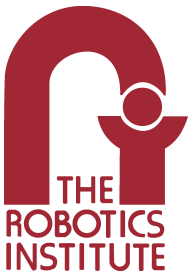
Long-range GPS-denied Aerial Inertial Navigation

Garrett Hemann

May 2016



Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

Technical Report CMU-RI-TR-16-11

Thesis Committee

Prof. Michael Kaess

Prof. George Kantor

Prof. Sanjiv Singh

Humphrey Hu

Contents

Abstract

Despite significant progress in GPS-denied autonomous flight, long-distance traversals (over 100 km) in the absence of GPS remain elusive. This work focuses on techniques that efficiently capture the full state dynamics of the air vehicle with semi-intermittent global corrections using LIDAR measurements matched against an a priori Digital Elevation Model (DEM). Using an error-state Kalman filter with IMU bias estimation, we are able to maintain a high-certainty state estimate, reducing the computation time to search over a global elevation map. A sub region of the DEM is scanned with the latest LIDAR projection providing a correlation map of landscape symmetry. The optimal position is extracted from the correlation map to produce a position correction that is applied to the state estimate in the filter. This method provides a GPS-denied state estimate for long-range drift-free navigation. We demonstrate this method on multiple data sets from a full-sized helicopter, showing significantly longer flight distances over the current state of the art.

1. Introduction

1.1. Motivation

Autonomous and manned aerial vehicles rely on accurate localization for safe navigation and for finding their destination. Relying on only inertial sensors for localization is not feasible because navigation drift accumulates over time without bounds. Typically, aerial vehicles use GPS to localize their positions with respect to the Earth. While this is a simple solution to integrate, GPS is not always a reliable sensing mechanism. For one, satellite coverage breaks down from obstructions and multipath caused by mountains or skyscrapers. Also, GPS is susceptible to adversarial jamming and spoofing, rendering it useless or dangerous for navigation. Additionally, the satellite system is not immune against system errors, such as the January 2016 incident[1] where wrong time information was temporarily broadcast after the decommissioning of a GPS satellite. Lastly, GPS only provides a navigation solution on Earth and extra-terrestrial navigation requires a different sensing strategy.

[1] http://www.insidegnss.com/node/4829

1.2. Problem Statement

GPS-denied navigation has gained much attention within the past decade, particularly in the indoor and underwater domains. The problem is challenging because in the absence of a global reference such as GPS, onboard sensors can only produce a drifting navigation solution with unbounded error. Strategies for remaining localized with onboard sensors involve using a priori maps (e.g. [12]) as well as using SLAM to build maps on the fly (e.g. [9]). Development in perception and efficient mapping algorithms have merged to form stable visual-based localization methods using feature-rich environments for GPS-denied navigation. These methods require repetitive landmark observations for loop-closure to eliminate longer term drift errors. This dependency, along with the payload limitations of small indoor air vehicles, limits the range with which localization algorithms can be demonstrated.

One area that has received less attention is extending the GPS-denied navigation capabilities to long distance outdoor missions (see fig. 1.1). This scenario presents two major challenges: (1) capture the dynamics of the vehicle at a high rate and (2) estimate its 6DOF global position and orientation in real-time. The first challenge has been addressed for air vehicles in shorter range missions using Kalman filters or smoothing techniques (e.g. [11]). The second challenge however has not been addressed for long distances without GPS, and therefore this work explores this challenge.

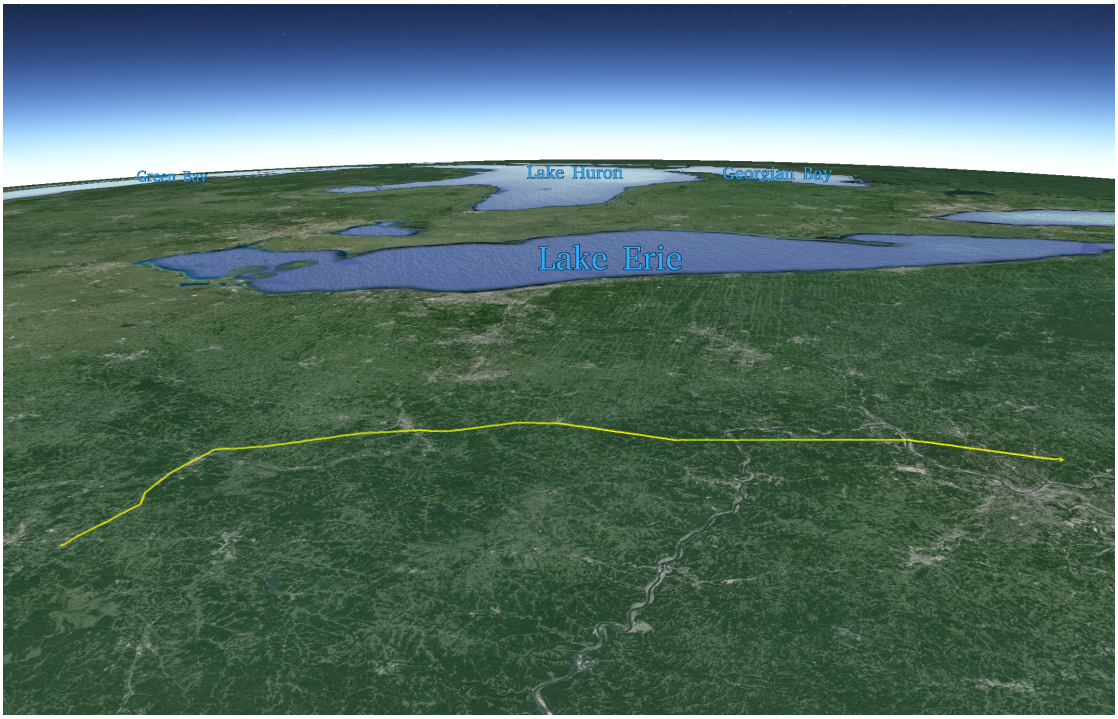


Figure 1.1.: One result of the work presented here; a GPS-denied 218 km helicopter flight (yellow trajectory), with a final position error of 37 m, 0.017% of the distance traveled. The helicopter took off from Zanesville airport, OH, and landed at Cedar Run airport, PA.

1.3. Technical Contributions

To achieve drift-free localization in the absence of GPS, we match LIDAR measurements against a Digital Elevation Model (DEM) for localization. Our technique fits into the category of Terrain Referenced Navigation (TRN) [24] in which terrain models are used for localization. The a priori DEM provides a surface elevation map at various resolutions for the entire desired flight plan. Instead of trying to scan a single LIDAR projection against the entire geographical region or world, we use our inertial propagation estimate to narrow the search space, reduce computational cost, and provide real-time global pose estimation. The low-drift estimate also accounts for periods of inactive corrections which may occur when flying at

low altitudes, over water where LIDAR returns fail, or in areas with low terrain variability.

This work makes the following contributions:

1. We present a LIDAR localization algorithm that eliminates the need for cameras and therefore works independent of lighting conditions.

2. We provide a tightly-coupled LIDAR-inertial integration that achieves low, bounded position and orientation error using intermittent position corrections in the absence of GPS.

These contributions demonstrate our ability for long distance drift-free navigation on two datasets from the flight of a full-sized helicopter, each covering around 200 kilometers from takeoff to landing, a significant increase in distance over the current state of the art.

This work has also contributed to a submitted paper under review [7].

1.4. Notation

The rest of the paper follows the following math notation conventions:

- A scalar value is a non-bold, lower case letter or symbol, p.
- A vector of values is a bold, lower case letter or symbol, $\mathbf{p}$.
- A matrix of values is a bold, upper case letter or symbol, $\mathbf{P}$.
- Estimated (not-directly measured or known) values are decorated with a hat, $\hat{p}$.

- Error values are decorated with a tilde, $\tilde{p}$.
- A capital superscript before a variable represents the frame of the value or vector, ${}^{G}p$.

2. Related Work

2.1. History of Long-range GPS-denied Solutions

The earliest known techniques in long-range navigation were for military missile tracking and guidance. Cruise missiles used a combination of sensors to navigate at high speeds over long distances. Given the lower computation power in the 1970's and 1980's, this technology had limited accuracy. With the creation of the GPS satellite network, intermittent coarse corrections via GPS could be used to improve the accuracy.

As computer memory and processing power increased, the focus shifted away from long-range navigation to indoor navigation. This shift towards indoor GPS-denied navigation therefore has focused primarily on smaller Unmanned Air Vehicles (UAVs) due to the ubiquity of cheap, easily available platforms like quadrotors. Smaller platforms offer limited payload and limit the range to which long distance autonomy can be tested. Without the emphasis on large environments, most recent research in GPS-denied autonomy have used and expanded the techniques of visual odometry (VO) which can provide reasonably accurate results in short-range flight, but drift over time.

2.1.1. Terrain Referenced Navigation

The earliest research in long-range GPS-denied navigation is Terrain Referenced Navigation (TRN, but also sometimes referred to as terrain aided navigation or terrain relative navigation), a technique that compares different measurements of "broken terrain" to a previously recorded database of terrain (concept patented [1] in 1958)[1]. If the stretch of terrain covers enough area and is "broken" (not flat), then two separate measurements would always be unique. Not until the 1960's and 1970's did computational power improve enough to validate this concept. The U.S. military started using this strategy for long-range cruise missiles such as the Tomahawk, and in the 1970's the first well-developed system TERCOM ([6][22]) was introduced.

TRN for shorter segment flights has recently been motivated by planetary landing, where GPS is not available. [13] provides a comprehensive analysis of TRN and a comparison of different TRN-based landing techniques. Some strategies use a pattern matching approach where cameras detect ground landmarks and match to satellite imagery. This approach is viable in static environments like the Moon or Mars, but for our Earth-based application, landmarks vary too frequently. The other major approach is position correcting with active range sensing that uses terrain structure. [13] shows results in simulation of an altimeter-based approach, but an actual 300 second landing sequence was tested in [3] using a laser scanner. The LIDAR points are converted to elevation estimates and aligned with the DEM using a sum of squared error minimum. While the landing drift error was considerably low (less than a meter in any direction), the system incorporated GPS and

[1] Historically, we only focus on digital systems, but there are some earlier analog TRN systems that are captured well in section 2.3.1 of [31].

radar altimeters and only used the laser when the data was reliable.

Another TRN-based method [12] evaluates the horizontal position accuracy of a lunar lander approach by converting a LIDAR scan to an elevation model. They perform the matching using the Fast Template matching algorithm [16] and finding the correction from the shift of the maximum value in the correlation map. We instead use a normalized cross-correlation matching method which does not rely on precomputed integral tables. This TRN approach was tested on Earth terrain models using $3\,m$ DEM and their landing trajectory guaranteed $90\,m$ accuracy. Their system however does not have a tightly coupled navigation solution. We extend this work to incorporate a high-rate fused state estimate, global corrections in 3 axes instead of 2, and operate over long distance flights as opposed to just landing sequences.

A slightly different approach that uses TRN creates pre-generated flight trajectory profiles and a binary search tree lookup method to find the most likely profile in real time [4]. This system is unique in that it is not an inertial navigation system (INS) meaning it has no dependency on an IMU. Because of this however, it requires pre-processing the estimated flight profiles and does not scale well for large, realistic systems.

2.1.2. Radar Odometry

Another type of ground localization is radar odometry, which uses artificial ground scatters to reflect signal from an onboard radar. Though not as common of a strategy, one in particular is [20]. This process uses a Hough transform on a radar signal to identify the targets and their relative distance. The flight test results

show that using a commercial-grade IMU is comparable in drift error to using a navigation-grade IMU. Their flight was on a 2.4 kilometer dataset and had a final drift accuracy of 2.3% over the distance flown. The strength of using radar in this method gives the system the ability to operate in more varied weather conditions than a LIDAR, however it depends on artificial radar scatterers to be placed in the environment, which are not readily available.

2.1.3. Long-range Visual Odometry

Visual odometry is a process that tracks feature changes across sequences of images to compute the trajectory of the camera. Without a global correction, visual odometry is prone to drift. Regardless, with the increasing interest in UAVs in the past decade, the state of the art in visual odometry has considerably reduced this drift, increasing the distance of feasible navigation. As a result, some visual odometry techniques for indoor GPS-denied autonomy have been extended to attempt longer range missions.

[33] discusses the state of the art for minimal payload aerial vehicles using a monocular-camera and IMU sensor setup. Their PTAM-based visual SLAM approach was demonstrated on a 350 meter flight with a final position error of only 1.47 meters, at which point the battery was depleted. Another long-range visual odometry technique by [32] uses a deformable stereo-rig baseline to account for the vibrations of the vehicle and improve depth accuracy. This was tested on a fixed-wing UAV for a 6.5 kilometer dataset. Currently the longest VO demonstration we know of is [35]. This work reduces translational drift of an inertial navigation solution by reparametrizing features of a downward-facing camera along a ground

plane normal, extracted from a laser altimeter. They tested this on trajectories over 30 km with 0.09% trajectory error. This impressive performance over the distance traveled is still unbounded in drift however. Vision-based methods like VO can improve on inertial navigation solutions, but do not work for significant distances where target position error is critical.

2.2. Sensor Fusion

The global correction method is just one half of the state estimation problem. Highly dynamic platforms like UAVs also require a continuous high-rate state estimate, typically at a higher rate than what the absolute position corrector provides. Most modern multi-sensor robotic platforms rely on efficient sensor fusion algorithms to capture a consistent state estimate from various input sensor rates. A common method is the Kalman filter which handles propagating uncertainty and providing a fused state estimate of an inertial navigation system. A modified version of the Kalman filter, or error-state Kalman filter (see sec. 4.2), has been used extensively by [30] for state estimation in planetary landing. The state is propagated with IMU input and the Kalman filter estimates the IMU biases. They use a camera to track features from craters and correct the state estimate.

The Kalman filter and IMU bias estimation has improved the performance of some of the previously stated GPS-denied algorithms. Most notable improvements have been in an area of work called visual inertial odometry (VIO). VIO uses an IMU along with visual odometry feature tracking and provides a tightly coupled state estimate. [17] introduces the multi-state constraint Kalman filter (MSCKF) which extends the error-state Kalman filter from [30] to include camera position

history. Using a fixed length history, the camera position estimates help track individual features over each sequence. These tracks are maintained in the state estimate and propagated, providing a fused visual inertial estimate, IMU biases, and a measurement for uncertainty. This reduces the long-term drift in a VO only solution. With the addition of inertial data, it also improves results when the camera moves in the direction of the optical axis, which degrades in a visual-only algorithm.

Another common method for localization is the particle filter. The particle filter localizes its state by sampling points throughout an a priori map and eliminating low-likelihood points to refine the estimate of the true position. [28] details the advantages of using particle filters over the Kalman filter for SLAM applications, which include the Kalman filter's single state estimate versus an entire path estimate, as well as its lack of a data association solution. In our long-range application, tracking the entire history is expensive and unnecessary. Data association is also unnecessary since we are only localizing and any noise is filtered out from our binning procedure (see sec. 3.2). One advantage of the particle filter is it does not rely on the Gaussian noise assumption, but since our IMU noise model has this property, the Kalman filter is more advantageous.

Previous approaches to the long-distance GPS-denied state estimation challenge have used various types of sensing modalities and filter techniques to reduce track error and total trajectory error. However, the limitations in the operational environment or platform have prevented testing the long distance robustness of these algorithms. Furthermore, most of the techniques have only utilized either a robust filter for IMU bias estimation or an efficient terrain matching algorithm. Here we present a fusion of several of these techniques and improve them individually for

considerable improvement in terrain matching and pushing the state of the art in long-distance GPS-denied navigation.

3. LIDAR Localization

LIDAR localization intermittently localizes the vehicle by aligning LIDAR measurements to a geo-referenced elevation model (see sec. 3.1). The goal is to estimate the vehicle's true 3D world position by searching for an optimal alignment between a 3D LIDAR point cloud as measured by the vehicle and the a priori DEM. We make use of the current position estimate to restrict the search area to a local DEM neighborhood. This requires a low-drift estimate, which is achieved by tightly coupled LIDAR-inertial fusion as discussed in sec. 4.2.

The 3D localization problem is condensed into a 2D translation and a 1D altitude problem that are solved sequentially. The DEM is given as a regular grid of elevation values, represented as a floating-point valued image $\mathbf{I}_D$. The point cloud from the vehicle's LIDAR is converted into an image $\mathbf{I}_L$ of the same grid size by binning, allowing for noise filtering in the process. The localization problem is thereby reduced to finding the 2D offset between both images that provides the best correlation between DEM and LIDAR data. Finally, to obtain a full 3D position correction, the difference in elevation between predicted and measured ground surface is estimated.

3.1. Digital Elevation Models

For robotic localization problems in large-scale environments, the ground surface needs to be modeled appropriately. Generally for small environment applications (and especially for indoor applications), vehicle altitude is represented as distance to a flat surface or ground plane. In large-scale environments however, the ground plane is no longer flat and therefore no longer a viable reference frame. Therefore, a new frame of reference is required and the terrain must be represented in the frame.

Digital elevation models accomplish this by representing the surface of terrain in 3D against some reference frame. For Earth-based surface models, a common reference frame is the WGS84 spheroid. DEMs are raster height maps where each pixel represents the distance of the terrain surface to the reference spheroid at some (x, y) position. The term "terrain surface" is used loosely here as DEMs can represent one of two different types of models, a Digital Terrain Model (DTM) where the top layer of ground is modeled, or a Digital Surface Model (DSM) where the highest point of any obstruction (ground, tree, man-made building) is modeled (see fig. 3.1). These models are generated by various means of LIDAR scanning and surveying techniques over the last 20 years and are provided in most areas of the United States. The DEM we use provided by USGS[1] is a DTM.

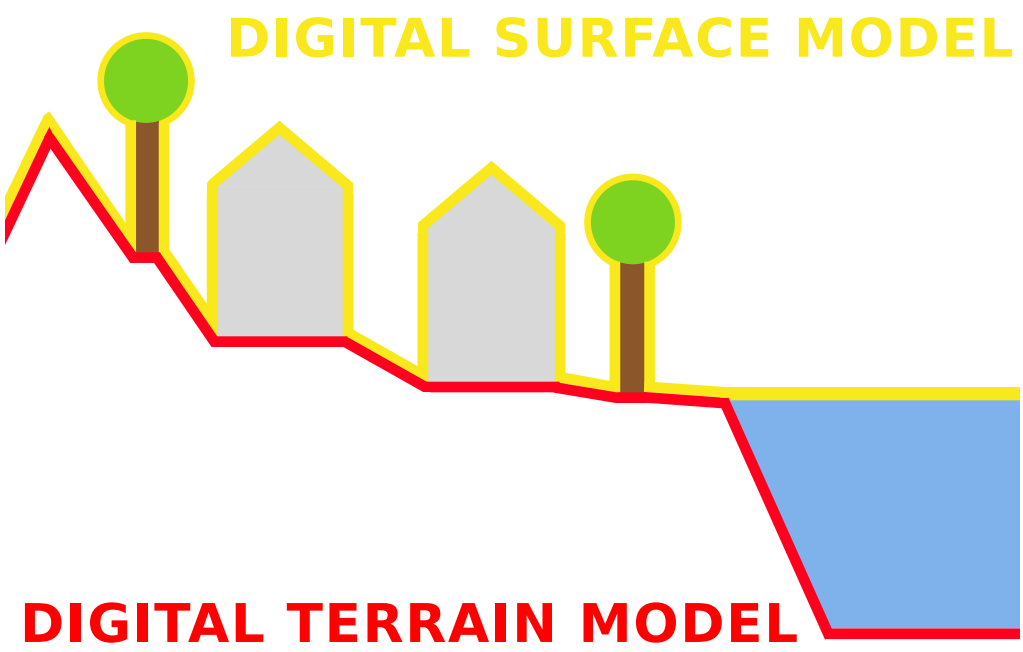


Figure 3.1.: DEM can be represented as a surface model (DSM, yellow) that follows the surface obstructions or as a terrain model (DTM, red) that follows the soil line.

3.2. Algorithm

3.2.1. LIDAR Binning

LIDAR binning takes the original LIDAR measurements and transforms them into a virtual DEM. The 3D-point measurements are obtained from a downward-facing 2D line LIDAR sensor rigidly attached to the vehicle. The individual timestamped measurements of each scan ${}^{L}\mathbf{p}_i$ are transformed from the laser frame into the global frame using a time stamped state estimate of the body-to-world transform ${}^{G}_{V}\hat{\mathbf{R}}(t)$ and a rigid transform of the sensor relative to the vehicle ${}^{V}_{L}\mathbf{R}$, using the

[1] `http://nationalmap.gov/elevation.html`

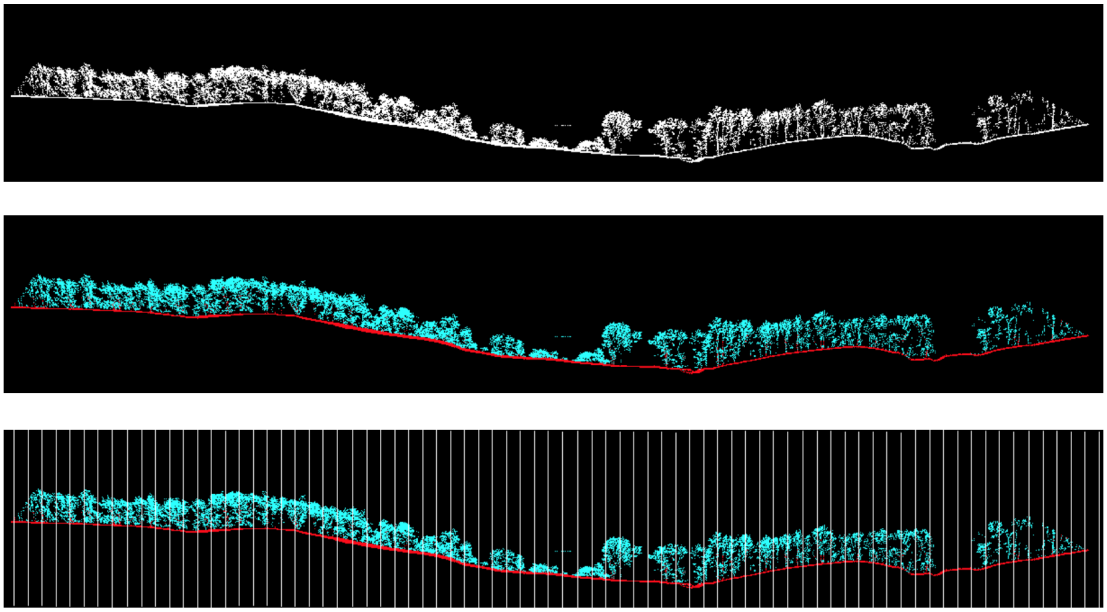


Figure 3.2.: A conceptual example of the binning process. The top row shows a raw cross section of the 3D LIDAR point cloud. The second row depicts the true ground profile (red) versus the obstructions (cyan). The bottom row demonstrates the binning procedure. All points except the lowest will be discarded per bin to extract the ground profile.

transformation chain

$$ {}^{G}\hat{\mathbf{p}}_i = {}^{G}_{V}\hat{\mathbf{R}}(t) \cdot {}^{V}_{L}\mathbf{R} \cdot {}^{L}\mathbf{p}_i, \tag{3.1} $$

where G denotes the global frame, V the vehicle frame and L the LIDAR frame. To cover a region of sufficient size for reliable offset estimation, a sufficient number of line scans are accumulated, depending on the vehicle's velocity.

A robust heightmap is computed by grouping the 3D points into bins along the x-y plane at the exact resolution of the DEM (fig. 3.2). These bins contain an array of estimated ground elevation values predicted from the LIDAR returns. Bins with a small range show an area with high certainty that the LIDAR beams reflected off

of the true ground surface. Bins with high variability on the other hand indicate occlusions such as trees, with only a few beams reflected off the desired ground surface. fig. 3.3 shows a cross-sectioned example of point bins that have as much as $30\,m$ in range variation due to foliage. The lowest elevation per bin (marked red) is the closest representation of the true ground elevation for that cell. To improve the likelihood that each bin has at least one return from the ground, we discard bins with less than 30 points. The result is a robust 2D heightmap $\mathbf{I}_L$ based on the lowest valued point of each valid bin.

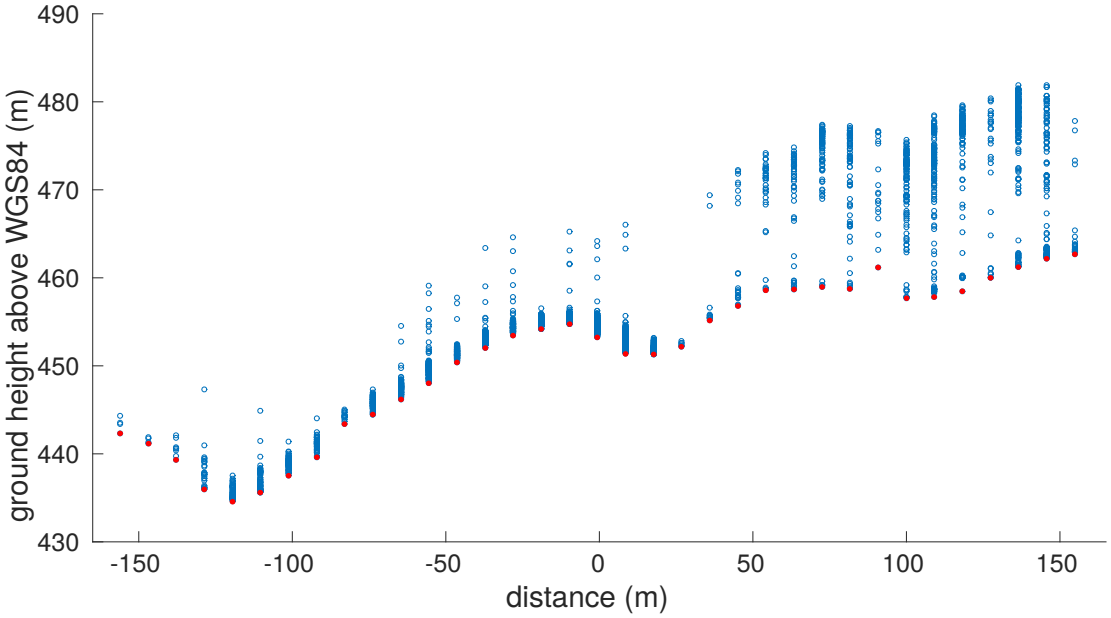


Figure 3.3.: A cross section of 3D LIDAR points (blue circles) after binning. The right half of the bins show high variability in height due to the presence of trees. The lowest point in each bin (marked in red) shows the smooth surface of the actual terrain. This is performed for every row of the x-y grid.

Removing invalid bins creates holes in the image. A binary mask $\mathbf{I}_v$ is used to track the active valid cells. Active regions are smoothed with a Gaussian kernel. This produces $\mathbf{I}_L$ to be a smooth surface-like heightmap generated from the 3D

LIDAR point cloud. Examples are shown in column (c) of fig. 3.4.

3.2.2. LIDAR DEM Matching

With both measured and ground truth heightmap available, the next step is to identify an offset based on the best alignment. The DEM image $\mathbf{I}_D$ is created from a local neighborhood window around the current state estimate. To account for the uncertainty of the state estimate, a region larger than the LIDAR image $\mathbf{I}_L$ is searched. An exhausive search is performed over all offsets between $\mathbf{I}_L$ along $\mathbf{I}_D$ with the goal of finding the offset with the best match. This is now a common image matching problem and there are various types of cost aggregation methods available to solve this (see [19]), such as Sum of Absolute Difference (SAD), Sum of Squared difference (SSD) and Normalized Cross-Correlation (NCC). We use the normalized cross-correlation matching because it is invariant to linear brightness shift meaning that if every pixel in one image were scaled by the same amount in another image, the matching would still succeed. For our purposes, brightness corresponds to height values. For the first step where matching is done only laterally, this invariance in altitude is important.

We calculate a normalized cross-correlation value

$$\mathbf{NCC}_{(i,j)} = \frac{1}{N(\mathbf{I}_v)} \frac{\sum(\mathbf{I}_{D,(i,j)} - \mu_{D,(i,j)})(\mathbf{I}_L - \mu_L)}{\sigma_D \sigma_L} \tag{3.2}$$

at each offset (i, j) to obtain a cross-correlation map (see column (d) of fig. 3.4 for examples). Note that the height images are multiplied element-wise. Also note that the summation is over all valid pixels of the image, where $N(\mathbf{I}_v)$ is the number of valid pixels in image mask $\mathbf{I}_v$.

Given this correlation map, we can now extract the position offset between the prior state estimate and the true position. The maximum value of this normalized cross-correlation cost map represents the highest correlation match between the LIDAR image $\mathbf{I}_L$ and a sub region of the DEM $\mathbf{I}_D$ at index (i^*,j^*). The shift from the center of $\mathbf{I}_D$ to (i^*,j^*) represents the horizontal correction of the vehicle. If max(**NCC**) is greater than an empirically determined threshold (we use 0.922), then it is a confident match and used to correct the position. Otherwise we treat it as a failed match and no correction is sent to the filter. Low confidence matches below this threshold are caused by dissimilar terrain features or low contrast areas (see fig. 3.4). This metric does not directly provide a confidence in the correctness of the match and is therefore not used as a noise model when updating the filter (see sec. 4.2.3).

For robustness, we also require some level of variability in the terrain before we accept a match. The matching algorithm only matches similarities in the images and ignores the actual shape and variation of the terrain. To ensure the terrain provides sufficient constraints, the standard deviation of all valid elevation values in the LIDAR image is computed. A successful match is returned only if this standard deviation is above a threshold (we use 2.5).

Two examples of LIDAR-DEM matches are shown in fig. 3.4. The top row shows a successful match with terrain variability in multiple directions, giving a single optimal location for the matching. The bottom row shows a failed match caused by nearly flat terrain within the matching region, seen by the low contrast of $\mathbf{I}_L$ and $\mathbf{I}_D$, and due to a river running through the center, leading to missing data in $\mathbf{I}_L$.

3.2.3. Altitude Correction

The final step is to compute a vertical correction (z) from the elevation images. The DEM image at the optimal index of the lateral matching $\mathbf{I}_{D,(i^*,j^*)}$ is subtracted element-wise from the measured heightmap $\mathbf{I}_L$. The median of the resulting difference image is the average elevation offset between the estimated and the true altitude. The offset represents the state estimation error as measured by the LIDAR against the ground truth DEM.

3.3. LIDAR Localization Failure Modes

The matching algorithm is susceptible to failures in specific environments. Dense urban environments fully occlude the laser returns from capturing the true ground plane providing little to no similarities between the LIDAR image and the DEM sub region. Also areas of thick vegetation (heavy forests or jungles) prevent the laser returns from penetrating the canopy and capturing the ground plane. Most long-range LIDAR sensors return invalid data when bouncing off water, so flying above a river, lake, or other large body of water causes matching failures. Lastly, large areas of flat terrain create a uniform cost map with no optimal location for correction, and so any attempted matches are discarded to avoid a degenerate solution.

A robust long-range navigation solution should account for these failure modes. We address this using a tightly-coupled inertial navigation solution that can dead reckon for periods of failed LIDAR localization matches. This method is discussed in chapter 4.

3.4. Comparison to ICP

Other algorithms, such as Iterative Closest Point (ICP) and its variants [25], can also solve the terrain matching problem. ICP minimizes the distance between two point clouds by fixing one point cloud and iteratively reducing a per-point distance measure to a source point cloud. Although this technique solves many localization problems, we chose the image matching algorithm over ICP for several reasons. The DEM's grid provides bins through which filtering can be used to isolate the ground plane. For ICP, filtering would run post-matching or in parallel, therefore processing more noisy points making the matching more expensive and less robust. Also, the alignment of image matching first in 2D reduces the complexity and later solving for altitude becomes trivial. ICP would require a full 3DOF iterative matching process and therefore is more computationally expensive.

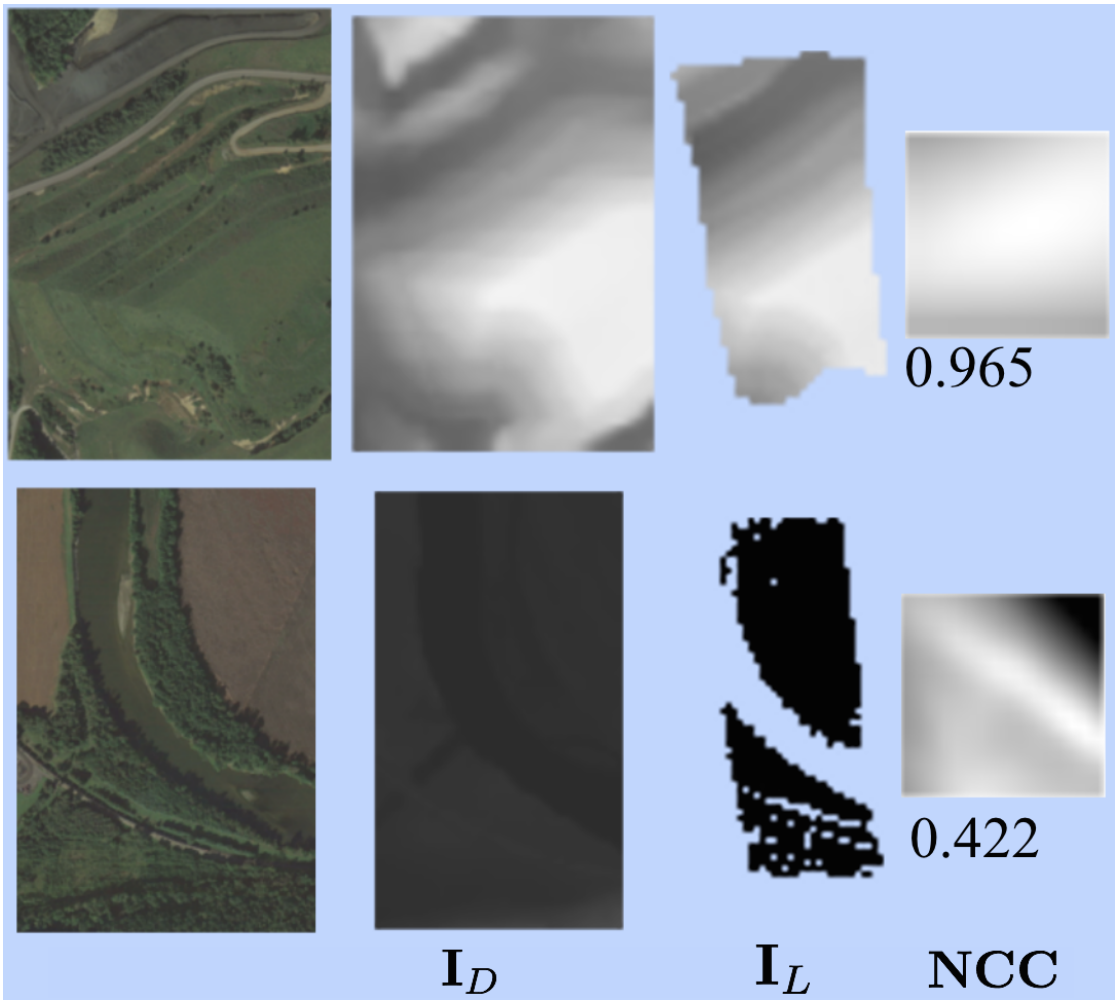


Figure 3.4.: Diagram of a successful match (top row) and an unsuccessful match (bottom row). The first column shows an aerial image of the terrain, the second shows a DEM region $\mathbf{I}_D$ of the same area at 10 m resolution, the third column shows a LIDAR projection $\mathbf{I}_L$ after binning and filtering, and the last column shows the normalized cross-correlation cost map of the matching (white denotes higher correlation) with the optimal value. The successful match (top) has an **NCC** image with a distinct optimum near its center, whereas the failed match (bottom) shows no clear optimum because of the low variation in terrain elevation.

4. State Estimation using Filtering

4.1. The Kalman Filter

Since its creation in the early 1960's, the Kalman filter has been used extensively in controls and state estimation. The low-cost incremental design of the filter and its ability to produce pseudo-optimal state estimates has made it a key component of many modern perception and mechanical systems.

Every type of Kalman filter is comprised of two steps; a prediction step that takes a measurement or control input to estimate its state, and an update step that uses a separate measurement modality to correct the previous prediction. Because of this simplicity, all Kalman filters can be derived from a model of basic stochastic equations

$$\begin{aligned} \mathbf{x}_k &= \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \\ \mathbf{z}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \end{aligned} \tag{4.1}$$

written in linearized discrete-time. $\mathbf{x}_k$ represents a state-vector at timestep k, $\mathbf{F}_{\mathrm{k}}$ is the discrete-time state transition matrix that propagates the state from time $k-1$ to k, $\mathbf{B}_k$ is the discrete time control-input model, $\mathbf{u}_{k-1}$is the deterministic

control input to the system, $\mathbf{z}_k$ is the observed measurement vector of the state $\mathbf{x}_k$, and $\mathbf{H}_k$ is the measurement model. Equation 4.1 is stochastic due to the process (or system) noise $\mathbf{w}_k$ and measurement noise $\mathbf{v}_k$, both assumed to be zero-mean, Gaussian white noise, and uncorrelated to each other, and are written as

$$\begin{aligned} \mathbf{w}_k &\sim \mathcal{N}(0, \mathbf{Q}_k) \\ \mathbf{v}_k &\sim \mathcal{N}(0, \mathbf{R}_k). \end{aligned} \tag{4.2}$$

$\mathbf{Q}_k$ is the process (system propagation) noise covariance matrix and $\mathbf{R}_k$ is the measurement noise covariance matrix, which can both be represented in their discrete outer product notation as $E[\mathbf{w}_k\mathbf{w}_j^T]$ and $E[\mathbf{v}_k\mathbf{v}_j^T]$ respectively.

Often, the basic equations will ignore the control-input model matrix $\mathbf{B}_k$ if it is a perception or state-estimation only component with

$$\begin{aligned} \mathbf{x}_k &= \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{w}_{k-1} \\ \mathbf{z}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k. \end{aligned} \tag{4.3}$$

For a background on the probability and derivation of these equations, see [27].

The Kalman filter uses equations 4.1 to propagate the state and covariance

$$\begin{aligned} \mathbf{x}_{k|k-1} &= \mathbf{F}_k\mathbf{x}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_{k-1} \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{Q}_k \end{aligned} \tag{4.4}$$

where $\mathbf{P}_{n|m}$ is the state covariance. n is the discrete timestamp of the state before processing the measurement (a priori state) and m is the discrete timestamp of the state after processing the measurement (a posteriori state). The correction step is

then defined as

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \mathbf{x}_{k|k} &= \mathbf{x}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_{k|k-1}) \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1} \end{aligned} \tag{4.5}$$

where $\mathbf{K}_k$ is the discrete-time Kalman gain and $\mathbf{I}$ is simply the identity matrix with the same dimensionality as the Kalman gain.

4.1.1. Types of Kalman Filters

Various forms of the Kalman filter have been developed to suit different types of systems. The four base-type filters are the linear Kalman filter, the extended Kalman filter (EKF), the unscented Kalman filter (UKF), and the error-state Kalman filter (ESKF). The primary difference between these is how the covariance is modeled and propagated.

4.1.1.1. Linear Kalman Filter

Equation sets 4.4 and 4.5 represent the standard linear Kalman filter for a linear system modeled in equation 4.1. While this is the simplest Kalman filter to implement, it typically only works for signal processing or single DOF systems. Any more complex system that requires rotations becomes non-linear and the basic Kalman filter cannot be used.

4.1.1.2. Extended Kalman Filter

To address the limitations of the linearity of the model, the EKF is used for complex, non-linear systems. The following model

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1} \\ \mathbf{z}_k &= h(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{4.6}$$

is a slight modification to the linear model 4.1 where instead of having linear transition matrices, a non-linear state transition function f and measurement model function h are introduced. The Kalman prediction and update equations now become

$$\begin{aligned} \mathbf{x}_{k|k-1} &= f(\mathbf{x}_{k-1|k-1}, \mathbf{u}_{k-1}) \\ \mathbf{P}_{k|k-1} &= \frac{\delta \mathbf{f}}{\delta \mathbf{x}} \mathbf{P}_{k-1|k-1} \Big(\frac{\delta \mathbf{f}}{\delta \mathbf{x}}\Big)^T + \mathbf{Q}_k \end{aligned} \tag{4.7}$$

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k|k-1} \Big(\frac{\delta \mathbf{h}}{\delta \mathbf{x}}\Big)^T \Big(\frac{\delta \mathbf{h}}{\delta \mathbf{x}} \mathbf{P}_{k|k-1} \Big(\frac{\delta \mathbf{h}}{\delta \mathbf{x}}\Big)^T + \mathbf{R}_k\Big)^{-1} \\ \mathbf{x}_{k|k} &= \mathbf{x}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_{k|k-1})) \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \frac{\delta \mathbf{h}}{\delta \mathbf{x}}) \mathbf{P}_{k|k-1}. \end{aligned} \tag{4.8}$$

where $\mathbf{F}_k$ and $\mathbf{H}_k$ have become Jacobians, $\frac{\delta \mathbf{f}}{\delta \mathbf{X}}$ and $\frac{\delta \mathbf{h}}{\delta \mathbf{x}}$ respectively, over the state vector $\mathbf{x}_k$. The challenge with an EKF is that these Jacobian matrix representations are non-trivial to derive and do not easily migrate between systems. Also, because the EKF is built on approximations of a non-linear model, it is not an optimal filter. A highly non-linear system will perform poorly with an EKF.

4.1.1.3. Unscented Kalman Filter

To accommodate highly non-linear systems, the UKF [14] was developed. Instead of propagating the full state through a non-linear function, individual samples of the state (called sigma points) are propagated through the non-linear equations and a mean and covariance can be extracted from these.

These sigma points are represented as a concatenation of $2L + 1$ sigma vectors, where L is the dimension of the state $\mathbf{x}_k$. They are calculated as

$$\mathcal{X}_{k-1} = \left\{ \begin{array}{c} \hat{\mathbf{x}}_{k-1} \\ \hat{\mathbf{x}}_{k-1} + \sqrt{(L+\lambda)\mathbf{P}_{k-1|k-1}} \text{ for i=1,...,L} \\ \hat{\mathbf{x}}_{k-1} - \sqrt{(L+\lambda)\mathbf{P}_{k-1|k-1}} \text{ for i=L+1...2L,} \end{array} \right\} \tag{4.9}$$

and associated weights

$$\begin{aligned} W_0^m &= \lambda/(L+\lambda) \\ W_0^c &= \lambda/(L+\lambda) + (1-\alpha^2+\beta) \\ W_1^m &= W_i^c = 1/\{2(L+\lambda)\} \text{ for i=1,...,2L} \end{aligned} \tag{4.10}$$

where $\lambda = \alpha^2(L+\kappa) - L$ and α, β, and κ help determine the spread of the sigma points. Three sigma matrices are used to maintain the state $\mathcal{X}^x$, the propagation noise $\mathcal{X}^w$, and the measurement noise $\mathcal{X}^v$.

The propagation equation sets then become

$$\begin{aligned}
\mathcal{X}^x_{k|k-1} &= \mathbf{F}[\mathcal{X}^x_{k-1}, \mathcal{X}^w_{k-1}] \\
\mathbf{x}_{k|k-1} &= \sum_{i=0}^{2L} W_i^m \mathcal{X}^x_{i,k|k-1} \\
\mathbf{P}_{k|k-1} &= \sum_{i=0}^{2L} W_i^c [\mathcal{X}^x_{i,k|k-1} - \mathbf{x}_{k|k-1}][\mathcal{X}^x_{i,k|k-1} - \mathbf{x}_{k|k-1}]^T \\
\mathcal{Y}_{k|k-1} &- \mathbf{H}[\mathcal{X}^x_{i,k|k-1}, \mathcal{X}^v_{i,k|k-1}] \\
\mathbf{y}_{k|k-1} &= \sum_{i=0}^{2L} W_i^m \mathcal{Y}_{i,k|k-1}
\end{aligned} \tag{4.11}$$

and

$$\begin{aligned}
\mathbf{P}_{\tilde{y}_k \tilde{y}_k} &= \sum_{i=0}^{2L} W_i^c [\mathcal{Y}_{i,k|k-1} - \mathbf{y}_{k|k-1}][\mathcal{Y}_{i,k|k-1} - \mathbf{y}_{k|k-1}]^T \\
\mathbf{P}_{\tilde{x}_k \tilde{y}_k} &= \sum_{i=0}^{2L} W_i^c [\mathcal{X}_{i,k|k-1} - \mathbf{x}_{k|k-1}][\mathcal{Y}_{i,k|k-1} - \mathbf{y}_{k|k-1}]^T \\
\mathcal{K} &= \mathbf{P}_{\tilde{x}_k \tilde{y}_k} \mathbf{P}^{-1}_{\tilde{y}_k \tilde{y}_k} \\
\mathbf{x}_{k|k} &= \mathbf{x}_{k|k-1} + \mathcal{K}(\mathbf{y}_{k|k} - \mathbf{y}_{k|k-1}) \\
\mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathcal{K} \mathbf{P}_{\tilde{y}_k \tilde{y}_k} \mathcal{K}^T
\end{aligned} \tag{4.12}$$

respectively. This version of the Kalman filter now can handle highly non-linear functions, but requires some tuning of the sigma point values and still a derivation of these propagation matrices.

4.1.1.4. Error-State Kalman Filter

Both the EKF and UKF provide solutions for non-linear systems, but introduce a lot of complexity and computation when modeling the dynamic system to create the state transition matrix $\mathbf{F}$. The ESKF [23] (also called an indirect Kalman

filter) solves that problem by returning to a linear model, except that the model propagates the errors of the state (denoted $\tilde{\mathbf{x}}_k$) instead of the actual state $\mathbf{x}_k$. The errors are assumed to be linear and therefore the standard linear Kalman filter model can be used. [34] compares the indirect to the direct approach, and demonstrates that both perform equally as well, but the ESKF is less computationally expensive[1].

The model for the indirect filter is

$$\begin{aligned} \tilde{\mathbf{x}}_k &= \mathbf{F}_k\tilde{\mathbf{x}}_{k-1} + \mathbf{w}_{k-1} \\ \tilde{\mathbf{z}}_k &= \mathbf{H}_k\tilde{\mathbf{x}}_k + \mathbf{v}_k. \end{aligned} \tag{4.13}$$

The second equation in the model now outputs a residual measurement $\tilde{\mathbf{z}}_k$ (more commonly referred to as $\mathbf{r}_k$) instead of a direct measurement. This residual is simply the difference between the measured state and the propagated state.

The ESKF prediction step equation

$$\begin{aligned} \tilde{\mathbf{x}}_{k|k-1} &= \mathbf{F}_k\tilde{\mathbf{x}}_{k-1|k-1} + \mathbf{G}_k\mathbf{n}_{k-1} \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{Q}_k \end{aligned} \tag{4.14}$$

introduces a new transition matrix $\mathbf{G}_k$ which is the discrete-time noise transition

[1]The conclusion of this work mentions that the dead-reckoning works better for the EKF, but since this propagation is done outside of the filter, dead-reckoning is independent of the filter and can perform just as well with the ESKF.

matrix. The new update step equations

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \mathbf{x}_{k|k} &= \mathbf{x}_{k|k-1} \oplus \mathbf{K}_k(\tilde{\mathbf{z}}_k - \mathbf{H}_k\tilde{\mathbf{x}}_{k|k-1}) \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1} \end{aligned} \tag{4.15}$$

only modify the second equation that computes the error and updates the actual state. Note the use of the paramaterized addition $\oplus$ as opposed to $+$ to signify this is handled uniquely for non-linear terms (see sec. 4.2.3). The full details of computing the various matrices needed for the ESKF are described in sec. 4.2.

4.2. Error-State Kalman Filter

The state we are estimating consists of vehicle orientation, velocity, position and IMU (accelerometer and gyroscope) biases. This is represented within a 16-dimensonal vector as

$$\mathbf{x}(t) = \begin{bmatrix} {}^G_L\mathbf{q}^T(t) & {}^L\mathbf{b_g}^T(t) & {}^G\mathbf{v}^T(t) & {}^L\mathbf{b_a}^T(t) & {}^G\mathbf{p}^T(t) \end{bmatrix}^T \tag{4.16}$$

where our notation closely follows [30]. We represent the orientation of the vehicle as a rotation of the global frame {G} with respect to the local frame {L} in the form of the quaternion ${}^G_L\mathbf{q}(t)$. We denote the equivalent 3x3 rotation matrix as $\mathbf{C_q}$. Therefore, our rotation representation transforms a vector from the local to the global frame as ${}^G\mathbf{v} = \mathbf{C_q} \cdot {}^L\mathbf{v}$. The gyroscope bias ${}^L\mathbf{b_g}^T(t)$ and accelerometer bias ${}^L\mathbf{b_a}^T(t)$ are represented in the local vehicle frame. The vehicle velocity ${}^G\mathbf{v}^T(t)$ and position ${}^G\mathbf{p}^T(t)$ are in the global frame. The global frame is an earth-

centered, earth-fixed (ECEF) reference frame so that long distance sprints are treated linearly and not corrupted by earth's curvature (see Appendix B). This also simplifies the model of earth's rotation, taken as ${}^{G}\boldsymbol{\omega}_{\mathbf{e}}$. The global gravity vector ${}^{G}\mathbf{g}$ is calculated at each time step using the WGS84 reference ellipsoid as described in [5].

4.2.1. State Propagation

State propagation, also called mechanization, is the process of predicting a new state estimate $\hat{\mathbf{x}}_{k+1}$ given a starting state $\hat{\mathbf{x}}_k$, transition measurements, and a time delta. For our purposes, this is estimating a new position, orientation, and velocity given input from the IMU, i.e. rotational velocity and linear acceleration. If the IMU had no noise and the system was linear, then the propagation would provide an exact solution and no filtering would be needed. Following our notation convention, where the hat '^' represents an estimated state, we rewrite our state estimate vector as

$$\hat{\mathbf{x}}(t) = \begin{bmatrix} {}^{G}_{L}\hat{\mathbf{q}}^{T}(t) & {}^{L}\hat{\mathbf{b}}_{\mathbf{g}}^{T}(t) & {}^{G}\hat{\mathbf{v}}^{T}(t) & {}^{L}\hat{\mathbf{b}}_{\mathbf{a}}^{T}(t) & {}^{G}\hat{\mathbf{p}}^{T}(t) \end{bmatrix}^{T}. \tag{4.17}$$

The vehicle state is first estimated by propagating the kinematic equations given the input IMU sensor data. The measured IMU data, ${}^{L}\boldsymbol{\omega}_{m}(t)$ (angular velocity) and ${}^{L}\mathbf{a}_{m}(t)$ (linear acceleration), is modeled as

$$\begin{aligned} {}^{L}\boldsymbol{\omega}_{\mathbf{m}}(t) &= {}^{L}\boldsymbol{\omega}_{\mathbf{true}}(t) + {}^{L}\,\mathbf{b}_{\mathbf{g}}(t) + \mathbf{n}_{\mathbf{g}}(t) \\ {}^{L}\mathbf{a}_{\mathbf{m}}(t) &= \mathbf{C}_{\mathbf{q}}{}^{T}({}^{G}\mathbf{a}_{\mathbf{true}}(t) - {}^{G}\,\mathbf{g}) + {}^{L}\,\mathbf{b}_{\mathbf{a}}(t) + \mathbf{n}_{\mathbf{a}}(t). \end{aligned} \tag{4.18}$$

For the gyroscope and accelerometer measurements, we assume zero-mean white Gaussian noise ($\mathbf{n_g}(t)$, $\mathbf{n_a}(t)$) and zero-mean first-order random walk ($^L\mathbf{b_g}(t)$, $^L\mathbf{b_a}(t)$) (see Appendix C for more details on IMU noise models). The estimated state $\hat{\mathbf{x}}(t)$ is propagated using the following kinematic equations

$$
\begin{aligned}
{}^G_L\dot{\hat{\mathbf{q}}}(t) &= \frac{1}{2}\mathbf{\Omega}(\hat{\boldsymbol{\omega}}(t)){}^G_L\hat{\mathbf{q}} \\
{}^L\dot{\hat{\mathbf{b}}}_{\mathbf{g}}(t) &= \mathbf{0}_{3x1} \\
{}^G\dot{\hat{\mathbf{v}}}(t) &= \mathbf{C}_{\hat{\mathbf{q}}} \cdot {}^L\hat{\mathbf{a}}(t) + {}^G\mathbf{g} - 2\lfloor {}^G\boldsymbol{\omega_e}\times\rfloor {}^G\hat{\mathbf{v}}(t) \\
&\quad - \lfloor {}^G\boldsymbol{\omega_e}\times\rfloor^2 {}^G\hat{\mathbf{p}}(t) \\
{}^L\dot{\hat{\mathbf{b}}}_{\mathbf{a}}(t) &= \mathbf{0}_{3x1} \\
{}^G\dot{\hat{\mathbf{p}}}(t) &= {}^G\hat{\mathbf{v}}(t)
\end{aligned}
\tag{4.19}
$$

where $\hat{\boldsymbol{\omega}}(t) = {}^L\boldsymbol{\omega}_{\mathbf{m}}(t) - {}^L\hat{\mathbf{b}}_{\mathbf{g}}(t) - C^T_{\hat{q}} \cdot {}^G\boldsymbol{\omega_e}$ and $\hat{\mathbf{a}}(t) = {}^L\mathbf{a}_{\mathbf{m}}(t) - {}^L\hat{\mathbf{b}}_{\mathbf{a}}(t)$. The quaternion derivative uses the matrix operation $\mathbf{\Omega}$

$$
\mathbf{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\times\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{4.20}
$$

where $\lfloor\boldsymbol{\omega}\times\rfloor$ is the skew-symmetric matrix of $\boldsymbol{\omega}$. This propagation is carried out using the Runge-Kutta integration method RK4, and it is important to note that this has significant improvement over other lower order methods. Pushing (4.19) directly into an RK4 solver as a coupled linear/rotational integration improves accuracy vital for long distance dead-reckoning, particularly in the linear velocity

estimate.

4.2.2. Error-State Prediction Step

After the estimated state vector $\hat{\mathbf{x}}(t)$ has been propagated for a single timestep, the covariance of the error-state is propagated. The measurement prediction step of the Kalman filter propagates the error-state and the respective state covariances. The error-states are a linearized version of our physical state. These states are represented in a 15-dimensional error vector defined as

$$\tilde{\mathbf{x}}(t) = \begin{bmatrix} \tilde{\boldsymbol{\delta\theta}}^T(t) & {}^L\tilde{\mathbf{b}}_{\mathbf{g}}^T(t) & {}^G\tilde{\mathbf{v}}^T(t) & {}^L\tilde{\mathbf{b}}_{\mathbf{a}}^T(t) & {}^G\tilde{\mathbf{p}}^T(t) \end{bmatrix}^T. \tag{4.21}$$

Note that the drop in dimensionality occurs in the orientation representation. Assuming orientation errors are small, the over-constrained four dimensional quaternion format is approximated by a three-dimensional vector, defined by

$$\boldsymbol{\delta q} \backsimeq \begin{bmatrix} \frac{1}{2}\tilde{\boldsymbol{\delta\theta}} \\ 1 \end{bmatrix}. \tag{4.22}$$

The continuous-time linearized dynamics of the error-state is written as

$$\dot{\tilde{\mathbf{x}}} = \mathbf{F}_c(\mathbf{x})\tilde{\mathbf{x}} + \mathbf{G}_c\mathbf{n}, \tag{4.23}$$

where $\mathbf{F}_c$ is the continuous-time error-state transition matrix and $\mathbf{G}_c$ is the continuous-time noise propagation matrix:

$$\mathbf{F}_c = \begin{bmatrix} -\lfloor\hat{\boldsymbol{\omega}}\times\rfloor & -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -\mathbf{C_q}\lfloor\hat{\mathbf{a}}\times\rfloor & \mathbf{0}_3 & -2\lfloor{}^G\boldsymbol{\omega}_e\times\rfloor & -\mathbf{C_q} & -\lfloor{}^G\boldsymbol{\omega}_e\times\rfloor^2 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \tag{4.24}$$

$$\mathbf{G}_c = \begin{bmatrix} -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{C_q} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}. \tag{4.25}$$

Both $\mathbf{F}_c$ and $\mathbf{G}_c$ are common state transition matrices, used in [30], [18], [17], a model that does not incorporate Earth's rotation in [8], and a slightly modified $\mathbf{G}_c$ in [10].

The continuous-time matrix 4.24 is converted to a discrete-time matrix Φ_k using Taylor series expansion on a matrix:

$$\boldsymbol{\Phi}_k = e^{\int \mathbf{F}_c(t)dt} = \mathbf{I}_{15} + \mathbf{F}_c dt + \frac{1}{2!}\mathbf{F}_c^2 dt^2 + \frac{1}{3!}\mathbf{F}_c^3 dt^3 + ... \tag{4.26}$$

where $\boldsymbol{\Phi}_0 = \mathbf{I}_{15}$. This is applied to the covariance update equation

$$\mathbf{P}_{k+1|k} = \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^T + \mathbf{Q}_d \tag{4.27}$$

where the discrete-time propagation noise matrix $\mathbf{Q}_d$ is updated by

$$\mathbf{Q}_d = \mathbf{\Phi}_k \mathbf{G}_c \mathbf{Q}_c \mathbf{G}_c^T \mathbf{\Phi}_k^T \cdot dt \tag{4.28}$$

and Q_c represents the process noise model matrix, defined as

$$\mathbf{Q}_c = \begin{bmatrix} \sigma_{g_n}^2 \cdot \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \sigma_{g_b}^2 \cdot \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \sigma_{a_n}^2 \cdot \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \sigma_{a_b}^2 \cdot \mathbf{I}_3 \end{bmatrix}. \tag{4.29}$$

The σ terms are found from the sensor specs of the IMU with white noise terms as σ_{g_n}, σ_{a_n} and bias stability (random walk) as σ_{g_b}, σ_{a_b}, each pair for gyroscope and accelerometer respectively.

4.2.3. Error-State Update Step

The update step (or correction step) takes a second sensor measurement with a corresponding noise model and updates the first propagated measurement and its covariance.

The measurement update step of the error-state Kalman filter is used to update the uncertainty of the state given a global correction. These global corrections are supplied to the filter from a three dimensional position correction from the LIDAR localization described in sec. 3.2. The measurement directly (and only) affects the position estimate, reflected by the Jacobian $\mathbf{H} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}$. The rest

of the update follows the standard Kalman filter equations

$$\begin{aligned} \mathbf{S} &= \mathbf{H}\mathbf{P}_{k+1|k}\mathbf{H}^T + \mathbf{R} \\ \mathbf{K} &= \mathbf{P}_{k+1|k}\mathbf{H}^T\mathbf{S}^{-1} \\ \tilde{\mathbf{x}}_k &= \mathbf{K}\mathbf{r} \\ \mathbf{P}_{k+1|k+1} &= (\mathbf{I}_{15} - \mathbf{K}\mathbf{H})\mathbf{P}_{k+1|k}(\mathbf{I}_{15} - \mathbf{K}\mathbf{H})^T + \mathbf{K}\mathbf{R}\mathbf{K}^T \end{aligned} \tag{4.30}$$

where $\mathbf{R}$ is the measurement noise matrix of the correction and the correction residual is $\mathbf{r} = \mathbf{p}_{correction} - \tilde{\mathbf{p}}$. We treat the measurement noise matrix as a static uncertainty on the resolution of the DEM. Although orientation is not computed or corrected from LIDAR localization, the measurement update indirectly estimates the orientation error from the position error over time. The newly computed covariance $\mathbf{P}_{k+1|k+1}$ is then used for the prediction step as described previously in sec. 4.2.2.

The last step is to update the actual state estimate. The linear terms of the state vector $\hat{\mathbf{x}}(t)$ are updated by $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \tilde{\mathbf{x}}_k$. The orientation is updated using Quaternion multiplication

$${}^G_L\hat{\mathbf{q}}_{k+1} = {}^G_L\hat{\mathbf{q}}_k \otimes \delta\mathbf{q}_{k+1}. \tag{4.31}$$

4.3. Addressing Long-range Challenges

Filters are susceptible to linearization errors and these errors become more apparent for long-range missions. To mitigate these issues and maintain stability, we apply the following five techniques.

First, the matching procedure in the LIDAR localization algorithm (for one scan) exceeds that of a single propagation step in the filter. When the position correction is received by the filter, the state has already been propagated multiple times. Therefore, the filter is rolled back to the state at which the correction occurred, applied with the correction, and the propagation is recomputed. We keep a history of states that is longer in duration than the amount of processing time required for the LIDAR localization. This history tracks state estimates, uncertainties, and input IMU measurements.

Second, the Kalman gain $\mathbf{K}$ includes the inversion of the $\mathbf{S}$ matrix (see equation 4.30). Doing inversions on poorly condition matrices can cause the covariance matrix to diverge. Ill-condition matrices can occur from iteratively inverting large high-precision numbers, such as what the ECEF reference frame uses. To prevent the ill-conditioning of these matrices, a position offset is applied in the initialization of the filter and added back outside of the filter.

Third, even with an offset applied, long distance traversals will accumulate larger numbers in the matrix inversion. As an additional layer of precision, we use a more expensive matrix inversion method, the full householder QR inverse in Eigen[2]. Because of the lightweight nature of the Kalman filter, this more expensive computation is temporally negligible, even at 200Hz.

Fourth, numerical stability is improved by using a more stable version of the measurement update, called the Joseph form, derived as

$$\begin{aligned} \mathbf{P}_{k+1|k+1} &= (\mathbf{I}_{15} - \mathbf{KH})\mathbf{P}_{k+1|k} \\ &= (\mathbf{I}_{15} - \mathbf{KH})\mathbf{P}_{k+1|k}(\mathbf{I}_{15} - \mathbf{KH})^T + \mathbf{KRK}^T. \end{aligned} \tag{4.32}$$

[2]http://eigen.tuxfamily.org/index.php?title=Main_Page

Lastly, the covariance matrix $\mathbf{P}$ must always be symmetric. While the Joseph form of the Kalman update does help prevent numerical issues, we ensure the covariance matrix maintains symmetry by regularly re-symmetrizing using

$$\mathbf{P} := \frac{1}{2}(\mathbf{P} + \mathbf{P}^T) \tag{4.33}$$

at every correction step.

4.4. Stationary Monitoring

For multi-hop missions (missions in which the vehicle will land for a period of time at various points along its trajectory), the stationary vehicle's INS solution will continue to drift. This presents a problem for state estimation. Both the LIDAR and camera are too close to the ground for correcting our estimate. After a period of time, the INS solution could drift outside of the feasible search region for LIDAR localization when the vehicle is back in flight.

To address the drifting INS issue, the IMU can be used to monitor a stationary vehicle. Vibrations from the vehicle's engines disrupt the accelerometer data. The gyroscope however still provides a clear separation to classify a stationary versus a dynamic state. At each timestep, if all three gyro data channels are below a threshold, the vehicle is determined to be in a stationary state, and a zero-movement correction is applied to the filter.

The vehicle's transition from a landed state is highly sensitive to errors. If the vehicle is falsely declared in a landed state when it is in motion, the velocity estimate at initial takeoff can severely corrupt the trajectory estimate. In practice,

it is better to apply a strict threshold so false positives (estimate moving when physically stationary) occur more often than true negatives (estimate stationary when physically moving). To prevent noisy data from repeatedly jumping our state estimate between stationary and moving, a voting scheme is used and if 95 or more of the last 100 samples are within the threshold, we declare a landing state. For the 200Hz IMU, this window covers a half-second interval.

5. Experimental Results

5.1. Hardware Setup

Figure 5.1.: The Near Earth Autonomy m4 sensor suite on a Bell 206L helicopter.

Our datasets were collected from a Bell 206L (LongRanger) helicopter outfitted with the Near Earth Autonomy m4 sensor suite (see fig. 5.1) which includes a 2D LIDAR and a strapdown fiber-optic IMU. The LIDAR provides roughly 10^5 point measurements per second, each with a separate time stamp. Additionally, a GPS inertial solution is collected to serve as ground truth in our evaluation. All sensors are time synchronized. The initial vehicle position and orientation in ECEF is provided, as well as the position and orientation uncertainty.

We use publicly available $10\,m$ resolution DEMs from USGS as prior elevation maps to localize against. To build a local 3D elevation model from LIDAR, we accumulate 350 sequential scans. It is important to note that this number can be adjusted based on vehicle speed to ensure a sufficiently large scan region. As shown in tab. 5.2, the vehicle averaged at about 50 m/s (180 km/hr).

We tested on a quad-core 2.5GHz i7 processor, running two parallel threads for filtering and matching separately. The filter is updated at $200\,Hz$, taking a constant $0.1\,ms$ for propagation, while the less frequent LIDAR measurement updates take $11\,ms$. The time required for the LIDAR localization algorithm is on average 22 ms with a standard deviation of 3 ms for the presented results.

Three separate datasets were collected for the analysis (see tab. 5.1). Two datasets were long-range (A,B) and one dataset (C) was a multi-hop mission (landing at various points along the trajectory).

Flight	trajectory distance	dataset duration	# of datapoints
A	196 km	01:12:31	~894,000
B	218 km	01:06:47	~890,000
C	56.7 km	00:30:09	~360,000

Table 5.1.: 3 datasets that our algorithm performed on.

5.2. Long-range LIDAR Localization Missions

Table 5.2.: Statistics for the two long distance flights, including LIDAR localization and state estimate performance.

Flight	Trajectory distance	Avg. flight speed	# of match attempts	% of match successes	Longest duration without a match	Landing position error	Max. position error
A	196 km	44.5 m/s	853	83.4%	62.8 s	38.6 m	90.2 m
B	218 km	55.6 m/s	851	71.3%	92.2 s	37.8 m	77.9 m

The fusion of IMU with LIDAR localization helped to reduce drift and to maintain a good state estimate for periods of poor localization performance. fig. 5.2 shows the matches for the entire trajectory of flight B, with an inset showing a period that had very few corrections due to buildings, flat terrain, and a river. The rest of the dataset shows very frequent successful matches, despite the mostly dense vegetation encountered.

tab. 5.2 shows the success ratio for two separate flights. The LIDAR matching succeeded at least 70% of the time. The robustness of the Kalman filter is tested during longer periods of unsuccessful matches. The system kept the state estimate within $100\,m$ of ground truth throughout the entirety of both trajectories. fig. 5.3 plots the errors in individual axes against the ground truth. The blue line is the LIDAR-inertial solution estimate and the red line represents the IMU-only dead reckoning attempt over the same duration. Although the maximum position error for this trajectory is $77.9\,m$, this magnitude occurs very rarely, and the solution stays within a $20\,m$ error laterally and $5\,m$ error vertically for most of the flight. Our maximum position error of 90 m is comparable to the results found in [12], however we demonstrate smaller position errors on average over the entire trajectory, as well as significantly longer distances achieved.

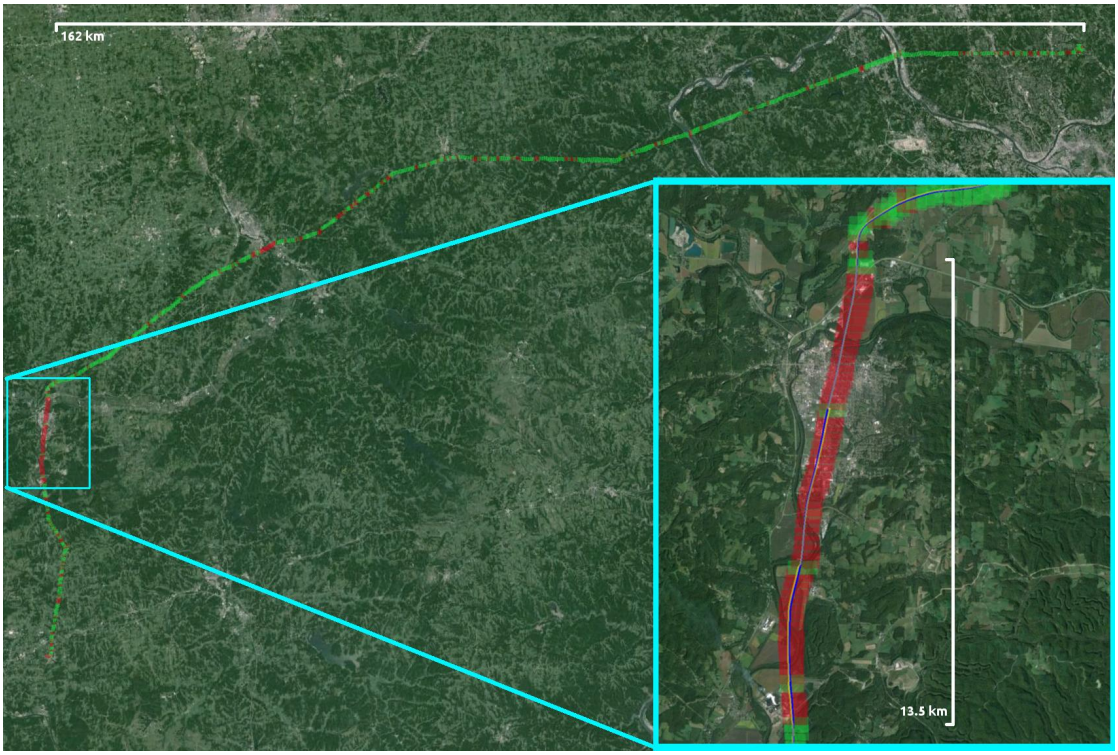


Figure 5.2.: LIDAR localization match successes (green rectangles) and failures (red rectangles) for the entire trajectory of flight B. The inset shows an enlarged section of the trajectory in which very few successes occurred and therefore the navigation relied on the bias-corrected inertial solution. At this scale, the ground truth trajectory (blue line) is nearly identical to the estimated trajectory (yellow line) and just barely visible in the inset.

fig. 5.4 shows Euclidean position error in relation to successful and failed matching events. As expected, during flight segments in which we fail to successfully match the LIDAR with the terrain, the position error grows over time. The growth is determined by the robustness of the IMU bias estimate and state propagation. Note that we stop acquiring successful matches near the end of the trajectory. This is due to the landing sequence, where the vehicle is too close to the ground to observe the terrain shape. Before that point, the error is less than $20\,m$ from ground truth, while the final position error is close to $40\,m$.

We visualize the resulting percentage of error over distance flown in fig. 5.5, comparing it to an IMU-only dead reckoning estimate. The flight lands with a distance error ratio of 0.019% over the entire 196 km trajectory. The data shows that the position error is bounded independent of the length of the flight. In contrast, the IMU-only solution drifts without bounds, showing that even for a high quality fiber-optic IMU, an IMU-only solution is infeasible and integration of additional information such as from our LIDAR localization is needed.

5.3. Multi-hop Missions

Longer missions require short stops to refuel, so we have tested the robustness of the state estimation in periods of LIDAR localization blackout, i.e., when landed on the ground. VO methods work at lower altitudes than the LIDAR localization technique, so for a better comparison, we have also run a VO method [35].

fig. 5.6 shows the overhead profile of a two-stop mission. The bottom right corner of the image was the initial takeoff and final landing position, and the vehicle flew in a clockwise direction, landing at both of the furthest points of the triangle. The mission was 56.7 km and the final landing error for the LIDAR localization was less than 10 m and no more than a total of 26.1 m at any point during the mission. fig. 5.7 shows a closer look at the final landing position.

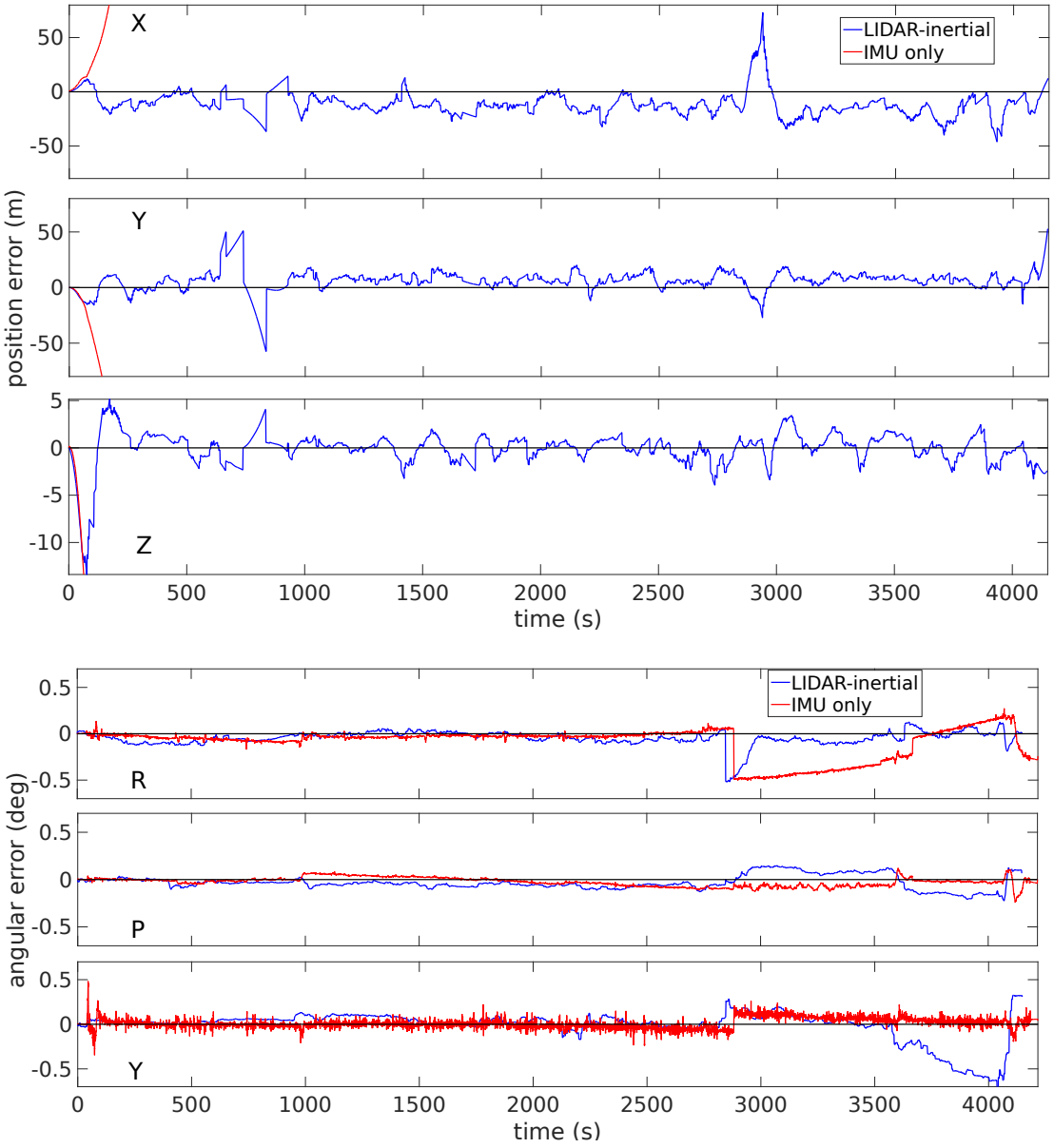


Figure 5.3.: Linear and angular errors of the state estimate for flight B. Our LIDAR-inertial solution (blue) is compared against an IMU-only dead reckoning attempt (red).

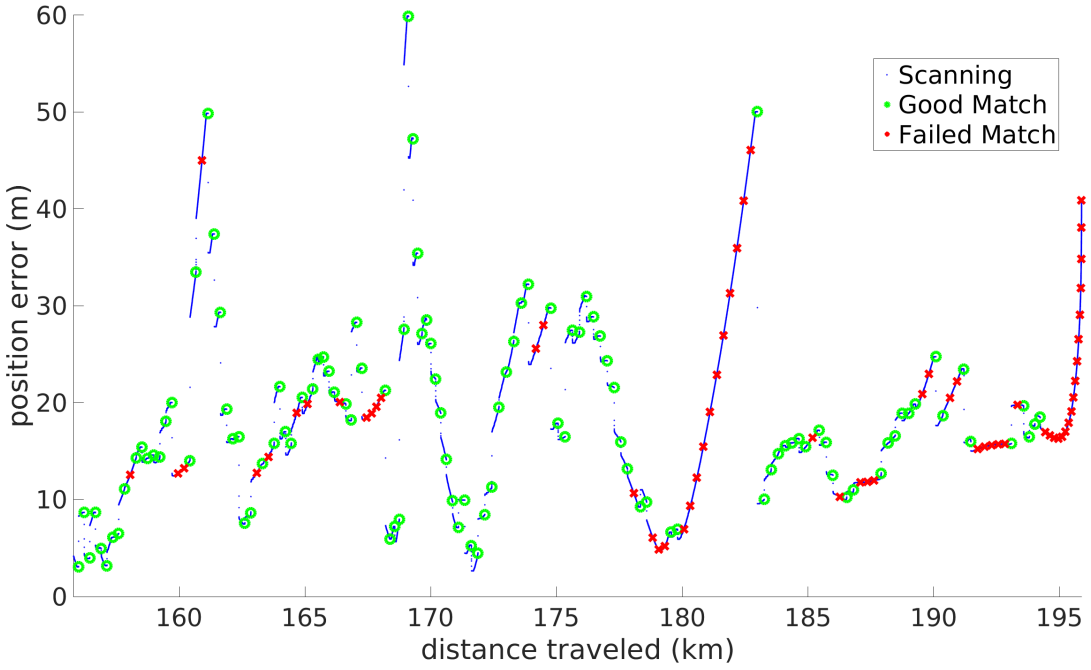


Figure 5.4.: Euclidean position error for the last 40 kilometers of flight A, with successful (green) and failed (red) matching attempts marked. The drift from the inertial estimate grows the position error over periods of failed LIDAR matches until a successful match resets the error.

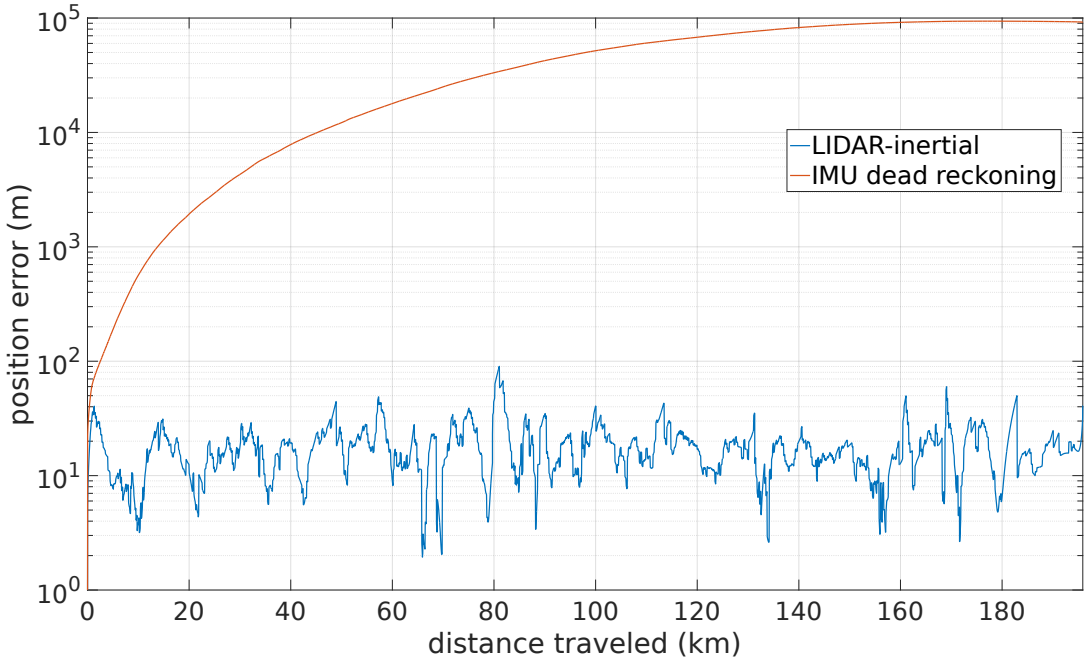


Figure 5.5.: Euclidean position error over the entirety of flight A. We compare our solution (blue) against an IMU-only estimate (red) which quickly drifts away from the correct position. Our LIDAR-inertial solution keeps the position error bounded independent of distance.

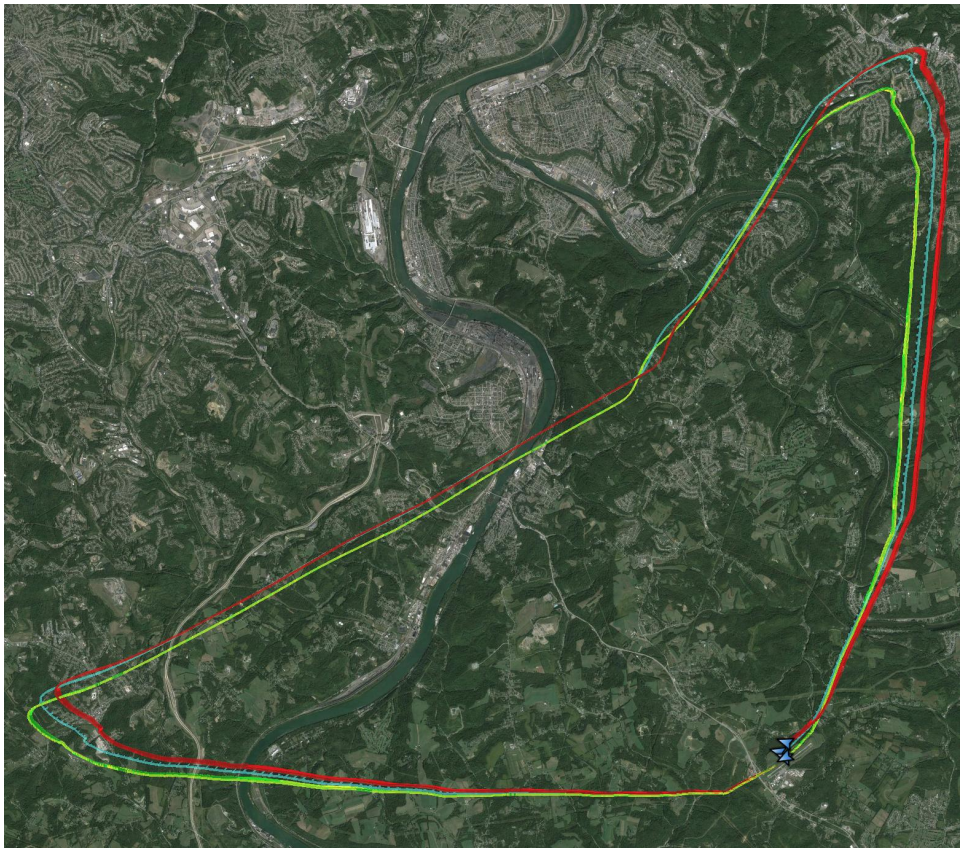


Figure 5.6.: The overhead flight plan for the multi-hop mission, flight C. The red trajectory is an IMU-only (with stationary monitoring) dead-reckoning. The cyan trajectory is the visual odometry comparison. The yellow trajectory is our LIDAR DEM localization and the green trajectory is the ground truth reference. This full flight was 56.7 km.

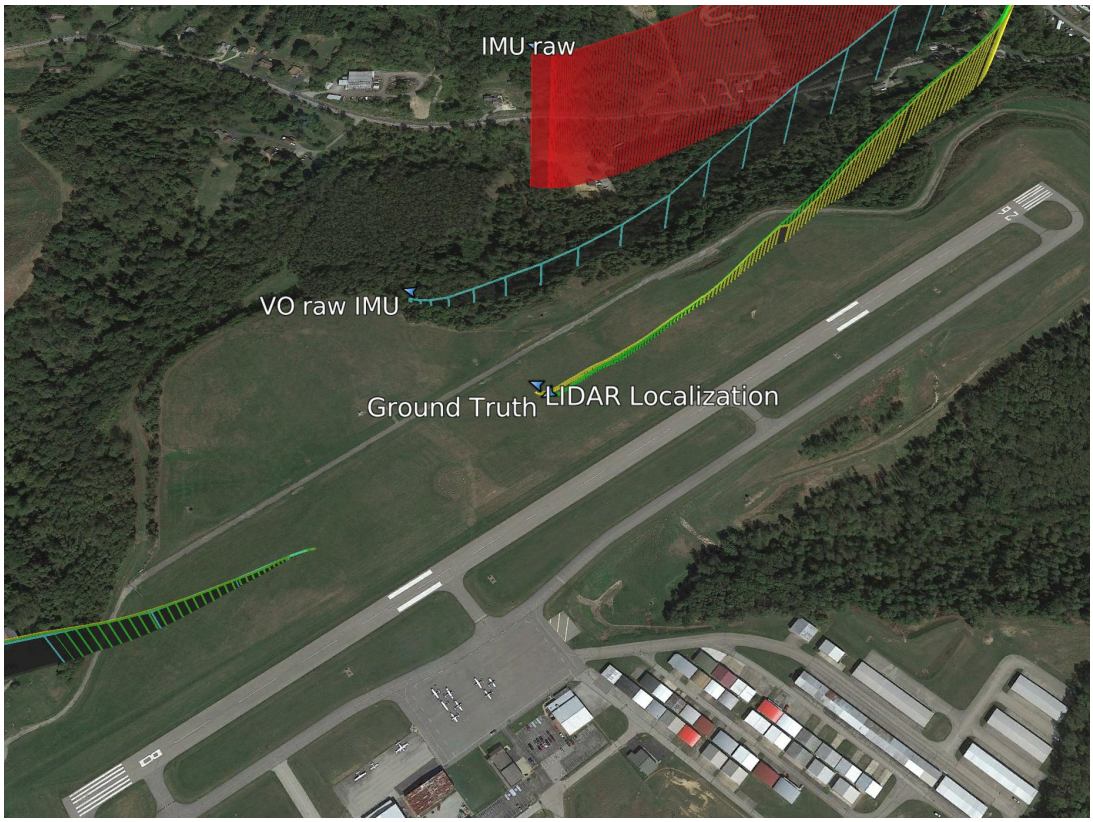


Figure 5.7.: A closeup of the initial and final landing point for flight C. The red trajectory is an IMU-only (with stationary monitoring) dead-reckoning. The cyan trajectory is the visual odometry comparison. The yellow trajectory is our LIDAR DEM localization and the green trajectory is the ground truth reference.

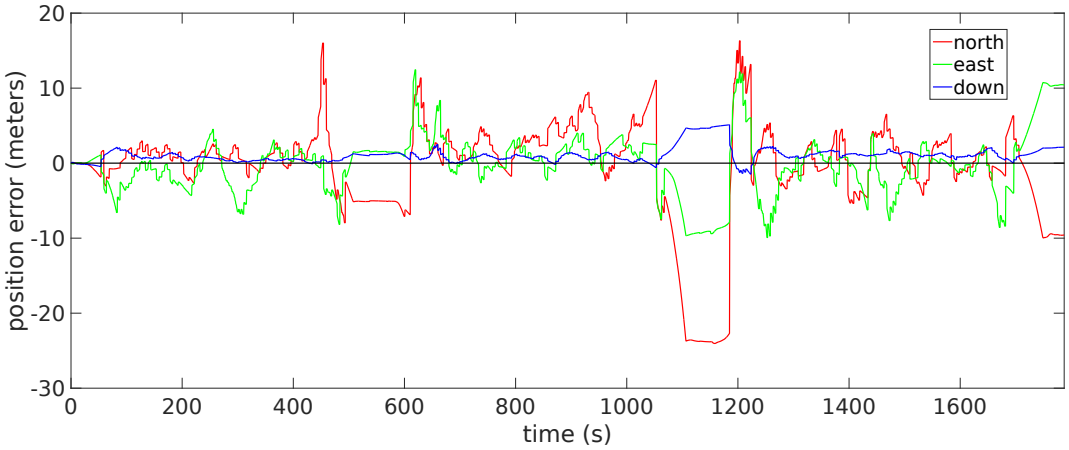


Figure 5.8.: The linear trajectory error for the multi-hop mission, flight C. The largest position errors occur when in a takeoff or landing sequence, where the LIDAR is too close to the ground to perform corrections. This plot also shows that the stationary monitoring (described in sec. 4.4) keeps the error from growing when landed.

6. Conclusion

This thesis presented a solution to long-distance aerial state estimation for LIDAR-based sensor systems. Using available a priori DEM, we are able to match a LIDAR scan region against this elevation model efficiently in real-time. By using an error-state Kalman filter, we can estimate IMU biases and provide a better position estimate. This allows us to reduce the search region of the DEM as well as fly longer distances without LIDAR position corrections.

6.1. Lessons Learned

A few key components that significantly improved the results:

1. Using the rectangular coordinate frame ECEF instead of the warped trapezoidal UTM coordinate frame eliminated position errors caused from Earth's curvature, UTM grid scaling, and true north versus UTM north misalignments. For more details on ECEF, see Appendix B.

2. The dead-reckoning / state propagation had considerable improvements using a robust 4th-order Runge Kutta integration, as mentioned in sec. 4.2.1. Previously, both Euler and a decoupled RK4 implementation provided a

higher drifting state estimate that was erroneously perceived as incorrect bias calculations. Having a coupled RK4 method that recomputes the linear vectors from the rotation estimates at each state provides smaller linearization errors. There exists higher order versions of the Runge Kutta integrator [21] that could provide even better results.

3. The frame transformation direction as expressed by the rotation matrix is critical. Most research papers express rotation as a global-to-local rotation which can be less intuitive for real data. Rotations expressed with Euler angles (e.g. an aircraft with 30° pitch) is a rotation expressed from the local (body) to the global (inertial) frame. In other words, the matrix representation of this orientation would convert a vector in the local frame to a vector in the global frame. This is briefly shown in sec. 4.2.

7. Future Work

Additional sensors can be added to improve robustness or reduce the requirement of a high-cost IMU. Drift depends on both the quality of the IMU and the accuracy of the bias estimation. To improve bias estimation accuracy, visual odometry can be fused with the system. This would provide continuous bias estimate corrections for periods of unsuccessful LIDAR localization possibly due to flight over water, low terrain variability, or low-altitude flight (see failure cases in sec. 3.3). Therefore less drift is expected over time, allowing for longer duration missions in a wider range of terrains.

The $10\,m$ resolution DEM provides too coarse of a correction estimate if a smooth trajectory is desired. We have tested our method on a $3\,m$ resolution DEM over shorter trajectories and have achieved tighter position estimates. The associated 10-fold increase in memory requirements necessitate dynamic DEM management solutions to cover long trajectories.

Kalman filters are susceptible to linearization errors, but there are additional techniques beyond the presented work that can improve their performance. The discrete-time error-state transition matrix $\mathbf{\Phi}_k$ is currently approximated using the Taylor series expansion of $\mathbf{F}_c$. To increase accuracy and reduce computational cost a closed-form solution to $\mathbf{\Phi}_k$ can be derived as in [33].

Smoothing is an extension of filtering where the entire trajectory is recomputed instead of a single position [15]. This reduces linearization errors and provides better trajectory estimates. While this is expensive for long-trajectories, a fixed-lag smoother which only recomputes a short history could provide better position estimates without the expense of recomputing the entire trajectory.

Acknowledgments

I would like to thank my advisor, Dr. Michael Kaess, for his patience with me in my journey through this research. I would also like to thank my other advisor, Dr. Sanjiv Singh, for giving me the opportunity to work on this project. Thank you as well to my other committee members, Dr. George Kantor and Humphrey Hu for reviewing and supporting my work. Thank you to Jeff Mishler of Near Earth Autonomy for providing me with data and various tools to help me progress with my work, as well as everyone else at Near Earth Autonomy who contributed to the overarching GPS-denied effort. I also extend thanks to Aaron Acton of Astrobotic for validating some of my uncertainty (no pun intended) in bias estimation.

A. Quaternions

Quaternions are rotations represented by 4 numbers instead of 3 (Euler angles) or 9 (rotation matrices). They are preferred over Euler angles because they do not suffer from gimbal lock and are preferred over rotation matrices because of the lower space requirement (4 scalars per rotation instead of 9).

A quaternion orientation or rotation is represented by a single scalar value and an imaginary vector

$$\mathbf{q} = (\mathbf{q}_i, q_3) = q_0 i + q_1 j + q_2 k + q_3 \tag{A.1}$$

or represented in vector form

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_i \\ q_3 \end{bmatrix} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \tag{A.2}$$

where q is the real/scalar component and the other values represent an axis in an imaginary plane. The imaginary bases i, j, and k follow the non-commutative properties

$\times$	i	j	k
i	-1	k	$-j$
j	$-k$	-1	i
k	j	$-i$	-1

Table A.1.: Imaginary bases that define the Hamiltonian form (right hand coordinate frame) of quaternion operations.

A.1. Operations with Quaternions

Quaternion multiplication is defined as

$$\begin{aligned} \mathbf{q} \otimes \mathbf{p} &= (q_0 i + q_1 j + q_2 k + q_3)(p_0 i + p_1 j + p_2 k + p_3) \\ &= \begin{bmatrix} q_3 p_0 - q_2 p_1 + q_1 p_2 + q_0 p_3 \\ q_2 p_0 + q_3 p_1 - q_0 p_2 + q_1 p_3 \\ -q_1 p_0 + q_0 p_1 + q_3 p_2 + q_2 p_3 \\ -q_0 p_0 - q_1 p_1 - q_2 p_2 + q_3 p_3 \end{bmatrix} \end{aligned} \tag{A.3}$$

which is a non-commutative operation.

For a more thorough analysis of the quaternion derivations, see [29]. Note that they do not use Hamiltonian form, but instead the "natural order". This is a left-handed coordinate convention, and while that does not follow most modern engineering practices, the derivation for the quaternion math is more natural (see pg. 473 of [26]).

$\times$	i	j	k
i	-1	$-k$	j
j	k	-1	$-i$
k	$-j$	i	-1

Table A.2.: Imaginary bases in a non-Hamiltonian, left hand coordinate frame.

B. Global Coordinate Frames

Coordinate frames play a significant role in integrating a Kalman filter into an actual physical system. In many cases, taking a Cartesian frame with respect to an arbitrary origin works. When using external references, external forces, and a large environment, such as the Earth, a more robust reference frame is necessary.

Global navigation coordinates can be provided in a geographic frame by a latitude and longitude (sometimes called geodetic if in reference to a spheroid, such as WGS84). However, for propagation of position and orientation, a rectangular or Cartesian coordinate frame is preferred. In the following sections, two common global/inertial coordinate frames are discussed. Note that for the body/vehicle frame, either East-North-Up (ENU) or North-East-Down (NED) axis conventions are fine. This work uses NED.

B.1. UTM

One common global coordinate frame for Earth navigation is Universal Transverse Mercator (UTM) coordinate system. This projection divides up the surface of the Earth into pseudo-rectangular zones and provides a distance from the lower left corner. UTM is preferred for its simplicity and the coordinates usually make sense

physically.

Unfortunately UTM is not truly Cartesian. The grid that aligns to the earth's surface actually warps with the surface of the Earth. This means that the length of a meter changes as you traverse along the North-South direction. Additionally, the North lines all point to the same point on the Earth, so a heading of North in a UTM frame will shift the further from the center of a UTM zone you are, by as much as a half degree. Both of these cause significant problems for long distance, high-precision state estimation. Therefore, a different Cartesian reference is preferred.

B.2. ECEF

The correct coordinate frame to use for the Earth is earth-centered, earth-fixed. This creates a rectangular coordinate frame in which the origin sits at the center of the Earth, with X and Y axes pointing out from the Earth at the equator and the Z axis pointing up through the North pole (see fig. B.1). Although this reference frame does not suffer from non-linearities, it is less intuitive given position relative to the center of the Earth in meters. There are standard equations that can be found in [5] that convert a local tangent plane or geographic frame to ECEF. Also note that now gravity will need to be converted, but several high-fidelity software libraries exist that express gravity in the ECEF frame.

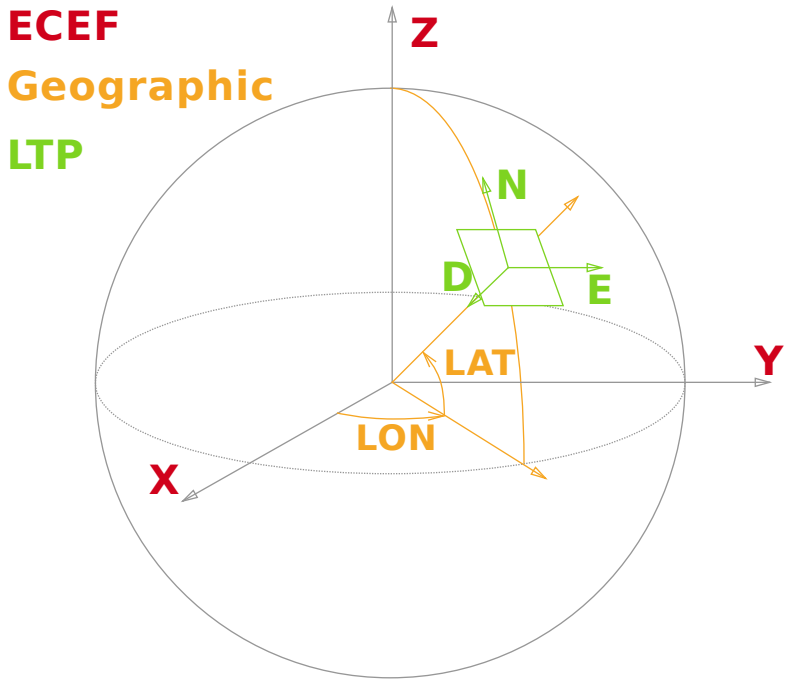


Figure B.1.: The Earth expressed in an ECEF frame, geographic frame, and showing a local tangent plane.

C. IMU Modeling

MEMS IMUs suffer from various noises and manufacturing defects which lead to time-varying biases and integration errors. In order to extract meaningful data from the IMU, modeling the noise is critical. IMUs range in quality and cost, and typically the more expensive the sensor, the lower the biases will deviate. Regardless of the cost however, modeling is still required to estimate the biases.

The gyroscope noise model

$$\begin{aligned} \boldsymbol{\omega}_m(t) &= \boldsymbol{\omega}_{true}(t) + \mathbf{b}_g(t) + \mathbf{n}_{wn}(t) \\ \dot{\mathbf{b}}_g(t) &= \mathbf{n}_{rw}(t) \end{aligned} \tag{C.1}$$

and the equivalent accelerometer noise model

$$\begin{aligned} \mathbf{a}_m(t) &= \mathbf{a}_{true}(t) + \mathbf{b}_a(t) + \mathbf{n}_{wn}(t) \\ \dot{\mathbf{b}}_a(t) &= \mathbf{n}_{rw}(t) \end{aligned} \tag{C.2}$$

show the measured value ($\boldsymbol{\omega}_m$,$\mathbf{a}_m$) is equal to the true value with some bias and a zero-mean white Gaussian noise. The second lines show that their respective biases change over time based on a zero-mean first-order random walk.

A more complex model of the IMU that takes into account an additional noise

parameter (see sec. C.1) is

$$\begin{aligned} \boldsymbol{\omega}_m(t) &= (1+\mathbf{k})\boldsymbol{\omega}_{true}(t) + \mathbf{b}_g(t) + \mathbf{d}(t) + \mathbf{n}_{wn}(t) \\ \dot{\mathbf{b}}_g(t) &= -\alpha \mathbf{b}_g(t) + \mathbf{n}_{rw}(t) \\ \dot{\mathbf{d}}(t) &= 0 \\ \dot{\mathbf{k}}(t) &= 0 \end{aligned} \tag{C.3}$$

for the gyroscope and

$$\begin{aligned} \mathbf{a}_m(t) &= (1+\mathbf{k})\mathbf{a}_{true}(t) + \mathbf{b}_a(t) + \mathbf{d}(t) + \mathbf{n}_{wn}(t) \\ \dot{\mathbf{b}}_a(t) &= -\alpha \mathbf{b}_a(t) + \mathbf{n}_{rw}(t) \\ \dot{\mathbf{d}}(t) &= 0 \\ \dot{\mathbf{k}(t)} &= 0 \end{aligned} \tag{C.4}$$

for the accelerometer. Here $\mathbf{k}$ is the scale factor (see sec. C.1.3) and $\mathbf{d}$ is the bias repeatability.

For the derivation of these noise models, see the appendix of [2].

C.1. Types of IMU Noise

C.1.1. White Noise

White noise, denoted here as $\mathbf{n}_{wn}(t)$, is a standard Gaussian noise applied to both the gyroscope and accelerometer signal. It is sometimes referred to as random noise, rate noise, angle or velocity random walk (for gyroscope and accelerometer respectively), noise density and band noise. In fig. C.1, this is labeled as “white

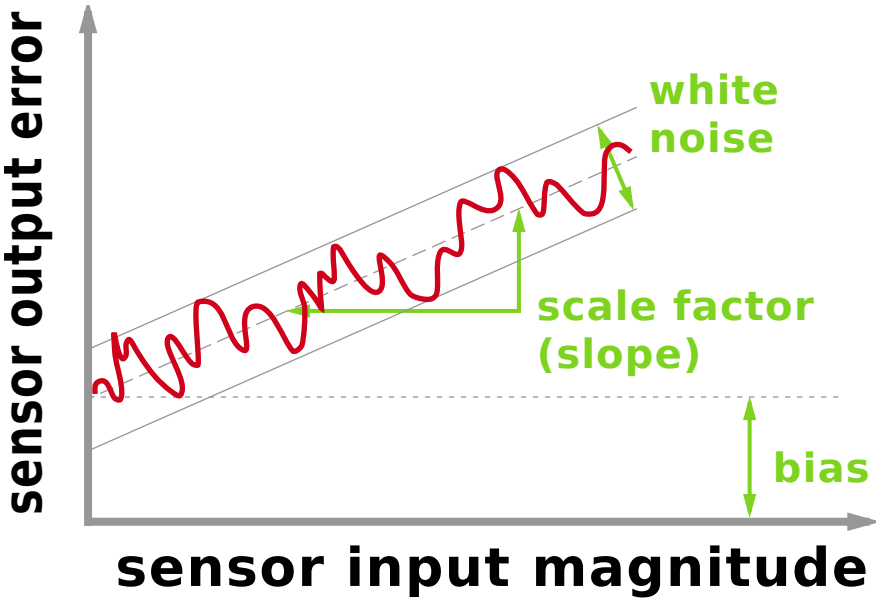


Figure C.1.: IMU noises for a single timestep. This shows bias as a constant value, although in realty bias is time varying, and therefore this is an instantaneous snapshot of the sensor (accelerometer or gyroscope). The x-axis represents the input of the true value of the measurement, and the y-axis is any error added to the true output.

noise".

C.1.2. Flicker Noise / Bias Walk

Flicker noise or bias random walk is the bias stability measure. It directly determines how fast the bias ($\mathbf{n}_{rw}(t)$) will change over time. It is sometimes referred to as bias stability, bias instability, bias variation, random walk, or Brownian motion. Because fig. C.1 is a single snapshot in history, flicker noise is not depicted in the graphic.

C.1.3. Scale Factor

Scale factor, denoted here as **k** is a multiplier of the input. In other words, the input will be proportionally scaled to some output, determined by the value of **k**. Most often scale factor is measured in ppm (parts per million), and therefore a scale factor of say 30000 would mean an input of 10 units becomes 10.3 units (3%). fig. C.1 shows scale factor as the slope.

Bibliography

[1] F. B. Berger, "Aircraft navigation system," U.S. Patent US2 847 855 A, Aug. 19, 1958. [Online]. Available: http://www.google.com/patents/US2847855

[2] J. Crassidis, "Sigma-point kalman filtering for integrated GPS and inertial navigation," in *AIAA Guidance, Navigation, and Control Conference*, San Francisco, CA, Aug. 2005.

[3] M. U. de Haag, A. Vadlamani, J. L. Campbell, and J. Dickman, "Application of laser range scanner based terrain referenced navigation systems for aircraft guidance," *Electronic Design, Test, and Applications, Proceedings of the Third IEEE International Workshop*, 2006.

[4] O. Eroglu and G. Yilmaz, "A terrain referenced navigation UAV localization algorithm using binary search method," in *Journal of Intelligent and Robotic Systems*, vol. 73, Jan. 2014, pp. 309–323.

[5] J. Farrell, *Aided navigation: GPS with high rate sensors*, ser. Electronic Engineering. McGraw-Hill New York, 2008.

[6] J. P. Golden, "Terrain contour matching (tercom): A cruise missile guidance aid," in *The Intl. Society for Optics and Photonics*, Dec. 1980.

[7] G. Hemann, S. Singh, and M. Kaess, "Long-range GPS-denied aerial inertial navigation with LIDAR localization," 2016, under submission.

[8] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Transactions on Robotics*, vol. 30, pp. 158–176, Jan. 2014.

[9] F. Hover, R. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. Leonard, "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *Intl. Journal of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, Oct. 2012.

[10] G. Huang, M. Kaess, and J. Leonard, "Towards consistent visual-inertial navigation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Hong Kong, Jun. 2014, pp. 4926–4933.

[11] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Journal of Robotics and Autonomous Systems, RAS*, vol. 61, no. 8, pp. 721–738, Aug. 2013.

[12] A. Johnson and T. Ivanov, "Analysis and testing of a LIDAR-based approach to terrain relative navigation for precise lunar landing," in *AIAA Guidance, Navigation, and Control Conference*, Portland, Oregon, Aug. 2011.

[13] A. Johnson and J. Montgomery, "Overview of terrain relative navigation approaches for precise lunar landing," in *IEEE Aerospace Conference*, 2008, pp. 1–10.

[14] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *The Intl. Society for Optics and Photonics*, Jul. 1997.

[15] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. on Robotics (TRO)*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[16] J. Lewis, "Fast template matching," in *Vision Interface*, Quebec City, Canada, May 1995.

[17] A. Mourikis and S. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2007, pp. 3565–3572.

[18] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *IEEE Transactions on Robotics*, vol. 25, pp. 264–280, Apr. 2009.

[19] S. Patil, J. S. Nadar, J. Gada, S. Motghare, and S. S. Nair, "Comparison of various stereo vision cost aggregation methods," in *International Journal of Engineering and Innovative Technology*, vol. 2, Feb. 2013.

[20] E. Quist, "UAV navigation and radar odometry," Ph.D. dissertation, Bringham Young University, 2015.

[21] F. Rabiei and F. Ismail, "Fifth-order improved Runge Kutta method with reduced number of function evaluations," in *Australian Journal of Basic and Applied Sciences*, 2012, pp. 97–105.

[22] F. Riedel, S. Hall, J. Barton, J. Christ, B. Funk, T. Milnes, P. Neperud, and D. Stark, "Guidance and navigation in the global engagement department," Johns Hopkins APL Technical Digest, Tech. Rep. 2, 2010.

[23] S. Roumeliotis, G. Sukhatme, and G. Bekey, "Circumventing dynamic modeling: Evaluation of the error-state Kalman filter applied to mobile robot localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, Jan. 1999.

[24] A. R. Runnalls, P. D. Groves, and R. J. Handley, "Terrain-referenced navigation using the IGMAP data fusion algorithm," in *ION Annual Meeting*, Jun. 2005.

[25] A. Segal, D. Hahnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems (RSS)*, Jun. 2009.

[26] M. D. Shuster, "A survey of attitude representations," in *Journal of the Astronautical Sciences*, vol. 41, no. 4, oct-dec 1993, pp. 439–517.

[27] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, 2006.

[28] S. Thrun, "Particle filters in robotics," in *in Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[29] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," Department of Computing Science and Engineering, University of Minnesota, Tech. Rep. 2005-002, Jan. 2005.

[30] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery, "Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks: Research articles," *Journal of Field Robotics*, vol. 24, no. 5, pp. 357–378, May 2007.

[31] D. Vaman, "A GPS inspired terrain referenced navigation algorithm," Ph.D. dissertation, TU Delft, Nov. 2014.

[32] M. Warren, P. Corke, and B. Upcroft, "Long-range stereo visual odometry for extended altitude flight of unmanned aerial vehicles," *Intl. Journal of Robotics Research*, 2015.

[33] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: A compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, Sep. 2013.

[34] J. Wendel, C. Schlaile, and G. Trommer, "Direct Kalman filtering of GPS/INS for aerospace applications," in *International Symposium on Kinematic System in Geodesy, Geomatics, and Navigation*, 2001.

[35] J. Zhang and S. Singh, "Visual-inertial combined odometry system for aerial vehicles," *Journal of Field Robotics*, vol. 32, no. 8, pp. 1043–1055, Dec. 2015.

Nomenclature

Abbreviations

DEM	digital elevation model
DOF	degree(s) of freedom
DSM	digital surface model
DTM	digital terrain model
ECEF	earth-centered, earth-fixed
EKF	extended Kalman filter
ENU	east-north-up
ESKF	error-state Kalman filter
GPS	global positioning system
ICP	iterative closest point
IMU	inertial measurement unit
INS	inertial navigation system

LIDAR	light detection and ranging, light radar
MEMS	micro-electro-mechanical system
MSCKF	multi-state constraint Kalman filter
NCC	normalized cross correlation
NED	north-east-down
RK4	Runge Kutta 4th order
SAD	sum of absolute difference
SINS	strapdown inertial navigation system
SLAM	simultaneous localization and mapping
SSD	sum of squared difference
TAN	terrain aided navigation
TRN	terrain referenced navigation
UAV	unmanned air vehicle
UKF	unscented Kalman filter
USGS	United States Geological Survey
UTM	universal transverse mercator
VIO	visual inertial odometry
VO	visual odometry

WGS84	World Geodetic System 1984

Mathematical Symbols

$\hat{\mathbf{x}}_k$	vehicle state estimate at timestep k
$\tilde{\mathbf{x}}_k$	error-state at timestep k
$\tilde{\mathbf{z}}_k$	measurement residual at timestep k
$\mathbf{B}_k$	discrete-time control input model
$\mathbf{F}_k$	discrete-time state transition matrix
$\mathbf{G}_k$	discrete-time noise transition matrix
$\mathbf{H}_k$	discrete-time measurement model
$\mathbf{I}_D$	DEM image matrix
$\mathbf{I}_L$	LIDAR image matrix
$\mathbf{I}_v$	binary mask image matrix
$\mathbf{K}_k$	discrete-time Kalman gain
$\mathbf{P}_k$	discrete-time state covariance
$\mathbf{Q}_k$	discrete-time process noise covariance
$\mathbf{R}_k$	discrete-time measurement noise covariance
$\mathbf{v}_k$	measurement noise (zero-mean Gaussian white noise)
$\mathbf{w}_k$	process noise (zero-mean Gaussian white noise)

$\mathbf{x}_k$	vehicle state at timestep k
$\mathbf{z}_k$	measurement vector at timestep k