AERIAL VISUAL-INERTIAL ODOMETRY PERFORMANCE EVALUATION

THESIS

Daniel Carson

AFIT-ENG-MS-17-M-010

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-17-M-010

AERIAL VISUAL-INERTIAL ODOMETRY PERFORMANCE EVALUATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Daniel Carson, B.S.E.E.

23 March 2017

AFIT-ENG-MS-17-M-010

AERIAL VISUAL-INERTIAL ODOMETRY PERFORMANCE EVALUATION

THESIS

Daniel Carson, B.S.E.E.

Committee Membership:

John F. Raquet, PhD
Chair

Kyle J. Kauffman, PhD
Member

Scott L. Nykl, PhD
Member

AFIT-ENG-MS-17-M-010

Abstract

With the cheapening of optical camera technology, computer vision-based navigation algorithms have grown in popularity. Alternative navigation techniques are invaluable in Global Positioning System (GPS)-denied environments such as in urban canyons, underwater, other planets, or in the presence of jamming or spoofing. Given the proliferation of optical alternative navigation techniques, such as Visual Odometry (VO), it is necessary to evaluate these algorithms in a way where they can be compared to each other on common ground. While benchmarks like the KITTI Vision Benchmark Suite compare many VO algorithms, this paper compares three top-down monocular VO algorithms—a purely feature-based VO method, a purely optical flow-based VO method, and a hybrid method—fused with an inertial sensor on much more customizable and detailed level than before.

The feature-based method uses an Accelerated-KAZE (AKAZE) feature detection and description and match descriptors using a brute force method. The optical flow method creates a grid of points which are tracked with a Lucas-Kanade tracker in order to create matches. Both methods rely on RANdom SAmple Consensus (RANSAC) for outlier rejection and to build the essential matrix, which motion estimates are derived from. With the assumption of known height above ground, the translation can be scaled to match real-world translations, allowing for a three-dimensional velocity update. The hybrid method will be based on a version of Semi-Direct Visual Odometry (SVO). SVO uses a combination of feature-level and pixel-level tracking. This method relies on the same assumption as the other methods (known height above ground) and provides the same type of update.

Scorpion, a tool developed by the Automation Navigation Technology Center at

the Air Force Institute of Technology, is a framework for Kalman filtering and estimation. Scorpion enables the implementation of multiple VO algorithms into the same framework where sensor inputs, filter parameters, filter types, and solution outputs can be quickly and easily adjusted and allows each algorithm to be evaluated on a level playing field. Each VO algorithm's velocities update the INS using an extended Kalman filter and an unscented Kalman filter. The algorithms' measurement accuracy and ability to estimate the rotation between frames is evaluated. The effects of multiple grades of IMUs, a full and partial INS reset, different kinds of filters, and image preprocessing are also analyzed.

Acknowledgements

I would like to thank my adviser Dr. Raquet for all his investments in me as a student as well as Dr. Kauffman, Dr. Nykl, Daniel Marietta, who also made significant technical contributions. I would also like to thank my wife for her support during this entire process. This research was funded by Lockheed Martin through USAF CRADA No. 16-AFIT-02.

Daniel Carson

Table of Contents

Page

List of Figures

List of Tables

Table Page

AERIAL VISUAL-INERTIAL ODOMETRY PERFORMANCE EVALUATION

I. Introduction

Machine vision localization algorithms have become more feasible to implement because of recent cheapening of camera technology and an increase of computing power available to process imagery. In order to best take advantage of machine vision technology, it is necessary to use the algorithm best suited for a given scenario.

1.1 Problem Statement

This research effort implemented five Visual Odometry (VO) algorithms into a common framework, and evaluates them in a specific scenario in order to generate a recommended use-case for each algorithm. The framework consisted of a Kalman Filter (KF) with measurements from an Inertial Measurement Unit (IMU) and barometer. This research was motivated by a desire to understand how various types of VO algorithms would perform relative to each other and how each algorithm would affect the position solution.

The algorithms are limited to monocular visual odometry algorithms which supported a top-down perspective. The algorithms implemented are either open source or designed using open source libraries. These algorithms have been characterized on a single set of real-world data. The scope of this project does not include run time optimization.

1.2 Thesis Overview

Chapter II of this thesis will provide the background information pertinent to the project in order to aide the reader during the following chapters. This chapter also includes mathematical notation, definitions of mathematical operations, and reference frame definitions. Last, Chapter II summarizes some of the state-of-the-art machine vision localization approaches.

Chapter III will cover the setup, or methodology, of this experiment. This includes information about the filter states and covariances. The data sources, as well as data preprocessing, will be covered here. Chapter III will also detail the structure of each VO algorithm, how VO results are used to produce useful measurements, and how the measurements are rotated into the correct frame in order to be used by the filter.

Chapter IV will present and analyze the results of this experiment. Here, the filter errors and covariances will be shown to demonstrate the validity of the filter design. Next, the VO measurement accuracy for each algorithm and the impact of each algorithm's measurements on the filter solution will be evaluated. At this time, various other cases will be analyzed including various filter types, multiple IMU sources, and data preprocessing techniques. Based on the results, use cases for each algorithm will be recommended.

At the end, in Chapter V, the results from Chapter IV will summarized. Future work will be recommended, and this thesis will be concluded.

II. Background

This chapter serves a few purposes. The first is to define mathematical notation and operations which are utilized by later chapters. Building upon these definitions it is useful to provide background concepts to aid the reader in understanding the later discussions. Lastly, examples of related work will be provided in order to show how this work fits in with and builds off of similar projects in this field.

2.1 Mathematical Notation and Definitions

All scalar values will be represented by plain symbols, such as "z". Bolded symbols, like "$\mathbf{z}$", represent vectors. If a vector has more than one dimension, such as a matrix, then it will be represented by a capital bolded symbol such as "$\mathbf{Z}$".

When referring to a filtering time t_i, t_i^- is the propagation time prior to t_i^+, which is the update time.

DCM's will be represented by the symbol "$\mathbf{R}_{\text{frame1}}^{\text{frame2}}$", which should be read as "the rotation from frame1 to frame2." Sets of coordinates will be denoted by the symbol "$\mathbf{x}_{\text{frame}}$," where "frame" is the reference frame containing those coordinates.

The Root Mean Square (RMS) operation will be used to quantify error. For a set of errors $\delta\mathbf{x}$, the RMS is

$$\delta_{RMS} = \sqrt{\frac{(\delta x_1^2 + \delta x_2^2 + \cdots + \delta x_n^2)}{n}} \tag{1}$$

where n is the number of elements of $\delta\mathbf{x}$.

To quantify two sets of one-dimensional errors (such as horizontal position) as a single scalar value, the 2DRMS function will be used. Given two sets of errors $\delta\mathbf{x}$ and $\delta\mathbf{y}$, the 2x Distance Root Mean Square (2DRMS) is

$$\delta_{2DRMS} = 2\sqrt{\frac{(\delta x_1^2 + \delta y_1^2) + (\delta x_2^2 + \delta y_2^2) + \cdots + (\delta x_n^2 + \delta y_n^2)}{n}} \tag{2}$$

A percentage reduction of error from δ_1 to δ_2 will be given as a percentage by

$$\frac{\delta_1 - \delta_2}{\delta_1} \cdot 100\% \tag{3}$$

Similarly, a percentage increase in error from δ_1 to δ_2 will be

$$\frac{\delta_2 - \delta_1}{\delta_1} \cdot 100\% \tag{4}$$

The nose-right wing-down (NRD) body frame is a frame fixed to the aircraft where the first dimension points from the center to the nose of the aircraft, the second dimension points from the origin to the right wing, and the last dimension points from the center to the bottom of the aircraft. Its counterpart, the right-wing-nose-up (RNU) body frame is the same as the NRD body frame except that the first and second dimensions are switched and the third dimension points from the center of the aircraft to the top of the aircraft.

The sensor frames include the INS frame and the camera frame. These frames are fixed to their respective sensor. A fixed rotation to convert to the body frame is required to convert the sensor frame to the body frame. These DCM's will be included in Chapter III.

The navigation frame is a North-East-down (NED) frame with an origin which is aligned with the body frame initially. The first dimension points from the origin to the North, the second to the East, and the third down towards the center of the Earth. The DCM which relates the NRD body frame to the navigation frame is the aircraft's attitude.

Attitude errors, or tilt errors, are defined as the off-diagonal terms in the DCM

which completes the estimated rotation from the body frame to the navigation frame. These DCMs are related by

$$\mathbf{R}_1^{2,\text{ true}} = \mathbf{R}_{2,\text{ error}}^{2,\text{ true}} \mathbf{R}_1^{2,\text{ error}} \tag{5}$$

where $\mathbf{R}_1^{2,\text{ true}}$ is the true rotation, $\mathbf{R}_1^{2,\text{ error}}$ is the estimated rotation, and $\mathbf{R}_{2,\text{ error}}^{2,\text{ true}}$ is the error in the estimated rotation. The off-diagonal elements of $\mathbf{R}_{2,\text{ error}}^{2,\text{ true}}$ are the tilt errors, which are also approximately $\mathbf{R}_{2,\text{ error}}^{2,\text{ true}}$ expressed in Euler angle form.

2.2 Estimating Egomotion

Estimating egomotion, or the displacement of a camera through space, is a necessary component of unmanned and autonomous vehicle applications. A combination of a Global Positioning System (GPS) and an Inertial Measurement System (INS) is commonly used to estimate an agent's pose because of its ability to provide an accurate global update to almost anywhere on Earth. In many cases though, because of jamming, spoofing, or poor satellite reception, GPS is unavailable. For example, GPS systems perform poorly in natural or urban canyons, underwater, or on other planets. In any of these situations an alternative method is necessary.

A few common alternatives include using just inertial sensors, wheel odometry, and visual odometry. All of these techniques are subject to growing error, or drift, since each provides the system with a relative position update instead of a global position update. By itself, INS suffers from exponential errors, making it unreliable for more than short periods of aiding by itself. On slippery surfaces wheel odometry is subject to slippage, and it is altogether useless with aerial systems. Visual odometry, on the other hand, is becoming more viable because of the proliferation of low-cost cameras and technological advancements of VO algorithms.

Some of the main optical-based algorithms which estimate egomotion are orga-

nized based on categorization in Figure 1.

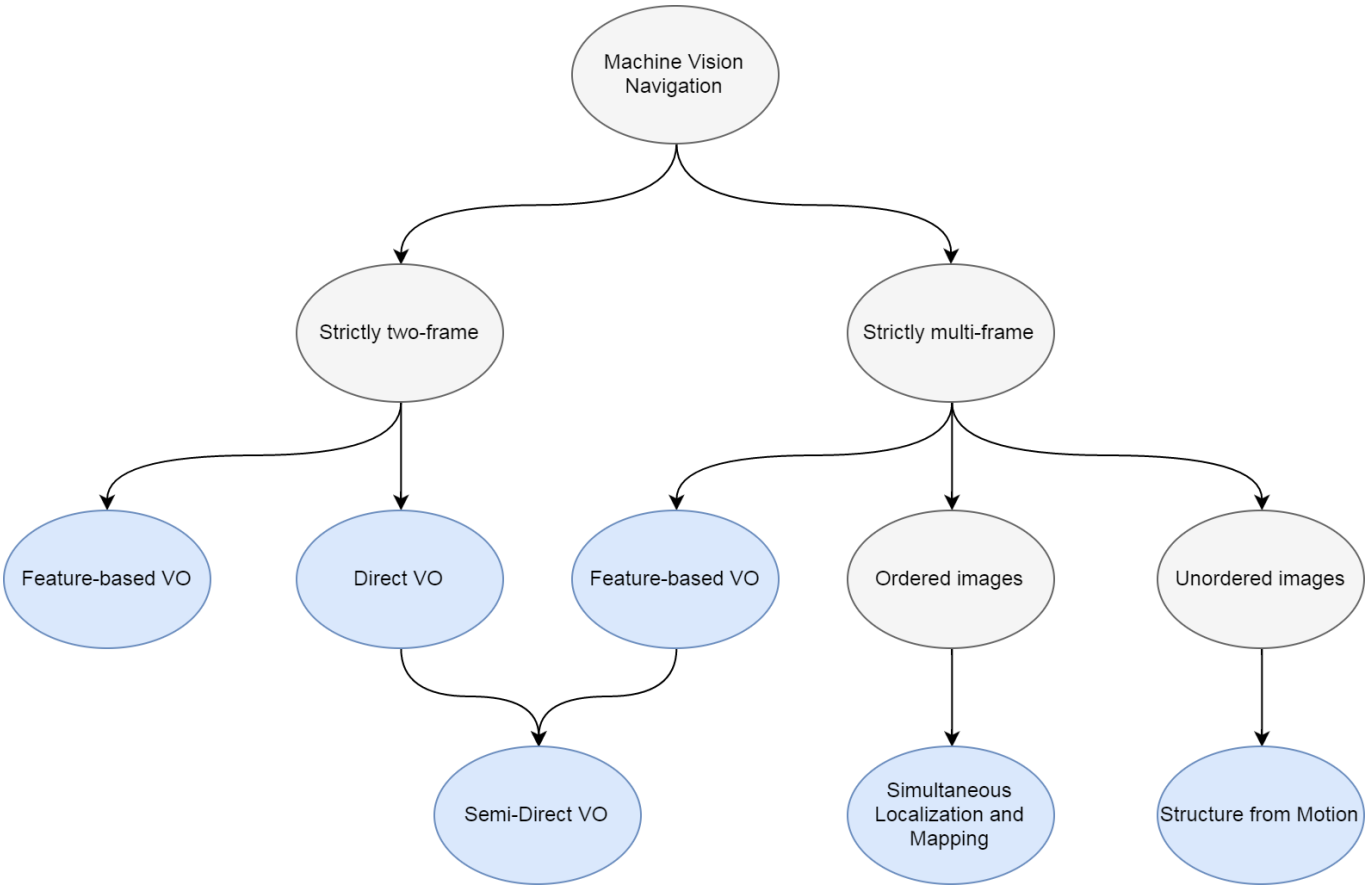


Figure 1. Organization of algorithms (blue) based of categorization (gray)

This figure visually categorizes the methods which will be discussed in the remainder of this chapter.

2.3 Structure From Motion

In the case of a camera moving through a scene, Structure from Motion (SfM) attempts to simultaneously recover both the three-dimensional point cloud structure of a scene and the motion of the camera. SfM can accomplish this sequentially or from an unordered series of frames [8]. The point clouds which represent structures can be used to create digital representations of physical structures in scanning applications.

In terms of navigation, this data can be used to create maps in order to perform localization or object avoidance. If localization is the priority then estimating structure can be ignored and the algorithm becomes similar to Visual Odometry.

Figure 2 shows the results of a SfM operation.

Figure 2. A visual representation of a the generated point cloud (made up of the black dots) and the camera path (made up of the black squares) [1].

Lhuillier applies SfM to a catadioptric (omnidirectional) camera in order to recover both structure and motion [1]. This figure shows one of the input frames (top) as well as the outputted structure and motion. The point cloud forms dense, straight clusters of points marking the walls of the building, especially in the areas that are close to the camera path.

2.4 Visual Simultaneous Localization and Mapping

Visual Simultaneous Localization and Mapping (VSLAM, though often just referred to as just SLAM) differs from SfM in that the images are necessarily processed in order, updating the trajectory and map as the images are added. As SLAM operates it may detect a group of features that it has detected before, recognizing that the camera resides in a previously-mapped location. When SLAM recognizes this, it performs loop closure, adjusting its trajectory to match the current location. For example, a camera may travel in a large circle. When the camera returns to its starting point SLAM adjust its current location to match its starting location.

Figure 3 shows an example of loop closure.

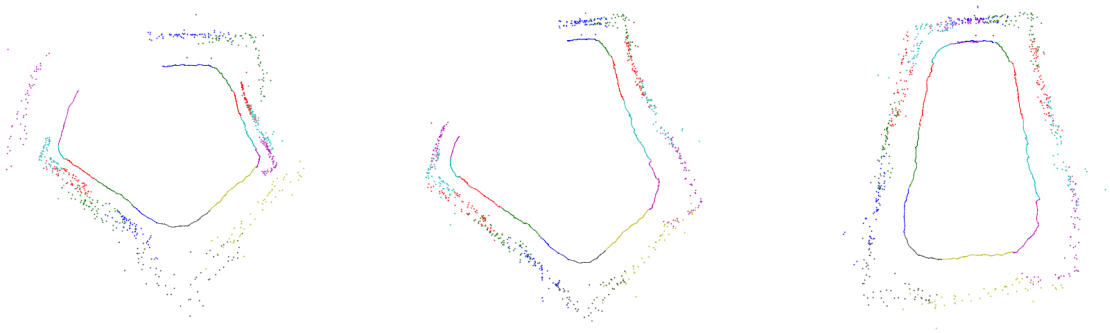


Figure 3. Map and trajectory being adjusted after identifying a loop closure [2]

In this figure, Williams' SLAM algorithm identifies that loop closure has occurred, maps are auto scaled, then the trajectory and maps are adjusted to account for loop closure, removing any drift error that have occurred.

2.5 Visual Odometry

Visual odometry estimates the relative pose of a camera from a sequentially-ordered set of frames. It can either be used to operate on two images at a time, or one at a time on an ordered set of images like SLAM does. It is generally faster than

SfM or SLAM, since it concerns itself solely with the problem of localization, rather than focusing on mapping or loop closure. For this reason, VO can process images more quickly, making it easier to implement for real-time applications.

The result of nearly every VO algorithm is a rotational matrix, $\mathbf{R}_{k,k-1}$, and translation vector, $t_{k,k-1}$. $t_{k,k-1}$ describes the difference between the camera's $k-1$ position and k position. $\mathbf{R}_{k,k-1}$ is a DCM which describes the attitude-change of the camera from the $k-1$ pose to the k pose. For simplicity, $\mathbf{R}_{k,k-1}$, will be referred to as the rotation matrix or just $\mathbf{R}$.

Number of Camera Inputs.

Monocular VO is a case of VO where a single camera provides all image inputs. Monocular VO offers greater hardware flexibility and lower cost, since many vehicles do not already have the multiple cameras with overlapping views and sufficient baseline which stereo VO requires to get scale. Monocular VO's major drawback is that depth information (the distance from a given point to the camera) cannot be directly calculated from a single camera, creating a scale ambiguity [9]. Monocular VO can be supplemented by fusing the data with data from other sensors in order to resolve scale.

Stereo visual odometry is able to resolve the scale ambiguity without the aid of other sensors, provided the baseline between the cameras is large enough. If the distance from the scene to each camera grows much larger than the distance between each camera then the system degrades to that of monocular VO and the scale of movement becomes ambiguous [8].

Determining Motion Across Frames.

Whether performed with one or two cameras, VO can also be broken up into feature-based and direct methods. As seen in Figure 1, feature-based VO can either be used to operate on two images at a time or an ordered set of images, while direct visual odometry is strictly two-frame. In feature-based algorithms, features are extracted from the image using a feature detector. Commonly-used feature detectors include Scale Invariant Feature Transform (SIFT) [3], Speeded-Up Robust Features (SURF) [10], Features from accelerated segment test (FAST), and Harris corners [11]. Other feature detectors which are used include the Shi-Tomasi [12] feature detector which selects "good features to track" and Center Surround Extrema (CenSurE) [13] which performs well in offroad scenes where standard feature detectors yield few features. Konolige compared the performance of these feature detectors in Table 1.

Table 1. Matching and tracking for various feature detectors [7].

	Harris	FAST	SIFT	CenSurE
Fail	0.53%	2.3%	2.6%	0.17%
Length	3.0	3.1	3.4	3.8
Error (m)	4.65	12.75	14.77	2.952
Error (%)	1.55%	4.25%	4.92%	0.97%

The rows in Table 1, from the top to the bottom, show what percentage of feature matches failed, the average length (in images) of a feature track, the position error, and the relative position error.

Of these feature detectors, one of the most commonly used is SIFT [14]. Given an initial image, SIFT calculates a set of Difference Of Gaussian (DOG) images for a given image. Each of these DOG images is scanned for local maxima and minima

so that if a given pixel is higher or lower than all of its neighbors it is selected as a feature. Once a feature is selected, it needs to be given a unique descriptor. Figure 4 illustrates the process used by SIFT to represent each feature by a descriptor.

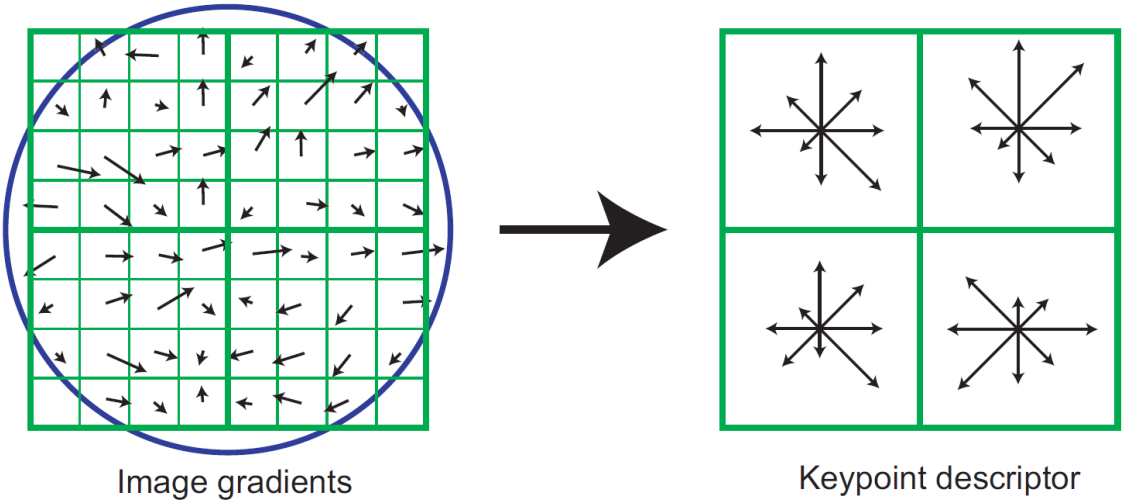


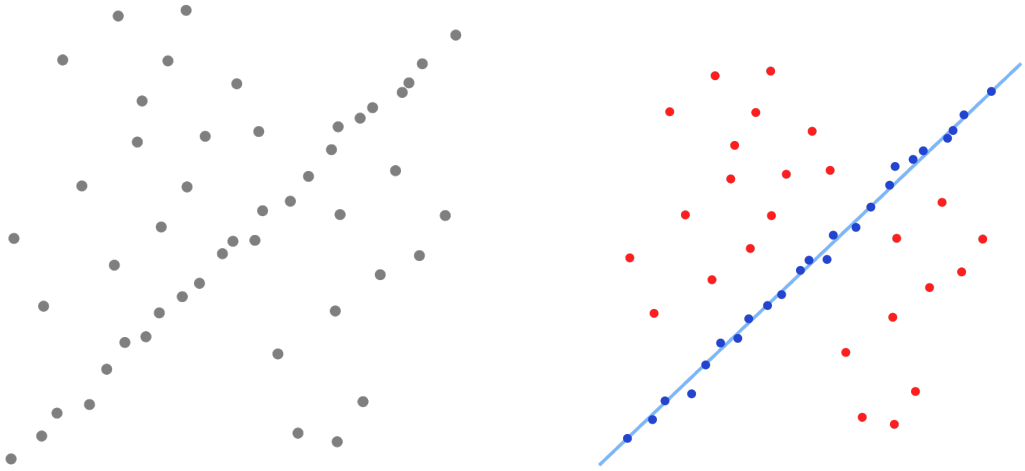
Figure 4. A descriptor is calculated from a matrix of gradients [3]

On the left, the gradient magnitude and orientation around a feature location is calculated. These are weighted by a Gaussian window represented by the circle. The gradients are summed according to orientation and location relative to the feature, resulting in the descriptor in the right of Figure 4.

Once features are extracted from an image, they are tracked from one image to the next via their descriptors. For example, a brute force feature tracker attempts to match the first descriptor in one image with every other descriptor in the second, then match the second descriptor with every remaining descriptor in the second image, and so on.

Once features have been matched from the first image to the second, outlier rejection is performed in order to reduce the number of bad matches. The quintessential outlier rejection method uses RANdom SAmple Consensus (RANSAC) [15] in addition to applying the geometric epipolar constraint [16].

In general, RANSAC is a iterative method to determine the parameters of a model from a set of data which includes inliers with noise and outliers. Figure 5 shows an example of RANSAC being used to estimate the parameters of a linear fit model.



(a) Collection of data with outliers

(b) Data separated into inliers (blue) and outliers (red), with the calculated model displayed as a blue line

Figure 5. RANSAC used to find the best-fit line

In this example, RANSAC is able to find a best-fit line even though almost none of the inliers fall exactly on the line. In VO, RANSAC typically builds the essential matrix using 8 random descriptor matches at a time. Depending on the number of matches, RANSAC will build hundreds of essential matrices and evaluate which matrix produces the highest consensus of inliers.

The epipolar constraint forces inliers to be geometrically consistent with each other, adding another laying of outlier rejection. In Figure 6, c_0 and c_1 refer to the position of a camera at two different locations, separated by a rotation R and translation t.

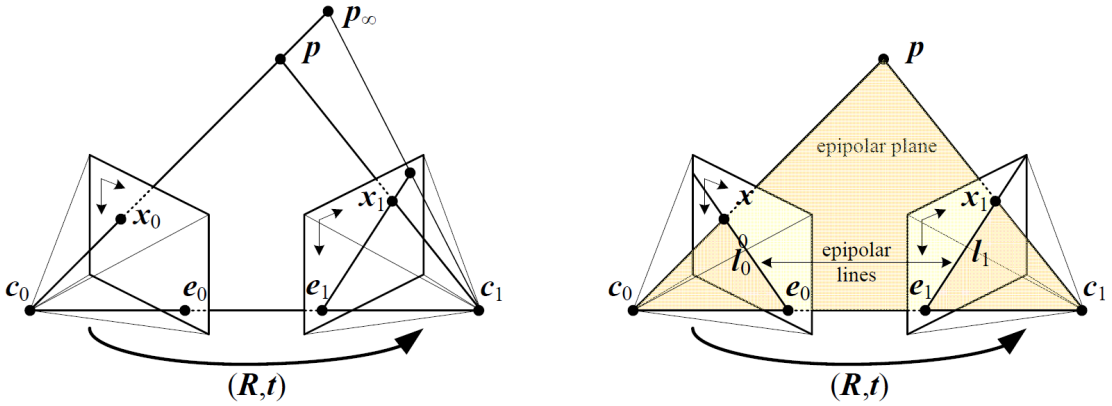


Figure 6. The epipolar constraint [4]

A point p can be seen from the perspective of each camera. A point in one image projects as a line segment on the other. This constraint is used to limit a feature in one image to being matched with features that reside along a line segment in the other image.

The Essential Matrix, $\mathbf{E}$, contains both the rotation and translation from one camera frame to another by

$$\mathbf{E} = \mathbf{t} \times \mathbf{R} \tag{6}$$

Once the essential matrix, $\mathbf{E}$, has been calculated, it can be decomposed into $\mathbf{R}$ and $\mathbf{t}$. One method of decomposition is detailed in [17]. Mathematically, decomposing $\mathbf{E}$ into $\mathbf{R}$ and $\mathbf{t}$ results in two solutions for $\mathbf{R}$, requiring some context information in order to select the correct $\mathbf{R}$.

Direct VO methods do not rely on any sort of feature detection. Instead, they calculate movement directly from pixel intensities. One such method, Optical Flow (OF), operates similar to feature-based VO by matching pixel intensities of the whole image or a subsection of the image from one frame to the next. These matches produce an optical flow field, like the one pictured in Figure 7.

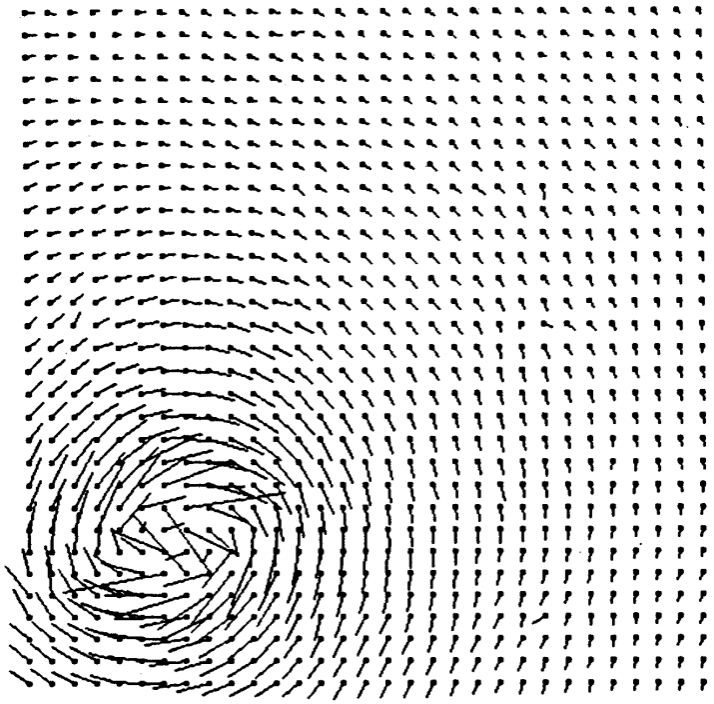


Figure 7. The flow pattern computed of flow around a line vortex [5]

Once individual pixels are matched between frames, the movement of each pixel is calculated. Optical flow requires little movement between frames; otherwise it produces unreliable results [14].

Hybrid methods use a combination of feature-based and appearance-based approaches, to estimate motion. These methods attempt to gain the speed of direct methods and the accuracy of feature-based methods while not heavily relying on the presence of sufficient features in scene. One such method, called Semi-Direct Visual Odometry (SVO) [6], tracks features from frame to frame but also analyzes the flow of pixels from one frame to the next. Notice from Figure 1 that SVO is a combination of two-frame optical flow and multi-frame feature-based VO. Forster's open source implementation of SVO is further discussed in Section 3.3.

Reducing Error Drift.

Since VO only provides a relative update, any pose generated from VO is subject to error drift. In order to increase accuracy and robustness some algorithms incorporate a global or local bundle adjustment, in addition to outlier rejection. A bundle adjustment of any scope requires the algorithm to retain the features extracted from previous frames. In Kalman filtering, this technique bears a strong resemblance to smoothing.

The number of frames that are adjusted determines whether the adjustment is a global or local bundle adjustment. If every frame is stored and adjusted whenever a bundle adjustment occurs then it is called a global bundle adjustment. Global bundle adjustments are often associated with SfM and Simultaneous Localization and Mapping (SLAM).

If only a subsection of the frames are brought up again, then it is a local bundle adjustment. By only correcting a portion of previous results, this method saves computational time. In [7], introducing a Sparse Bundle Adjustment (SBA) decreased the error by 50%, roughly.

2.6 Visual-Inertial Odometry

Often in a system other sensors are available in addition to a camera or set of cameras. The solution can then be improved by fusing data from these other sensors. An Inertial Navigation System (INS), for example, is often paired with many different sensors. Since INS suffers from exponential drift, its results are not very valuable except over short periods of time. They are convenient, however, because they do not depend on an external signal to operate (as is the case with a GPS receiver or camera) and provide a quick update. When an inertia sensor is coupled with visual odometry, it can be referred to as Visual-Inertial Odometry (VIO).

Sensor data can be loosely-coupled or tightly-coupled with other data. While loosely-coupled algorithms are simpler and easier to implement, tightly-coupled algorithms have the potential for more accurate measurements. Figure 8 demonstrates an example of tight coupling and loose coupling.

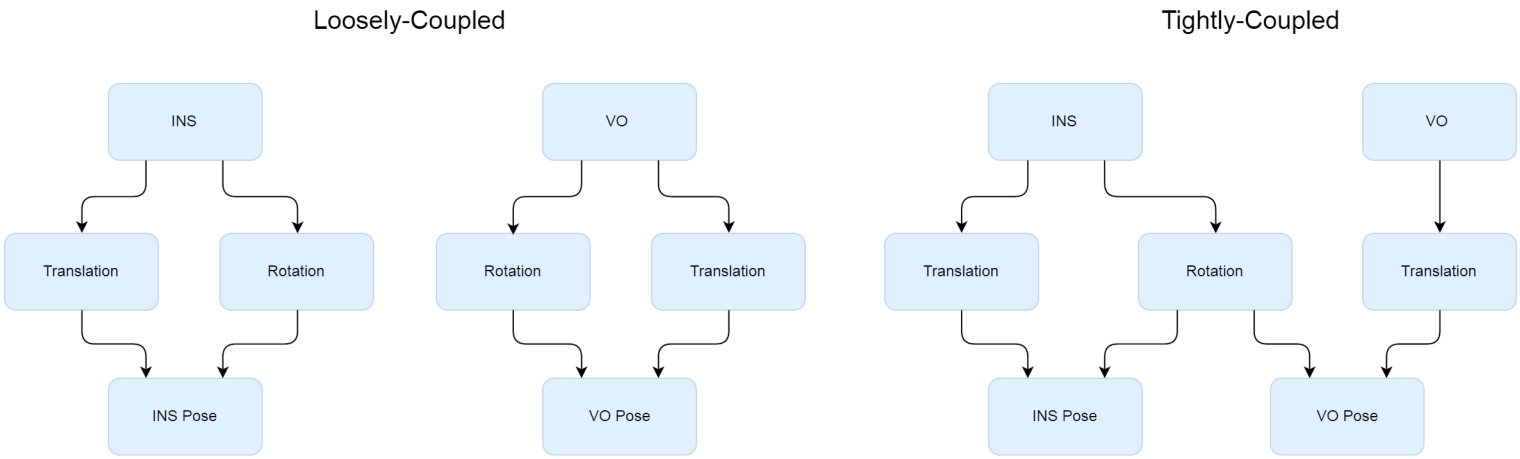


Figure 8. A loosely-coupled and a tightly-coupled visual-inertial odometry system.

On the left, an INS and VO system generate pose estimations independently of each other. On the right, INS and VO data are coupled to generate a more accurate VO pose estimation. INS estimates of $\mathbf{R}$ tend to be superior to those of VO. If the VO is given $\mathbf{R}$ by the INS then, instead of simultaneously calculating $\mathbf{R}$ and $\mathbf{t}$, the VO can produce a more accurate $\mathbf{t}$, as shown by [18]. For the purposes of this experiment, the term "VIO" will only be used to refer to algorithms which are tightly coupled with inertial data and rely on the filter solution for $\mathbf{R}$ and the term "VO" will refer to algorithms that estimate $\mathbf{R}$, even though these algorithms are still loosely coupled with inertial data. INS and VO are typically coupled using an Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) [19].

Linearized Filters.

The Kalman Filter (KF) optimally combines data from multiple sensors with predicted values in order to estimate a number of states. A Linear Kalman Filter (LKF) is the simplest to implement, but is only valid when the state propagation and sensor measurement equations are linearly defined with respect to the states.

The EKF linearizes propagation and update equations via a matrix of partial derivatives, or a Jacobian. While this step adds complexity in implementation, it enables the KF to accurately estimate most nonlinear systems. The linearization breaks down if the system is highly nonlinear, requiring more complicated filters to be used. In this case, a UKF may perform better. For example, Kitt used both an EKF and a UKF when filtering the results of a stereo VO algorithm [20]. In this case, the UKF provided a 68% reduction in positioning error and a 50% reduction in standard deviation from that of an EKF.

A UKF operates by using the unscented transform to pick a minimal set of points about the mean to avoid poor performance when observation models are highly nonlinear [19]. Whether or not a UKF or EKF will perform better than the other is often case-specific.

2.7 Benchmarks

With so many VO variations and methods, it is useful to be able to compare them in order to determine which one is most effective in a given situation. Benchmarks are a general way of accomplishing this. The KITTI Vision Benchmark Suite [21] hosts a VO/SLAM evaluation which is very commonly referenced by VO implementations. The KITTI benchmark allows users to download sets of stereo image data from a camera rig mounted on a car which drives through a variety of urban and rural scenes. Half of the data sets are accompanied by truth data, so that users can test

their algorithm. Users can then submit their results from the half without truth data in order to be evaluated. The KITTI benchmark publishes each submission on their website, ranked by accuracy.

This benchmark is effective at evaluating VO algorithms via translation and rotation error and runtime, for specific scenes. However it is currently limited to ground vehicle application. It cannot be used to evaluate algorithms tailored to offroad, underwater, or aerial environments where features are scarce or the camera views the scene from a different perspective. It also does not evaluate VIO algorithms.

2.8 Notable Visual Odometry Algorithms

Several existing VO implementations were examined before selecting one to integrate with this experiment. Table 2 includes some of the most notable algorithms.

Table 2. A number of noteworthy VO Implementations

Author/Ref	Mono / Stereo	Feature Extraction	Sensors, scale	Applied to	Technical Summary	Open Source	Misc. Notes
Vehicle Navigation in Dense Urban Environments Using GPS , Image , and IMU Measurements from Commercial Off-The-Shelf Sensors, ION 2015 Pacific PNT							
Rohde [18]	Mono	SIFT	GPS, IMU	Ground vehicle	Degradation of GPS, low cost sensors	No	
Large Scale Visual Odometry for Rough Terrain, 2010 Robotics Research							
Konolige, Agrawal, Sola [7]	Stereo	"CenSurE"	IMU	Ground vehicle	Real-time. 0.1% error over 9km. Off-road focus. Possibly good in feature-poor environments.	No	
SVO: Fast Semi-Direct Monocular Visual Odometry, 2014 IEEE International Conference on Robotics and Automation							
Forster, Pizzoli, Scaramuzza [6]	Mono	FAST, optical flow	IMU. Scale of first 10 measurements given, drift is estimated	MAV	SVO (coined). Fast run-time. Downward-facing camera. Mapping.	Yes	https://github.com/uzh-rpg/rpg_svo Implemented in C++.
Real-time stereo visual odometry for autonomous ground vehicles, 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems							
Howard [22]	Stereo	Harris, FAST	N/A	Ground vehicle	"Fast, accurate, robust." Relative error < 0.25%. 20ms per 512x384 image.	Yes	Implemented by a third party, Matlab https://github.com/avisingh599/vo-howard08.
Visual Odometry based on Stereo Image Sequences with RANSAC-based Outlier Rejection Scheme, 2010 Intelligent Vehicles Symposium							
Kitt, Geiger, Lategahn [20]	Stereo	corners	N/A	Ground vehicle	Compares results from using 4 different KF's	Yes	One of KITTI benchmark contributors. Developed computer vision library http://www.cvlibs.net/software/libviso/
Motion-without-Structure: Real-time Multipose Optimization for Accurate Visual Odometry, 2012 Intelligent Vehicles Symposium (IV)							
Lategahn et al [23]	Stereo	Blob and edge kernels	N/A	Ground vehicle	SfM without estimating scene structure, calculates uncertainty of each pose displacement	No	
Stereo-Based Ego-Motion Estimation Using Pixel Tracking and Iterative Closest Point, 2006 IEEE International Conference on Computer Vision							
Milella, Siegwart [24]	Stereo	Shi-Tomasi	N/A	Ground vehicle	Stereo algorithm that uses Iterative Closest Point (ICP)	No	
Real time localization and 3D reconstruction, 2006 Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition							
Mouragnon et al [25]	Stereo	Harris	N/A	Ground vehicle	Structure from Motion (SfM), local bundle adjustment	No	
Monocular Vision for Mobile Robot Localization and Autonomous Navigation, 2006 International Journal of Computer Vision							
Royer, et al [26]	Mono	Harris	Manually sets length of path	Ground vehicle	SfM, builds map from manually guiding then uses map on second run.	No	This algorithm is similar to a setup where one vehicle in a convoy produces a map which is used by other convoy vehicles
A robust visual odometry and precipice detection system using consumer-grade monocular vision, 2005 EEE International Conference on robotics and automation							
Campbell et al [27]	Mono	Low corner threshold, optical flow	Assumes fixed height from ground	Ground vehicle	3DoF, SVO, closed-loop, obstacle detection, variety of terrain testing, consumer-grade hardware. Closed Loop.	Yes	Original open source code not available, https://github.com/cgddrd/Monocular-Visual-Odometry-Emgu
Semi-dense visual odometry for a monocular camera, 2013 IEEE international conference on computer vision							
Engel, Sturm, Cremers [28]	Mono	N/A	Estimates depth map	MAV	SVO which does not use features, matches pairs, propagates a depth map	No	
Real-time Quadrifocal Visual Odometry, 2010 The International Journal of Robotics Research							
Comport, Malis, Rives [29]	Stereo	Dense	N/A	Ground vehicle, blimp	Dense VO (appearance-based), 3D structure constraints, M-estimation outlier rejection	No	
Visual Vehicle Egomotion Estimation using the Fourier-Mellin Transform, 2007 IEEE Intelligent Vehicles Symposium							
Goecke et al [30]	Mono	N/A	Fourier-Mellin Transform	Ground Vehicle	Dense VO, Fourier-Mellin Transform (global method)	No	
Correcting Angle of Visual Odometry System by Fusing Monocular and Stereo Methods in Untextured Dynamic Environment, 2010 10th International Symposium on Communications and Information Technologies							
Ishii et al [31]	Mono and Stereo	Shi-Tomasi corner detector	N/A	Ground Vehicle	3.2% error in untextured (feature-poor) environment. Mono and stereo data fused. Ground plane information used	No	
Real-Time Monocular Visual Odometry for On-Road Vehicles with 1-Point RANSAC, 2009 IEEE International Conference on Robotics and Automation							
Scaramuzza, Fraundorfer, Siegwart [32]	Mono	SIFT, Harris, KLT	speedometer	Ground vehicle	1-point RANSAC (limits to cars), SfM, tests three feature detectors	No	
Semi-Direct EKF-based Monocular Visual-Inertial Odometry, 2015 Intelligent Robots and Systems (IROS) IEEE							
Tanskanen et al [33]	Mono	"external feature tracker"	IMU		SVO, VIO	No	
A New Approach to Vision-Aided Inertial Navigation, 2010 Intelligent Robots and Systems IEEE							
Tardif et al [34]	Stereo	Harris	IMU	Ground vehicle	VIO, SBA, very low error	No	

These algorithms were selected with a focus on ease of implementation, accuracy, monocular VO, and algorithms which might perform well in feature-poor terrain. The first column specifies the last name of each author. The next column indicates whether the given algorithm uses a monocular or stereo camera setup. The "Feature Extraction" column specifies the feature detectors used. "Sensors, Scale" details how each monocular algorithm solves the scale ambiguity present in monocular systems. The next column specifies what kind of vehicle the algorithm was used with. The technical summary includes some brief descriptions of each algorithm. The next column specifies whether the source code is available. "Cited by IEEE/Google" gives the number of citations (according to both the IEEE database and Google Scholar), which is a measure of the seminality of each paper. The miscellaneous notes include things like where the source code is available and what language the algorithm was implemented in.

Ultimately, Forster's SVO algorithm was selected for use with this experiment for its low error, fast run-time, open-source code, and novelty. However this table serves as a good reference to select another algorithm for integration during future work. It also serves as a starting point for someone who is interested in finding significant visual odometry methods.

2.9 Chapter Conclusion

So far various visual odometry methods, including combining VO with other sensors in various filters, have been defined and described. Given the previous definitions and background information, the experiment set-up can now be described.

III. Methodology

This chapter describes the setup of the experiment including filter design, visual odometry methods implemented, and any sensor data preprocessing or characterization.

3.1 Framework

Each VO algorithm is implemented in C++. Java Native Interface (JNI) is used to call the C++ libraries from a Java environment, which is where the filter is implemented. Data is passed to the VO via a byte buffer in order to avoid making a copy. The input data and VO results are filtered using either an EKF or UKF in Scorpion, a plug-and-play framework for Kalman filtering developed by the Autonomy Navigation Technology (ANT) Center at the Air Force Institute of Technology (AFIT). Each filter has 16 error states, represented by a 15 state Pinson model plus an extra bias state to account for and track errors in the barometer. Equation (7) shows the filter states, $\mathbf{x}$.

$$\mathbf{x} = \begin{bmatrix} \delta\mathbf{p} \\ \delta\mathbf{v} \\ \delta\boldsymbol{\theta} \\ \mathbf{a} \\ \mathbf{b} \\ \delta h_{\text{baro}} \end{bmatrix} \tag{7}$$

In this array, $\delta\mathbf{p}$ is the position error (m), $\delta\mathbf{v}$ is the velocity error (m/s), $\delta\boldsymbol{\theta}$ is the tilt error (rad), $\mathbf{a}$ is the accelerometer bias (m/s^2), $\mathbf{b}$ is the gyroscope bias (rad/s),

and δh_{baro} is the barometer bias (m). The position errors states are expressed in a north-east-up (NEU) frame while the velocity and tilt errors are expressed in a north-east-down (NED) frame.

Each position state is initialized with the same covariance, each velocity state is initialized with the same covariance, and so on. All cross-covariances are initialized as zero. The initial covariances for each state are shown in Table 3.

Table 3. Initial sigmas for each state.

	$\delta\mathbf{p}$ (m)	$\delta\mathbf{v}$ (m/s)	$\delta\boldsymbol{\theta}$ (rad)	$\mathbf{a}$ (m/s^2)	$\mathbf{b}$ (rad/s)	δh_{baro} (m)
σ	0	0	0	0.098	4.85e-06	0

The position, velocity, and attitude covariances are initialized with small values because the INS mechanization is initialized with the corresponding truth data. The accelerometer and gyro biases are initialized with values relevant to the INS model used. The barometer bias initial covariance is derived in Section 3.2.

The top-level structure of the system is represented in block-diagram form in Figure 9.

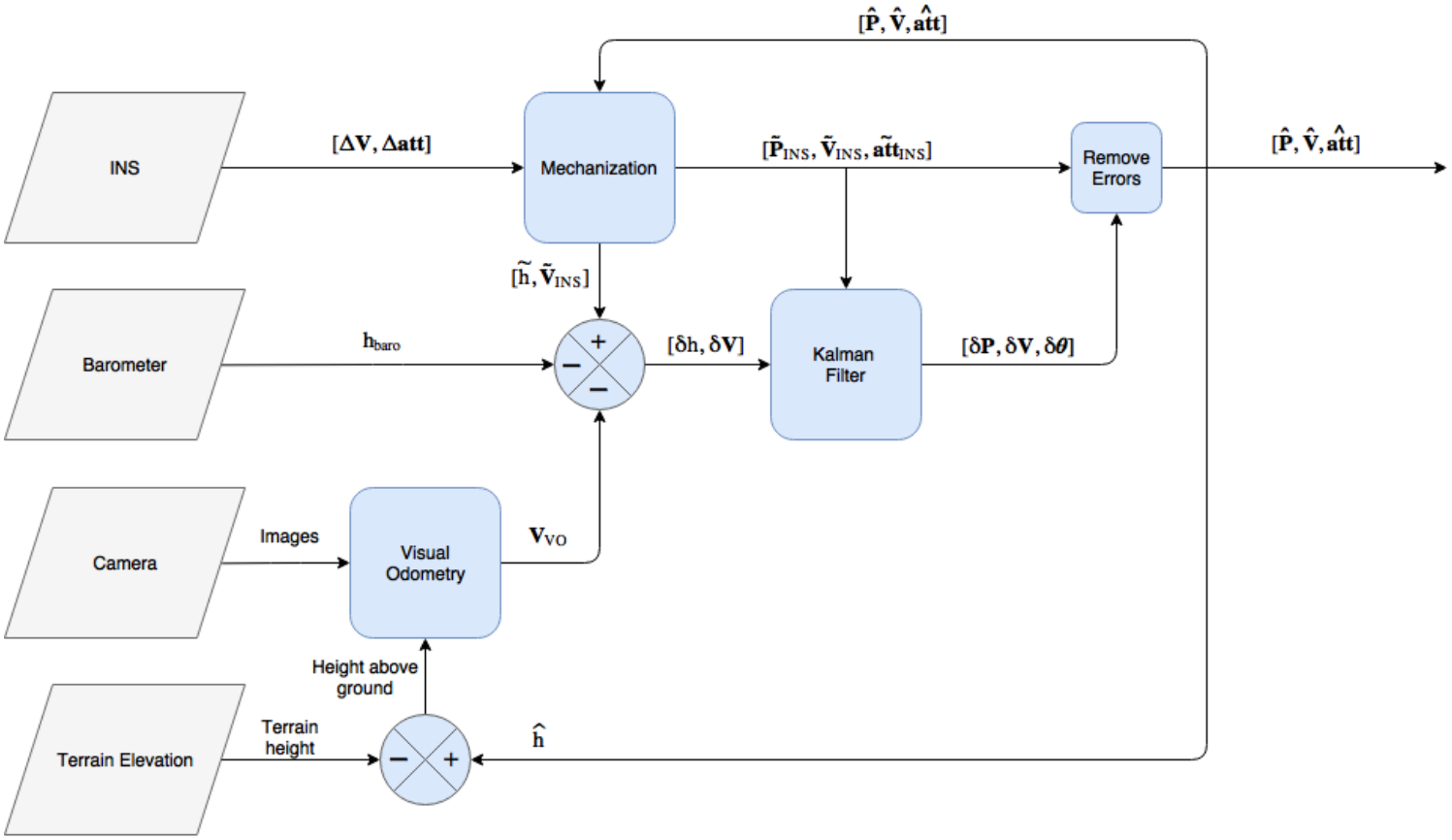


Figure 9. Top-level block diagram of the filter, visual odometry, and sensors

The raw INS measurements are mechanized to calculate a position, velocity, and attitude. The Kalman filter estimates the errors in each of these 9 states in addition to the 7 accelerometer, gyroscope, and barometer bias states. The INS estimate is corrected with the error estimate in order to calculate the final solution. Every 100 seconds the INS solution is reset to this solution and the corresponding error states are zeroed out in order to keep the error models operating in a range that yields accurate results. Measurements of the INS error are calculated by subtracting the INS solution of the altitude (height) and velocity by the barometer altitude and VO velocity, respectively. The two-frame VO algorithms produce a two-dimensional velocity from images and height above the ground. The multi-frame VO produces a three-dimensional velocity measurement. The height above the ground is calculated

by subtracting the terrain height from the estimated MSL height.

Scorpion.

Scorpion contains several modules which handle different kinds of measurements and implement multiple kinds of filter types. This makes it easy to add new sensors to the experiment or swap out an EKF for a UKF, for example. Figure 10 shows a top-level block diagram of the Scorpion modules used and sensor inputs.

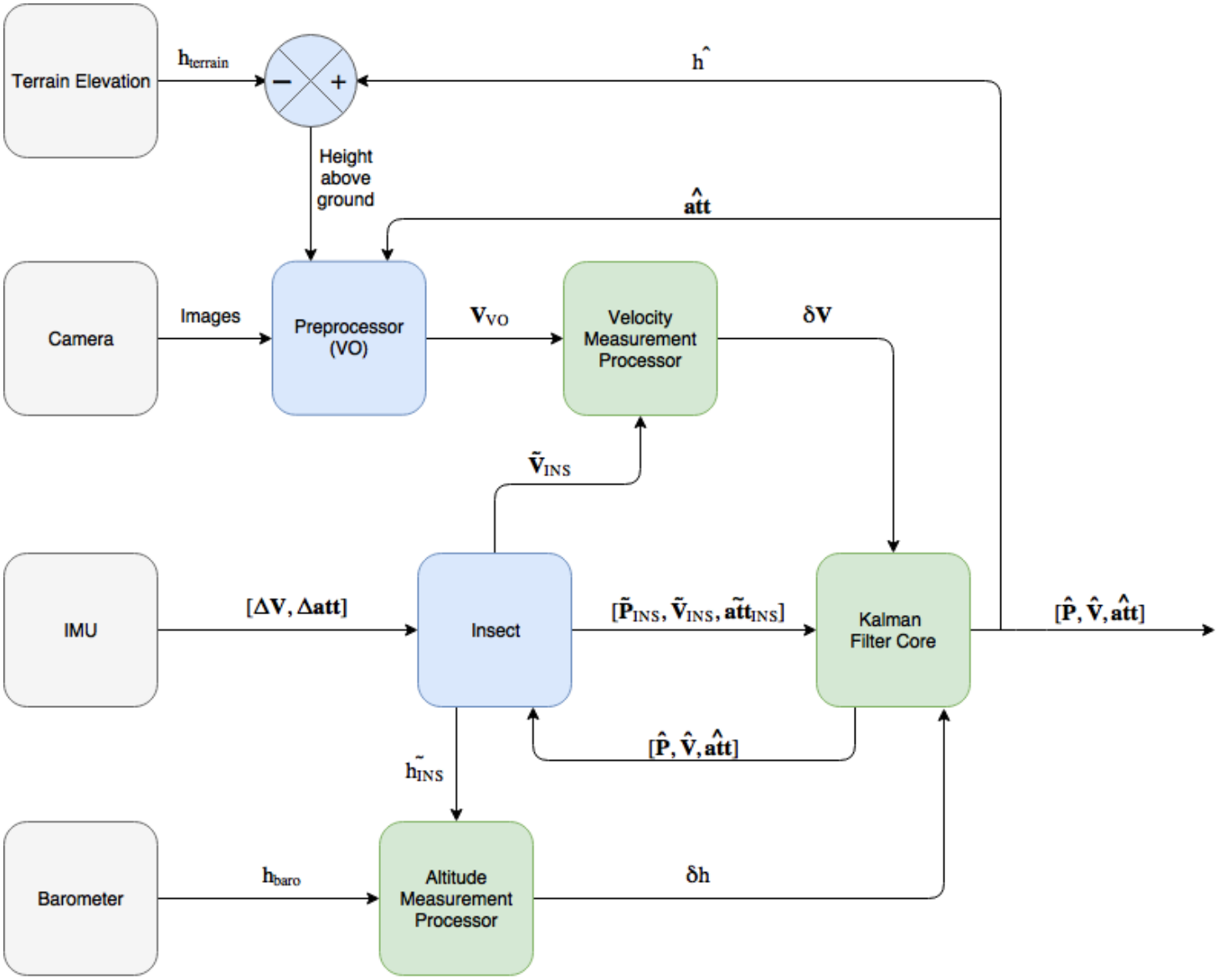


Figure 10. Top-level block diagram of the sensors (gray) and Scorpion modules (blue) including the filter (green)

In the middle of the figure, IMU measurements are mechanized using Insect and the INS solution is used to propagate the filter. The filter (in this case an EKF or UKF) estimates the errors in the INS solution and sensor biases and feeds the INS solution back to Insect, performing an INS reset. The kalman filter applies the estimated errors to the INS solution in order to produce a solution.

At the bottom, the altitude measurement processor creates a measurement by

subtracting the INS solution's altitude by the barometer's altitude. This measurement is used to directly update the filter's altitude state.

At the top, the filter's solution altitude is subtracted by the terrain elevation, resulting in height above ground. The preprocessor module performs visual odometry on the input images, resulting in a velocity which is scaled by the height above ground and rotated into the navigation frame using the solution attitude. The velocity measurement processor creates a measurement by subtracting the INS solution velocity by the visual odometry velocity. The measurement directly updates the filter velocity states.

Visual Odometry Measurement Covariance.

The error between the VO-measured velocity and true velocity was first assumed to be white Gaussian noise. Applying this assumption, the covariance associated with the VO measurements was derived by calculating the squared standard deviation of the measurement error. The initial measurements were filtered using a conservative covariance of

$$\sigma_N = 20 \qquad \sigma_E = 20 \tag{8}$$

The standard deviation of the measurements was then calculated to be

$$\sigma_N = 2.543 \qquad \sigma_E = 2.495 \tag{9}$$

However, when the latter covariance was used to filter the velocity measurements the tighter covariances actually degraded the integrity of the filter solution which, in turn, degraded the VO measurements since they rely on the attitude solution.

The covariance was reexamined by calculating the autocorrelation function of the measurement error. For simplicity, just the autocorrelation of the error of the velocity

in the North direction is displayed in Figure 11.

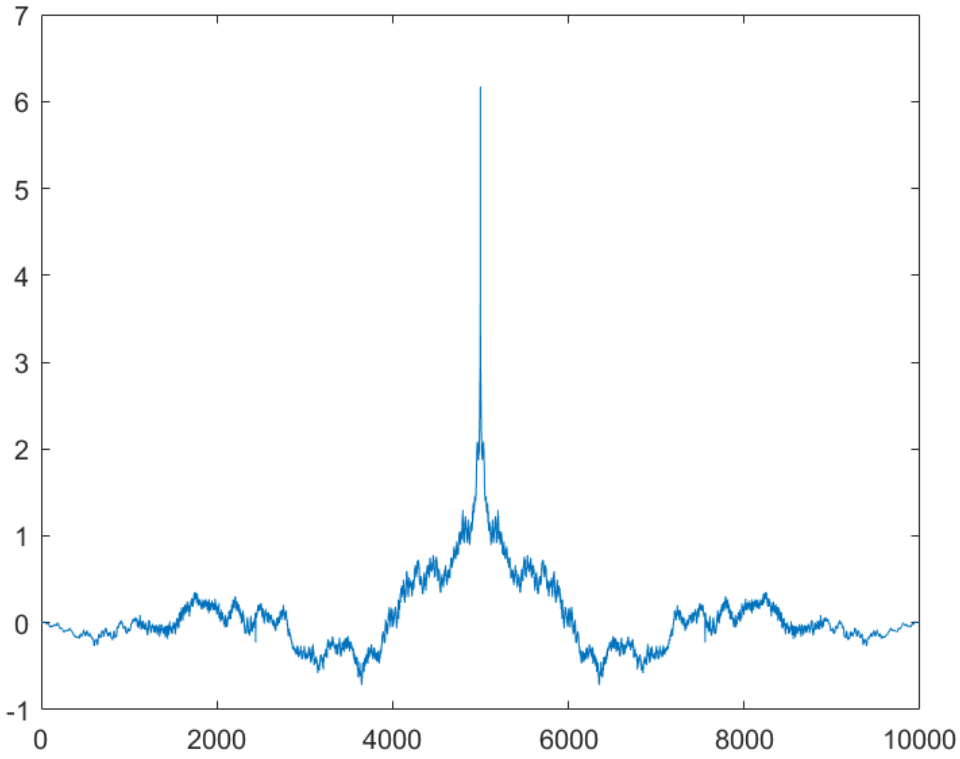


Figure 11. Autocorrelation of North velocity measurements filtered with low covariance

The autocorrelation function contains a sharp point at the center, indicated that most of the error is not time-correlated so each new measurement mostly contains new information. However the edges of Figure 11 taper outwards indicating that some of each measurement has a small amount of time-correlated error. Although most of each measurement is time-independent, the small amount that is not was enough to invalidate the initial approach of calculating the measurement covariance.

In order to account for this non-white noise the measurements can be downsampled so that each measurement contains a higher percentage of new information or the covariance associated with each measurement can be increased so that the filter relies on each measurement less heavily. Since the former would result in a significant

loss of measurement information, the latter approach is used. Through tuning, the covariances in Table 4 were found to produce good results for each algorithm type.

Table 4. Diagonal Covariances of Each VO Algorithm Type

	Inertial-aided (VIO)	Solves for $\mathbf{R}_{C_1}^{C_2}$ (VO)	SVO
$\sigma^2\ (m/s)^2$	20^2	25^2	50^2

The measurement covariance matrix for each class of algorithm can be calculated by multiplying the appropriate value in Table 4 by the identity matrix.

Residual Monitoring.

Since VO accuracy is highly dependent on imagery, it is advantageous to screen measurements before applying them to the filter. Residual monitoring [35] is used to detect measurement outliers with degraded accuracy. If a measurement fails this test then it is ignored. The residual, $\mathbf{r}(t_i)$, is the difference between the measurement (in this case velocity error) and the state (the estimated velocity error, in this case):

$$\mathbf{r}(t_i) = \mathbf{z}(t_i) - \hat{\mathbf{x}}(t_i^-) \tag{10}$$

Since the measurement matrix, $\mathbf{H}$, is identity, the residual covariance is a summation of the estimated velocity error covariance and the measurement covariance:

$$\boldsymbol{\sigma_r}^2 = E\{\mathbf{r}(t_i)\mathbf{r}(t_i)^T\} = \mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i) \tag{11}$$

Each residual is compared to a scaled version of the residual covariance

$$\mathbf{r}(t_i) < n\boldsymbol{\sigma_r} \tag{12}$$

so that if the inequality is false then for any of the residuals then the entire measurement is ignored. For the results presented in this experiment, a multiplier of $n = 2$ was used.

3.2 Data Sources

Most of the input data is derived from a collection of sensors mounted to a fixed-wing aircraft. Each sensor is fixed to the aircraft with a known orientation relative to the origin of the ENU body frame, which is defined as the position of the INS. The shape of the trajectory is shown in Figure 12.

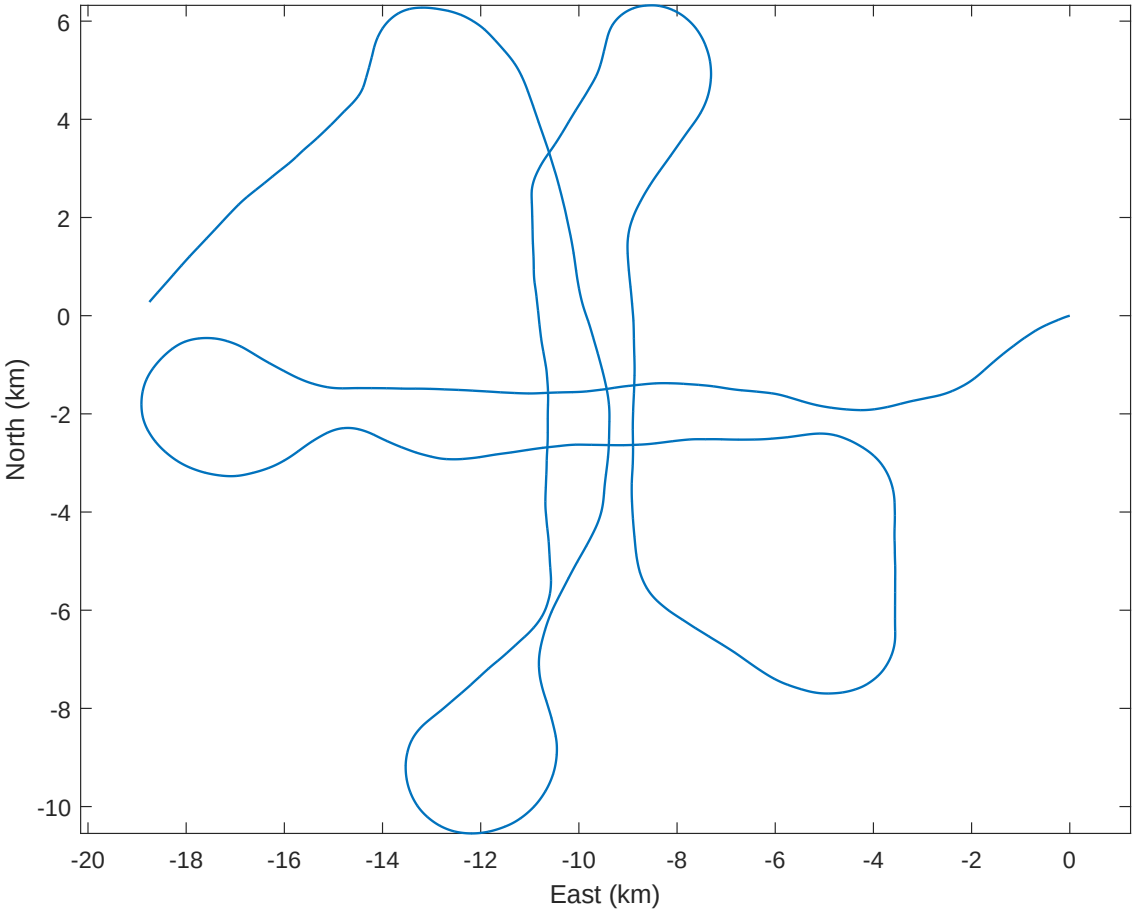


Figure 12. Truth trajectory

This figure shows the changes in position relative to the origin, which is the

position at 0 seconds. The duration of this trajectory is 1800 seconds. Figure 13 shows this same trajectory overlaid onto aerial imagery.

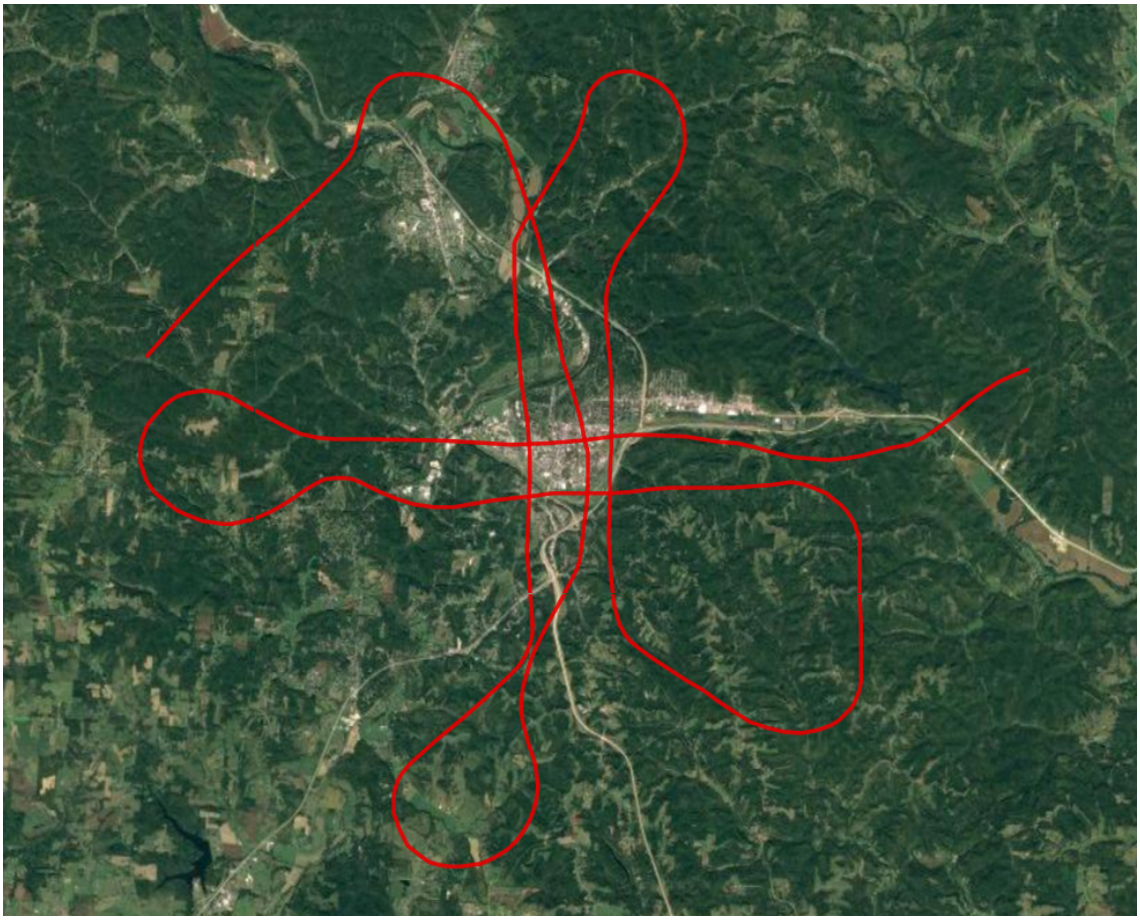


Figure 13. Truth trajectory overlaid on Google Maps using gpsvisualizer.com

This figure shows the diversity of the terrain featured in the imagery, including both rural and urban settings. The city underneath the center of the trajectory is Athens, Ohio.

Imagery.

The imagery data is stored in the Tagged Image File Format (TIFF) and was captured by a Prosilica GC1350 model camera, which has a horizontal resolution of 1360 pixels and a vertical resolution of 1024 pixels. The images were collected at a

rate of 5Hz, although small gaps are present in the imagery set. The intrinsic camera calibration parameters for the camera are shown in Table 5, where the parameter f corresponds to two-dimensional focal length, the parameter c refers to the optical center of the image (which is roughly half of the resolution), and α is the skew coefficient, which only has one dimension.

Table 5. Intrinsic Camera Calibration Parameters

	f	c	α
x	1335.0277	718.1614	
			-0.002
y	1334.8041	492.1758	

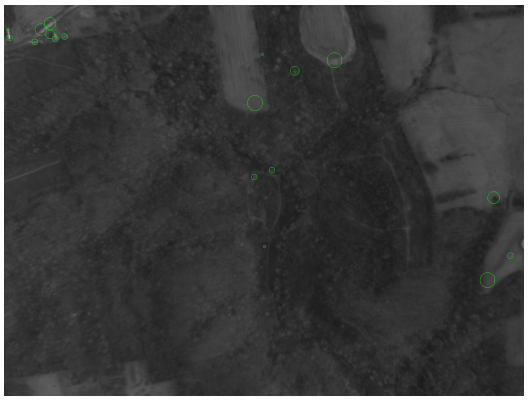
These parameters constitute the camera calibration matrix, $\mathbf{K}$.

$$\mathbf{K} = \begin{bmatrix} f_x & \alpha f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

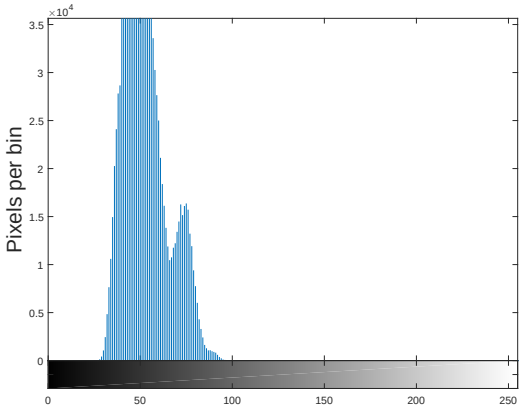
$\mathbf{K}$ is used to convert between pixel coordinates and normalized coordinates:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} \tag{14}$$

In order to improve feature detection these images undergo a histogram equalization. An example image is shown in Figure 14.



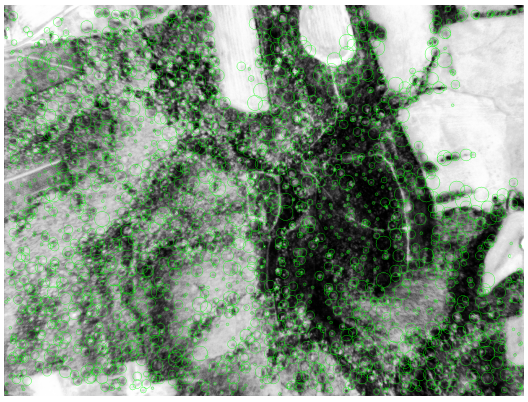
(a) The image with FAST features overlaid in green.



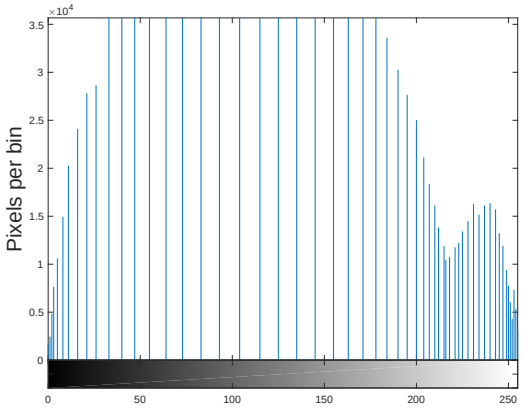
(b) A histogram of the image.

Figure 14. An uncorrected image.

Few features are detected from this image because most of the pixel values are concentrated in one region of the histogram. In order to improve the distribution of pixel values images are processed using OpenCV's equalizeHist algorithm [36], which spreads the intensity values to utilize more of the possible pixel values and increase contrast. The result can be seem in Figure 15.



(a) The image with FAST features overlaid in green.



(b) A histogram of the image.

Figure 15. Image after histogram equalization.

As evident in Figure 15, this process enables the feature detector to locate many more features than before.

Next, the images can be undistorted. The five camera calibrartion distortion coefficients (defined in [37]) are included in Table 6.

Table 6. Camera Distortion Coefficients

k_1	k_2	p_1	p_2	k_3
-0.2655	0.3841	-0.0020	-0.0030	0.0000

Inertial.

The primary inertial sensor used is an HG1700 Honeywell tactical-grade IMU, although there is one experimentation scenario which uses Systron Donner's commercial-grade MMQ-50 INS and one which simulates an HG9900 IMU. The inertial measurements begin as raw $\Delta\mathbf{v}$'s and $\Delta\boldsymbol{\theta}$'s in the frame of the sensor and are available at a rate of 100Hz. Once mechanized, these measurements describe the rotation from the INS frame to the navigation frame. In order to be relevant to the filter, however, they must be rotated so that they are relative to the nose-right wing-down (NRD) body frame. Equation (15) demonstrates this process.

$$\Delta\mathbf{v}_{B_{NRD}} = \mathbf{R}_{B_{RNU}}^{B_{NRD}}\mathbf{R}_{INS}^{B_{RNU}}\Delta\mathbf{v}_{INS} \tag{15}$$

The fixed rotation $\mathbf{R}_{INS}^{B_{RNU}}$ describes the orientation of the INS relative to the right wing-nose-up (RNU) body frame and $\mathbf{R}_{B_{RNU}}^{B_{NRD}}$ is a fixed rotation to convert between the data's RNU body frame and Scorpion's NRD body frame. This DCM is

$$\mathbf{R}_{B_{RNU}}^{B_{NRD}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{16}$$

The orientation of the tactical-grade IMU is

$$\mathbf{R}_{INS}^{B_{RNU}} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

and the orientation of the commercial-grade IMU is

$$\mathbf{R}_{INS}^{B_{RNU}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{18}$$

The NRD raw measurements are converted into pose and velocity estimates using standard INS mechanization equations [38].

Barometer.

The barometer data consists of a series of Mean Sea Level (MSL) altitude measurements at a variable rate which is close to 25Hz. The barometer has a time-varying bias as shown in Figure 16. The bias has been calibrated so that it is equal to zero initially.

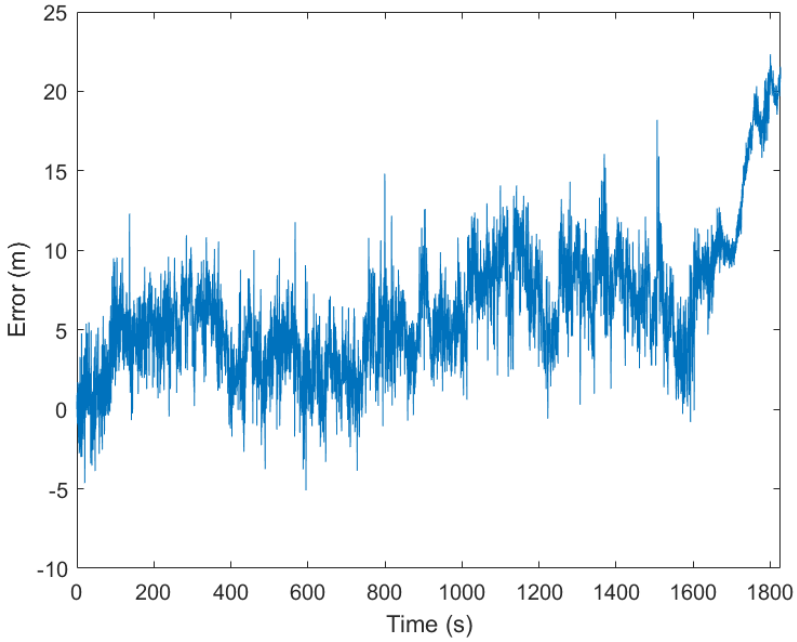


Figure 16. Barometer bias example (m)

This bias is modeled as a time-correlated FOGM with a variance σ and time constant τ. The maximum value of the autocorrelation function of the bias yields $\sigma^2 = 7.60^2 m^2$. The time constant τ can be calculated by finding the value of the autocorrelation function at $0.368\sigma^2$, resulting in $\tau = 21.25s$.

Truth.

The truth data comes from a Novatel SPAN GPS/INS system. It is available at a rate of 10Hz. This data was used to initialize the INS mechanization result and to evaluate the experimental results.

Terrain Elevation.

During some of the preliminary results large, time-correlated errors were observed. Through debugging, it was discovered that the original assumption that the local ground was a plane with a constant elevation was invalid for the data set. In order to

rectify these errors, it was assumed that the the terrain consisted of a series of planes such that at every second the true position of camera was above the center of a new plane. The elevation of each plane was pulled from Google Maps API.

3.3 Visual Odometry Algorithms

Five VO algorithms have been implemented. The first four algorithms are strictly two-frame visual odometry and share many of the same processes. The last algorithm, SVO, stores information from previous images making it a multi-frame visual odometry method although it is incorporated into the filter on a two-frame by two-frame basis.

Two-Frame Visual Odometry.

These four algorithms rely on OpenCV algorithms [36] for tracking, feature detection and description, outlier rejection, and rotation estimation. Two of the algorithms match feature descriptors while the other two track pixel intensities. Two rely on the INS to estimate the rotation between frames while the other two estimate this rotation themselves. Their overall structure is summarized in Figure 17.

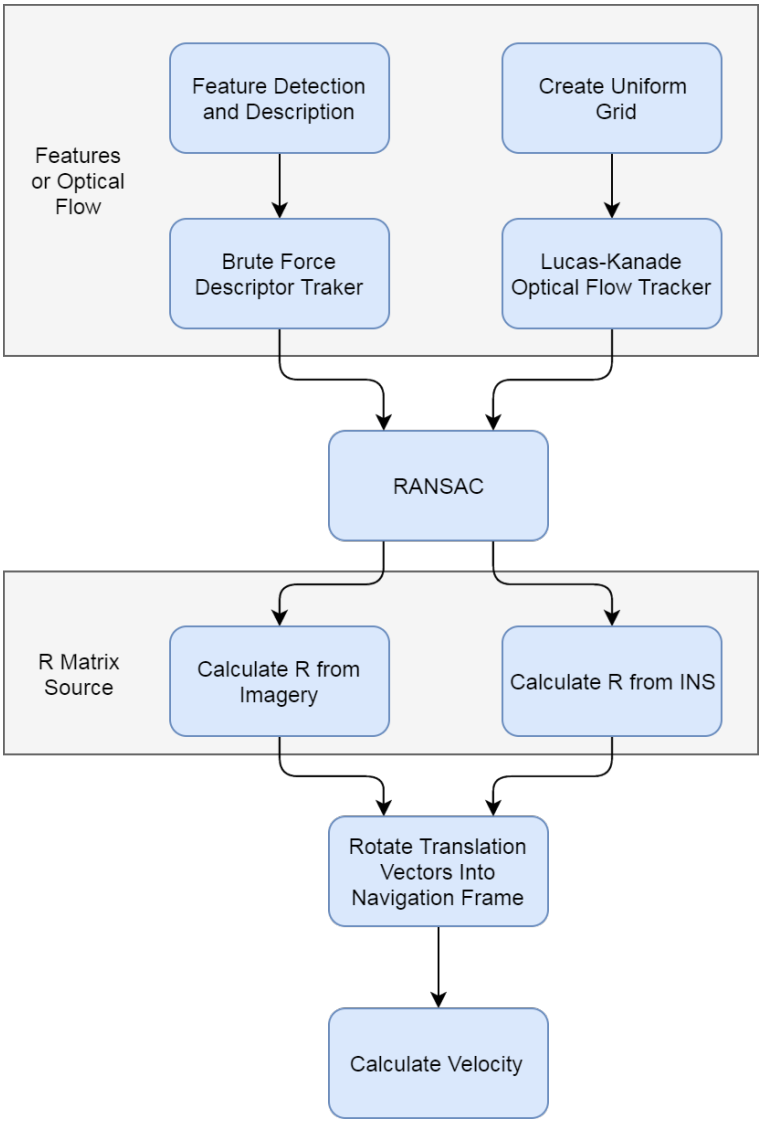


Figure 17. The Four Two-Frame Visual Odometry Algorithms Represented By A Single Flowchart

The square-cornered boxes represent a choice between the two tracks of smooth-edged boxes inside. The first choice is between feature-based VO and optical flow VO. The first option uses openCV's AKAZE feature detector and descriptor [36], followed by a brute force descriptor matcher. AKAZE stands for "Accelerated KAZE" [39], where "kaze" translates to "wind" in English. The default feature detector threshold (defined by OpenCV's AKAZE class reference) is set to 0.008, but it is adjusted in increments of 0.003 at a time to ensure that there are always at least 200

features detected but no more than 2000 features detected in the first image. The matcher operates based on Hamming distance, which is appropriate for binary-string-based descriptors like those that AKAZE produces. The matches are subject to a preliminary outlier rejection check where the first distance of every match (d_1) must be less than 0.8 times the second distance (d_2):

$$d_1 < 0.8 \cdot d_2 \tag{19}$$

If this statement is false, then the match is discarded. This eliminates the worst matches before RANSAC, improving the accuracy of the result and reducing the number of matches that RANSAC needs to evaluate.

The second option (optical flow) selects pixel-features along an evenly spaced grid of predetermined pixel locations. These points are matched using OpenCV's iterative Lucas-Kanade tracker with pyramids, "calcOpticalFlowPyrLK", to track the points from the first frame to the second frame. The tracker is set to use four pyramid levels with a search window size set to 31x31 for each level. The tracker's "minEigThreshold" is set to 0.001. This threshold is compared to a "spatial gradient matrix" [40] divided by the number of pixels in a window. If this value is less than 0.001, then the point is not used. The tracker is set to run until it performs 20 iterations or until the search window moves less than an epsilon of 0.03.

After this, every method employs RANSAC for outlier rejection and to calculate the essential matrix. This is done using OpenCV's "findEssentialMat" function with a confidence level set to 0.9999 and a maximum distance from the epipolar line set to 0.1.

At this point in the structure is the second choice which distinguishes the algorithms from each other. The second choice determines how the rotation matrix, $\mathbf{R}$, which rotates from the first camera frame to the second camera frame, is determined.

For one, it can be determined by the VO algorithm by decomposing the essential matrix that is calculated using RANSAC. Alternatively, it can be determined by the INS's filtered measurements. Thus the four possible combinations of two binary options make up the four two-frame algorithms.

At this point every algorithm rotates the set of features from pixel coordinates to navigation frame coordinates. The first step in this process is to convert the pixel coordinates to normalized camera coordinates

$$\mathbf{x}_n = \frac{\mathbf{x}_{n_0}}{|\mathbf{x}_{n_0}|} \tag{20}$$

where $\mathbf{x}_{n_0}$ is the un-normalized and unscaled camera coordinates given by

$$\mathbf{x}_{n_0} = \begin{bmatrix} (x_p - c_x)/f_x \\ (y_p - c_y)/f_y \\ 1 \end{bmatrix} \tag{21}$$

In Equation (21), f_x, f_y, c_x, and c_y are camera calibration parameters detailed in Section 3.2 and x_p and y_p are the pixel coordinates.

The normalized camera coordinates need to be scaled, requiring a depth vector to be calculated for each point. The first step to do this is to rotate the normalized camera coordinates into the NED navigation frame, resulting in unscaled navigation coordinates. The rotation can be accomplished with the DCM $\mathbf{R}_C^{Nav}$. This DCM is applied to the normalized camera coordinates

$$\mathbf{x}_{Nav} = \mathbf{R}_C^{Nav}\mathbf{x}_n \tag{22}$$

Two different DCM's, $\mathbf{R}_{C_1}^{Nav}$ and $\mathbf{R}_{C_2}^{Nav}$, are actually necessary to rotate each set of points. If the an initial camera-to-navigation DCM $\mathbf{R}_{C_1}^{Nav}$ is known, then the second

camera-to-navigation DCM can be calculated from the first using the transpose of the camera rotation matrix, $\mathbf{R}_{\mathbf{C_1}}^{\mathbf{C_2}}$, according to Equation (23).

$$\mathbf{R}_{C_2}^{Nav} = \mathbf{R}_{C_1}^{Nav}(\mathbf{R}_{C_1}^{C_2})^T \tag{23}$$

This DCM can then be used at the following timestep as the initial camera-to-navigation DCM, eliminating dependence on the INS. However, since this method will result in a significant attitude drift (and thus velocity drift), this experiment does not propagate the camera-to-navigation matrix using the VO. Instead, a new initial camera-to-navigation matrix is calculated from the INS at each timestep. This calculation is described by Equation (24).

$$\mathbf{R}_{C}^{Nav} = \mathbf{R}_{B_{NRD}}^{Nav}\mathbf{R}_{B_{RNU}}^{B_{NRD}}\mathbf{R}_{C}^{B_{RNU}} \tag{24}$$

$\mathbf{R}_{C}^{B_{RNU}}$ is a fixed rotation describing the orientation of the camera relative to the body frame, $\mathbf{R}_{B_{RNU}}^{B_{NRD}}$ is a conversion from the RNU body frame to the NRD body frame, as described in Equation (16), and $\mathbf{R}_{B_{NRD}}^{Nav}$ is a variable DCM. $\mathbf{R}_{B_{NRD}}^{Nav}$ is taken from the filter attitude solution, which is primarily fed by the INS.

So far the calculations to obtain navigation coordinates have been described, but these coordinates are not scaled. The depth vector to each unscaled camera coordinate can be obtained if the angle, θ is known. Since θ is also the angle between each unscaled navigation coordinates and the unit depth vector

$$\mathbf{h}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{25}$$

$\cos\theta$ can be calculated by relating $\mathbf{h}_0$ and $\mathbf{x}_{Nav}$.

This relation is

$$\cos\theta = \frac{\mathbf{x}_n^{Nav} \cdot \mathbf{h}_0}{|\mathbf{x}_n^{Nav}||\mathbf{h}_0|}$$

$$\cos\theta = \frac{z_n^{Nav}}{|\mathbf{x}_n^{Nav}|}$$

$$\cos\theta = \frac{z_{Nav}}{\sqrt{x_{Nav}^2 + y_{Nav}^2 + z_{Nav}^2}} \tag{26}$$

From here, θ is used to calculate the scaled depth d to each feature, given h, the height of the camera above the terrain. Their relationship is visualized in Figure 18.

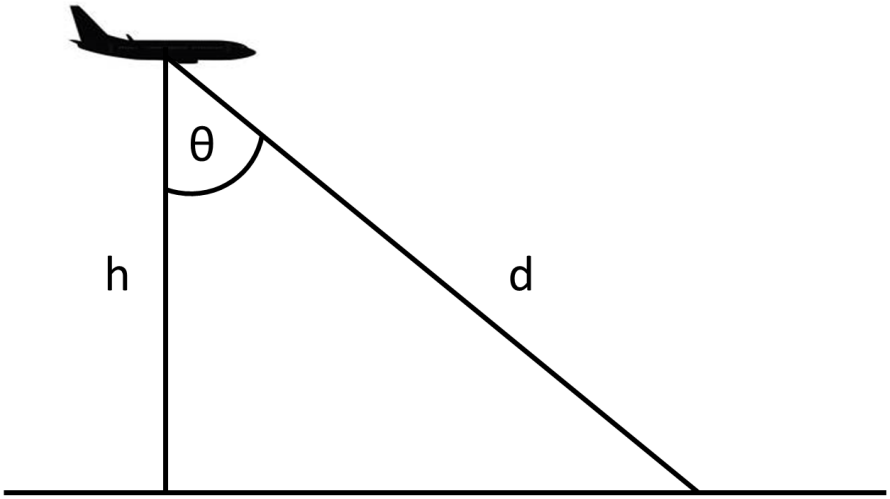


Figure 18. Geometric relationship between the height above ground and depth vector as well as the angle that they form

Mathematically, this is

$$cos\theta = \frac{h}{d}$$

$$d = \frac{h}{cos\theta} \tag{27}$$

The depth is used to scale the normalized camera coordinates:

$$\mathbf{x}_C = d \cdot \mathbf{x}_n \tag{28}$$

where $\mathbf{x}_C$ are the scaled camera coordinates.

Once again, both sets of coordinates are rotated into the navigation frame:

$$\mathbf{x}_{Nav} = \mathbf{R}_C^{Nav}\mathbf{x}_C \tag{29}$$

this time resulting in scaled navigation frame coordinates. Now that both sets of coordinates are scaled and represented in a common frame, the velocity measurement can be calculated. The first set is subtracted by the second set, resulting in a vector field.

$$\mathbf{t}_i = \mathbf{x}_{Nav_{1_i}} - \mathbf{x}_{Nav_{2_i}} \tag{30}$$

Each vector in this field is averaged together, excluding the outliers that were detected earlier, to get a single translation vector:

$$\mathbf{t} = \frac{\sum_{i=1}^{n} \mathbf{t}_i}{n} \tag{31}$$

where n is the number of inliers and $\mathbf{t}_i$ contains no outliers. In the last step of generating the velocity measurement, the translation vector is differentiated to get the mean velocity between frames. This is done by dividing the translation vector by the time which elapsed from the first image to the second:

$$\mathbf{v} = \frac{\mathbf{t}}{\Delta t} \tag{32}$$

This three dimensional velocity is the final result of the VO. It should be noted that using these methods gives little insight into the third dimension, so it will be

ignored by the filter.

Multi-Frame Visual Odometry.

The last algorithm, and the only multi-frame algorithm, is Forster's open source SVO implementation [6]. This semi-direct visual odometry algorithm is a good example of a hybrid between feature-based VO algorithms and optical flow VO algorithms. This algorithm contains two threads: one which maps out the environment and one which estimates the motion between frames. Forster's flowchart detailing the structure of these two threads is shown in Figure 19.

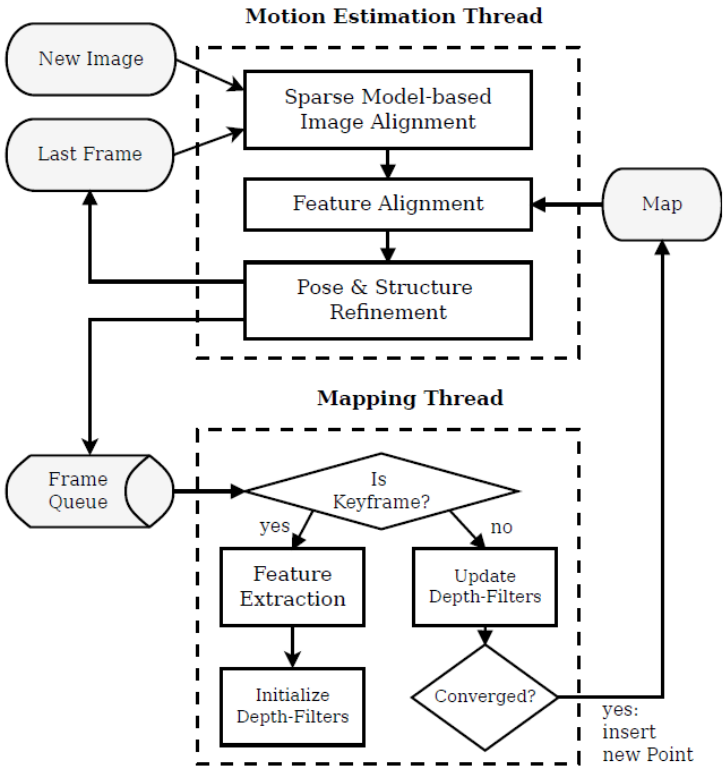


Figure 19. Forster's SVO Algorithm [6]

This method utilizes point-features for most of the motion estimation and only uses feature detection when a keyframe is selected in order to initialize new 3D points, saving computational time.

While the SVO is designed to perform accurate localization and mapping for an entire trajectory, it may still provide a better estimation of motion than the other non-inertial-aided two-frame algorithms. The SVO is used to generate a series of trajectory points in the "world" frame, which is the frame of the first camera keyframe, and a set of rotations, which provide the DCM's necessary to rotate from the world frame to the current camera frame. Unlike the other algorithms, this algorithm does not rely on the filter for continual height solutions. Instead, it is initialized with the height above ground and estimates this value afterwards.

The velocity measurement is calculated by taking the difference between two sets of world coordinates ($\mathbf{x}_{w,t_1}$ and $\mathbf{x}_{w,t_2}$), dividing by the time elapsed (Δt), then rotating the resulting velocity from the world frame to the current camera frame:

$$\mathbf{v}_{C_2} = (\frac{\mathbf{x}_{w,t_2} - \mathbf{x}_{w,t_1}}{\Delta t})\mathbf{R}_w^{C_2} \tag{33}$$

where $\mathbf{R}_w^{C_2}$ is the rotation from the world frame to the last camera frame and $\mathbf{v}_{C_2}$ is the velocity in the last camera frame. From here, the velocity is rotated from the camera frame to the navigation frame using

$$\mathbf{v}_{Nav} = \mathbf{R}_C^{Nav}\mathbf{v}_C \tag{34}$$

This velocity can then be used to update the filter.

In order to avoid memory issues, SVO is reset approximately every 500 seconds.

Visual Odometry Algorithm Summary.

These VO algorithms can be described by whether they feature-based VO or optical flow VO and whether they rely on an inertial system to estimate the rotation matrix or attempt to estimate it directly. Every algorithm is categorized based on

these descriptions in Table 7.

Table 7. Categorization of VO Algorithms

	OF-VIO	OF-VO	Feature-VIO	Feature-VO	SVO
Tracks Feature Descriptors			✓	✓	✓
Tracks Grid of Pixels	✓	✓			✓
Inertial-Aiding	✓		✓		
Estimates R Matrix		✓		✓	✓

These names will be used from this point on to refer to the algorithms defined in this section. The "OF" prefix indicates that the algorithm performs optical flow visual odometry while the "feature" prefix denotes that the algorithm performs feature-based visual odometry. The "VIO" suffix indicates that the algorithm relies on inertial data to calculate $\mathbf{R}$ while the "VO" suffix indicates that the algorithm directly estimates $\mathbf{R}$. SVO is simply the hybrid method's acronym.

3.4 Conclusions

This chapter has detailed the framework of this experiment, including the filter design. The data sources and any data preprocessing has been described. Also, the visual odometry algorithms have been classified, named, and explained.

IV. Analysis

1As the results of this experiment are analyzed, it will be helpful to define a nominal case as a reference point to compare all other cases against. The nominal case will use OF-VIO with rectified images and the tactical-grade INS. Variations of the nominal case include the use of a lower-quality INS (commercial INS case), a higher-quality INS (navigation INS case), the exclusion of image rectification (image distortion case), the use of a partial INS reset (vertical-only INS reset case), and the use of a UKF (UKF case).

4.1 Filter Performance

In order to demonstrate that the filter is operating properly, the state errors will be examined. The nominal case is used to generate these results.

Figure 20 shows the error in the position and velocity states with the filter-computed 1-σ value.

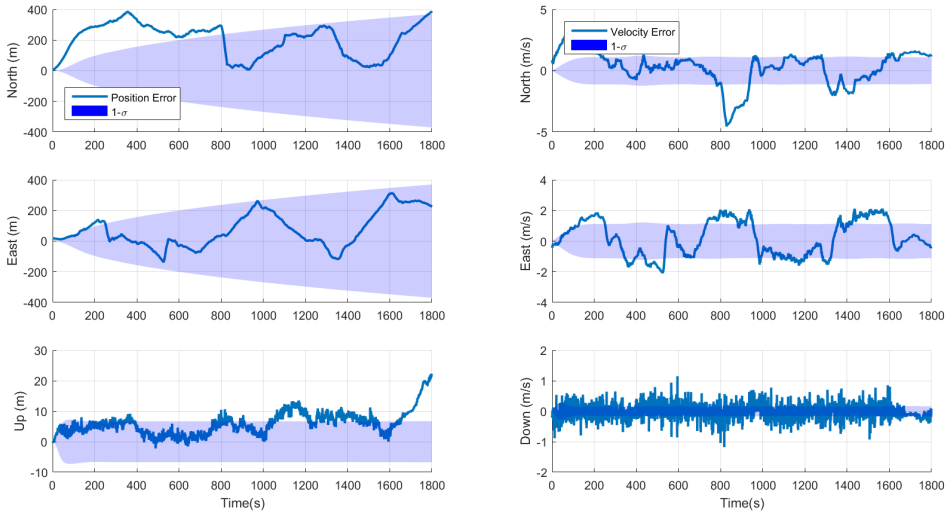


Figure 20. Error on the Position and Velocity States, Nominal Case

The position and velocity error appears to match the $1-\sigma$ covariance, indicating that the filter is reasonably tuned. The Up position, East velocity, and West velocity covariances do not grow since they are bound by (essentially absolute) direct measurements from the barometer and VO, respectively. There is a sharp error in the up position state, however, due to the sharp increase in the barometer bias, as seen in Figure 16. Although the down velocity state is not bound by a direct update, this dimension benefits from the absolute altitude update as well as a smaller scale of velocities.

Figure 21 shows the error on the attitude states, or the tilt error.

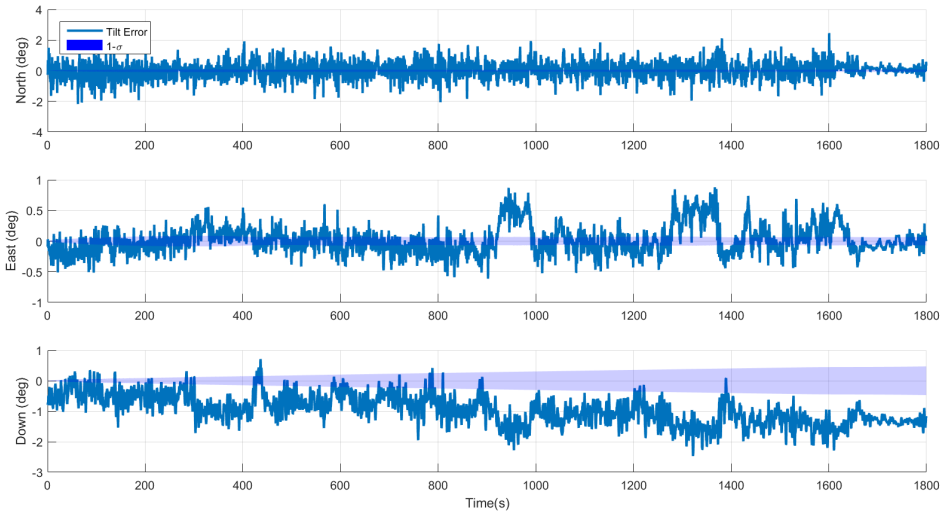


Figure 21. Error on the Attitude States, Nominal Case

The tilt errors about the North and East directions are significant, but remain centered around zero. The Down tilt error, however, does experience a significant drifting bias. This occurs because the roll and pitch of the aircraft vary but any tilt about the North or East axes will result in an observable velocity drift due to misapplication of gravity. The yaw of the aircraft, on the other hand, is unobservable without an absolute horizontal acceleration update.

Figure 22 shows the last states, which are the bias states.

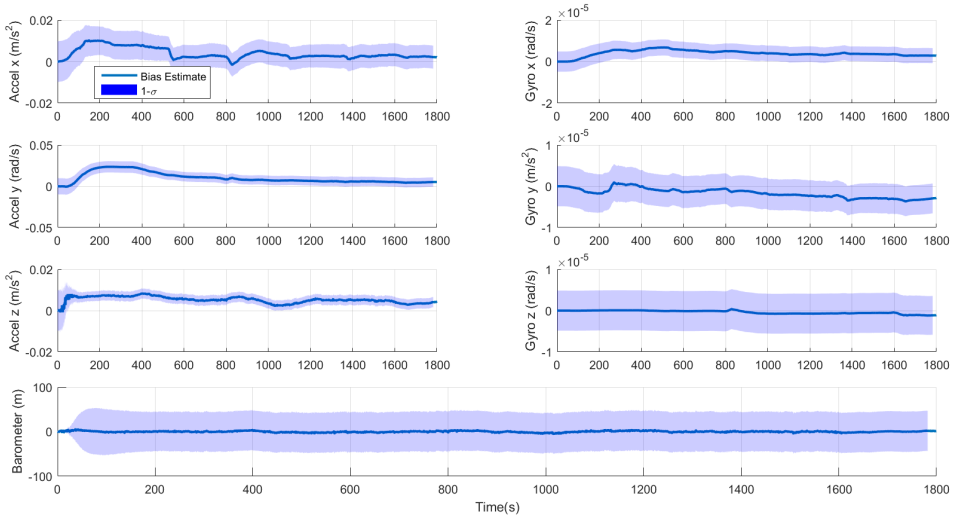


Figure 22. Accelerometer, Gyroscope, and Barometer Bias States, Nominal Case

These states do not have any truth data associated with them since the sensors they are associated with are real and not simulated. Thus Figure 22 does not contain error plots but plots of the actual state estimates.

4.2 Measurement Accuracy

The accuracy of each VO algorithm's measurements will be evaluated by calculating the RMS of the measurement error along each axis and the tilt errors of the estimated **R** matrix. The difference between the measured velocity and true velocity is defined as the measurement error. The measurement error for each algorithm is shown in Figure 23.

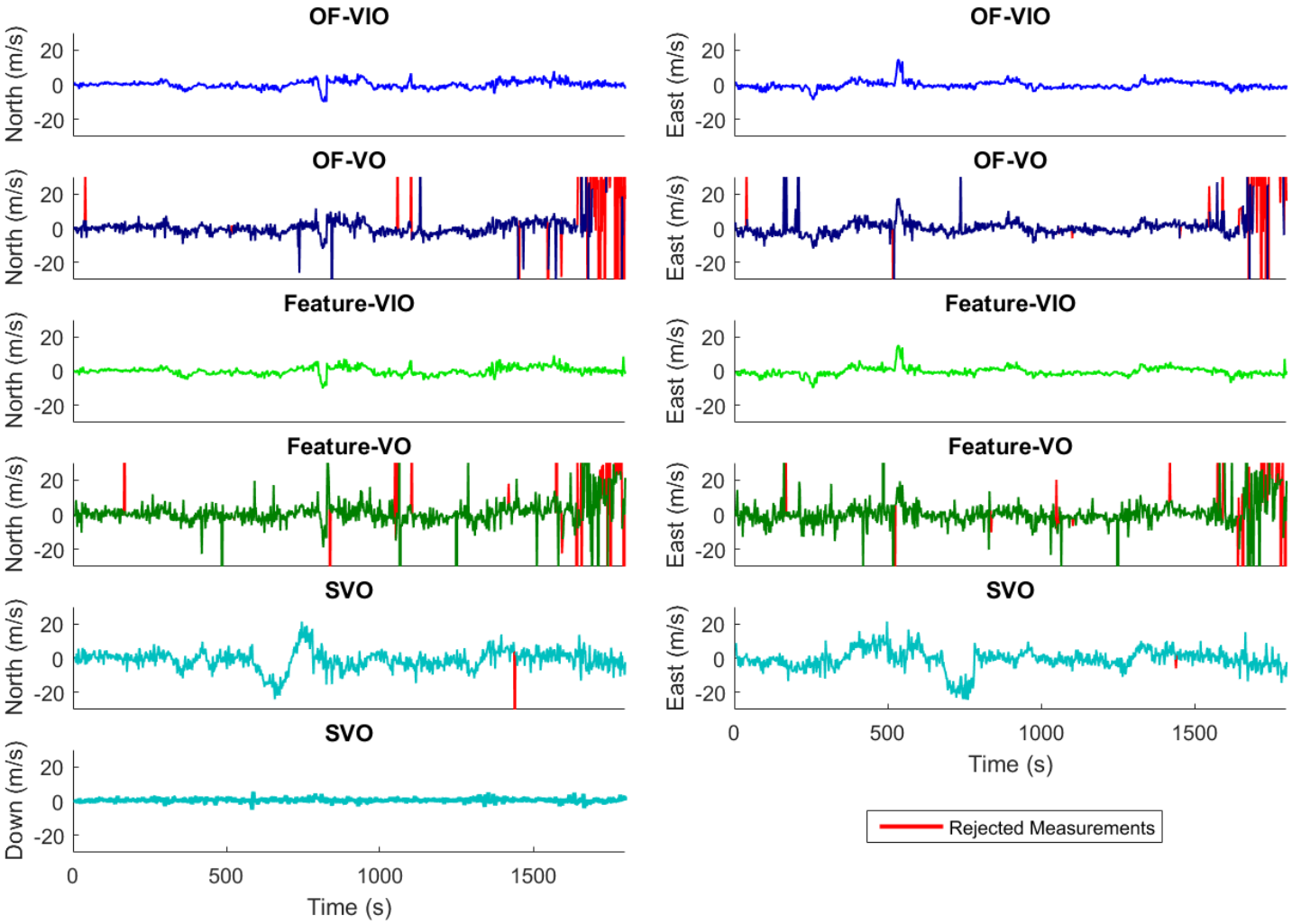


Figure 23. Velocity measurement error with for each algorithm

The color blue is associated with the OF algorithms, the color green plots with the feature-based algorithms, and blue-green, or cyan, is associated with the hybrid SVO method. The darker shades of blue and green indicate that the algorithm estimated the $\mathbf{R}$ matrix while the lighter shades indicate that the algorithm is given $\mathbf{R}$ by the filtered INS measurements. A red value indicates that the plotted measurement was rejected by the filter using the residual monitoring discussed in Section 3.1. Remember that SVO is the only algorithm which provides a velocity in the down direction, which

is why there is only one plot associated with the down velocity.

From this figure it can easily be seen that the inertial-aided algorithms outperform every other algorithm. In order to analyze these errors more precisely, the RMS of each error plot was calculated. These values are displayed in Table 8.

Table 8. RMS Error of Accepted VO Measurements

	North (m/s)	East (m/s)	Down (m/s)
OF-VIO	2.24	2.32	
OF-VO	3.62	3.74	
Feature-VIO	2.13	2.11	
Feature-VO	5.01	5.19	
SVO	6.37	6.67	1.42

From this table we can see that Feature-VIO measurements have the highest accuracy, followed closely by OF-VIO. Behind them are OF-VO and Feature-VO, respectively. The least accurate algorithm is the SVO, which is understandable since this algorithm attempts to track the height above the ground. Note that measurements rejected by the filter are omitted from the RMS calculation.

Since the only difference between the inertial-aided (VIO) and the loosely-coupled (VO) algorithms is in the way the $\mathbf{R}$ rotation matrix is calculated by each, it can be inferred that the difference in measurement error is influenced by the $\mathbf{R}$ used in each algorithm

For reference, the Euler angle form of the truth $\mathbf{R}$ is shown in Figure 24.

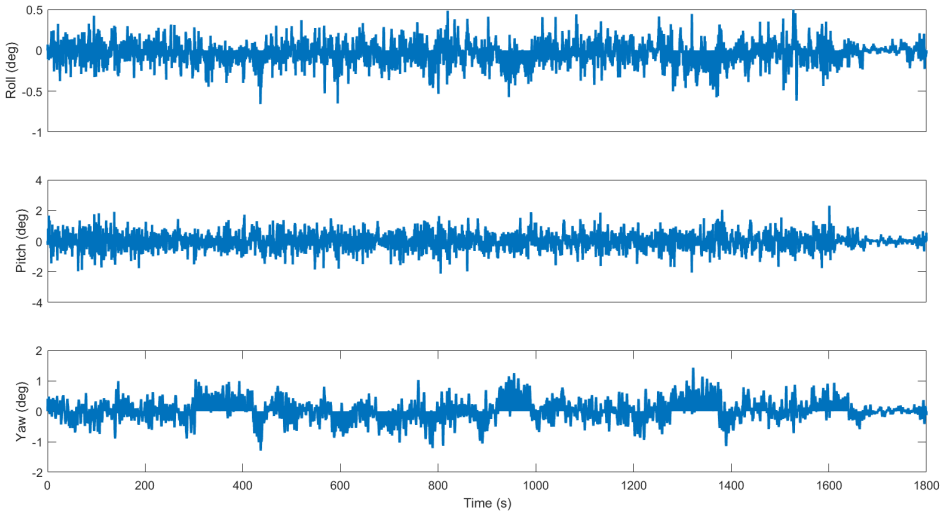


Figure 24. R calculated from truth data in Euler angle form

From this figure, it can be noted that **R** is small, with values on the order of a degree.

Figure 25 shows the tilt errors of the **R** matrix utilized by each visual odometry algorithm. This is a measure of how accurately each algorithm is able to estimate the rotation between camera frames. For the inertial-aided algorithms, this is calculated from the filter's attitude solution and not by the VO.

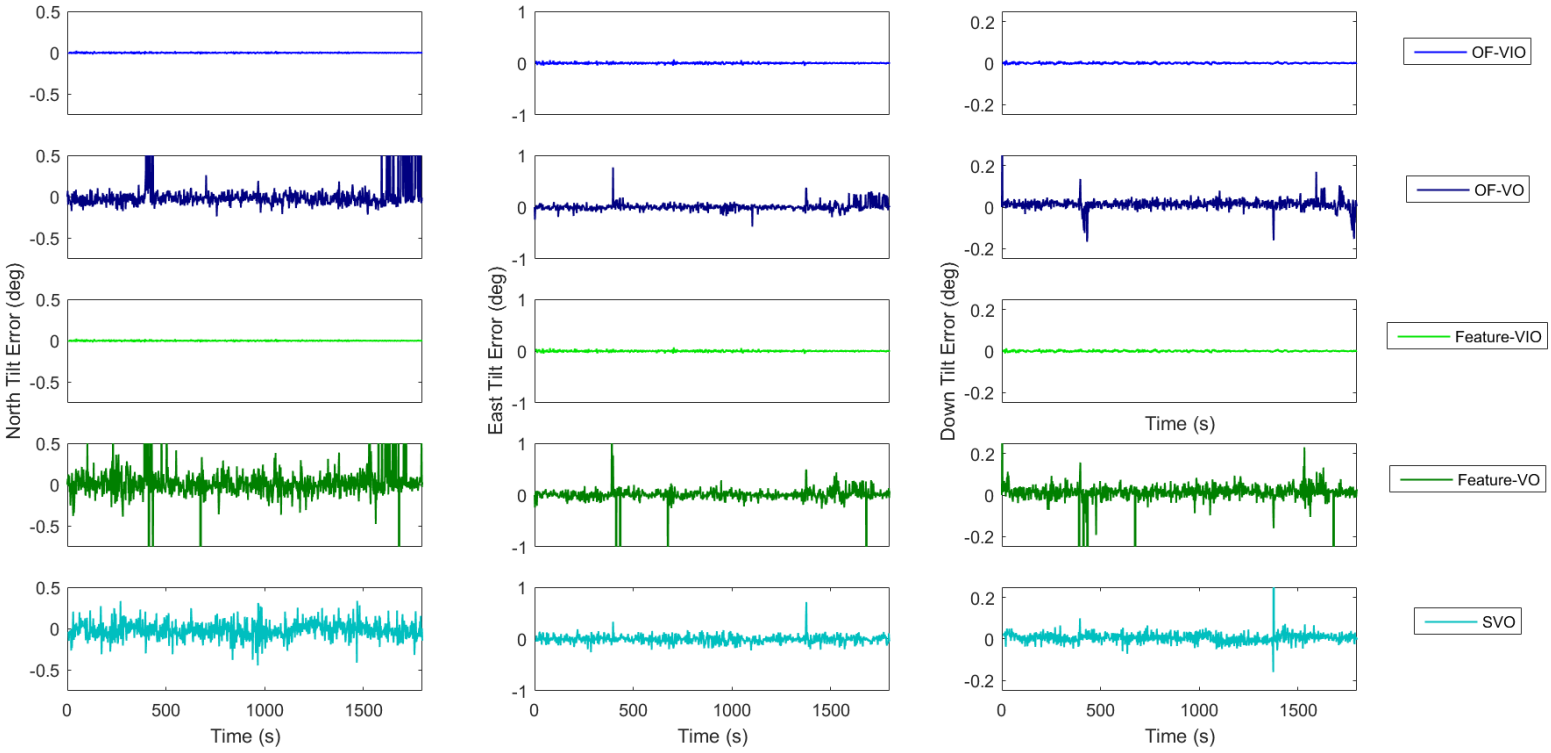


Figure 25. δR used by each algorithm in Euler angle form

As expected, the filter solution results in the most accurate **R**, which is used by both of the VIO algorithms. This figure reveals part of the source of Feature-VO's measurement error, since this algorithm is not able to estimate the **R** matrix as accurately as OF-VO can, evidenced by the greater tilt errors about the x-axis. This might be because optical flow tracks points from all over the images while the feature detector might only detect small patches of features in localized areas of each image.

Table 9 contains the RMS values of the tilt error.

Table 9. RMS of R tilt error

	X (deg)	Y (deg)	Z (deg)
OF-VIO	0.01	0.04	0.00
OF-VO	12.8	0.45	18.6
Feature-VIO	0.01	0.04	0.00
Feature-VO	10.0	0.98	21.1
SVO	0.12	0.25	0.07

From this chart we can see that the inertial-aided algorithms use an almost identical $\mathbf{R}$, which means that differences in the VO-VIO and Feature-VIO measurements do not affect the attitude solution to a significant degree.

For the most part, OF-VO is able to more accurately estimate $\mathbf{R}$ than Feature-VO. This is one of the reasons that OF-VIO outperforms Feature-VIO.

SVO's $\mathbf{R}$ matrix is more accurate than that of the algorithms which independently calculate $\mathbf{R}$. This is surprising given that the velocity error of SVO is greater than every other algorithm. This is partially due to the fact that SVO attempts to estimate the change in height, which works against it since the filter can provide a more accurate altitude.

For comparison purposes, Figure 26 shows the truth trajectory alongside the trajectory generated by INS and barometer measurements.

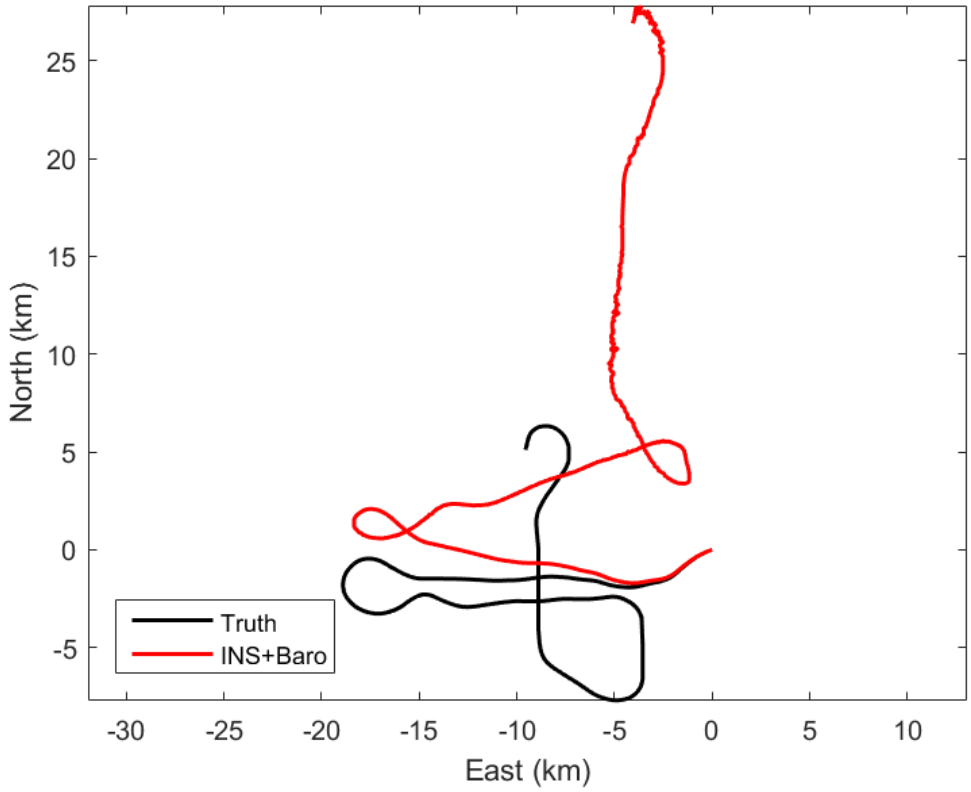


Figure 26. Trajectory resulting from INS and barometer measurements only, 1000 seconds

These trajectories include only the first 1000 seconds of the trajectory results since the solution without VO measurements becomes unstable by this time.

Figure 27 illustrates the difference in resulting trajectories from filtering each VO algorithm's measurements, in addition to barometer and IMU data.

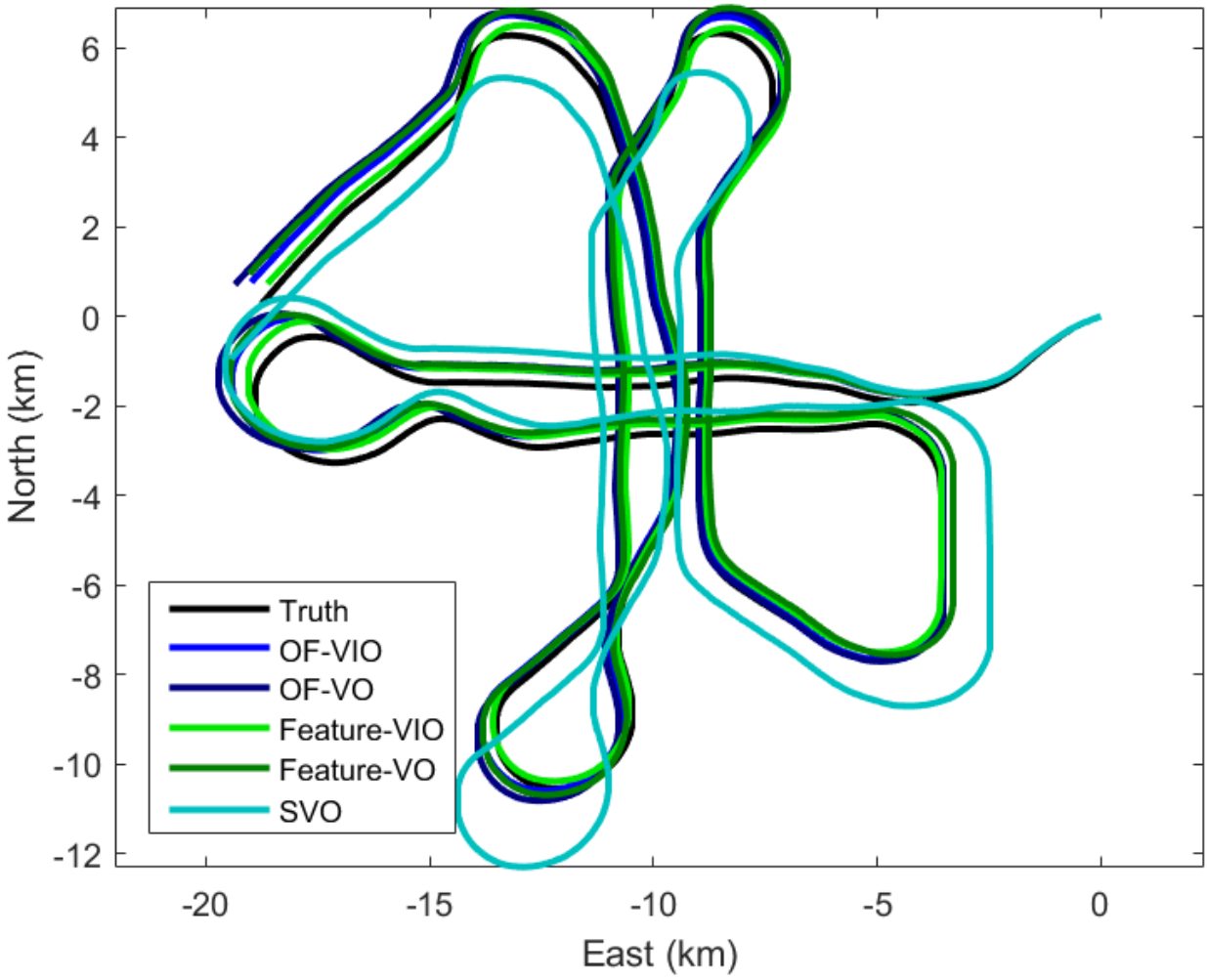


Figure 27. Trajectories resulting from each VO algorithm's measurements

This figure shows that each VO algorithm enables the filter to follow the truth trajectory for the entire 1800 seconds, although the solution which uses SVO measurements contains a little more error that that of the other algorithms.

The 2DRMS of the position error is shown for each algorithm in Table 10.

Table 10. 2DRMS of Position Error

	2DRMS	Change from nominal
OF-VIO	646	
OF-VO	867	34%
Feature-VIO	704	9%
Feature-VO	837	29%
SVO	2068	219%

For the most part, the 2DRMS of the position error follows the RMS of the measurement errors for each algorithm. However, even though the RMS of the measurement errors for Feature-VO was higher than that of OF-VIO, using Feature-VO actually happened to result in a lower 2DRMS of the position error.

4.3 Commercial-Grade IMU

In order to evaluate the necessity of a HG1700 INS, the commercial-grade MMQ 50 INS is substituted for the tactical-grade INS in the nominal case. The state covariances are adjusted to the specific INS specifications but all other covariances are unchanged. This case is limited to 300 seconds since the measurements become unstable shortly after this time, as seen in Figure 28, which depicts the solution for this time. All other aspects of the nominal case apply to the commercial INS case.

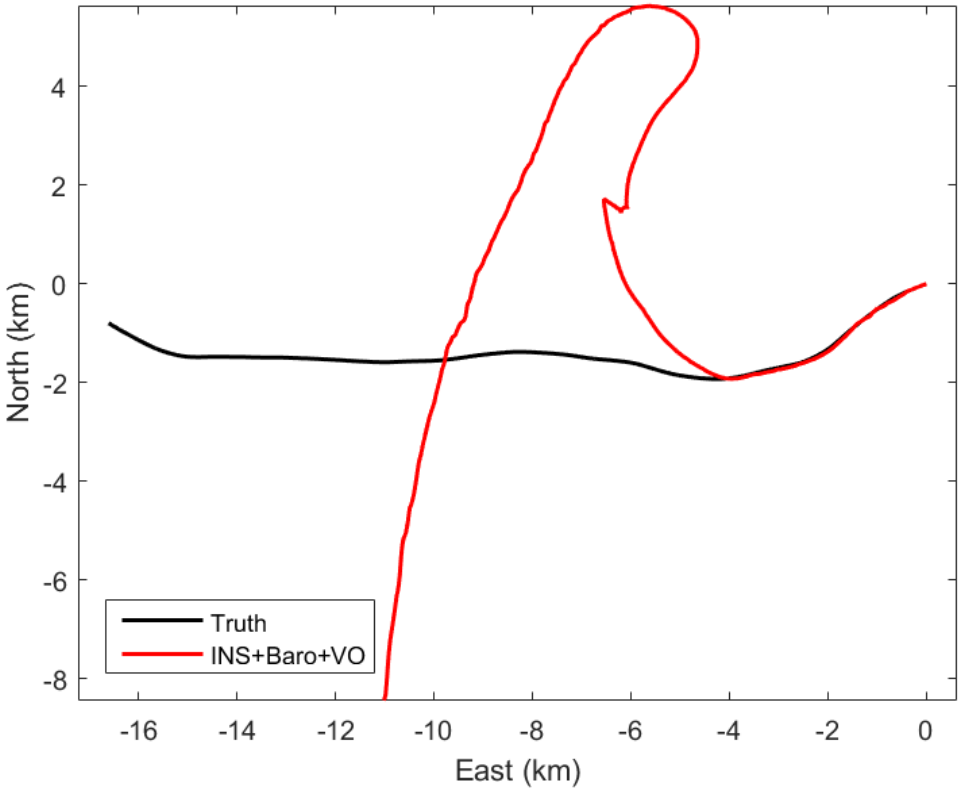


Figure 28. Commercial Case Trajectory, 300 Seconds

The accuracy of the VO measurements in the nominal case is dependent on two inputs from the filter: the altitude and attitude. Since the altitude is constrained by the barometer, any change in performance will mainly result from a degraded attitude solution. Figure 29 shows the tilt errors for this case.

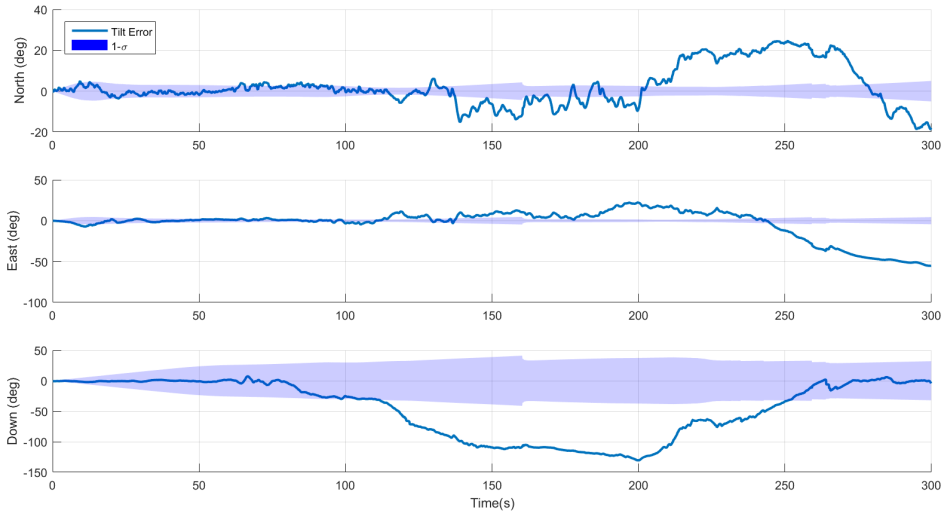


Figure 29. Error on the Attitude States, Commercial INS Case

As expected, the attitude solution is significantly degraded. Errors have gone from the order of a a couple degrees to hundreds of degrees.

Figure 30 shows the error in the measurements produced for this case.

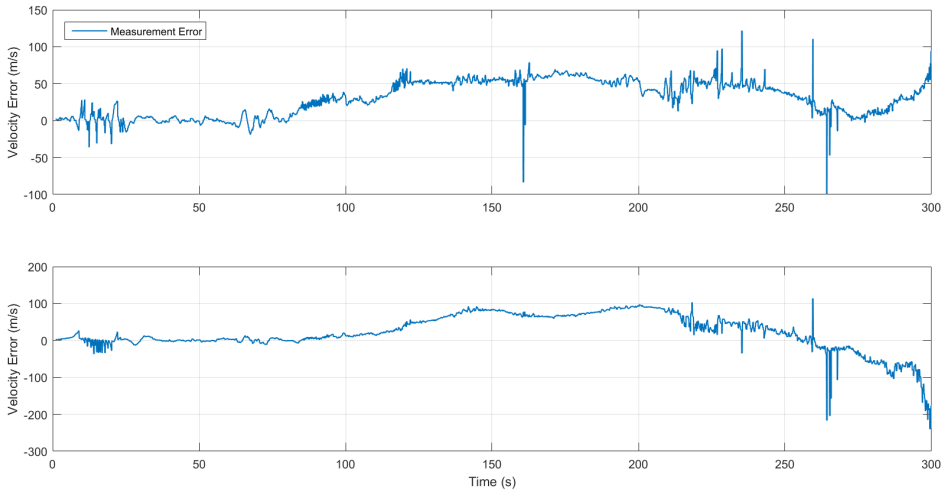


Figure 30. VO Measurement Error, Commercial INS Case

During the first 300 seconds, the measurements are noisy but stable. It is easy to see that the measurement velocity accuracy is severely degraded. Table 11 quantifies this

degradation.

Table 11. RMS Error of Measurements of Commercial Case, First 300 Seconds

	North	East
Commercial	33.9 m/s	45.6 m/s
Tactical	2.29 m/s	2.37 m/s
Reduction	93%	95%

In all, using a tactical INS over a commercial tactical INS reduces the RMS measurement error by over 90% in each direction.

Part of the difference in error between these two cases is caused by increased errors in the attitude solution, which will result in a degraded $\mathbf{R}$ being used by OF-VIO. Table 12 quantifies the degradation of $\mathbf{R}$, showing the RMS of the Euler angle tilt errors of $\mathbf{R}$ for the commercial case and the first 350 seconds of the nominal case.

Table 12. RMS of R Tilt Error as RPY For Each IMU, First 300 Seconds

	X	Y	Z
Commercial	0.264°	0.286°	0.970°
Tactical	0.0085°	0.0040°	0.0181°
Reduction	97%	99%	98%

From this table, we see that the tilt error present in $\mathbf{R}$ increases by nearly a factor of to 100 in every axis.

Table 13 shows the 2DRMS of the position error for the commercial INS case and the first 300 seconds of the nominal case.

Table 13. 2DRMS of Position Error for Commercial and Tactical IMUs, First 300 Seconds

	2DRMS
Tactical	646 m
Commercial	11,429 m
Reduction	94%

From this table, it can again be seen that the use of the commercial INS severely degrades performance. The errors in the commercial INS case have begun to grown exponentially without bounds, even after only 300 seconds.

In the case of the commercial tactical INS, the VO measurements are not accurate enough to prevent the INS from degrading and the INS measurements are not accurate enough to retain stable VO measurements. After a few hundred seconds, errors in the commercial INS case begin to grow exponentially and this case becomes unstable. Because of this, the tactical INS is used in every other case.

4.4 Navigation-Grade IMU

In order to further evaluate the effects that the inertial sensor has on the solution, a simulated navigation-grade IMU is substituted for the tactical-grade INS in the nominal case. The state covariances are adjusted to the HG9900 specifications but all other covariances are unchanged. All other aspects of the nominal case apply to the commercial INS case. The trajectory resulting from this case is shown in Figure 31.

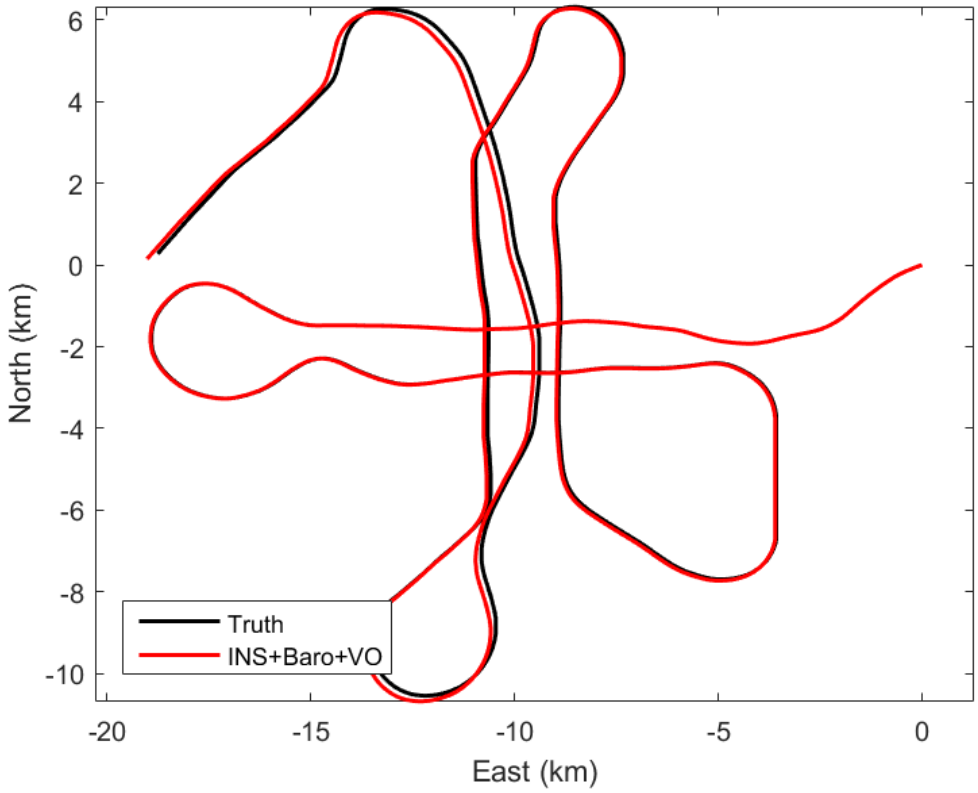


Figure 31. Navigation case trajectory

This figure is quantified in Table 14.

1

Table 14. 2DRMS of Position Error for Navigation and Tactical IMU's

	2DRMS
Tactical	646 m
Navigation	261 m
Reduction	60%

From this table we see that the horizontal position solution is greatly improved by upgrading the IMU.

The effects that the navigation-grade IMU has on the VO measurement accuracy is summarized in Table 15.

Table 15. RMS Error of Measurements of Navigation Case,

	North	East
Navigation	2.24 m/s	1.91 m/s
Tactical	2.29 m/s	2.37 m/s
Reduction	2%	19%

Switching to a navigation-grade IMU reduces the VO measurement error significantly. Since a higher grade IMU is able to more accurately measure the rotation necessary to rotate VO measurements into the correct from and the **R** matrix, this case results in more accurate measurements. The navigation-grade IMU was not used in other cases, however, because the tactical-grade IMU was proven to be sufficient.

4.5 Image Distortion

In order to verify the benefits of image rectification, the nominal case is compared to image distortion case, which processes raw (distorted) images instead of rectified images. The error of the velocity measurements resulting from processing each set of images using OF-VO is shown in Figure 32.

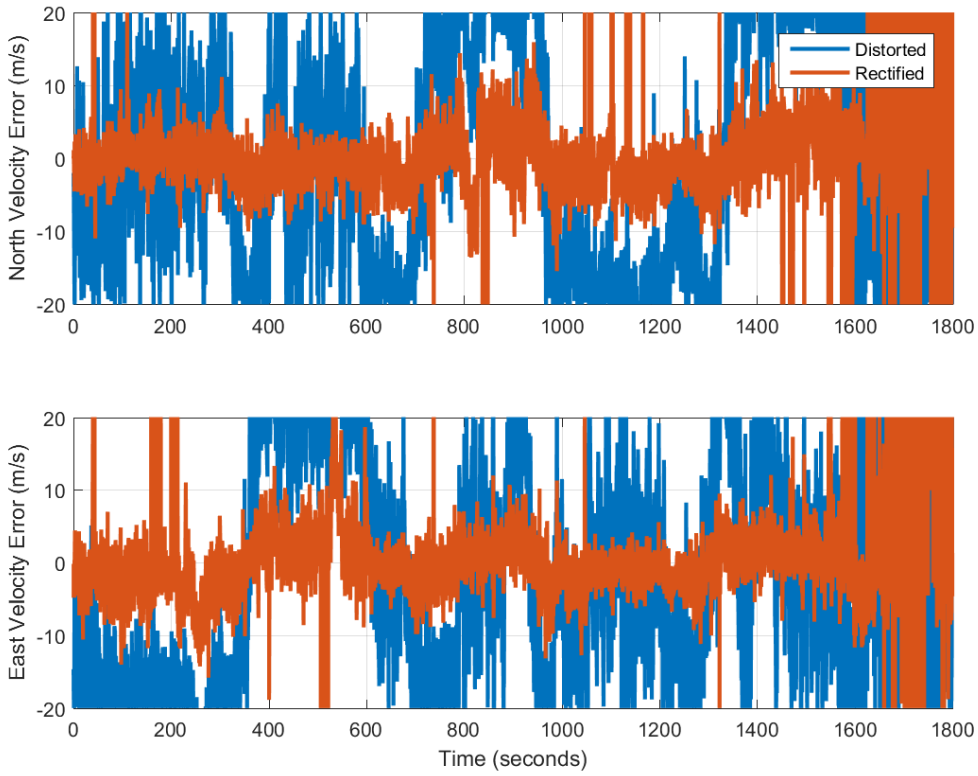


Figure 32. Velocity measurement error with and without image rectification

From the figure it can be observed that the image rectification makes a large difference in measurement accuracy. This is not the case for every algorithm though. The RMS of the measurement error for each algorithm is shown in Table 16.

Table 16. RMS of Measurement Error for Each Algorithm Before With and Without Image Distortion

	OF-VIO		OF-VO		Feature-VIO		Feature-VO		SVO	
	North	East	North	East	North	East	North	East	North	East
Distorted	2.14	1.99	11.5	19.1	2.14	2.12	14.2	16.3	8.33	8.95
Rectified	2.25	2.33	3.63	3.75	2.41	2.59	5.02	5.20	6.38	6.70
Change	-5%	-15%	80%	79%	-11%	-18%	65%	68%	23%	25%

Image rectification improved the measurements for the algorithms which solve for **R**, but hurt their counterparts. Overall a net gain is seen from undistorting the images, however.

4.6 Vertical-Only INS Reset

The vertical INS reset case is used to evaluate the effect that the full INS reset has in the nominal case. In the partial reset case, only the up position and down velocity solution states are fed back to the INS, as opposed to the nominal case which resets all nine solution states.

The RMS values of the measurement error for this case and the nominal case are shown in Table 17.

Table 17. RMS of Measurement Error from Full and Vertical INS Reset

	North	East
Full	2.29 m/s	2.37 m/s
Vertical	2.40 m/s	2.42 m/s
Reduction	5%	2%

Implementing a full INS reset over a partial INS reset reduced the measurement error by a small amount.

In order to further explore the differences of each type of reset, the filter states will be examined for each case. Figures 33 and 34 shows the first six raw filter error states, which estimate the error in the INS position and velocity solution.

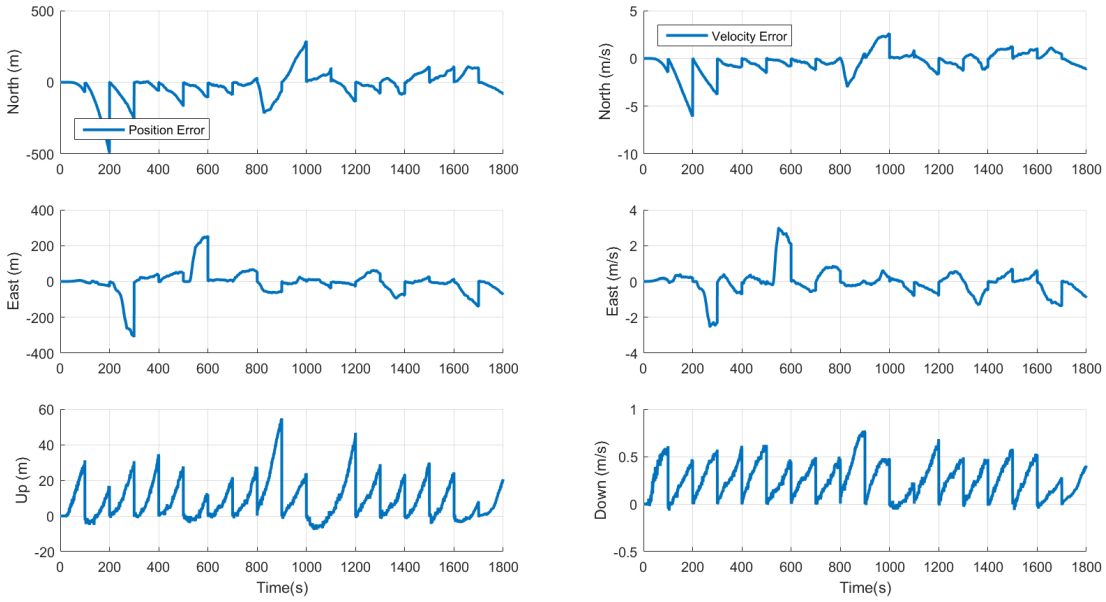


Figure 33. The Raw Filter Position and Velocity States, Full Reset

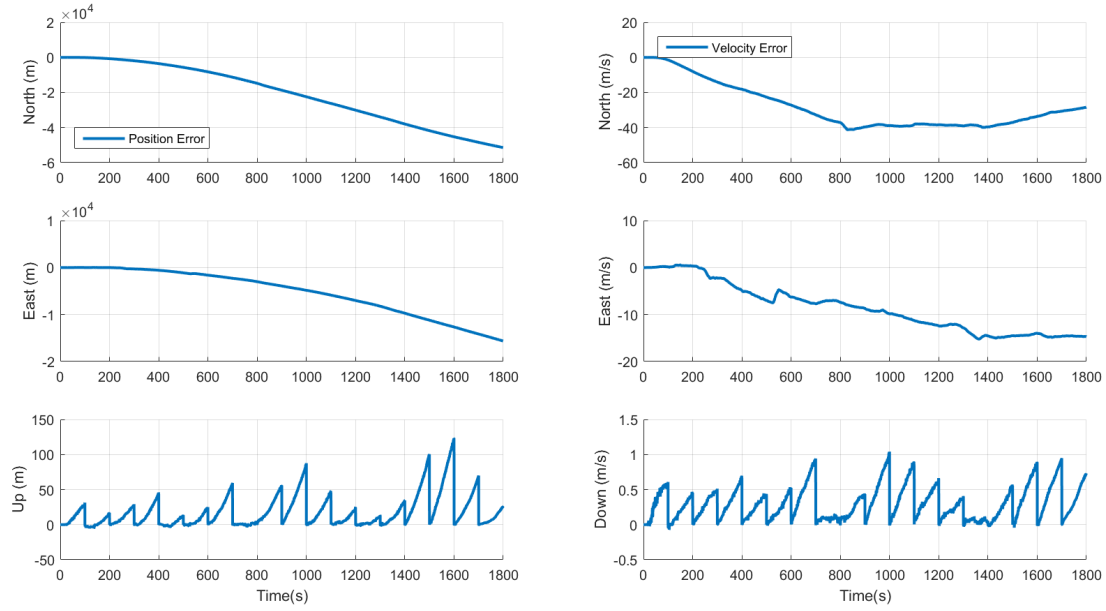


Figure 34. The Raw Filter Position and Velocity States, Partial Reset

From these figures it can been seen that states which are reset track smaller errors. When only a partial reset is performed, the error in the horizontal states grows into

the tens of kilometers, many factors higher than when all states are reset. This magnitude of error will cause the INS error models to become less accurate. It is also interesting that the vertical errors grow larger for that of the partial reset than that of the full reset, even though they are both reset at the same times.

Figures 35 and 36 shows the errors in the state estimates for both cases.

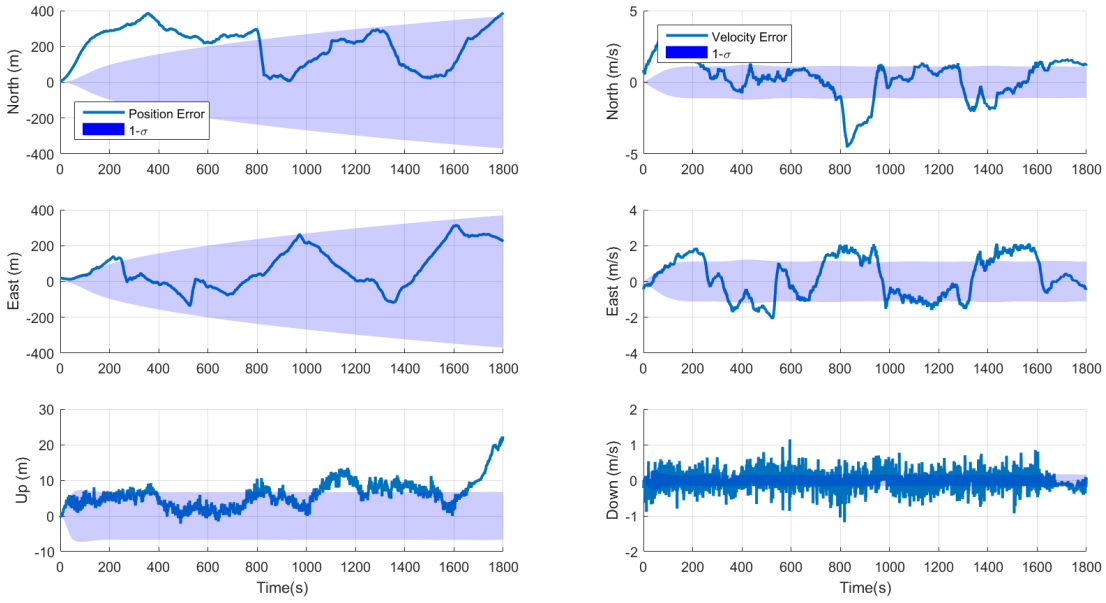


Figure 35. The Filter Error in the Position and Velocity States, Full Reset

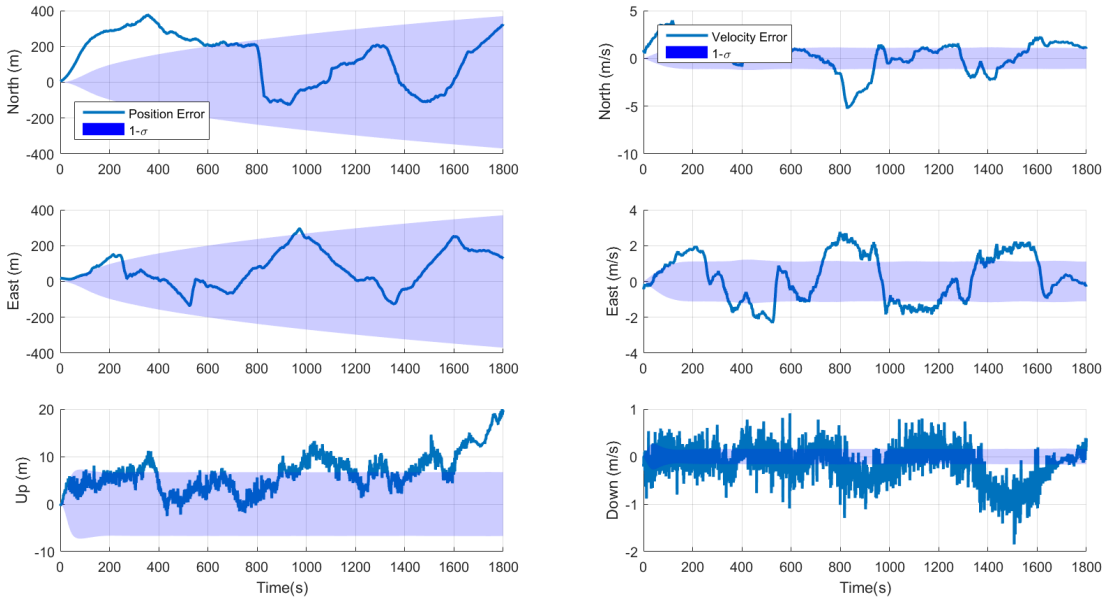


Figure 36. The Filter Error in the Position and Velocity States, Partial Reset

From these figures, it can be seen that while most states are visually identical, the North position error is lower for the partial reset compared to the full reset. This causes the horizontal position error to be closer to zero for the partial reset case. This plot also shows that the vertical solutions are degraded for the partial reset case.

The RMS of each error is shown in Table 18.

Table 18. RMS of Position Solution Error from Full and Vertical INS Reset

	East	North	Up
Full	266 m	189 m	7.1 m
Vertical	213 m	196 m	7.7 m
Reduction	-20%	4%	8%

From this table it can be seen that, although the East and Up positions are slightly improved by resetting every INS state, the horizontal position is actually significantly

degraded. It is possible that the extra error in the up position would have resulted in a smaller scaling factor for the VO measurements, which would have affected the horizontal position.

The effect that each case has on the overall position solution can be seen from Table 19.

Table 19. 2DRMS of Position Error from Full and Vertical INS Reset

	2DRMS	Decrease from nominal
Full	646 m	
Vertical	580 m	10%

Since the East and North position solutions are degraded, the horizontal position as a whole is degraded by resetting all INS states.

Table 20 quantifies the velocity errors.

Table 20. RMS of Velocity Solution Error from Full and Vertical INS Reset

	North	East	Down
Full	1.86 m/s	1.70 m/s	0.21 m/s
Vertical	2.05 m/s	1.80 m/s	0.38 m/s
Reduction	9%	6%	45%

This table demonstrates that the full reset significantly improves the velocity solution for every axis.

Figures 37 and 38 shows the tilt errors for each case.

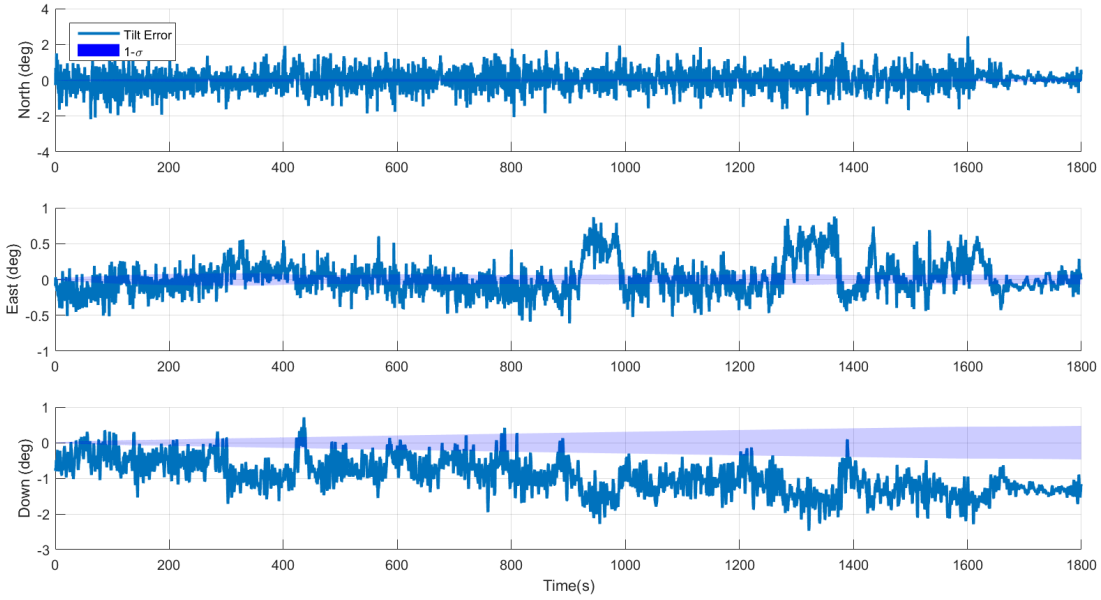


Figure 37. Tilt Errors, Full Reset

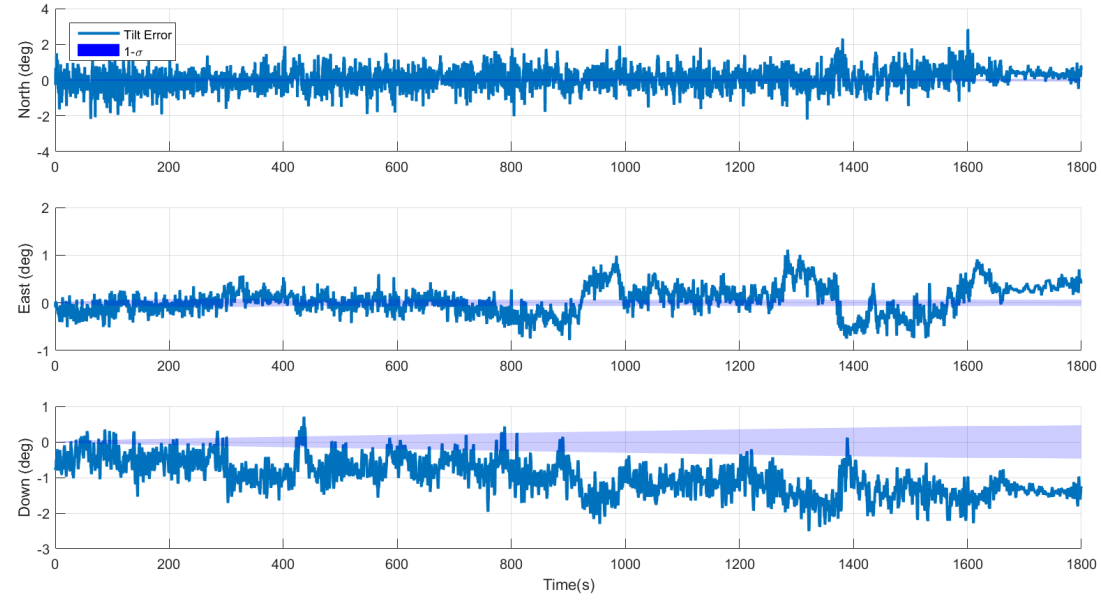


Figure 38. Tilt Errors, Partial Reset

From this figure it can be seen that resetting every INS state decreases the tilt error about the East axis.

Table 18 quantifies these errors.

Table 21. RMS of Attitude Solution Tilt Error from Full and Vertical INS Reset

	North	East	Down
Full	0.0085 °	0.0040 °	0.0181 °
Vertical	0.0091 °	0.0053 °	0.0188 °
Reduction	7%	25%	4%

Going from a partial INS reset to a full reset improved the tilt errors about every axis. It is possible that in this scenario the extra tilt errors actually resulting in a position closer to that of the truth position.

To sum up the differences between the partial reset case and the nominal, the 2DRMS of the error of the horizontal position was actually smaller for the partial reset than for the full reset even though every state except for the horizontal position states. This could have occurred because the partial reset degraded the attitude solution, which happened to counteract velocity errors, improving the position. An interaction could have also occurred from the vertical position error influencing the VO measurements. More data is necessary to further analyze this strange result. Either way, the full reset case produced more accurate VO measurements and filter states which is why the full reset is included in the nominal case.

4.7 Unscented Kalman Filter

In order to evaluate the effectiveness of the EKF in the nominal case, the results of a UKF were also calculated. The UKF case consists of the nominal case, substituting the EKF for a UKF. Table 22 shows the RMS of the measurement error that results from filtering the VO, barometer, and IMU measurements with an EKF and a UKF.

Table 22. RMS of Measurement Error from EKF and UKF

	North	East
EKF	2.24 m/s	2.32 m/s
UKF	2.24 m/s	2.32 m/s
Reduction	0%	0%

There is no significant difference in VO measurements between these two cases.

Table 23 shows the resulting 2DRMS of the position for nominal and UKF cases.

Table 23. 2DRMS of Position Error for UKF and EKF

	2DRMS
EKF	646 m
UKF	646 m
Reduction	0%

Again, the difference between these cases is insignificant. Since the measurement equations and state dynamics used for this experiment are linear, there is no difference in performance between an EKF and UKF. For this reason, the EKF is used in the nominal case.

4.8 Best-Use Cases

Given the previous analyses, it is possible to recommend a use case for each algorithm.

OF-VIO gives the best all-around performance in speed and accuracy. If INS data is available, then this algorithm will leverage the inertial data since it provides tight-

coupling with the inertial. This algorithm is recommended for use in feature-sparse imagery or when computation time is desired.

OF-VO gives the best accuracy when an IMU is not available. It outperformed its feature-based counterpart, although this may not be the case for another data set.

Although Feature-VIO is slower than OF-VIO, it still gives slightly more accurate results. This may be especially true in cases of large changes in the camera's attitude or lower quality imagery, such as in [41].

Feature-VO is best used when INS data is not available. Although in this experiment it was outperformed by OF-VO, Feature-VO may perform better than OF-VO on other imagery sets.

SVO performed the least accurately in this experiment. It is however, recommended for use when inertial data is not available and when height information is not continually available. This algorithm was also able to better estimate $\mathbf{R}$ than either OF-VO or Feature-VO. Last, in part because it is the only computationally optimized algorithm, SVO is the fastest out of every algorithm, making it a good choice if real-time performance is desired.

V. Conclusions

The first part of this chapter summarizes the research project. The second part makes recommendations for future work.

5.1 Summary

In a novel comparison, five top-down visual odometry algorithms were implemented into a common framework in order to develop recommended use-cases using real-world data. Velocity measurements from these algorithms were filtered with inertial and barometric data. The filter state error for the nominal case was examined in order to justify filter design. The RMS of the measurement error and 2DRMS of the horizontal position were evaluated for each algorithm, two grades of IMUs, distorted and rectified images, a full and vertical-only INS reset, and an EKF and UKF. Each algorithm's ability to estimate the rotation between camera frames was also evaluated.

After this analysis was completed, each algorithm was recommended for a best-use case. The VIO algorithms were recommended when inertial data is available. OF algorithms performed slightly more accurately and faster than the feature-based methods, although feature-based algorithms may perform better in less-ideal imagery. Last, SVO was found to be most useful when processing power is limited and when neither inertial nor altitude information is available.

5.2 Future Work

Although the effects of a commercial-grade IMU and tactical-grade IMU were analyzed, it would be interesting to see how substituting the tactical-grade with a navigation-grade IMU would affect the filter solution. It would also be useful to

experiment with different ways of updating the filter with VO measurements. For example, if the attitude solution is not accurate to rotate velocity measurements into the filter's frame then the VO could be used to generate speed measurements. This kind of measurement would require a non-linear measurement equation, which would also make the UKF and EKF comparison nontrivial. Velocity measurements could also be used to provide the INS with a zero-velocity update in two dimensions.

It would also be useful to see how a different data set affects the results. Since the full INS reset and partial INS result had mixed results, another data set would give more insight into how each reset affects the results. Another imagery set may help better distinguish feature-based and optical flow methods. A data set which had sparse height information would give SVO a better comparison to the other methods. A precise grid of terrain elevation would eliminate errors due to inaccurate height-above-ground calculations.

This project would be improved if more algorithms were implemented into the framework. Other types of machine vision navigation, such as SLAM, would show the performance gained by mapping and loop closure. It would be interesting to evaluate other feature detectors, feature descriptors, and descriptor trackers for the feature-based algorithms. Similarly, other kinds of direct visual odometry, other than optical flow, could be implemented for comparison.

Bibliography

1. Maxime Lhuillier. Automatic scene structure and camera motion using a catadioptric system. *Computer Vision and Image Understanding*, 109(2):186–203, 2008.

2. Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. An image-to-map loop closing method for monocular slam. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2053–2059. IEEE, 2008.

3. David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

4. Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

5. Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

6. Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

7. Kurt Konolige, Motilal Agrawal, and Joan Sola. Large-scale visual odometry for rough terrain. In *Robotics research*, pages 201–212. Springer, 2010.

8. Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

9. David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. Ieee, 2004.

10. Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

11. Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

12. Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

13. Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer, 2008.

14. Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, 2012.

15. Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

16. Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

17. Berthold KP Horn. Recovering baseline and orientation from essential matrix. *J. Opt. Soc. Am*, 110, 1990.

18. J Rohde. Real-time implementation of vision, inertial, and gps sensors to navigate in an urban environment. Technical report, Air Force Institute of Technology, 2015.

19. SY Chen. Kalman filter for robot vision: a survey. *IEEE Transactions on Industrial Electronics*, 59(11):4409–4420, 2012.

20. Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 486–492. IEEE, 2010.

21. Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

22. Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3946–3952. IEEE, 2008.

23. Henning Lategahn, Andreas Geiger, Bernd Kitt, and Christoph Stiller. Motion-without-structure: Real-time multipose optimization for accurate visual odometry. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 649–654. IEEE, 2012.

24. Annalisa Milella and Roland Siegwart. Stereo-based ego-motion estimation using pixel tracking and iterative closest point. In *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*, pages 21–21. IEEE, 2006.

25. Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *Computer Vision*

and Pattern Recognition, 2006 IEEE Computer Society Conference on, volume 1, pages 363–370. IEEE, 2006.

26. Eric Royer, Maxime Lhuillier, Michel Dhome, and Jean-Marc Lavest. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3):237–260, 2007.

27. Jason Campbell, Rahul Sukthankar, Illah Nourbakhsh, and Aroon Pahwa. A robust visual odometry and precipice detection system using consumer-grade monocular vision. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3421–3427. IEEE, 2005.

28. Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

29. Andrew I Comport, Ezio Malis, and Patrick Rives. Real-time quadrifocal visual odometry. *The International Journal of Robotics Research*, 29(2-3):245–266, 2010.

30. Roland Goecke, Akshay Asthana, Niklas Pettersson, and Lars Petersson. Visual vehicle egomotion estimation using the fourier-mellin transform. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 450–455. IEEE, 2007.

31. Akira Ishii, Atsushi Sakai, Masahito Mitsuhashi, and Yoji Kuroda. Correcting angle of visual odometry system by fusing monocular and stereo methods in untextured dynamic environment. In *Communications and Information Technologies (ISCIT), 2010 International Symposium on*, pages 950–955. IEEE, 2010.

32. Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *Robotics*

and Automation, 2009. ICRA'09. IEEE International Conference on, pages 4293–4299. Ieee, 2009.

33. Petri Tanskanen, Tobias Naegeli, Marc Pollefeys, and Otmar Hilliges. Semi-direct ekf-based monocular visual-inertial odometry. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 6073–6078. IEEE, 2015.

34. Jean-Philippe Tardif, Michael George, Michel Laverne, Alonzo Kelly, and Anthony Stentz. A new approach to vision-aided inertial navigation. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4161–4168. IEEE, 2010.

35. Peter S Maybeck. *Stochastic models, estimation, and control*, volume 3. Academic press, 1982.

36. G. Bradski. *Dr. Dobb's Journal of Software Tools*.

37. Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106–1112. IEEE, 1997.

38. David Titterton and John L Weston. *Strapdown inertial navigation technology*, volume 17. IET, 2004.

39. Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *European Conference on Computer Vision*, pages 214–227. Springer, 2012.

40. Ming Zhao, Jian Zhang, and Renato Figueiredo. Distributed file system support for virtual machines in grid computing. In *High performance Distributed Comput-*

ing, 2004. Proceedings. 13th IEEE International Symposium on, pages 202–211. IEEE, 2004.

41. Benjamin Fain. Small fixed-wing aerial positioning using intra-vehicle ranging combined with visual odometry. Master's thesis, 2017.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
23–03–2017	Master's Thesis	Sept 2015 — Mar 2017

4. TITLE AND SUBTITLE

Aerial Visual-Inertial Odometry Performance Evaluation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Carson, Daniel, J.

5d. PROJECT NUMBER

16G758A, 17G758A

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-17-M-010

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Lockheed Martin Corporation, USAF CRADA No. 16-AFIT-02
1801 State Route 17C, Maildrop 0210
Owego, NY 13827
(607) 751-5383
Email: scott.sorber@lmco.com

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

THIS MATERIAL IS DECLARED A WORK OF THE U.S. GOVERNMENT AND IS NOT SUBJECT TO COPYRIGHT PROTECTION IN THE UNITED STATES.

14. ABSTRACT

With the cheapening of optical camera technology and an increasing interest in reducing GPS dependency, computer vision-based navigation algorithms have grown in popularity. This experiment implements five visual odometry algorithms into a common framework, evaluating their accuracy in a variety of situations on an unprecedented level. A variety of techniques are compared and contrasted including FAST-features versus gridded pixel-features, inertial-aided and inertial-independent rotation estimation, the effectiveness of image histogram equalization, the benefits of image rectification, and two-frame versus multi-frame visual odometry.

15. SUBJECT TERMS

Visual Odometry, Kalman Filter, Sensor Fusion, Vision Navigation, Optical Flow

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. John F. Raquet, AFIT/ENG
U	U	U	U	95	19b. TELEPHONE NUMBER *(include area code)* 937-255-3636, x4580; john.raquet@afit.edu