04	1.596E–01	3.540E–08

TABLE III
COMPUTATION TIME

	OpenCV poly	LM Basis	Algorithm 2
Hypothesis generation	100 × 0.404 ms = 40.4 ms	100 × 27.9 us = 2.79 ms	100 × 23.4 us = 2.34 ms
Hypothesis scoring (avg 400 pts)	400 × 17.3 ns = 6.91 ms	182 × 13.9 ns = 2.52 ms	100 × 8.99 ns = 0.90 ms
Refinement	-	2.17 ms	5.91 ns
Pose disambiguation	0.32 ms	1.86 ms	0.14 ms
Total	47.7 ms	9.57 ms	3.97 ms

Basis [7], and Algorithm 2. The algorithms were both implemented on a laptop with a 2.1 GHz Intel i7 CPU running Linux. The breakdown of the time required to generate each hypothesis set is shown in Fig. 8. The most time-consuming part of the OpenCV solver is finding the zeros of the tenth-order polynomial. The most time-consuming part of Algorithm 2 is the Eigen matrix solver. Note that while LM Basis and Algorithm 2 take about the same amount of time to generate hypotheses per subset, LM Basis generates up to 10 hypotheses per subset, while Algorithm 2 only generates one hypothesis. The faster optimization used in the LM Basis algorithm is likely due to solving simpler equations which require inverting a 3×3 matrix instead of a 5×5 matrix.

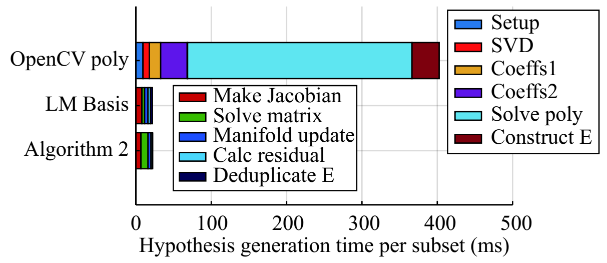

Fig. 8. Time required to generate each hypothesis set.

B. Motion Detection and Tracking - Flight Video

The motion detection algorithm was tested on a moving camera video sequence taken from a multi-rotor UAV. Fig. 9 shows the results of the motion detection algorithm. Note that the stationary points have zero perpendicular velocity and a positive parallax velocity, while the moving points have a non-zero perpendicular velocity component. Fig. 10 shows the results of tracking these moving points using R-RANSAC [22].

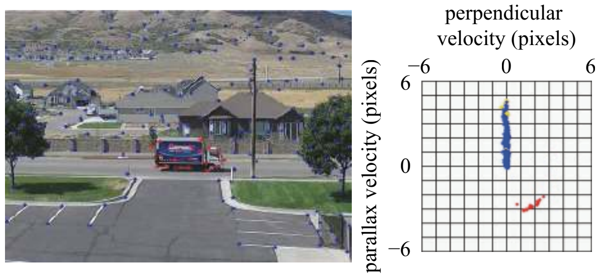

Fig. 9. Video motion detection results. Each point position (left) and its corresponding net velocity (right) are plotted. Points with a net perpendicular velocity greater than one pixel are classified as moving points (red), while points with a velocity below this threshold are classified as stationary points (blue).

The computation times of the motion detection and tracking algorithm are shown in Table IV. For faster processing the video was scaled to 640×480. The motion detection and tracking algorithm is running on a Linux desktop computer with a 4 GHz Intel i7 CPU. On average about 800 points are detected, tracked, and fed to Algorithm 2 each frame. Notice that the OpenCV feature detection and tracking are the most time-consuming components of the tracking algorithm and consume 70% of the total CPU usage. The complete algorithm takes 29 milliseconds to run per frame, which means it is capable of running in real-time at 34 frames per second (FPS).

TABLE IV
MOTION DETECTION AND TRACKING COMPUTATION TIMES

Tracking component	Computation time
Good Features to Track	9.2 ms
LK optical flow	12 ms
Compute $(\tilde{R}, \tilde{q})$ (Algorithm 2)	3.0 ms
R-RANSAC	0.4 ms
Other	4.4 ms
Total	29 ms (34 FPS)

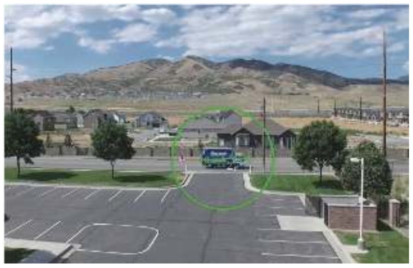

Fig. 10. R-RANSAC tracks.

VI. CONCLUSION

In this paper we have presented a relative pose estimation algorithm for solving the rotation and translation between consecutive frames, that requires at least five matching feature points per frame, and is capable of running in real-time. We show the importance of seeding the LM optimizer with an initial pose estimate and demonstrate that this initial estimate significantly improves the performance of the algorithm. We have applied the algorithm to detecting motion and tracking multiple targets from a UAV and demonstrated real-time performance of this tracking algorithm on a 640×480 video sequence. Future work includes applications to 3D scene reconstruction and more complex tracking methods.

References

[1] 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*, vol. 24, no. 6, pp. 381–395, 1981.

[2] P. J. Rousseeuw, "Least median of squares regression," *J. American Statistical Association*, vol. 79, no. 388, pp. 871–880, 1984.

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

[4] "OpenCV 3.1: Open source computer vision library," [Online]. Available: https://github.com/opencv/opencv/releases/tag/3.1.0, 2015.

[5] Y. Ma, J. Košecká, and S. Sastry, "Optimization criteria and geometric algorithms for motion and structure estimation," *Int. J. Computer Vision*, vol. 44, no. 3, pp. 219–249, 2001.

[6] T. Botterill, S. Mills, and R. Green, "Refining essential matrix estimates from RANSAC," in *Proc. Image and Vision Computing New Zealand*, 2011, pp. 1–6.

[7] T. Botterill, S. Mills, and R. Green, "Fast RANSAC hypothesis generation for essential matrix estimation," in *Proc. IEEE Int. Conf. Digital Image Computing Techniques and Applications*, 2011, pp. 561–566.

[8] U. Helmke, K. Hüper, P. Y. Lee, and J. Moore, "Essential matrix estimation using Gauss-Newton iterations on a manifold," *Int. J. Computer Vision*, vol. 74, no. 2, pp. 117–136, 2007.

[9] M. Sarkis, K. Diepold, and K. Hüper, "A fast and robust solution to the five-point relative pose problem using Gauss-Newton optimization on a manifold," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2007. DOI: 10.1109/ICASSP.2007.365999.

[10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cabridge University Press, 2003.

[11] J. Shi and Tomasi, "Good features to track," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1994. DOI: 10.1109/CVPR.1994.323794.

[12] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *Int. J. Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[13] S. Belongie, "Cse 252b: Computer vision II, lecture 11," [Online]. Available: https://cseweb.ucsd.edu/classes/sp04/cse252b/notes/lec11/lec11.pdf, 2006.

[14] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, Fe. 2017.

[16] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. John Wiley & Sons, 2005, vol. 589.

[17] V. Santhaseelan and V. K. Asari, "Moving object detection and tracking in wide area motion imagery," in *Wide Area Surveillance*. Springer, 2014, pp. 49–70.

[18] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *Proc. IEEE 12th Int. Conf. Computer Vision*, 2009, pp. 1219–1225.

[19] A. Elqursh and A. Elgammal, "Online moving camera background subtraction," in *Proc. European Conf. Computer Vision*. Springer, 2012, pp. 228–241.

[20] J. Kang, I. Cohen, G. Medioni, and C. Yuan, "Detection and tracking of moving objects from a moving platform in presence of strong parallax," in *Proc. 10th IEEE Int. Conf. Computer Vision*, vol. 1, pp. 10–17, 2005.

[21] S. Dey, V. Reilly, I. Saleemi, and M. Shah, "Detection of independently moving objects in non-planar scenes via multi-frame monocular epipolar constraint," in *Proc. 12th European Conf. Computer Vision*, vol. 7576, pp. 860–873, 2012.

[22] P. C. Niedfeldt and R. W. Beard, "Multiple target tracking using recursive RANSAC," in *Proc. American Control Conf.*, Jun. 2014, pp. 3393–3398.

[23] P. C. Niedfeldt, K. Ingersoll, and R. W. Beard, "Comparison and analysis of recursive-RANSAC for multiple target tracking," *IEEE Trans Aerospace and Electronic Systems*, vol. 53, no. 1, pp. 461–476, Feb. 2017.

[24] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *IEEE Control Systems*, vol. 29, no. 6, 2009.

[25] "BYU holodeck: A high-fidelity simulator for deep reinforcement learning," [Online]. Available: https://github.com/byu-pccl/holodeck, 2018.

Jacob H. White received the B.S. and M.S. degrees in electrical engineering from Brigham Young University in 2015 and 2019, respectively. His research interests include robotics, multiple-target tracking, computer vision, and deep learning.

Randal W. Beard (F'15) received the B.S. degree in electrical engineering from the University of Utah, USA, in 1991, the M.S. degree in electrical engineering in 1993, the M.S. degree in mathematics in 1994, and the Ph.D. degree in electrical engineering in 1995, all from Rensselaer Polytechnic Institute, USA. Since 1996, he has been with the Electrical and Computer Engineering Department at Brigham Young University, USA, where he is currently a Professor. His primary research interests include autonomous control of unmanned air vehicles, multiple target tracking, and multivehicle coordination and control. He is a past Associate Editor for the *IEEE Control Systems Magazine*, *Journal of Intelligent and Robotic Systems*, and *IEEE Transactions on Automatic Control.*