2020 American Control Conference
Denver, CO, USA, July 1-3, 2020

VEst: An Efficient Solution to the Camera Velocity Estimation from Minimal Feature Points

Yujie Zhang[1] and Kaveh Fathian[2] and Nicholas R. Gans[3]

Abstract— **This paper presents a new method to determine camera velocity from image data. Velocity estimation has been underexplored, compared to estimating rotation and translation. Existing methods suffer from various drawbacks in terms of necessary conditions on the 3D structure of the scene, the types of camera velocities they can solve for, convergence of estimates, or necessary partial knowledge of the scene or velocity. Our approach uses optical flow vectors and delivers estimates of angular and linear velocities. Rigid body velocity equations are used to derive a polynomial system, which is then solved for the desired velocity terms as well as the current depths of feature points that root the optical flow vectors. Simulations are conducted to validate the correctness and accuracy of the proposed algorithm. The real-world KITTI dataset is used further to demonstrate the performance in practical problem.**

I. Introduction

A classic problem in computer vision is estimating the motion of a camera and the 3D structure of the surrounding environment from the images it captures. Vision-based motion and structure estimation have a central role in visual servoing [1] and visual odometry [2], as well as many practical applications in control and navigation of autonomous vehicles [3], surveillance and person detection [4], and simultaneous localization and mapping (SLAM) [5].

Methods to estimate the rotation and translation motion of a camera, while estimating structure, have been studied for some time. There are two dominant algorithms to solve this problem: finding the *essential matrix* [6] or *homography matrix* [7] and decoupling rotation and translation. The widely-used *eight-point algorithm* to estimate the essential matrix requires at least eight feature points matched in two images. This approach fails if there is no translation movement of the camera or scene, which limits its implementations in real world. The homography matrix needs at least four points, but can only be applied in planar scenes. More recent explorations have developed approaches to estimate the essential matrix from only five points [8]–[10]. These methods do not suffer from the degeneracies that are associated with the eight-point algorithm. However, they require solving the roots of a 10th order polynomial system, thus they are generally slower and admit more solutions, which can be difficult to distinguish. We recently developed the QuEst algorithm [11]. QuEst also requires five points, but it eschews the essential matrix and separately solves for rotation (in the form of quaternion) and translation.

Estimating *camera velocity* or continuous motion from images has received considerably less attention to date. Some existing methods are derived from the essential matrix or the homography matrix [12]. These methods do not need to track points over time, since they use optical flow vectors as algorithm inputs. However, they have similar drawbacks as the traditional matrices, in that the continuous essential matrix needs optical flow from noncoplanar points and nonzero linear velocities, and the continuous homography matrix needs optical flow from coplanar feature points. Another class of methods for vision-based velocity estimation is based on the Extended Kalman Filters (EKF) [13] or EKF combined with continuous homography matrix estimation [14]. EKF methods take time to converge, and cannot generally guarantee convergence. A third class of methods for vision-based velocity estimation is based on nonlinear estimators [15], which can offer convergence guarantees, but take time to converge and often need partial knowledge of the velocity. EKF and nonlinear estimation methods typically take tracked feature points over time, rather than optical flow.

Inspired by the above velocity estimation methods, and our previous work in QuEst, we present, an algorithm for vision-based velocity estimation. Our algorithm requires five optical flow vectors derived from camera images. Rigid body velocity and the pinhole camera model are used to create a system of polynomial equations, the variables of which are camera angular velocities, and the coefficients are composed of the feature coordinates and optical flow vector elements. We solve the system using a novel polynomial solver first described in QuEst. Having solved for angular velocities, we then solve a system of linear equations to obtain the camera linear velocity and the depth and derivative of depth of the 3D points. VEst has the following noteworthy properties:

- It works for any configuration of 3D points (i.e., for coplanar and noncoplanar points) and any camera velocity (e.g., zero linear velocity does not hurt performance).
- Velocity estimation occurs at each time step as a unique nonlinear system of equations. Convergence of the estimate is not an issue, and there is no need to track or match feature points over time, as the five features and their corresponding optical flow vectors need not correspond to the same points frame to frame.
- Simulations and experiments with real-world datasets show that the proposed method works well.

*This work was supported in part by the U.S. Air Force Research Laboratory under Grant FA8651-17-1-0001.

[1]Y. Zhang is with the Deportment of Electrical Engineering, University of Texas at Dallas, USA. Email: yxz162530@utdallas.edu.

[2]K. Fathian and is with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, USA. Email: kavehf@mit.edu.

[3]N. R. Gans is with the UT Arlington Research Institute at the University of Texas at Arlington, USA. Email: ngans@uta.edu

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on February 14,2022 at 01:14:08 UTC from IEEE Xplore. Restrictions apply.

This paper is organized as follows. Section II introduces notation and assumptions. Section III explains the algorithm step-by-step and references numerical examples provided in the Appendix to elaborate each step. Section IV evaluates the algorithm using simulated and real-world datasets.

II. Preliminaries

A. Assumption and Notations

We assume the camera calibration matrix is known using existing calibration routines [16]. Scalars are written in lower cases (e.g., s). Vectors are written in small bold case (e.g., $\mathbf{v}$). Matrices are written in upper bold letters (e.g., $\mathbf{M}$). The Moore-Penrose pseudoinverse of a matrix $\mathbf{M}$ is represented by $\mathbf{M}^{\dagger}$. Given a vector $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^{\top} \in \mathbb{R}^3$, we denote by $\boldsymbol{\omega}_{\times} := \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \in so(3)$ its corresponding skew-symmetric matrix. Binomial coefficients are written as $\binom{n}{k} := \frac{n!}{k!(n-k)!}$.

B. Problem Formulation

Consider a camera moving smoothly, with body velocity $[\boldsymbol{\omega}^{\top}(t), \mathbf{v}^{\top}(t)]^{\top} \in \mathbb{R}^6$, where $\boldsymbol{\omega}(t)$ is the angular velocity and $\mathbf{v}(t)$ is the linear velocity. For any 3D point, its coordinate in the world frame $\mathbf{n}(t) \in \mathbb{R}^3$ obeys the constraint

$$\boldsymbol{\omega}_{\times}(t)\,\mathbf{n}(t) + \mathbf{v}(t) = \dot{\mathbf{n}}(t). \tag{1}$$

In the image frame, point coordinate $\mathbf{n}(t)$ must satisfy

$$\mathbf{n}(t) = u(t)\,\mathbf{m}(t) \tag{2}$$

where $u(t)$ is the distance from the camera's center of projection to the 3D point, and $\mathbf{m}(t) \in \mathbb{R}^3$ is the homogeneous coordinate of the point in the image plane (i.e., the last element of $\mathbf{m}(t)$ is 1). We drop the time dependency henceforth for notation brevity. Substituting (2) into (1) yields

$$u\,\boldsymbol{\omega}_{\times}\,\mathbf{m} + \mathbf{v} = u\,\dot{\mathbf{m}} + \dot{u}\,\mathbf{m}, \tag{3}$$

in which $\dot{\mathbf{m}} \in \mathbb{R}^3$ (with 0 as the last element) is the velocity of the point in the image plane, which is often referred to as *optical flow* [17].

Our goal is to find $\boldsymbol{\omega}$, $\mathbf{v}$, u, and $\dot{u}$ from (3), where feature points $\mathbf{m}$ and optical flow vectors $\dot{\mathbf{m}}$ are given from two or more sequential camera images. A minimum of five feature points and optical flow vectors are needed in our method. In the following section, we explain our algorithm for estimating the unknown variables in (3).

III. The VEst Algorithm

The first step is to find a set of feature points $\mathbf{m}$ and optical flow vectors $\dot{\mathbf{m}}$ given a pair of sequential images taken from the camera. There are many techniques that can be used here, including corner features [18], SIFT features [19], SURF features [20], or ORB features [21]. In our simulation and experiment, we use corner features as they can be found more efficiently than other approaches. Optical flow vectors are computed from the difference of tracked points [22], [23] from each pair of sequential images. An example of

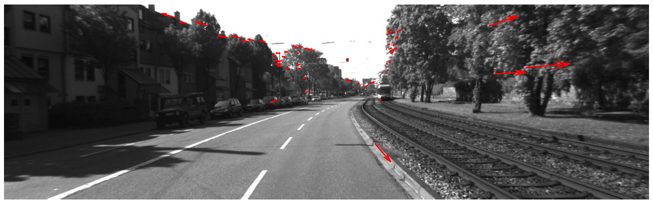

Fig. 1. An image from the KITTI data set, in which the camera faces forward as a car drives along a road. Feature points and optical flow vectors are drawn in red.

optical flow vectors extracted from a sequence of images is shown in fig:example. Five sets of feature points $\mathbf{m}_i$ and their corresponding optical flow vectors $\dot{\mathbf{m}}_i$, $i = 1, \ldots, 5$ are the minimum requirement for solving the velocity estimation problem. Each set can be written in the form of (3) as

$$u_i\,\boldsymbol{\omega}_{\times}\,\mathbf{m}_i + \mathbf{v} = u_i\,\dot{\mathbf{m}}_i + \dot{u}_i\,\mathbf{m}_i, \tag{4}$$

where u_i, $\dot{u}_i$, $\boldsymbol{\omega}_{\times}$, and $\mathbf{v}$ are unknowns to be solved. Note that in (4) $\boldsymbol{\omega}_{\times}$ and $\mathbf{v}$ are the same for all i. Example 1 in Appendix presents a numerical example of (4).

We proceed by proposing a technique to solve for the angular velocity $\boldsymbol{\omega}$, followed by the linear velocity $\mathbf{v}$ and depths of all points. In order to estimate $\boldsymbol{\omega}$, we first eliminate the unknowns u, $\dot{u}$, and $\mathbf{v}$ from (4). This is done by taking two equations of the form (4) for two different features points, i and j, and subtracting one from the other to eliminate $\mathbf{v}$ and obtain

$$u_i\,\boldsymbol{\omega}_{\times}\,\mathbf{m}_i - u_j\,\boldsymbol{\omega}_{\times}\,\mathbf{m}_j = u_i\,\dot{\mathbf{m}}_i + \dot{u}_i\,\mathbf{m}_i - u_j\,\dot{\mathbf{m}}_j - \dot{u}_j\,\mathbf{m}_j. \tag{5}$$

Using two equations of the form (5) gives us a matrix-vector form equation

$$\underbrace{\begin{bmatrix} \boldsymbol{\omega}_{\times}\mathbf{m}_i\text{-}\dot{\mathbf{m}}_i & \text{-}\mathbf{m}_i & \text{-}\boldsymbol{\omega}_{\times}\mathbf{m}_j\text{+}\dot{\mathbf{m}}_j & \mathbf{m}_j & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\omega}_{\times}\mathbf{m}_i\text{-}\dot{\mathbf{m}}_i & \text{-}\mathbf{m}_i & \mathbf{0} & \mathbf{0} & \text{-}\boldsymbol{\omega}_{\times}\mathbf{m}_k\text{+}\dot{\mathbf{m}}_k & \mathbf{m}_k \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} u_i \\ \dot{u}_i \\ u_j \\ \dot{u}_j \\ u_k \\ \dot{u}_k \end{bmatrix} = \mathbf{0}. \tag{6}$$

Equation (6) implies that the determinate of the matrix $\mathbf{M}$ must be equal to zero. Computing the determinant gives a multivariate polynomial equation with 20 degree-three monomials in variables ω_x, ω_y, and ω_z. Coefficients of these monomials are in terms of the known $\mathbf{m}$ and $\dot{\mathbf{m}}$ coordinates. The closed-form expression of these coefficients is obtained from evaluating the determinant symbolically, hence, in the implementation they are computed directly from the points coordinates. Example 2 in the Appendix illustrates this procedure.

Since every three points generate a polynomial equation, from five points we obtain $\binom{5}{3} = 10$ equations. To recover the angular velocity from this system, we first multiply each equation by ω_x, ω_y, and ω_z to obtain a total of 30 equations. These new equations consist of 34 degree-four monomials (rather than 20 degree-three monomials in the original system). These equations can be written as

$$\mathbf{A}\mathbf{x} = \mathbf{0} \tag{7}$$

where $\mathbf{A} \in \mathbb{R}^{30\times 34}$ contains the coefficients and $\mathbf{x} \in \mathbb{R}^{34}$ consists of all monomials in ω_x, ω_y, and ω_z. Example 3 in the Appendix helps to clarify this action. Vector of monomials

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on February 14,2022 at 01:14:08 UTC from IEEE Xplore. Restrictions apply.

$\mathbf{x} \in \mathbb{R}^{34}$ can be split into $\mathbf{x}_1 \in \mathbb{R}^{20}$ and $\mathbf{x}_2 \in \mathbb{R}^{14}$, where $\mathbf{x}_1$ consists of monomials with at least one ω_x term and $\mathbf{x}_2$ contains the remaining monomials. Matrix $\mathbf{A}$ can similarly be split into $\mathbf{A}_1 \in \mathbb{R}^{30\times 20}$ and $\mathbf{A}_2 \in \mathbb{R}^{30\times 14}$ corresponding to $\mathbf{x}_1$, $\mathbf{x}_2$. Thus, (7) can be expressed as

$$\mathbf{A}_1\,\mathbf{x}_1 + \mathbf{A}_2\,\mathbf{x}_2 = \mathbf{0} \tag{8}$$

from which $\mathbf{x}_2$ is given by

$$\mathbf{x}_2 = -\mathbf{A}_2^{\dagger}\mathbf{A}_1\,\mathbf{x}_1 := \bar{\mathbf{B}}\,\mathbf{x}_1, \tag{9}$$

where $\mathbf{A}_2^{\dagger}$ is the pseudoinverse of $\mathbf{A}_2$, and $\bar{\mathbf{B}} \in \mathbb{R}^{14\times 20}$.

In the last step, we construct an eigenvalue problem to recover the vector of monomials $\mathbf{x}_1$ (hence, angular velocities ω_x, ω_y, and ω_z). We pick the eigenvalue $\lambda = \frac{\omega_y}{\omega_x}$ (or other fractions in terms of ω_x, ω_y, and ω_z) and define an eigenvector $\mathbf{w} = \frac{1}{\omega_x}\mathbf{x}_1$ to obtain the eigenvalue problem

$$\omega_y\,\mathbf{w} = \omega_x\,\mathbf{B}\,\mathbf{w}, \tag{10}$$

where $\mathbf{B} \in \mathbb{R}^{20\times 20}$. Each row in $\mathbf{B}$ is chosen from either $\bar{\mathbf{B}}$ or natural basis row vector (i.e., a vector including only one 1 and the rest of this vector is 0). The predefined vector $\mathbf{w}$ is an eigenvector of matrix $\mathbf{B}$. Example 4 in the Appendix elaborates this operation. Eigendecomposition of matrix $\mathbf{B}$ yields 20 eigenvectors $\mathbf{w}$. Each vector gives us a solution for ω_x, ω_y, and ω_z. To find the true solution among them, we note that a valid solution must satisfy the original system of polynomials obtained from the determinant of $\mathbf{M}$. For each candidate solution, we evaluate this system and store the result as a residual. The solution with the smallest residual is chosen as the correct answer.

Note that choosing $\lambda = \frac{\omega_y}{\omega_x}$ is not valid if the camera has no angular velocity along the x direction. The same issue occurs when picking $\lambda = \frac{\omega_x}{\omega_y}$ when there is no angular movement in the y direction, and picking $\lambda = \frac{\omega_x}{\omega_z}$ when there is no angular movement in the z direction. This issue is resolved by solving the eigenvector problem in (10) using all six possible eigenvalues (i.e., $\frac{\omega_y}{\omega_x}$, $\frac{\omega_z}{\omega_x}$, $\frac{\omega_x}{\omega_y}$, $\frac{\omega_z}{\omega_y}$, $\frac{\omega_x}{\omega_z}$, and $\frac{\omega_y}{\omega_z}$). For each eigenvalue, we select the solution with the least residual. We then compare the six residuals and pick the final solution with the smallest residual. A special case occurs when the camera moves with pure linear velocity, in which case all eigenvalues are undefined. Our simulation results show that our algorithm still works under this condition. We admit we cannot explain this apparent incongruity, and more investigations are remained for future work.

Having recovered the angular velocity $\boldsymbol{\omega}$, the last step is to recover the linear velocity $\mathbf{v}$ based on $\mathbf{m}$, and $\dot{\mathbf{m}}$. Recall that (3) is the rigid motion constraint, which is satisfied for every set of feature points and optical flow. Stacking all five pairs together gives the matrix-vector form

$$\underbrace{\begin{bmatrix} \mathbf{I} & \boldsymbol{\omega}_{\times}\mathbf{m}_1\text{-}\dot{\mathbf{m}}_1 & \text{-}\mathbf{m}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \boldsymbol{\omega}_{\times}\mathbf{m}_2\text{-}\dot{\mathbf{m}}_2 & \text{-}\mathbf{m}_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & & & \ddots & \vdots & \vdots \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\omega}_{\times}\mathbf{m}_5\text{-}\dot{\mathbf{m}}_5 & \text{-}\mathbf{m}_2 \end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} \mathbf{v} \\ u_1 \\ \dot{u}_1 \\ u_2 \\ \dot{u}_2 \\ \vdots \\ u_5 \\ \dot{u}_5 \end{bmatrix}}_{\mathbf{y}} = \mathbf{0} \tag{11}$$

where $\mathbf{I} \in \mathbb{R}^{3\times 3}$ is an identity matrix, $\mathbf{C} \in \mathbb{R}^{15\times 13}$, and $\mathbf{y} \in \mathbb{R}^{13}$. Vector $\mathbf{y}$ is the null space of matrix $\mathbf{C}$ and can be recovered by taking the singular value decomposition of C and selecting the vector corresponding to the smallest singular value. Note that vector $\mathbf{y}$ can be recovered up to an unknown a real scalar, since any real scalar can be absorbed into $\mathbf{y}$ while (11) still holds. Therefore, linear velocity and the depths of the feature points can only be recovered up to a constant, unknown scale factor. This is a drawback fundamental to all estimation methods that use a single camera.

IV. Simulations and Experiments

In this section, we verify VEst through simulations and experiments. In the simulation, synthetic points and a simulated camera are used from [24]. Noise is added to test the robustness of our method. In the experiment, the real-world KITTI Vision Benchmark Suite [25] is used.

A. Simulation

Seven synthetic points, fixed in the world frame, and a simulated camera are created. These points are distributed to not lie in the same plane. Simulated images of the feature points are generated. The known calibration matrix is

$$\mathbf{K} = \begin{bmatrix} 800 & 0 & 512 \\ 0 & 800 & 512 \\ 0 & 0 & 1 \end{bmatrix}.$$

As the camera is moved with velocity $[\boldsymbol{\omega}(t)^{\top}, \mathbf{v}(t)^{\top}]^{\top}$, the optical flow vectors are computed using

$$\dot{\mathbf{m}} = J_p\,[\boldsymbol{\omega}(t)^{\top}, \mathbf{v}(t)^{\top}]^{\top}$$

where J_p is the Jacobian matrix [24].

We present three different simulation tests under different camera velocities. In the first case, the camera has three non-zero angular velocities and three non-zero linear velocities. The second case limits the linear velocity to be nearly zero, while the angular velocity remains the same as in the first case (methods based on the continuous essential matrix generally fail as linear velocity approaches zero). In the last case, no angular velocity exists from the camera (recall that this condition would appear to have an undefined eigenvalue in (10)). In order to evaluate the robustness of our method during each test, Gaussian noise with zero mean and standard deviation ranging from 0 to 2 pixels is added to the image coordinates in increments of 0.1 pixels. For each level of pixel noise, $\binom{7}{5}$ = 21 independent trials are conducted, since 7 points can be seen and 5 points are used each time to do the estimation. We use metrics

$$\rho(\boldsymbol{\omega}, \hat{\boldsymbol{\omega}}) = ||\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}||, \tag{12}$$

$$\rho(\mathbf{v}, \hat{\mathbf{v}}) = \frac{1}{\pi}\arccos\left(\frac{\mathbf{v}^{\top}\hat{\mathbf{v}}}{||\mathbf{v}||\,||\hat{\mathbf{v}}||}\right) \in [0, 1], \tag{13}$$

to define the estimation errors, where $\hat{\boldsymbol{\omega}}$ is the estimated angular velocity, and $\hat{\mathbf{v}}$ is the estimated linear velocity. Note that $\rho(\mathbf{v}, \hat{\mathbf{v}})$ measures the angle between the estimated linear velocity and the ground truth. In $\rho(\mathbf{v}, \hat{\mathbf{v}})$, both $\mathbf{v}$ and $\hat{\mathbf{v}}$ are normalized because linear velocity can only be recovered up to a scalar. The metric $\rho(\mathbf{v}^{\top}, \hat{\mathbf{v}})$ is also unitless.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on February 14,2022 at 01:14:08 UTC from IEEE Xplore. Restrictions apply.

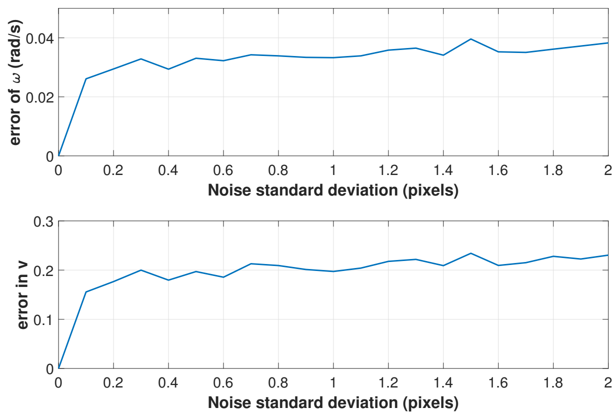

Fig. 2. Angular velocity error and linear velocity error over increasing magnitude of additive Gaussian noise. The camera has both linear and angular movements.

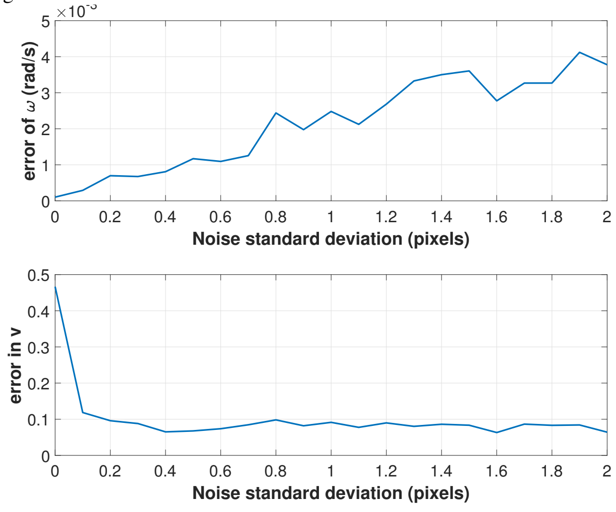

Fig. 3. Angular velocity error and linear velocity error over increasing magnitude of additive Gaussian noise. The camera has only angular velocity.

The first test has camera velocity $[\boldsymbol{\omega}(t)^\top, \mathbf{v}(t)^\top]^\top = [0.8, 1.3, 0.5, 0.2, 0.1, 0.3]^\top$. The average error over the 21 tests for each noise level is shown in fig:case1. Under no noise, our estimates have zero error. As noise increases, the errors increase but remain below 0.04 for $\rho(\boldsymbol{\omega}, \hat{\boldsymbol{\omega}})$ and under 0.3 for $\rho(\mathbf{v}, \hat{\mathbf{v}})$. The second test has camera velocity $[\boldsymbol{\omega}(t)^\top, \mathbf{v}(t)^\top]^\top = [0.8, 1.3, 0.5, 0, 0, 0.000001]^\top$ and the test result is shown in fig:case2. The infinitesimal value for v_z prevents a divide by zero in $\rho(\mathbf{v}, \hat{\mathbf{v}})$. Note when there is no noise, $\mathbf{v} \cdot \hat{\mathbf{v}} \approx 0$ and $\rho(\mathbf{v}, \hat{\mathbf{v}}) = 0.5$ even for a flawless estimate. When noise is introduced, the estimated linear velocity becomes non-zero. The linear velocity estimation error remains smaller than the first test. The third case contains pure linear velocity where $[\boldsymbol{\omega}(t)^\top, \mathbf{v}(t)^\top]^\top = [0, 0, 0, 0.2, 0.1, 0.3]^\top$, and the plot is shown in fig:case3. Our estimation has no error when there is no noise. As noise increases, the estimation error increases but remains under 0.03 for $\rho(\boldsymbol{\omega}, \hat{\boldsymbol{\omega}})$ and under 0.2 for $\rho(\mathbf{v}, \hat{\mathbf{v}})$. These results show that our method is robust under noise.

B. Experiments

We use the KITTI Vision Benchmark Suite in our experiments. The KITTI dataset consists of thousands of images

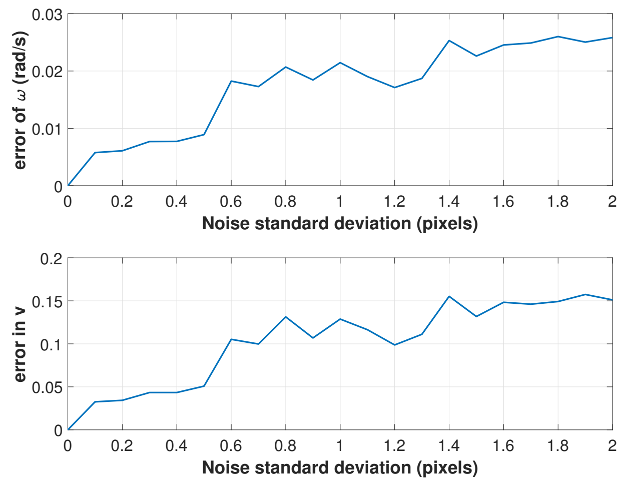

Fig. 4. Angular velocity error and linear velocity error over increasing magnitude of additive Gaussian noise. The camera has only linear velocity.

taken from a forward-facing, on-car camera. The car is equipped with GPS and IMU, which provides ground truth of camera motion. The car drives in different environments such as city roads and country roads. We first present a result from an urban scene (data set: 2011_09_26_drive_0009_sync). The car was driving forward during the first 17 seconds, then made a left turn and continued to drive forward. The total time of this segment is 35 seconds.

For each sequential pair of images, the Harris corner detection [18] is used to find feature points. Optical flow vectors are computed by subtracting two matched feature points in the two images [26]. No outlier rejection or optical flow smoothing was conducted. The result is shown in fig:KITTI, which are processed using Robust Locally Weighted Scatterplot Smoothing [27]. The upper plot shows the estimated angular velocity and the angular velocity reported by KITTI. The bottom plot shows the estimated and ground truth linear velocities. The car made a left turn around 170th frame, which can be clearly seen in the angular velocity plot. The angular velocity estimates are accurate throughout the sequence, though the effects of noise are apparent. The error in linear velocity estimation increases slightly when the car makes a turn, which can be attributed to the fact that optical flow induced by a left turn is similar to that induced by a rightward velocity. Nevertheless, our estimate remains accurate.

We ran our proposed algorithm on six different image sequences from the KITTI Data Suit *2011_09_26_drive*. As feature points and optical flow vectors are selected randomly, we did 30 independent trials through each data set. Following the 30 trials, we found the mean and standard deviation of the the error metrics (12) and (13) for each sequence. We present the error statistics in Table I. The average angular velocity estimation errors are under 0.12, except for dataset-0009, which has the highest angular velocity estimation error 0.1559. The variance of each dataset is under 0.04, which proves that our method has consistent performance among different datasets. Note that we used dataset-0009 in the experiment reported above, which reported good results

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on February 14,2022 at 01:14:08 UTC from IEEE Xplore. Restrictions apply.

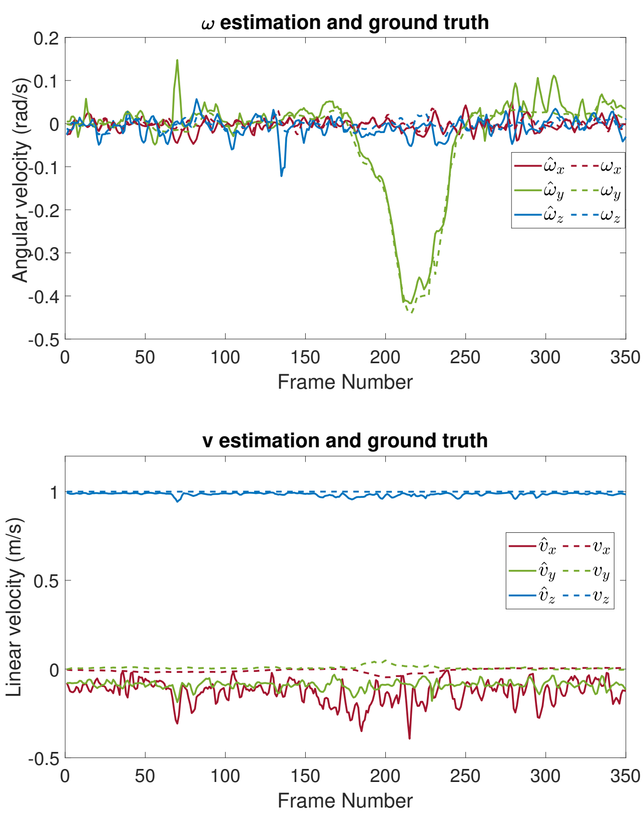

Fig. 5. Camera velocity estimation for a passage in the KITTI dataset.

15pt **sequence**	$\overline{\rho(\boldsymbol{\omega},\hat{\boldsymbol{\omega}})}$	$\sigma^2(\rho(\boldsymbol{\omega},\hat{\boldsymbol{\omega}}))$	$\overline{\rho(\mathbf{v},\hat{\mathbf{v}})}$	$\sigma^2(\rho(\mathbf{v},\hat{\mathbf{v}}))$
8pt **0001**	0.0649	4..3366e-4	0.0631	1.1170e-4
8pt **0002**	0.1185	0.0018	0.0711	3.5542e-4
8pt **0009**	0.1559	0.0357	0.0626	2.1071e-4
8pt **0011**	0.0697	0.0011	0.0529	1.5328e-4
8pt **0013**	0.0785	7.3639e-4	0.0660	2.4643e-4
8pt **0051**	0.1042	0.0025	0.0688	1.1552e-4

TABLE I
STATISTICAL RESULTS OF EXPERIMENTS ON SIX DATA SETS

compared with the ground truth. We conclude that our approach shows strong results in estimating angular velocity. The average errors of linear velocity estimation are under 0.08, and the variances of linear estimation errors are under 0.0004. This is similar to the performance in our simulations.

V. Conclusion

We proposed a novel method to solve the vision-based velocity estimation problem. Our method overcomes weaknesses of other existing methods without relying on the essential matrix or the homography matrix. A minimum of five points and their corresponding optical flow vectors are needed to to create a polynomial system with roots corresponding to the angular velocity terms. We solve the polynomial system using a novel polynomial solver first described in our previous work. We used both synthetic point simulation and real-world experiment to show that our approach is accurate and robust to noise. Future work includes implementing outlier rejection method and associating with visual servo applications.

Appendix

Example 1: As described in Section III, a set of five features points $\mathbf{m}$ and their corresponding optical flow vectors $\dot{\mathbf{m}}$ are extracted from a sequence of images. Each pair satisfies (3). Let the first pair be $\mathbf{m}_1 = \begin{bmatrix}-0.4177\\-0.0775\\1\end{bmatrix}$ and $\dot{\mathbf{m}}_1 = \begin{bmatrix}-0.0051\\-0.0010\\0\end{bmatrix}$, which give the equation

$$u_1\boldsymbol{\omega}_\times\begin{bmatrix}-0.4177\\-0.0775\\1\end{bmatrix} + \mathbf{v} = u_1\begin{bmatrix}-0.0051\\-0.0010\\0\end{bmatrix} + \dot{u}_1\begin{bmatrix}-0.4177\\-0.0775\\1\end{bmatrix} \quad (14)$$

where scalar u_1 is the depth of first point, scalar $\dot{u}_1$ is the velocity of first point. Similarly, the second and third pairs can be written as

$$u_2\boldsymbol{\omega}_\times\begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix} + \mathbf{v} = u_2\begin{bmatrix}-0.0193\\-0.0041\\0\end{bmatrix} + \dot{u}_2\begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix} \quad (15)$$

$$u_3\boldsymbol{\omega}_\times\begin{bmatrix}-0.1230\\-0.0561\\1\end{bmatrix} + \mathbf{v} = u_3\begin{bmatrix}0\\0.0020\\0\end{bmatrix} + \dot{u}_3\begin{bmatrix}-0.1230\\-0.0561\\1\end{bmatrix} \quad (16)$$

where subscripts denote different points. $\boldsymbol{\omega}_\times$ and $\mathbf{v}$ stay the same for all points.

Example 2: Subtract (15) from (14) and move everything to the left, we get

$$\left[\boldsymbol{\omega}_\times\begin{bmatrix}-0.4177\\-0.0775\\1\end{bmatrix} - \begin{bmatrix}-0.0051\\-0.0010\\0\end{bmatrix}\right]u_1 - \begin{bmatrix}-0.4177\\-0.0775\\0\end{bmatrix}\dot{u}_1 \quad - \\ \left[\boldsymbol{\omega}_\times\begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix} - \begin{bmatrix}-0.0193\\-0.0041\\0\end{bmatrix}\right]u_2 + \begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix}\dot{u}_2 \quad = 0. \quad (17)$$

Subtract (16) from (14) and repeat the action to get

$$\left[\boldsymbol{\omega}_\times\begin{bmatrix}-0.4177\\-0.0775\\1\end{bmatrix} - \begin{bmatrix}-0.0051\\-0.0010\\0\end{bmatrix}\right]u_1 - \begin{bmatrix}-0.4177\\-0.0755\\1\end{bmatrix}\dot{u}_1 - \\ \left[\boldsymbol{\omega}_\times\begin{bmatrix}-0.1230\\-0.0561\\1\end{bmatrix} - \begin{bmatrix}0\\0.0020\\1\end{bmatrix}\right]u_3 + \begin{bmatrix}-0.1230\\-0.0561\\1\end{bmatrix}\dot{u}_3 = 0 \quad (18)$$

in which $\mathbf{v}$ is cancelled. (17) and (18) can be written together in a matrix form as seen in (19) where $\mathbf{M} \in \mathbb{R}^{6\times 6}$ has a null vector. Its determinant has to be zero, which is given by

$$-0.0154\omega_x^3\text{-}0.0018\omega_y^3 - 2.5994e^{-4}\omega_z^3 - 0.0380\omega_x^2\omega_y \\ -1.0911e^{-5}\omega_y\text{-}3.5115e^{-6}\omega_z - 2.5870e^{-8} = 0. \quad (20)$$

Equation (20) is a polynomial equation with highest degree-three order. All of its coefficients can be found from feature points and their optical flows. In practice, we calculate these coefficients using MATLAB symbolic toolbox.

Example 3: Multiplying ω_x, ω_y, and ω_z with (20) respectively, we get three equations:

$$\text{-}0.0154\omega_x^4\text{-}0.0018\omega_y^3\omega_x\text{-}2.5994e^{-4}\omega_z^3\omega_x\text{-}0.0380\omega_x^3\omega_y + \dots \\ \text{-}1.0911e^{-5}\omega_x\omega_y\text{-}3.5115e^{-6}\omega_x\omega_z\text{-}2.5870e^{-8}\omega_x = 0.$$

$$\text{-}0.0154\omega_x^3\omega_y\text{-}0.0018\omega_y^4\text{-}2.5994e^{-4}\omega_z^3\omega_y\text{-}0.0380\omega_x^2\omega_y^2 + \dots \\ \text{-}1.0911e^{-5}\omega_y^2\text{-}3.5115e^{-6}\omega_y\omega_z\text{-}2.5870e^{-8}\omega_y = 0.$$

$$\text{-}0.0154\omega_x^3\omega_z\text{-}0.0018\omega_y^3\omega_z\text{-}2.5994e^{-4}\omega_z^4\text{-}0.0380\omega_x^2\omega_y\omega_z + \dots \\ \text{-}1.0911e^{-5}\omega_y\omega_z\text{-}3.5115e^{-6}\omega_z^2\text{-}2.5870e^{-8}\omega_z = 0.$$

We have $\binom{5}{3} = 10$ equations, and each equation can yield three equations shown above. Stacking all equations together can yield a matrix-vector equation

$$\underbrace{\begin{bmatrix}\text{-}0.0154 & 0 & 0 & \cdots & 0\\ 0 & \text{-}0.0018 & 0 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \text{-}2.5994e^{-4} & \cdots & \text{-}2.5870e^{-8}\end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix}\omega_x^4\\ \omega_y^4\\ \omega_z^4\\ \vdots\\ \omega_z\end{bmatrix}}_{\mathbf{x}} = \mathbf{0} \quad (21)$$

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on February 14,2022 at 01:14:08 UTC from IEEE Xplore. Restrictions apply.

$$\underbrace{\begin{bmatrix} \hat{\omega}\left(\begin{bmatrix}-0.4177\\-0.0775\\1\end{bmatrix}-\begin{bmatrix}-0.0051\\-0.0010\\0\end{bmatrix}\right) & \begin{bmatrix}0.4177\\0.0775\\0\end{bmatrix} & -\hat{\omega}\left(\begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix}+\begin{bmatrix}-0.0193\\-0.0041\\1\end{bmatrix}\right) & \begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix} & \mathbf{0} & \mathbf{0} \\ \hat{\omega}\left(\begin{bmatrix}-0.4177\\-0.0775\\1\end{bmatrix}-\begin{bmatrix}-0.0051\\-0.0010\\0\end{bmatrix}\right) & \begin{bmatrix}-0.4177\\-0.0775\\0\end{bmatrix} & \mathbf{0} & \mathbf{0} & \hat{\omega}\left(\begin{bmatrix}-0.1230\\-0.0561\\1\end{bmatrix}-\begin{bmatrix}0\\0.0020\\1\end{bmatrix}\right) & \begin{bmatrix}-0.1230\\-0.0561\\1\end{bmatrix} \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix}u_1\\\dot{u}_1\\u_2\\\dot{u}_2\\u_3\\\dot{u}_3\end{bmatrix} = \mathbf{0} \qquad (19)$$

where matrix $\mathbf{A} \in \mathbb{R}^{30\times 34}$ and vector $\mathbf{x} \in \mathbb{R}^{34}$. Our goal is to solve for ω_x, ω_y, and ω_z from this equation.

Example 4: We split $\mathbf{x}$ into $\mathbf{x}_1$ and $\mathbf{x}_2$ and choose eigenvalue $\lambda = \frac{\omega_y}{\omega_x}$, eigenvector $\mathbf{w} = \frac{\mathbf{x}_1}{\omega_x}$ shown as

$$\mathbf{x}_1 = \begin{bmatrix}\omega_x^4\\\omega_x^3\omega_y\\\omega_x^3\omega_z\\\vdots\\\omega_x\end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix}\omega_y^4\\\omega_z^4\\\omega_y^3\omega_z\\\vdots\\\omega_z\end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix}\omega_x^3\\\omega^2\omega_y\\\omega^x\omega_z\\\vdots\\\omega_x\\\omega_y\\\omega_z\\1\end{bmatrix} \qquad (22)$$

Using (9) for matrix $\mathbf{A}$ above in (21) we can get

$$\underbrace{\begin{bmatrix}\omega_y^4\\\omega_z^4\\\omega_y^3\omega_z\\\vdots\\\omega_z\end{bmatrix}}_{\bar{\mathbf{x}_2}} = \underbrace{\begin{bmatrix}0 & -58.7978 & -62.9465 & \cdots & 0\\0 & 6.6234 & 8.2964 & \cdots & 0\\0 & -5.0563 & -5.8116 & \cdots & 0\\\vdots & \vdots & \vdots & \ddots & \vdots\\0 & 5.2099e^5 & 1.0540e^6 & \cdots & 0\end{bmatrix}}_{\bar{\mathbf{B}}} \underbrace{\begin{bmatrix}\omega_x^4\\\omega_x^3\omega_y\\\omega_x^3\omega_y\\\vdots\\\omega_x\end{bmatrix}}_{\mathbf{x}_1} \qquad (23)$$

The first row in $\mathbf{B}$ is a natural basic vector, in which the second element is 1 and other elements are 0, since $\omega_x^3\omega_y$ can be found in $\omega_x\mathbf{w}$ in the second row. The second row in $\mathbf{B}$ is also a natural basic vector with 1 in the sixth place. The third row in $\mathbf{B}$ is a natural basic vector with 1 in the eighth position. The forth row in $\mathbf{B}$ is the first row in $\bar{\mathbf{B}}$ since ω_y^4 matches with the first row in (24). Therefore, we build an eigenvalue-eigenvector problem as

$$\underbrace{\begin{bmatrix}\omega_x^3\omega_y\\\omega_x^2\omega_y^2\\\omega_x^2\omega_y\omega_z\\\vdots\\\omega_y\end{bmatrix}}_{\omega_y\mathbf{w}} = \underbrace{\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\0 & 0 & 0 & \cdots & 0\\0 & 0 & 0 & \cdots & 0\\0 & -58.7978 & -62.9465 & \cdots & 0\\\vdots & \vdots & \vdots & \ddots & \vdots\\0 & -4.3256e^7 & -4.5851e^7 & \cdots & 0\end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix}\omega_x^4\\\omega_x^3\omega_y\\\omega_x^3\omega_z\\\vdots\\\omega_x\end{bmatrix}}_{\omega_x\mathbf{w}} \qquad (24)$$

By solving eigenvector of matrix $\mathbf{B}$, we recover angular velocity (i.e., ω_x, ω_y, and ω_z) from $\mathbf{w}$, correspondingly the last fourth element, last third element, and the last second element as shown in (22).

REFERENCES

[1] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[2] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, pp. I–I.

[3] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle," *Journal of Field Robotics*, vol. 33, no. 4, pp. 431–450, 2016.

[4] H. Wei and N. Kehtarnavaz, "Semi-supervised faster rcnn-based person detection and load classification for far field video surveillance," *Machine Learning and Knowledge Extraction*, vol. 1, no. 3, pp. 756–767, 2019.

[5] E. Garcia-Fidalgo and A. Ortiz, "Vision-based topological mapping and localization methods: A survey," *Robotics and Autonomous Systems*, vol. 64, pp. 1–20, 2015.

[6] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, p. 133, 1981.

[7] O. Faugeras, "Three-dimensional computer vision: a geometric viewpoint. 1993."

[8] D. Nister, "An efficient solution to the five-point relative pose problem," in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. II–195.

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

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

[11] K. Fathian, J. P. Ramirez-Paredes, E. A. Doucette, J. W. Curtis, and N. R. Gans, "Quest: A quaternion-based approach for camera motion estimation from minimal feature points," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 857–864, 2018.

[12] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer Science & Business Media, 2012, vol. 26.

[13] S. Soatto, R. Frezza, and P. Perona, "Motion estimation via dynamic vision," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 393–413, 1996.

[14] D. Tick, A. C. Satici, J. Shen, and N. Gans, "Tracking control of mobile robots localized via chained fusion of discrete and continuous epipolar geometry, imu and odometry," *IEEE Transactions on Cybernetics*, vol. 43, no. 4, pp. 1237–1250, 2013.

[15] A. P. Dani, N. R. Fischer, and W. E. Dixon, "Single camera structure and motion," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 238–243, 2012.

[16] J. Bowman and P. Mihelich, "Camera calibration," *URL http://www. ros. org/wiki/camera_ calibration*, 2012.

[17] D. Fortun, P. Bouthemy, and C. Kervrann, "Optical flow modeling and computation: a survey," *Computer Vision and Image Understanding*, vol. 134, pp. 1–21, 2015.

[18] C. G. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, no. 50, 1988, pp. 10–5244.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[20] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc.of IEEE International conference on computer vision*, 2011, pp. 2564–2571.

[22] J.-Y. Bouguet *et al.*, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[23] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[24] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. Springer, 2017, vol. 118.

[25] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research*, 2013.

[26] M. Muja and D. G. Lowe, "Fast matching of binary features," in *Proc. IEEE Conference on computer and robot vision*, 2012, pp. 404–410.

[27] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *Journal of the American statistical association*, vol. 74, no. 368, pp. 829–836, 1979.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on February 14,2022 at 01:14:08 UTC from IEEE Xplore. Restrictions apply.