The Pennsylvania State University

The Graduate School

Department of Aerospace Engineering

LANDMARK-AIDED LOCALIZATION FOR AIR VEHICLES USING LEARNED OBJECT DETECTORS

A Dissertation in

Aerospace Engineering

by

Mark Patrick DeAngelo

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2016

The dissertation of Mark Patrick DeAngelo was reviewed and approved* by the following:

Joseph F. Horn
Professor of Aerospace Engineering
Dissertation Advisor
Chair of Committee

Robert G. Melton
Professor of Aerospace Engineering and Director of Undergraduate Studies

Sean N. Brennan
Associate Professor of Mechanical Engineering

Richard Lee Culver
Senior Research Associate/Associate Professor of Acoustics

Philip J. Morris
Professor and Interim Head of Aerospace Engineering

*Signatures are on file in the Graduate School

ABSTRACT

This research presents two methods to localize an aircraft without GPS using fixed landmarks observed from an optical sensor. Onboard absolute localization is useful for vehicle navigation free from an external network. The objective is to achieve practical navigation performance using available autopilot hardware and a downward pointing camera. The first method uses computer vision cascade object detectors, which are trained to detect predetermined, distinct landmarks prior to a flight. The first method also concurrently explores aircraft localization using roads between landmark updates. During a flight, the aircraft navigates with attitude, heading, airspeed, and altitude measurements and obtains measurement updates when landmarks are detected. The sensor measurements and landmark coordinates extracted from the aircraft's camera images are combined into an unscented Kalman filter to obtain an estimate of the aircraft's position and wind velocities. The second method uses computer vision object detectors to detect abundant generic landmarks referred as buildings, fields, trees, and road intersections from aerial perspectives. Various landmark attributes and spatial relationships to other landmarks are used to help associate observed landmarks with reference landmarks. The computer vision algorithms automatically extract reference landmarks from maps, which are processed offline before a flight. During a flight, the aircraft navigates with attitude, heading, airspeed, and altitude measurements and obtains measurement corrections by processing aerial photos with similar generic landmark detection techniques. The method also combines sensor measurements and landmark coordinates into an unscented Kalman filter to obtain an estimate of the aircraft's position and wind velocities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Nomenclature

$\vec{\mathbf{L}}_i^C$ = landmark location vector in camera coordinate frame

$\vec{\mathbf{L}}_i^W$ = landmark location vector in world coordinate frame

$\mathbf{P}$ = estimator covariance matrix

$\mathbf{Q}$ = process noise covariance matrix

$\mathbf{R}$ = measurement noise covariance matrix

$\mathbf{T}^A$ = translation matrix that shifts the origin from the center to upper left corner in the image frame

$\mathbf{T}^P$ = perspective transformation matrix

$\mathbf{T}^R$ = camera rotation matrix

$\mathbf{T}^\phi$ = aircraft body rotation matrix about roll axis

$\mathbf{T}^\theta$ = aircraft body rotation matrix about pitch axis

$\mathbf{T}^\psi$ = aircraft body rotation matrix about yaw axis

$\mathbf{T}^O$ = aircraft translation matrix

g = standard acceleration due to gravity

$\mathbf{u}$ = input vector

u = longitudinal velocity component in aircraft body coordinates

h = altitude above ground level

r = range distance from estimated aircraft location to landmark location

t = time

V = airspeed

V_x = velocity component in x-direction (east-west) measured by VICON motion capture system

V_y = velocity component in *y*-direction (north-south) measured by VICON motion capture system

v = lateral velocity component in aircraft body coordinates

W_w = wind speed from west to east

$\hat{W}_w$ = estimated wind speed from west to east

W_n = wind speed from north to south

$\hat{W}_n$ = estimated wind speed from north to south

w_x = process noise on *x* position

w_y = process noise on *y* position

w_w = process noise on W_w wind component

w_n = process noise on W_n wind component

$\mathbf{x}$ = state vector containing *x*, *y*, W_w, W_n

$\hat{\mathbf{x}}$ = state vector containing $\hat{x}$, $\hat{y}$, $\hat{W}_w$, and $\hat{W}_n$ estimated by the UKF

x = horizontal local coordinate in map

$\hat{x}$ = estimated horizontal local coordinate in map

y = vertical local coordinate in map

$\hat{y}$ = estimated vertical local coordinate in map

ϕ = roll angle

λ = camera focal length

θ = pitch angle

ψ = yaw angle

ACKNOWLEDGEMENTS

God has blessed me with wonderful parents, and I thank them for their encouragement to continue my education.

I sincerely thank the Penn State Applied Research Lab (PSU/ARL) for supporting me as I pursue this research. The Lab has given me the freedom and flexibility to combine my interests and think creatively. As a result, this dissertation combines practical elements of aviation and photography, which are two of my most passionate subjects. Dr. Joseph Horn and Mark Rothgeb have offered valuable guidance and feedback in their supervisory roles. Dr. Robert Collins has introduced me to the computer vision object classification papers, which became an important part of this work.

Next, I thank Yande Liu (Armstrong) for his help with the hardware demonstration. His tireless work in preparing both the motion capture lab and the quadcopter for flight data acquisition has saved me a tremendous amount of time.

Finally, I extend my thanks to the staff at SAE International for their encouragement and support. Their flexibility has given me the opportunity to be a part of their team while I pursue my graduate education.

Chapter 1

Introduction

Unmanned vehicles have operated for a span exceeding 75 years and the earliest began as remote controlled aircraft and later served the purposes of recreation, defense, or research. Over time with the advancement of electronics, materials, and computing hardware, the capabilities of those vehicles have increased. Then, with the aid of the global positioning system (GPS), inertial navigation, and programmable autopilots, unmanned vehicles became more autonomous such that precise navigation is now achievable, sparking greater awareness globally. However, vehicles may operate in areas where GPS is unreliable, denied, or nonexistent. For example, autonomous underwater vehicles (AUV) have their own set of challenges and inherently operate where GPS is nonexistent. Solutions such as resurfacing to obtain GPS fixes or remaining tethered to a surface vehicle severely limit (and compromise) the underwater vehicle's autonomy, stealth, and range. Leonard et al summarized the state-of-the art methods of AUV navigation, and much of it remains relevant today [1]. The paper stated, "*Good navigation information is essential for safe operation and recovery of an AUV. For the data gathered by an AUV to be of value, the location from which the data has been acquired must be accurately known*." In 2004, the Unmanned Undersea Vehicle Master Plan identified nine capabilities and expanded on six recommendations. Intelligence, surveillance, and reconnaissance was the first priority and inspection/identification was the fourth priority on the list [2]. An updated Master Plan was written in 2011 but remained classified.

In 2009, Rand Corporation published *A Survey of Missions for UUVs*, which was sponsored by the U.S. Navy. The survey comprehensively discusses both missions outlined in the 2004 Master Plan but also evaluates other missions and technologies [3].

According to the Federal Aviation Administration (FAA), approximately 50 companies, universities, and government organizations are developing and producing some 155 unmanned aircraft designs in the United States alone [4]. As of 2016, UAVGlobal.com lists 92 manufacturers in the United States and 354 manufacturers elsewhere [5].

In [6], the United States Department of Defense summarized the potential of unmanned systems with the following:

Unmanned aircraft will not achieve their full potential military utility to do what manned aircraft do unless they can go where manned aircraft go with the same freedom of navigation, responsiveness, and flexibility. Unmanned systems are ideally suited for many protection tasks that are deemed dull, dangerous or dirty. As the future enables greater automation with respect to both navigation and manipulation, unmanned systems will be able to perform tasks such as firefighting, decontamination, forward operating base security, installation security, obstacle construction and breaching, vehicle and personnel search and inspection, mine clearance and neutralization, sophisticated explosive ordnance disposal, casualty extraction and evacuation, and maritime interdiction.

A mission that can be deemed repetitive, dangerous, and dirty became very apparent when observing an urgent fire-fighting operation near the Pacific Ocean on May 14, 2014. The photograph in Figure 1-1 shows an Orange County Fire Authority helicopter dumping ocean water on a wildfire along Interstate 5. A swarm of UAVs could have appropriately performed this mission of low altitude, repetitive trips to/from the ocean.

Figure 1-1. An Orange County Fire Authority helicopter dumps nearby ocean water on a wildfire along Interstate 5 on May 14, 2014.

Both unmanned aerial vehicles (UAV) and Unmanned Underwater Vehicles (UUVs) will benefit from a reliable, drift-free navigation system independent of an external network of GPS satellites or acoustic beacons, ultimately leading to truer autonomous capabilities. A self-reliant vehicle with low cost sensors and computationally efficient algorithms capable of reliable localization is a desired outcome of this research.

Possible solutions to the localization problem involve an intuitive approach that models the way human pilots have visually navigated since the first flight until today. Human pilots flying under visual flight rules (VFR) observe their surroundings and compare landmarks (both natural and artificial) to the landmarks printed in maps (sample depicted in Figure 1-2) along their pre-planned route. In some cases, the landmarks are distinct such that they are unique to the local geographical region. In other cases, the landmarks are generic such as urban regions (printed as solid yellow regions on VFR sectional aeronautical charts), major roads, and road intersections. Under the conditions of VFR navigation, pilots estimate their approximate location based on their perspective relative to the landmarks they perceive. Their expectation is not to determine exact coordinates as precise as GPS, but rather sufficiently approximate their location to ensure they are

on course and have enough fuel to reach their destination (adverse headwinds are a concern to available fuel calculations).

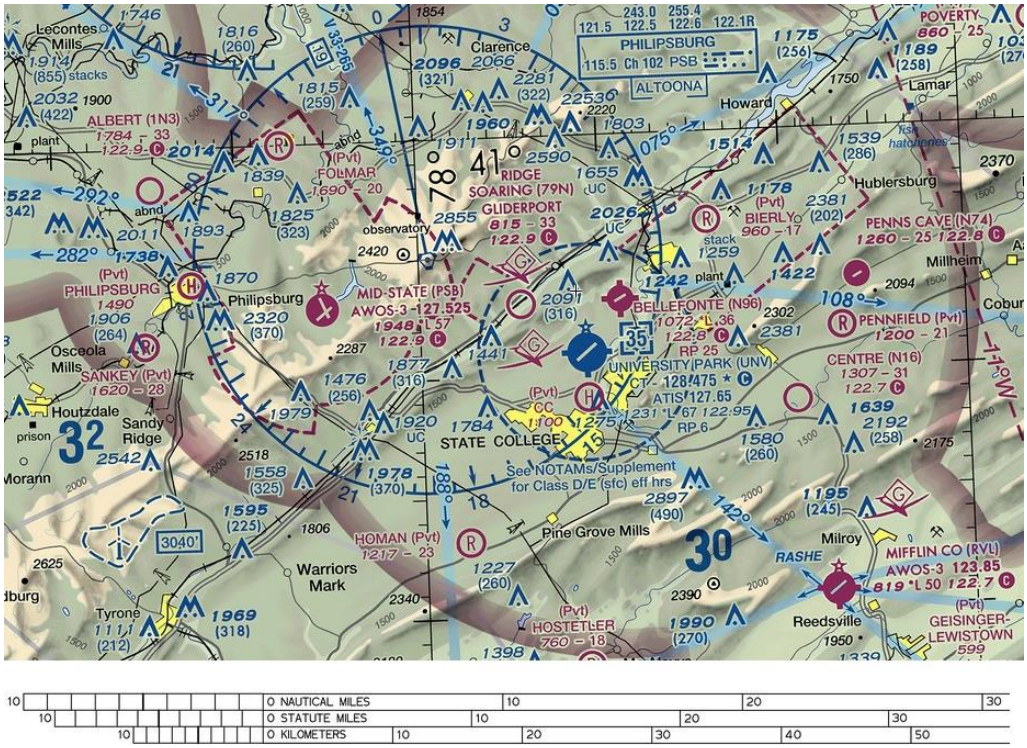


Figure 1-2. Sample depiction of an FAA sectional aeronautical chart commonly used for VFR navigation.

In this research, the concept of visual navigation is automated by combining information obtained from a downward-pointing optical sensor (a digital camera), vehicle orientation sensors, an altimetry sensor, an airspeed sensor, preprocessed digital maps, and appropriate processing algorithms programmed on an onboard computer. An important focus of this research is the development of algorithms that fuse the sensor data to estimate the coordinates of the vehicle mathematically. Furthermore, the automatic detection and classification of landmarks is easily verifiable by humans upon visual inspection. As a result, two unique localization methods are presented. The first method scans aerial photographs for distinct landmarks, and the second method scans aerial photographs for generic landmarks such as buildings, fields, trees, and road intersections. Simulations using photorealistic scenery and actual hardware-in-the-loop autopilot

sensor telemetry test both methods. Finally, the second method is evaluated with post-processed flight data obtained from an actual flight test in an indoor laboratory with a quadcopter that flies over an artificial environment.

Past Research

UAV Navigation and Mapping

An incredible amount of research has been performed in the area of navigation and mapping using vision sensors on UAVs. Researchers have used feature points (such as corners and other feature descriptors) obtained in optical images [7], [8], [9]. Feature points are usually more abundant and are extracted from local pixels. However, feature points are sensitive to image noise and distortion, leading to difficulties tracking them from frame to frame. As an alternative, some researchers have explored methods that take advantage of complex shapes in the environment. For example, [10] showed a method that identified building facades to navigate in urban environments. A user could take a picture of a building and then the algorithm would determine the pose and compare it against a database of building facades despite changes in viewpoint and lighting.

Other researchers have investigated simultaneous localization and mapping (SLAM) using vision data. In his dissertation, Wu [11] identified past research relating to aerial SLAM such as [12], [13], [14], [15], [16], [17]. More recently, Kümmerle showed that instead of learning maps from scratch in a typical SLAM problem, their algorithm uses publicly available aerial photographs as prior information [18].

Researchers such as [19], [20], [21] and [22] presented methods for landing small helicopters using vision. [19] and [21] demonstrated landing on a helipad of a known shape, [20]

demonstrated landing on unknown terrain, and [22] demonstrated landing on flat terrain using only optical flow and a barometric pressure sensor. When a monocular camera is used, range information is lost in the conversion from the 3D world to the 2D image plane, but known coordinates and target size simplify the relative positioning. Langelaan estimated vehicle state (position, orientation and velocity) as well as the position of obstacles using only inertial measurements and a monocular camera [23]. The images were used to provide bearings to nearby obstacles and landmarks, enabling obstacle avoidance and navigation through an open forest. Celik, et al used line perspectives and optical flow techniques for the purposes of navigating a small helicopter inside an office [24].

The following UAV-related papers will be discussed in greater detail because they relate closely to the objective of this research. In 2007, Madison et al presented a solution to detect, track, and geolocate visual landmarks with a single camera while GPS initially provided primary navigation [25]. Then, GPS was disabled and the small UAV combined estimated landmark observations with new observations to both navigate and geolocate new landmarks. During the GPS outage, new features were selected and filtered in a bootstrapped manner. The authors did not specify the actual types of features used. An extended Kalman filter was used to obtain the navigation solution. Results showed that vision-aided navigation maintained knowledge of vehicle position and orientation better than an inertial measurement unit (IMU) alone, and the accumulation of drift was significantly slower.

Figure 1-3 shows a block diagram of the vision-aided navigator architecture. Line of sight (LOS) measurements were extracted from camera images and transformed into Earth-centered-Earth-fixed (ECEF) coordinates. Then, the LOS measurements were augmented into a navigation filter to compute not only vehicle position, velocity, and attitude, but also 3D locations of landmarks. The navigation filter was initialized with GPS information, which then helped compute the initial landmark locations. More specifically, the navigation filter state vector was

updated with an extended Kalman filter and contained the vehicle ECEF position and velocity errors, body-referenced attitude misalignment vector, instrument calibration errors, and ECEF position errors of various landmarks currently being tracked. The authors stated that the use of error dynamics is more robust than estimating the full navigation state and thereby allows the error estimates to be reset to zero, keeping the linearized state equations closer to their operating region.

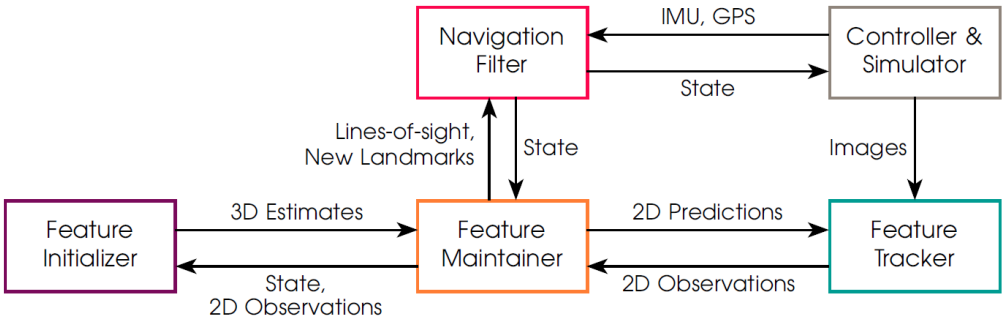


Figure 1-3. Vision-aided navigator architecture. Figure reproduced from Figure 1 in [25].

The *feature tracker* performed the image processing task of identifying and tracking 2D features within the images. The authors elected to use the Lucas-Kanade feature tracking method. If a feature is no longer visible in the image, then the feature tracker resets to estimate a different landmark. The feature tracker rejects a feature when it moves near the edge of the image or when there is significant mismatch.

The *feature initializer* was fed vehicle pose and 2D location of features in images. The output was 3D estimates of landmark locations, which are then used by the navigation filter. The lack of range information poses a challenge for obtaining a 3D position from a 2D feature location, and has led the authors to explore motion stereo.

As its name suggested, the *feature maintainer* managed the input and output of information supplied to the feature initializer, the navigation filter, and the feature tracker. When landmarks are no longer visible, the feature maintainer calls the feature tracker to select, track,

and replenish a pool of features. Then it tells the feature initializer to obtain 3D positions of the features, and finally sends high confidence 3D positions as measurements to the navigation filter.

Flight data was conducted in a controlled indoor lab with a quad-rotor UAV with a forward-looking camera. Flight trajectory composed of takeoff, small translational motion in two directions and landing. Figure 1-4 shows a performance comparison between inertial navigation and vision-aided navigation, which mitigates drift. The research, however, relied on GPS to first geolocate arbitrary features and did not run in real time.

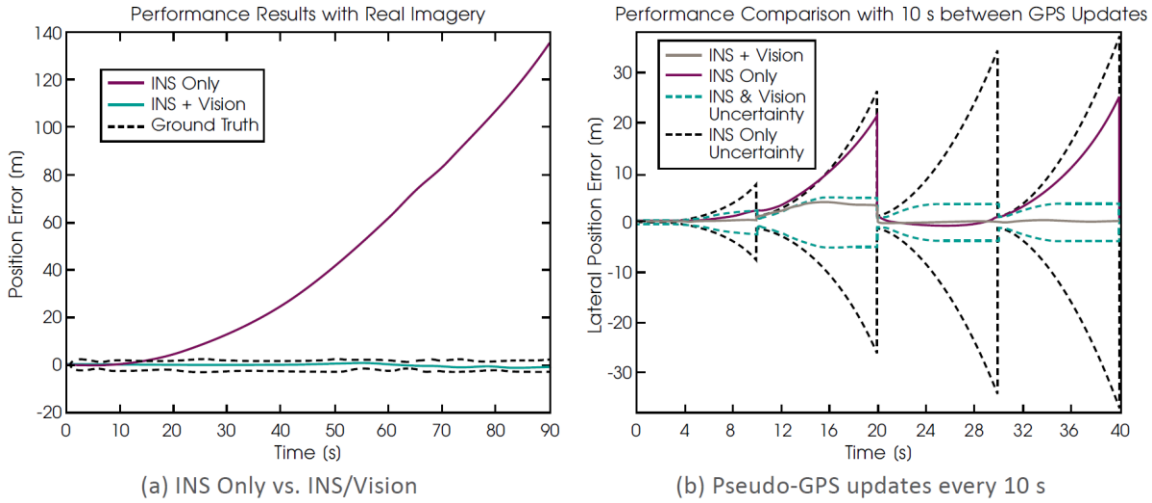


Figure 1-4. Flight data perfomance comparison. Figure reproduced from Figure 7 in [25].

Kümmerle presented a technique that uses aerial imagery (global priors) to provide more accurate SLAM solutions by limiting the error when visiting unknown locations [18]. The method works for mixed indoor/outdoor operations and preserves traditional SLAM in that it can still be used in the absence of any prior information. Thus it is an extension of the traditional SLAM problem. When incorporating prior information given from aerial images, a Monte-Carlo localization algorithm was combined with a novel sensor model that matched 3D laser range scans to the aerial images. Canny edge detection was used to extract edges in 2D images because edges in urban environments associate with rooftops. Then, a subset of points from a 3D scan was

projected onto a 2D ground plane and compared to the Canny edges. The authors also discussed the use of stereo images instead of 3D laser range data and discussed techniques for reducing false positives. Experimental results indicated that the stereo cameras outperform the laser range finder especially in vegetated areas where 3D structures are lacking. Both sensors outperformed GPS measurements. Figure 1-5 shows results comparing the standard SLAM approach to the global priors approach. Figure 1-6 shows results comparing both 3D laser scans and stereo vision data.

Figure 1-5. The robot trajectory is overlaid on a Google Earth image used as prior information. The standard SLAM approach is shown in yellow. The global priors approach is shown in red and accurately aligns with the aerial image. Figure reproduced from Fig. 7 in [18].

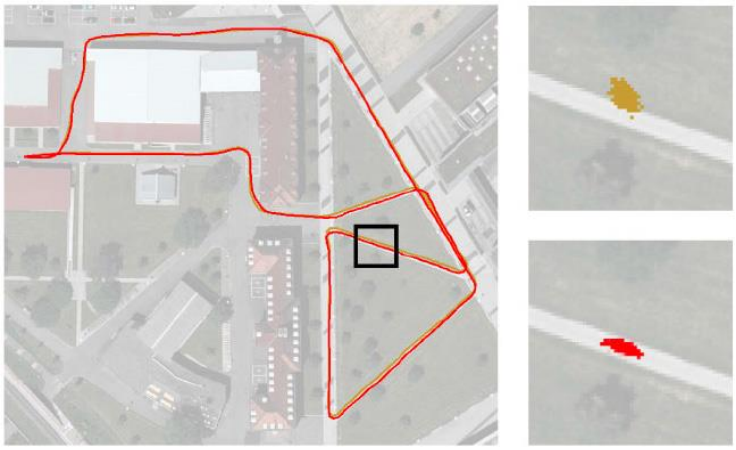


Figure 1-6. 3D vision results are shown in yellow. Stereo vision results are shown in red. The right column shows a magnified view of the black rectangle and shows the particle cloud for Monte Carlo Localization using 3D laser scans (top) and stereo data (bottom). Figure reproduced from Fig. 9 in [18].

In 2007, Dogruer decoupled the localization and mapping problem and focused on the localization problem over a pre-constructed map such as freely available satellite imagery (e.g. Google Earth) [26]. Unlike SLAM, maps of unexplored regions were available to the robot. The authors noted that SLAM algorithms are "not practical in large-scale environments". The robot was assumed to identify the city through a wireless LAN and download free satellite images for that city. Those images were used to create digital maps so that the robot can both localize itself and plan future actions. The bulk of the paper discussed methods to classify and segment images, and then proposed a technique using Fuzzy C-Means (FCM) and Adaptive Neuro-Fuzzy Inference System (ANFIS) methods on satellite images. Figure 1-7 shows the results of a satellite image segmented into four regions: buildings (black), roads (white), forest (light gray), and ground (dark gray).

Table 1 displays the accuracy of the classification. The authors considered the classification a success but noted that improvements were needed. Misclassifications were explained by some objects sharing similar colors.

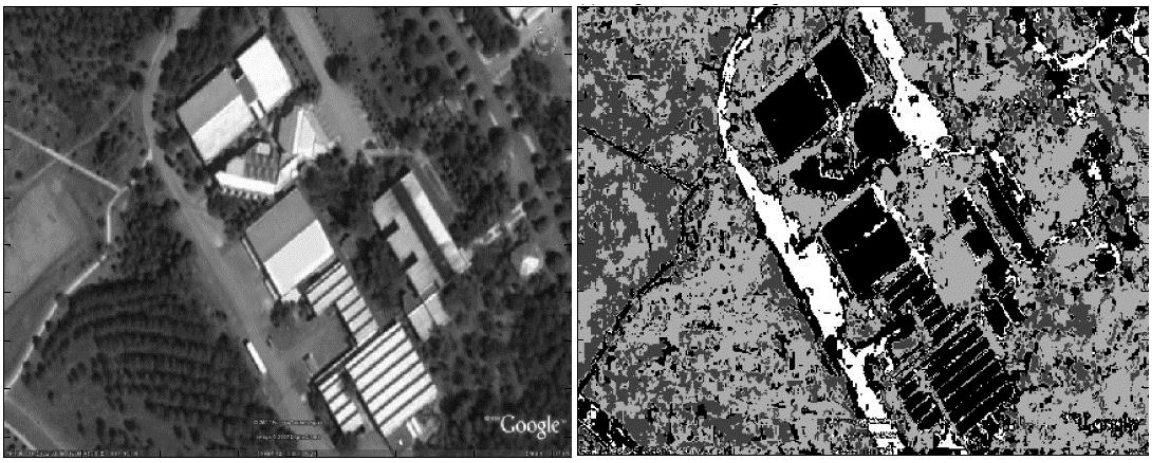


Figure 1-7. (left) Original satellite image. (right) Segmented satellite image. Figure reproduced from Fig. 5 in [26].

Table 1. Classification error ratios comparing computer and human interpretations of pixels. Table reproduced from [26].

	Ground Truth Classes (human interpretation)				
Thematic Map Classes		Building	Road	Ground	Forest
	Building	0.9746	0.2819	0.1669	0.1287
	Road	0.0211	0.7012	0.0244	0.0027
	Ground	0.0002	0.0046	0.4651	0.1172
	Forest	0.0041	0.0123	0.3436	0.7514
	# pixels	14717	17020	38448	63884

A year later, the same authors used those maps to localize a robot by applying the Monte Carlo Localization (MCL) technique, which was appropriate for large scale environments [27]. Attached to the robot, a laser scanner accumulated local range data, which was then compared to the maps. The authors chose the MCL technique instead of Markov localization or Extended Kalman Filter (EKF) based solutions because MCL was computationally faster than Markov localization and its statistics are not limited to unimodal distributions like the EKF. The MCL employs both a motion model and a measurement model. The motion model predicts how the states (Cartesian position and heading) change due to the robot's actions, and the measurement model relates the likelihood of measurements to the states. The initial probability density function

is assumed to be known beforehand. Experimental results showed that laser range scans were more precise in detecting buildings and less precise in detecting vegetation, trees, and bushes so additional filtering algorithms were applied. Since the robot traveled only on roads, however, the algorithm distributed particles on roads only as seen in Figure 1-8.

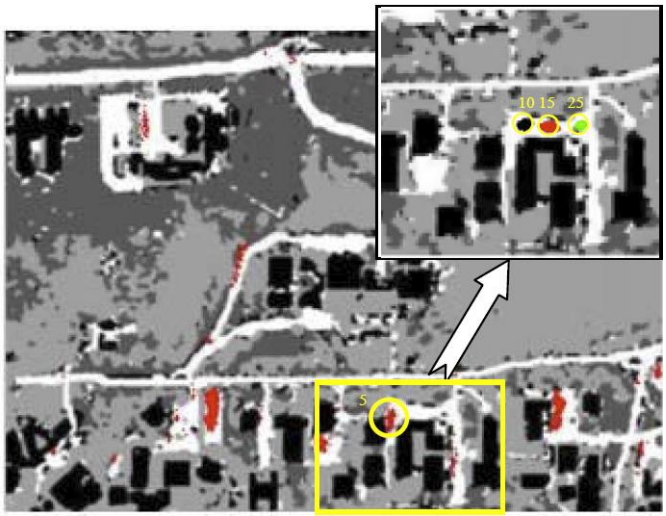


Figure 1-8. The particle distribution is shown by red dots. Figure reproduced from Fig. 3 in [27].

In 2014, Viswanathan et al presented the difficult problem of matching images captured from a ground vehicle to aerial imagery—even with drastic differences in perspectives—in a GPS-denied environment. The ground vehicle images were warped (Figure 1-9) to obtain a birds-eye view of the ground and compared to a grid of satellite locations. Ground-air matching is conducted within a particle-filter framework. The authors stated that the method can improve the location estimates of Google Street View [28].

Figure 1-9. Panoramic imagery is warped for comparision to sattelite imagery. A particle filter is used to localize the unmanned ground vehicle. Image reproduced from Fig. 1 in [28].

In 2016, van Dalen, et al presented a method for aircraft localization using image alignment and particle filtering. The method uses normalized cross-correlation (NCC) to compare aerial photographs to photographic maps of the region. The NCC results combined with particle filtering are used to provide absolute position updates to the SLAM-based navigation solution [29].

AUV Navigation and Mapping

Underwater vehicles inherently face localization challenges therefore it is worthwhile to research AUV literature. A concept of AUV navigation and mapping has been documented in the 1987 journal article, *Sonar-Based Real-World Mapping and Navigation* [30]. The intention was to develop an autonomous mobile robot operating in an unknown environment. Although the robot platform was a ground vehicle, the paper serves as a comprehensive introduction to the systems, sensors, and the general approach to mapping and navigation. The vehicle used a sonar array to gather range information about its environment and recorded multiple measurements to

cope with uncertainties and errors in the data. Multiple readings from a sonar sensor would then build a two-dimensional world model, which served as the basis for path planning, obstacle avoidance, landmark identification, position, and motion estimation. Sonar maps were represented by two-dimensional arrays with cells corresponding to a horizontal grid of space. The sonar beam probability profile was modeled to obtain a probability of empty and a probability of occupancy for each measurement. Next, the probabilities for multiple measurements were combined by superposition. Finally, the result was thresholded so each cell contained occupancy status using the following certainty convention:

Unknown	0
Empty	[-1, 0)
Occupied	(0, 1]

Figure 1-10 shows the results obtained by the method above. Each symbol represents $0.5 \times 0.5\, ft$ cells. High certainty empty space is represented by white space, lower certainty area is represented by + symbols. Occupied area is represented by $\times$ symbols and unknown area is represented by $\cdot$ symbols.

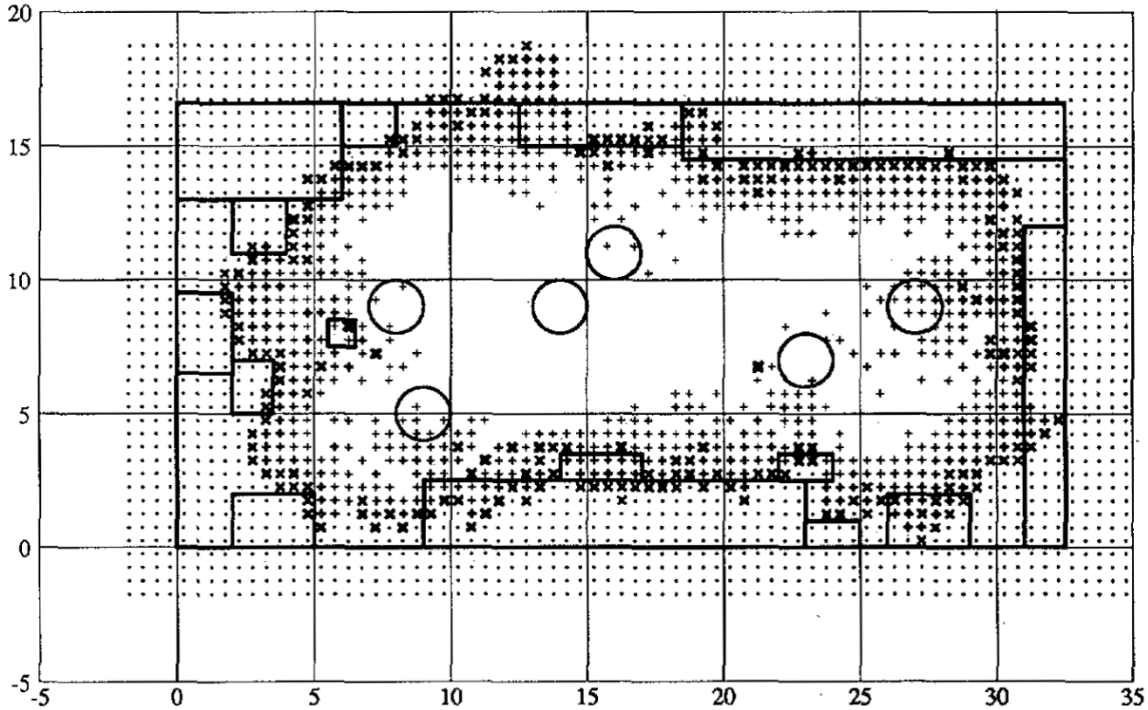


Figure 1-10. Two-dimensional sonar map after thresholding. Figure reproduced from Figure 8 in [30].

The paper also discussed map matching algorithms, which are useful for landmark recognition and for updating the robot's estimate of its position and orientation. One approach was to compute the sum of products of corresponding cells in the two maps. An occupied cell in the new map corresponding to an occupied cell in the old map contributed a positive increment to the sum (an empty cell corresponding to an empty cell also contributed a positive increment). Whereas an occupied cell corresponding to an empty cell contributed to a negative increment to the sum, and an unknown cell corresponding to an unknown cell neither increased nor decreased the sum. At that time, the approach was slow given the computational resources. Therefore the authors developed hierarchical maps with different layers of abstraction and resolution to speed up computation.

In the journal article by Lane et al, the objective was to identify different 3D objects (such as pier leg, anchor chain, jetty, and diver) based on 2D sector scan sonar imagery [31]. The method involved two main stages: 1) Perception—image processing for obtaining numerical

feature descriptors followed by their conversion to a high level abstraction that used qualitative representations, and 2) Classification—matching the observed features against *a priori* stored information. The qualitative representation of features used *linguistic variables* such as "size," "brightness," "elongation," and "variance." For example, the "size" variable can be assigned values of "small," "quite small," "midsize," or "big." The boundary between "midsize" and "big" was determined from all the observations in the image stored in rank order. The main advantage of the linguistic variables is that they *do not rely upon the specific operating conditions for the sonar*. The method followed the following steps:

1. Convolve images with a 3 by 3 or 5 by 5median filter to reduce noise.
2. Estimate background levels with a 21 by 21 median filter.
3. Segment image from 1) using gray level difference between pixel values in images from 1) and 2) to produce two binary images—one for object candidates and the other for shadow observations.
4. Apply erosion and dilation.
5. Assign bounding boxes to blobs.
6. Measure and record each blob's size, brightness, variance, and elongation to a feature vector.
7. Cluster the feature vectors using nearest neighbor cluster analysis by Euclidean distance.
8. Map the quantitative features to qualitative linguistic attributes.
9. Compare and classify the linguistic variables of the observations to the ones contained in a library of known objects.

As a result, the authors concluded that invariance between sonar parameters, environmental conditions and sonars was achievable by quantitative-to-qualitative feature mapping.

Classification, however, did not work well where distinct descriptions were not available. Therefore alternative feature measures were needed to provide the discrimination.

By 1998, sonar imaging for the purposes of detection, localization, identification or tracking of objects and targets of interest, has been studied [32, 33]. Lane, et al presented an approach to tracking multiple objects in sector-scan sonar image sequences using optical flow techniques. The experiment was to track two divers swimming near piers. The challenge was to detect and track the motion of the divers without confusing them with the static piers and noise. The algorithm can be summarized as follows:

1. Convolve images with a 5×5 median filter to reduce noise.
2. Convert images to the frequency domain using fast Fourier transform to distinguish between regions of moving and static objects.
3. Use low-pass and band-pass filters to obtain static and dynamic objects.
4. Segment significant objects with thresholding and apply the method of optical flow to the dynamic regions.
5. Average the optical flow vectors computed at pixels within the boundaries of the significant objects.
6. Employ tracking tree to record possible tracking solutions.
7. Return the best tracking tree solution.

For a sequence of images, the tracking tree used confidence values between predicted and measured optical flow results. The intention was to increase tracking robustness by deferring decision-making until more information is available. Tracking robustness is important because objects change appearance, merge, split, and maneuver in the noisy sector scan sonar images.

Cuschieri and Negahdaripour applied the technique of acoustic flow to as sequence of 50 image frames from a forward sonar as the vehicle traversed north to south over a sunken barge [34]. Acoustic flow is somewhat different than optical flow because acoustic flow uses range and

azimuth rates, dR/dt and $d\theta/dt$ obtained from sonar images $I(R,\theta,t)$ of the environmental features, whereas optical flow uses translational rates, dx/dt and dy/dt obtained from optical images $I(x,y,t)$. Furthermore, sonar imaging data is inherently much noisier than optical images despite pre-processing and filtering.

In 2004, Ruiz et al investigated concurrent mapping and localization (CML) using sidescan sonar [35]. The objective was to build a stochastic map of the environment and localize the AUV in absolute coordinates. The sonar was used to detect landmarks in the terrain and add each new landmark into a state vector for Kalman filtering. Reobservations of the landmarks provided state measurements and corrected drift, but only if the reobservations were perfectly associated with previous observations. The paper referenced other authors for details concerning data association, image segmentation, classification, and landmark extraction. It presented a simulation comparison between sidescan sonars and forward-looking sonars for CML. The forward-looking sonar performs 23 times more observations per landmark and is approximately two times better than using a sidescan sonar for estimating the vehicles position. Therefore the sidescan sonar sensors are not as useful for CML missions where only one pass of the sea floor is performed. However, for multi-pass missions, such as parallel and regularly spaced linear tracks, sidescan sonars are very useful when the parallel track spacing is less than the sonar's maximum range. Sidescan sonars have an advantage of higher quality images, which considerably help the data association task, than forward-looking sonars.

Next, the researchers used a Rauch-Tung-Striebel backward filter to smooth the output of the Kalman filter. Smoothing was required because observations correct drift and therefore create jerks in the estimation. The authors stated that jerks in the trajectory estimation "are unsuitable for some data-exploitation techniques, such as mosaicing. CML using sidescan sonar is less useful without an appropriate smoothing postprocess." Finally, the authors presented real-world

test results of AUV trajectories overlaying sidescan sonar imagery of the sea floor. Figure 1-11 displays images from a set of regularly spaced and parallel linear tracks of a REMUS AUV. The white lines indicate the track position of the AUV.

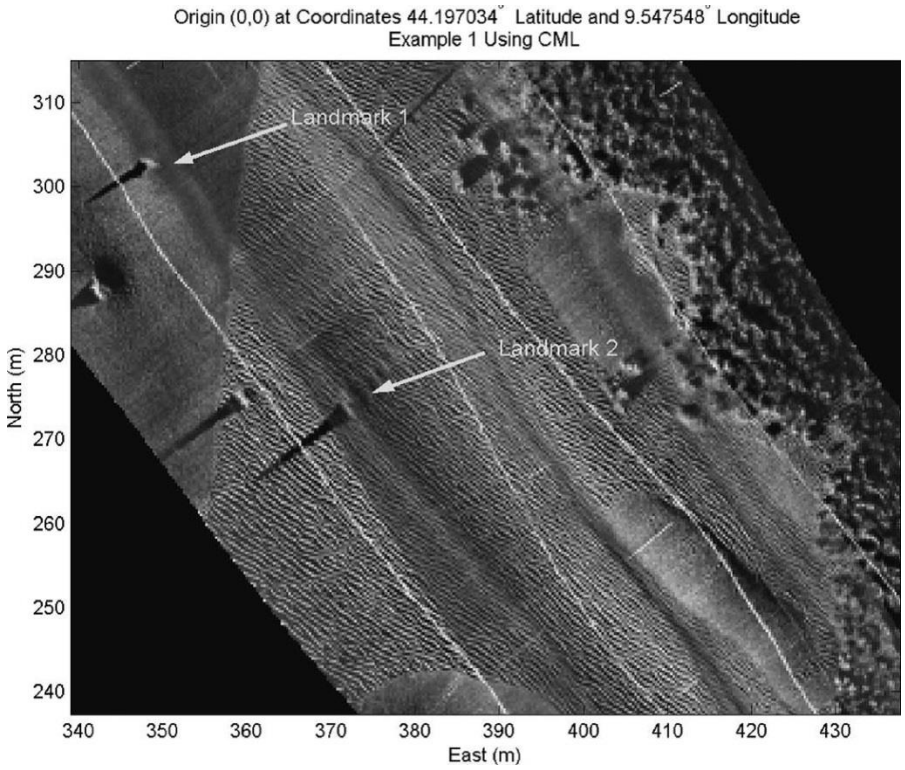


Figure 1-11. Sidescan sonar output using CML. Reproduced from Fig. 18 in [35].

In 2005, Zerr et al summarized a concept of scanning a seabed and storing its characteristics in a reference database. Each subsequent scan is used to detect changes in the environment for the purposes of homeland security.

For each new mission, the results are compared to a reference database to detect changes. Then, the database can be updated to include these changes. If the survey tasks are completed by AUVs, a copy of the database can be downloaded on-board and the navigation system can use this information to improve its accuracy. The AUV must also transform the sensor data into information suitable for a comparison with the database [36].

The paper mentioned symbol extraction for the various types of objects that can be encountered in the sea. These include individual objects, constellations of objects, large objects

(such as shipwrecks), homogeneous texture area, and geo-referenced sonar images. Next, the authors discussed a simple symbol matching technique that compared the Euclidean distance between their vector properties. Once successful matching results were achieved, the results can be included into the navigation filter. The authors referred to [35], which has extended the state filter of an extended Kalman filter with the fixes generated by the matches. However, the authors cautioned against sharp jerks in navigation, and instead proposed postponing the corrections for future legs.

That same year, Kerneis et al presented automatic seabed classification by fusing data from sidescan sonar images and digital elevation models (DEM) [37]. Although the main goal is somewhat unrelated to the others discussed above, it provided interesting insights and used classification tools that will be discussed later. Texture analysis was applied to the sonar image to obtain *N* texture feature image layers. Geomorphologic analysis was applied to the DEM to obtain *P* geomorphologic feature image layers. After analysis, each pixel location contains information represented by a vector, *X*, of *N+P* dimensions. A flowchart of this process is depicted in Figure 1-12.

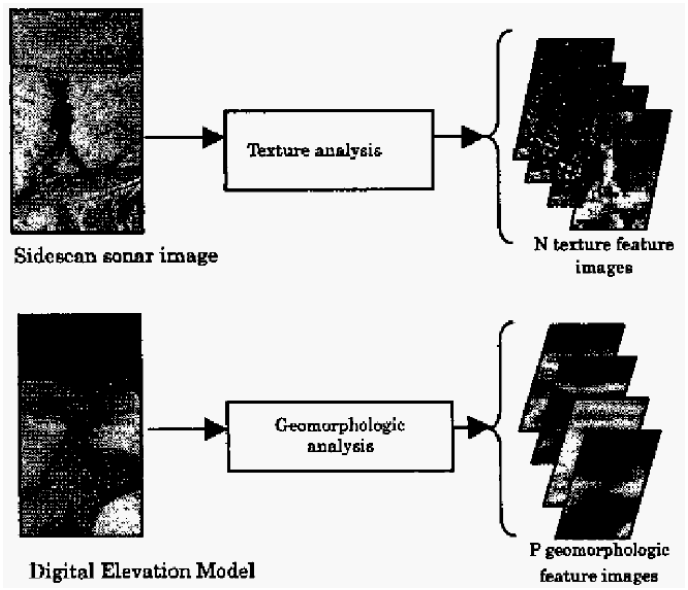


Figure 1-12. Feature extraction from a sidescan sonar image and a DEM. Figure reproduced from Fig. 1 in [37].

From this point, the authors presented three different fusion and classification methods. In the first method, the authors stated that information contained within each vector is “partially redundant or not useful” and they showed two ways to automatically select the best features and decrease computational load—1) Discriminant Analysis Feature Extraction (DAFE) and 2) Projection Pursuit. Both ways differ by the way they compute a matrix, A, which projects X to Y with the equation $Y=A^TX$. A reduction from 44 (30 texture features + 14 morphologic features) to 3 features were obtained. Three classes: shadow, sand and rock, were determined by Gaussian Maximum Likelihood Classification. Due to the linear projection, the data “has a greater tendency to be Gaussian”. The results indicate that the use of bathymetry improves the classification otherwise rocks were sometimes misclassified as sand due to similar micro texture.

In the second method, the authors used support vector machines (SVM) to classify the seabed. Since SVM is basically a two-class classifier but three classes (shadow, sand, rock) must be separated, a one vs one strategy was used. The final decision was the label that has been assigned most often. More information about SVM can be found in [38]. SVM works well with high-dimensional space, so there was no need to apply a dimension reduction technique.

In the third method, the authors used evidence theory. Dempster-Shafer theory allowed sensor data to be labeled with a compound class such as "sand or rock". This was useful because there were two types of sensor data: textural classes (from the sidescan sonar)—non-textured, micro-textured, and macro-textured, and geomorphological classes (from the DEM)—flat, smooth surface, steep surface. Therefore two symbol-level classified images were obtained from the sources. A look-up table then linked the intermediate classes to make a final decision—mud, sand, or rocks. The process is illustrated in Figure 1-13.

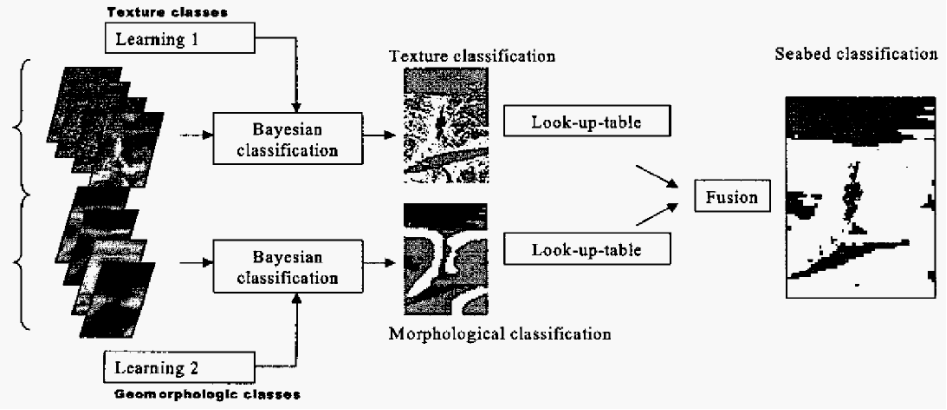


Figure 1-13. Fusion process for classifying the sea bed from two sources. Figure reproduced from Fig.8 in [37].

Kerneis and Zerr concluded that classification accuracy improved in all three methods compared to single-sensor classifications. Signal-level methods gave good results in ideal cases as long as all classes have been trained. The symbol-level fusion was "a little less accurate, but more robust for real applications."

Lanaaya has also presented work on seabed classification and have stated, “Detecting a kind of sediment can be important [39]. For example, rocks can be used as landmarks for image registration for underwater navigation or for the creation of an underwater map.” Of the four main steps of seabed identification—preprocessing images, feature extraction, dimensionality reduction, and classification—the authors have focused [39] on dimensionality reduction to increase classification speed. They have also presented papers pertaining to the other steps in [40] and [41]. In particular, [39] used a process called curvilinear component analysis (CCA). It maps D-dimensional vectors, x_i to M-dimensional, y_i vectors where $M<<D$ while preserving the local topology. The distance between all pairs of vectors in the original data is what defines the topology, but since the dimension is reduced, the local topology is favored at the expense of the global topology. After CCA is performed, SVM is used to classify the images as sand, ripple, rock, rock & sand, cobbles, or ripples & sand. Interestingly CCA improved classification rate on the artificial database (from 93.7% to 96.2%) but significantly hurt the classification accuracy on

the real database of sonar images (from as high as 67.6% down to 48.6%). However, the six classes of the sonar database "highly overlapped" and the classes were "unbalanced." A sample image of each class is depicted in Figure 1-14.

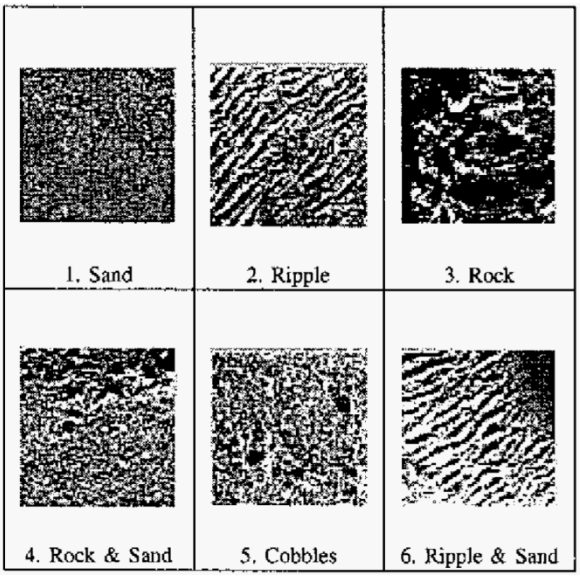


Figure 1-14. Six small images of each class. Figure reproduced from Table IV in [39].

In 2010, Tao et al applied the speeded up robust features (SURF) algorithm to sidescan sonar images and matched the keypoint descriptors with a registered seabed image for the purposes of navigation [42]. An experiment was conducted on a 3614×7416 pixel sidescan sonar relief image that was used as the reference data. Then, strips of 300×50 pixel images were randomly extracted, rotated, scaled and had their contrast altered. The strips were used to simulate real-time sidescan sonar images, which were then matched to the reference image as depicted in Figure 1-15. The authors also compared their results to the scale invariant feature transform (SIFT) algorithm, which had a slightly lower mismatch rate, but was much more computationally time consuming.

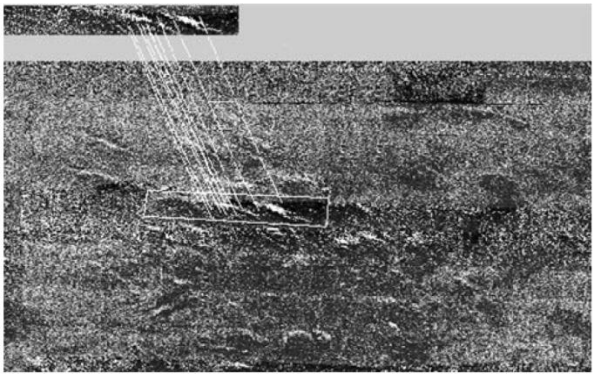


Figure 1-15. A small image strip is matched to the larger reference image. White lines connect corresponding features. Figure reproduced from Figure 5 in [42].

In 2011, Nad et al showed a different mission, which used a small, expendable UUV with minimal sensors—depth sensor, compass, and an acoustic receiver—to neutralize mines [43]. The small UUV was deployed from a parent vehicle such as a larger AUV or surface vehicle that remotely observed and guided the expendable UUV to the location of a mine. Prior to launch, the parent vehicle notified the armed UUV the location of the mine, and then the UUV proceeds to travel directly to the mine location. Once the UUV entered the forward-looking sonar image of the parent vehicle, the parent vehicle began sending position updates via acoustic link. The UUV's controller computes the input through a Kalman filter to maintain a trajectory along the line to the target. Figure 1-16 shows the parent vehicle's sonar image, which depicts the position of the UUV as it approached the target over time.

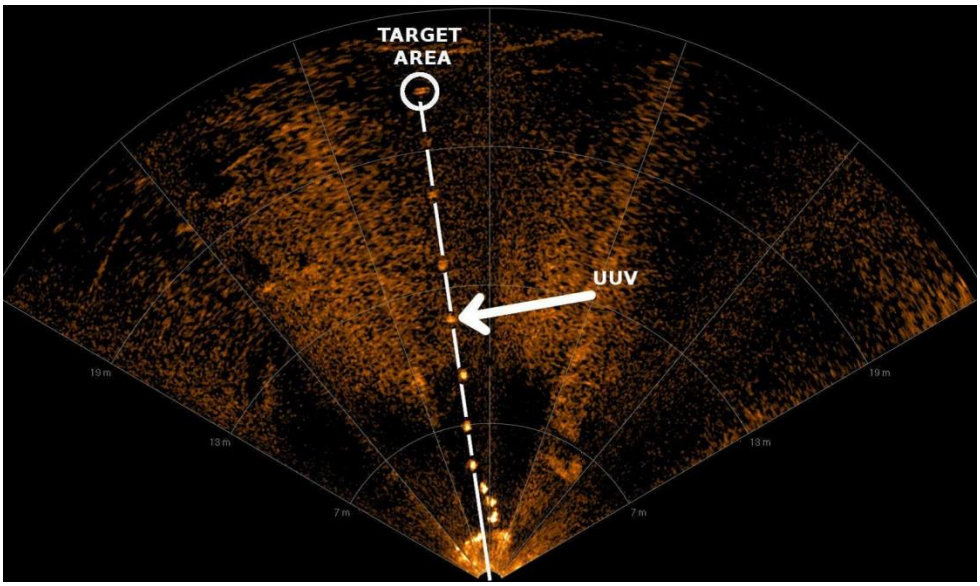


Figure 1-16. The UUV is tracked by the parent vehicle's sonar. Figure reproduced from Fig. 6 in [43].

Inspection and Surveillance

Other AUV research areas included underwater mine detection and ship hull inspection. For example, in 2009, Groen et al presented an automatic target recognition method for detecting mines [44]. The method matched a variety of objects expected in mine-hunting operations from high resolution synthetic aperture sonar to generate 2D image templates that modeled the objects. Matching was obtained by correlation and stochastically over a dataset of 6228 images with eight different objects--a cylindrical shape, a truncated cone shape, a wedge shape, a car wheel, a sphere, a truck wheel, a rock and an oil drum as seen in Figure 1-17. To summarize, the artificial templates were best matched with the sonar images to classify the objects.

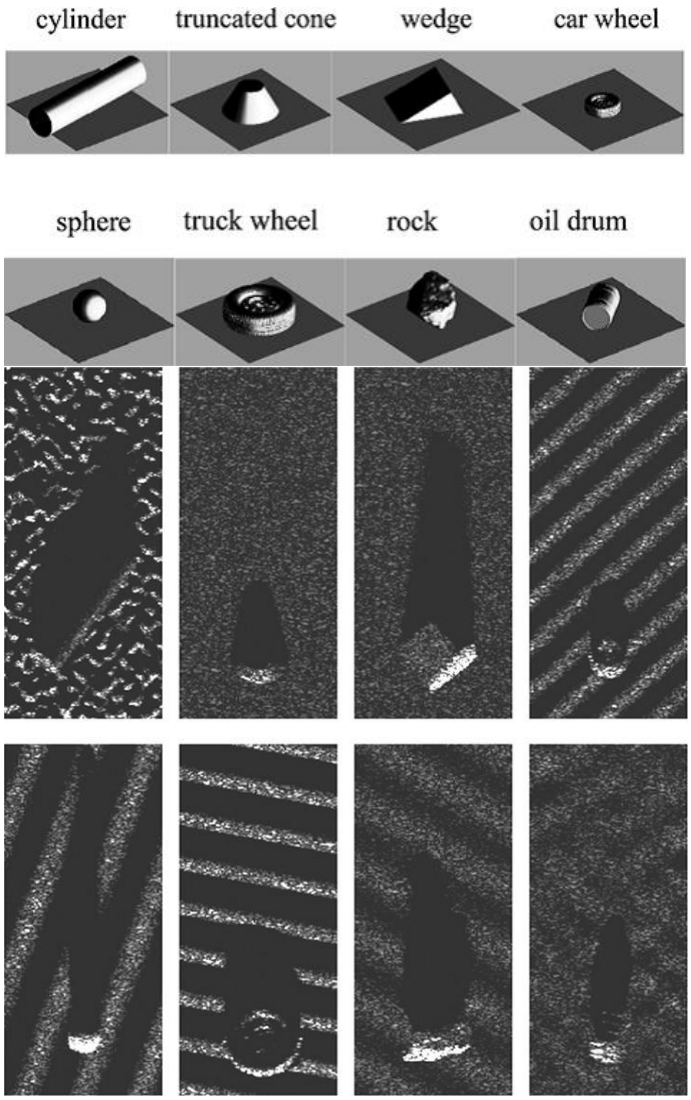


Figure 1-17. Objects used to generate the test data set. The cylinder, cone, and wedge are considered mines. Figure reproduced from Figure 2 and Figure 3 in [44].

In 2006, Vaganay et al demonstrated ship hull inspections in the United States and Italy with a hovering autonomous underwater vehicle (HAUV) [45]. Navigation and control was dead-reckoned relative to the hull with constant attitude and range using a Doppler velocity log (DVL). Image-based inspection was conducted with the vehicle's dual frequency identification sonar. Figure 1-18 shows a mosaic of the images stitched together as well as the path taken to obtain the images.

Later in 2010, Johannsson et al presented similar work with the goal of drift-free navigation using only onboard sensors such as the imaging sonar and DVL [46]. Image processing steps include:

1. Apply median filter to smooth the image
2. Calculate gradient
3. Threshold a top fraction as features
4. Cluster points and discard small clusters
5. Align two overlapping images by registration of extracted features using normal distribution transform (NDT).

For navigation, geometric measurements from image registration is used for the simultaneous localization and mapping (SLAM) problem. An estimate of the vehicle's complete trajectory is maintained so when the vehicle reaches a place it has seen before, "loop closure" occurs in the SLAM problem. Experiments compared the imaging localization results verses the dead reckoning using the DVL and gyro. However, the results would have been more meaningful if ground truth was compared.

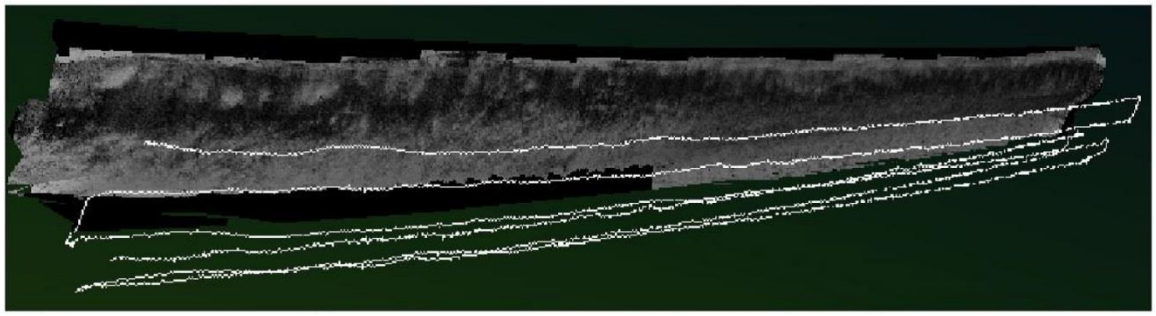


Figure 1-18. 3D mosaic on reconstructed hull shape. The vehicle's trajectory is shown as a solid white line (image courtesy of SeeByte Ltd). Figure reproduced from Figure 17 in [45].

Back in 2006, Montseny et al investigated underwater docks' anomalies detection. Dock anomalies indicate structural deficiencies, and dock inspection for maintenance is performed by divers [47]. The method in the paper uses an optical sensor for automatic visual monitoring based on color information. Anomalies appear within images as poorly illuminated regions (dark pixels)

and are referred as the region of interest. Results are determined from intensity histograms of segmented colors. As seen in Figure 1-19, the results in image 1 are better than image 2, because image 2 displays too many false detections.

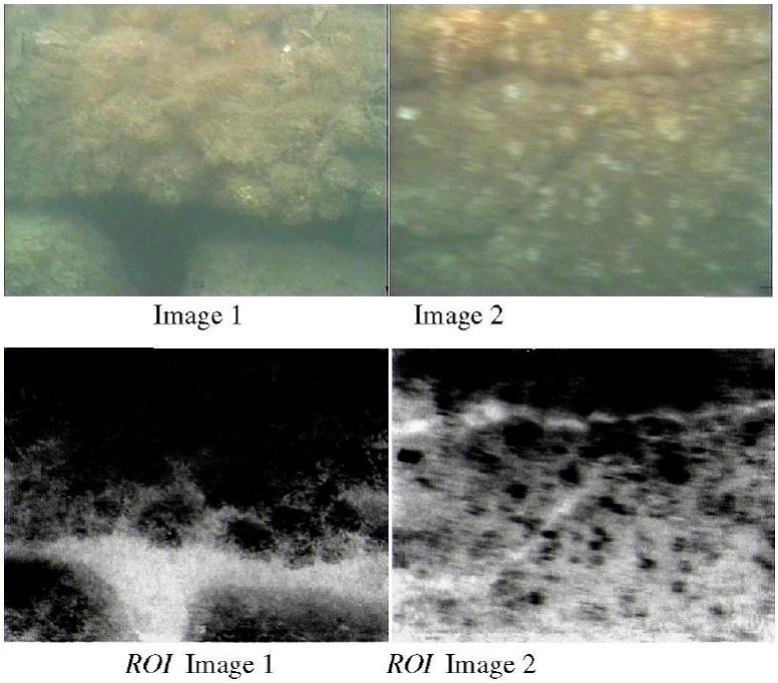


Figure 1-19. First row: original images. Second row: ROI obtained from the original images. Figure reproduced from Fig.8 in [47].

In 2010, Kim et al applied vision-based SLAM to autonomous hull inspection [48]. This means that the area of the ship's hull was mapped for foreign object detection and inspection. The sensor was a calibrated monocular camera mounted on a tilt actuator and SIFT and Harris features were used for image registration. The method allows for both navigation of the vehicle while also generating a texture-mapped 3D model of the ship hull. Results consist of 1300 images covering 30m by 5m of a USS aircraft carrier as seen in Figure 1-20.

Figure 1-20. Texture-mapped reconstruction of the ship's hull. Figure reproduced from Fig. 6 in [48].

Other AUV research involved path planning techniques for the inspection of marine structures. In other words, methods have been developed to generate paths for maximum sensor coverage over objects of interest such as ship hulls and propellers. Complete sensor coverage is necessary for foreign objects that may have been placed on the hull. In 2010 Englot and Hover investigated inspection planning for sensor coverage [49], and then in 2012 they published a paper that explores uncertainty-based inspection planning [50]. The later paper was different because it constructed a closed 3D mesh from acoustic range data and aimed to minimize uncertainty on the mesh surface rather than just seek maximum sensor coverage. Figure 1-21 illustrates the paths to be taken to obtain full sensor coverage of four example structures. Figure 1-22 illustrates the paths to be taken to minimize uncertainty surrounding two ship hulls.

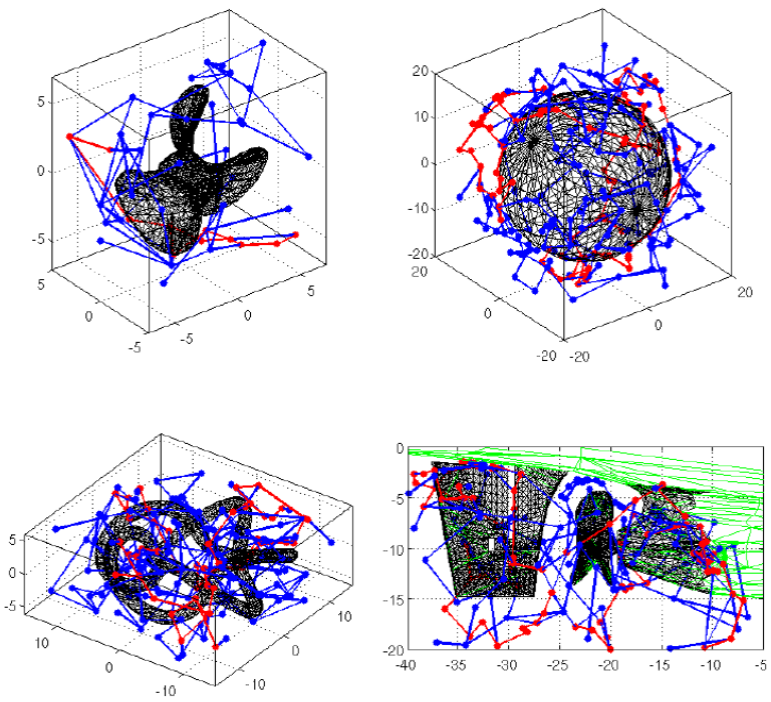
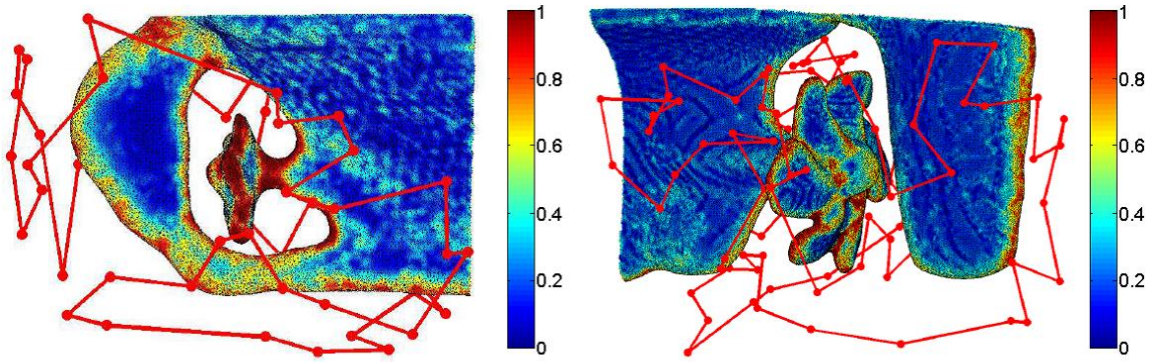


Figure 1-21. Inspection paths planned for the four example structures. Blue edges represent the subtours required for coverage, and red edges represent the paths selected by the algorithm that connects them. Figure reproduced from Fig. 5 in [49].



(a) Nantucket Lightship (b) SS Curtiss

Figure 1-22. The red lines indicated inspection paths. Full coverage of the Nantucket Lightship could be obtained within 5 minutes and full coverage of the SS Curtiss could be obtained within 15 minutes. Figure reproduced from Fig. 6 in [50].

Past Research Summary

Past researchers have investigated UAV and AUV localization and mapping, object classification, inspection of marine structures, and surveillance. Work involved optical sensors and sonars sensors that obtained raw images followed by significant image processing. Low level features were extracted either by SIFT or other vector representations followed by some form of clustering or dimensionality reduction if necessary. It became clear that high resolution images allowed for improved results.

Introduction to this Research

This research investigates detection of fixed landmarks on terrain to help determine current location of the vehicle (localization). The research can be applied to either underwater vehicles scanning the sea floor with a sonar/optical sensor or aircraft scanning terrain with a camera. Since underwater map data is not available for this research, simulations and experiments are limited to the scope of aerial vehicles. The concept can be applied to both manned and unmanned aircraft. The concept is similar to the way human pilots navigate visually from

landmark to landmark with sectional aeronautical charts. The concept could also apply to autonomous vehicles patrolling a region on a regular basis. The information could be shared across a fleet of unmanned systems operating in the same region. The database is used to aid in navigation, particularly in degraded modes such as GPS-denied environment.

Feature-based navigation has recently become a research topic of interest for aiding navigation of unmanned systems in a GPS-denied environment. UAV's operating close to the terrain may lose quality of GPS reception or their GPS systems might be jammed. As new images are captured, they are processed and matched to the database for vehicle localization. The research will investigate useful, abundant, and easily identifiable landmarks as well as a method for integrating the data into an efficient inertial navigation solution. This research is unique because it allows for the automatic detection of landmarks easily understandable by humans, rather than high dimensional abstract feature vectors. Overall, this research creates a framework for automatic aircraft localization and tests the method in both simulation and flight.

This research explores two different concepts of landmark detection and employs readily available object detectors (e.g. cascade object detectors or bag of visual words object detectors) during the image processing routine. The first concept involves the detection of *predetermined distinct* landmarks. This concept is similar to the flight planning stages familiar to pilots flying VFR routes. Computer vision cascade object detectors have been historically trained to detect human faces, but this research is first to apply those detectors to landmarks for the purposes of aircraft localization. More importantly, this research shows the *information required* for successful localization, and the cascade object detectors demonstrate one way, *but not the only way* of detecting landmarks. The architecture of the method is modular to allow more advanced computer vision techniques to provide the necessary measurements in the future.

The second concept involves the detection of *generic landmarks* categorized as buildings, fields, trees, and road intersections throughout the entire image. The second concept simplifies

imagery data into those four categories and uses their spatial relationships to associate the observed aerial scene to a preprocessed reference map. This research is the first to apply the bag of visual words computer vision technique to categorize generic landmarks for the purpose of aircraft localization. This research also presents a novel data association algorithm, which matches generic landmarks detected in aerial photographs to their respective landmarks detected in a reference map automatically. Again, the overall architecture is framed to allow for the application of different object detection methods and object classifiers in the future.

In both methods, an unscented Kalman filter (UKF) is used to integrate known landmark locations into standard inertial navigation solutions. As applied to this research, the UKF is used to compute estimated aircraft coordinates and wind velocity based on vehicle dynamics and sensor measurements. Furthermore, since map data is memory-demanding, techniques for managing large regions through smaller local regions are introduced.

Chapter 2 formulates the problem and presents the overall solution to aircraft localization using visual landmarks. Chapter 3 describes computer vision object detection techniques used in this research. Chapter 4 presents a UKF navigation solution that integrates the visual measurements obtained from the computer vision system. Chapter 5 explains the formulation of a photo-realistic flight simulator and presents results using both localization methods in simulation. Chapter 6 presents a hardware demonstration of the second method using a quadcopter in an indoor synthetic environment. Finally, Chapter 7 concludes the results and discusses opportunities for further advancements to this research.

Chapter 2

Problem Formulation

The ultimate goal is to pave the way for UAVs to localize themselves by intelligently recognizing landmarks as they fly through an environment. A simple concept is illustrated in Figure 2-1. As new optical images are captured, they are processed and matched to the database for vehicle localization.

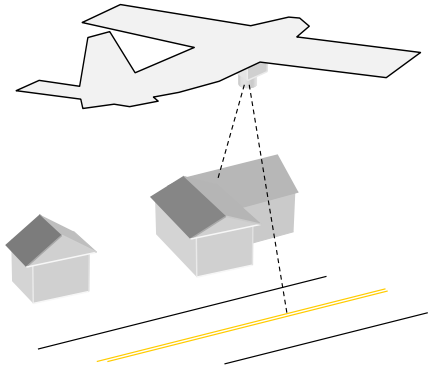


Figure 2-1. An aerial vehicle flies over predetermined landmarks to navigate without GPS.

Upon successful detection and association of a landmark, the pixel coordinates of the detected landmark is recorded as the measurement. Because the global coordinates of the landmarks are known, the absolute location of the aircraft can be determined based on the location of the landmark's projected image in the airborne 2D image plane.

A landmark's image plane coordinates can be predicted with knowledge of the following:

- landmark coordinates
- estimated aircraft location
- aircraft attitude (yaw, pitch, and roll) relative to inertial coordinates
- altitude

- camera location and orientation relative to the airframe
- camera intrinsics such as focal length

With the above information, a measurement model is constructed to predict the location of the landmark in the image plane coordinates, and the UKF compares it to the actual aerial photographic measurement. The first method of aircraft localization estimation is introduced in the following section titled, *Localization with Distinct Landmarks*. The second method is introduced in the section titled, *Localization with Generic Landmarks*.

Localization with Distinct Landmarks

Unique landmarks are chosen along a route or within a region prior to the flight and their coordinates are recorded. Next, landmark detectors are trained to detect the chosen landmarks. Then the aircraft flies over the scenery and the vision system scans for those unique landmarks using the trained detectors. The camera is fixed to the airframe and points down to the ground with its optical axis aligned with the z-axis of the aircraft body coordinates. The flow of information is depicted in Figure 2-2, which shows the overall architecture of the navigation solution using distinct landmarks. Blocks will be discussed in more detail in the following sections.

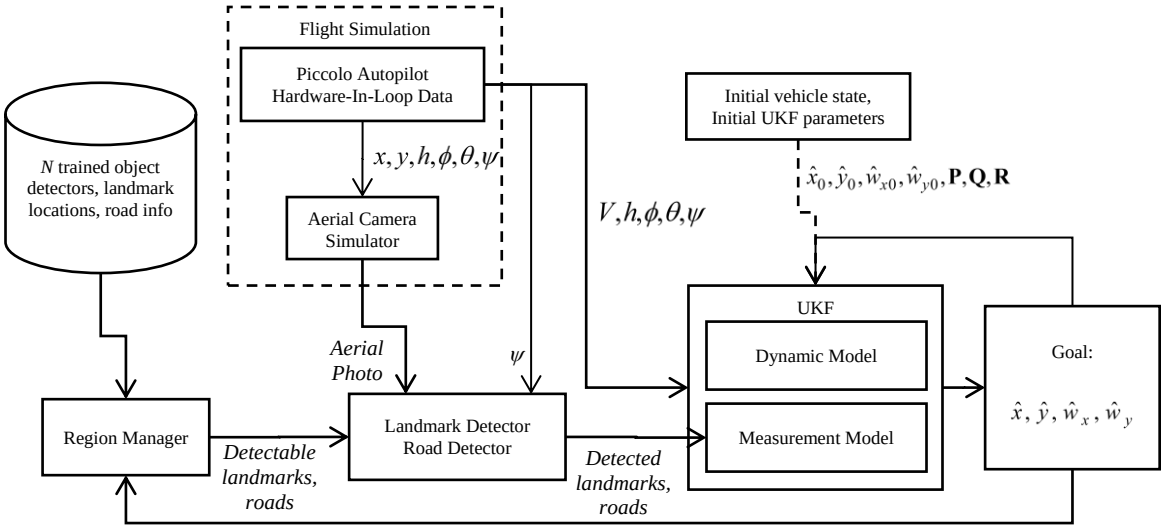


Figure 2-2. Flow of information used for estimating the navigation solution.

Before the flight begins, landmarks are chosen in a map and computer vision object detectors are trained to detect those landmarks. During a flight, the object detector scans each aerial photograph for an expected landmark. When successfully detected, the location of the landmark is reported as a pair of pixel coordinates in the photograph. The expectation is to obtain an improved location estimate when a landmark is detected and observe navigation drift when "dead reckoning" between landmarks. The amount and spread of landmarks can be chosen depending on accuracy of the inertial navigation system (INS).

Region Management

Because a reasonable estimate of aircraft location is known, the region manager feeds the object detector only with information about detectable landmarks. This purpose is to use appropriate landmark detectors for landmarks within a reasonable range from the vehicle—it would not make sense to scan an image with detectors trained for landmarks far out of range at the risk of false positive detections. For example, the region manager enforces the application of

landmark detectors that can detect landmarks within distance *r* from the estimated location of the aircraft. Furthermore, the initial location of the aircraft is assumed to be known when launched.

Landmark Detection

Each object detector is trained to detect a specific landmark, and each landmark's coordinates are known and recorded. The role of a landmark detector is to identify landmarks (if any) in the aerial photograph. If there are multiple landmarks within range as determined by the region manager, then the aerial photo is scanned multiple times with the goal of detecting the expected landmarks. When a landmark is detected in the photograph, a pair of pixel coordinates is recorded as the measured location of the landmark. The pair of pixel coordinates is determined by the center of a bounding box that encloses the landmark.

Road Localization

Road localization is the concept of determining the aircraft's position relative to an observed road. Road localization provides important position information in one dimension. The shortest distance from the image plane's origin to the road centerline and the direction of the road are measured to help localize the aircraft's position relative to the perpendicular distance from the road's centerline. To obtain two-dimensional localization, however, an intersection of two known roads is needed; otherwise localization parallel to the road is indeterminate. Still, successful road detection, even in one dimension, improves localization considerably rather than dead reckoning until a new landmark is detected. Visual road measurements record the road's distance (in pixels) and direction (unit vector), which are then integrated with the UKF. Road localization measurements are evaluated when no landmarks are detected.

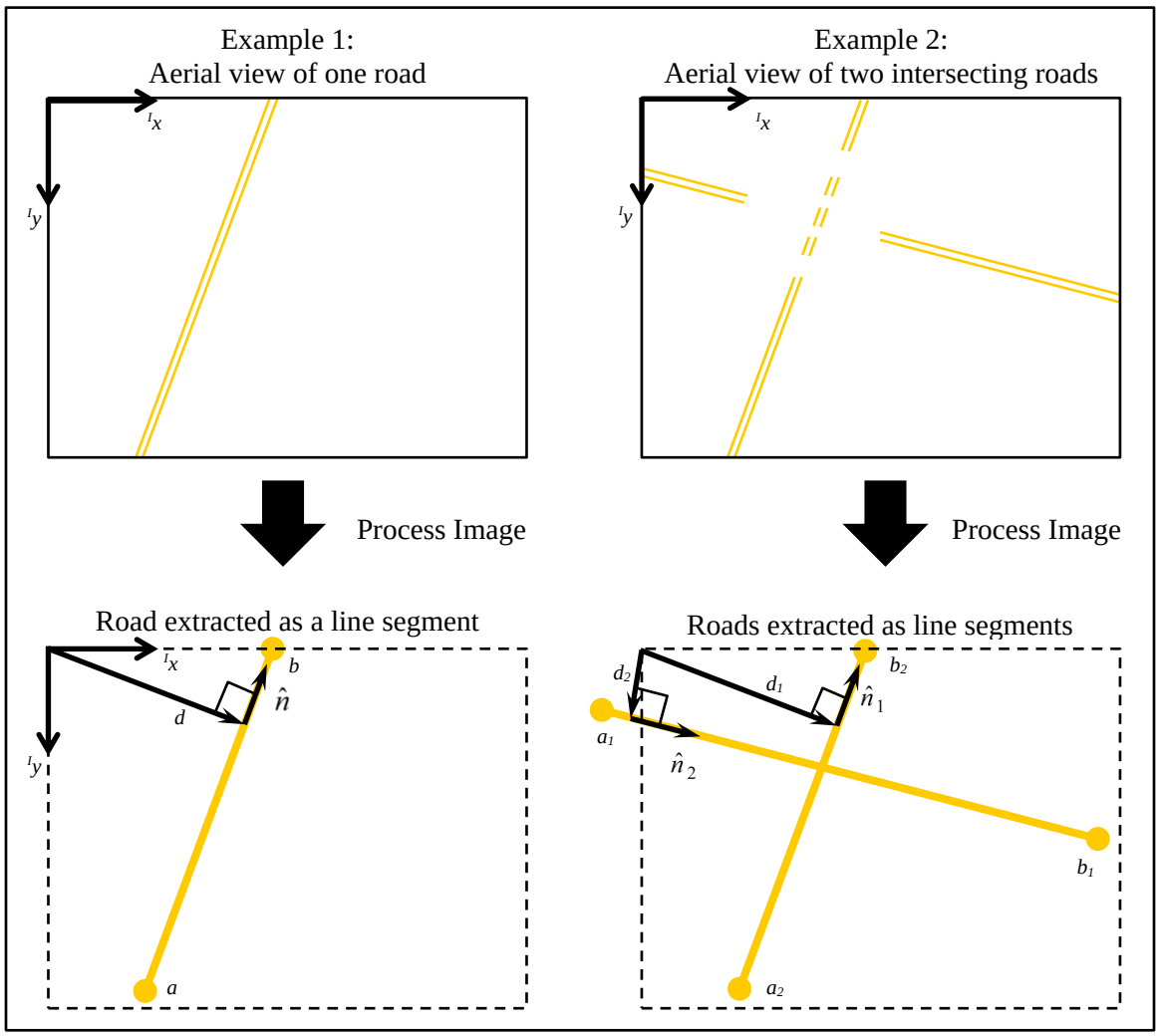


Figure 2-3. Roads are extracted and reduced to line segments. ${}^{I}x$ and ${}^{I}y$ are the image frame coordinate axes. Points *a* and *b* are endpoints of the line segments, *d* is the shortest distance from the origin of the image frame to the line segment, and $\hat{n}$ is the direction of the road expressed as a unit vector.

Road detection is also useful for the data association task when using the generic landmark detection method, which is described in the following section. Detected roads are used as a standard ground reference to help differentiate landmarks from one another and match them to their respective landmarks in the map. The side and distance from the road's centerline to the landmark serves as one attribute, *but not the only attribute*, for comparing landmarks during the data association routine. Figure 2-4 illustrates an example of how each landmark's distance from

the road differs. Additional landmark attributes are used for data association and will be discussed in detail in the next chapter.

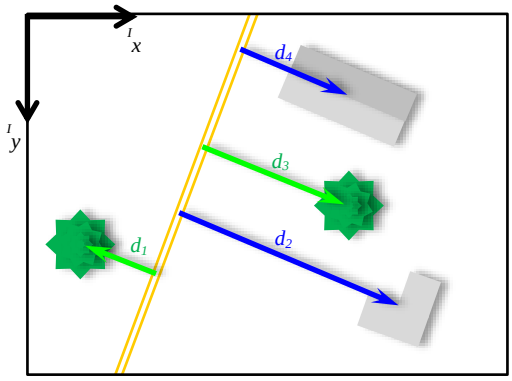


Figure 2-4. Roads are used as a standard ground reference to compare landmarks during the data association task.

Localization with Generic Landmarks

Generic landmarks are treated as the most abundant object types found in aerial images such as buildings, fields, and trees. The flow of information is depicted in Figure 2-5, which shows the overall architecture of the navigation solution using generic landmarks. This method includes a separate data association process that matches observed landmarks with landmarks recorded in the reference map. Note that the previously discussed section, *Localization with Distinct Landmarks*, inherently associates *distinct* landmarks because each detector is trained to detect a single landmark. A distinct landmark is unique and will match with only one possible landmark, and false detections must be resolved with the region manager. In this section, however, generic landmarks are detected and the system must match the observations with the correct landmarks in the reference map. A data association algorithm must match the detected landmarks in the aerial photo to the correct corresponding landmarks in the reference map. For example, when a building is detected in the aerial photograph, the data association task must

answer the question, “Which building is this?” Special landmark attributes and spatial relationships to other landmarks are used as features to help associate observed landmarks with reference landmarks.

The detectors are trained to detect each object type (buildings, fields, trees) within a scanning window across the image. High resolution orthoimagery (a map) is processed with the detector, and the object detector records the pixel location and object type in the map. Markers that indicate the location and object type are placed on the map. Therefore the arrangement of diverse objects types relative to one another provides useful information specific to the scene. A working example of a sample reference map is processed automatically and shows three categories of detected landmarks in Figure 2-6. In the figure, blue circles represent the location of detected buildings, yellow squares represent the location detected fields, and green asterisks represent the location of detected trees. The reference map is processed offline, and only the color-coded 2D markers are stored onboard the aircraft. Therefore memory storage requirements are minimized because the map’s RGB data can be discarded after the othoimagery is processed and no high dimensional feature vectors are stored either. After landmark detection, for example,

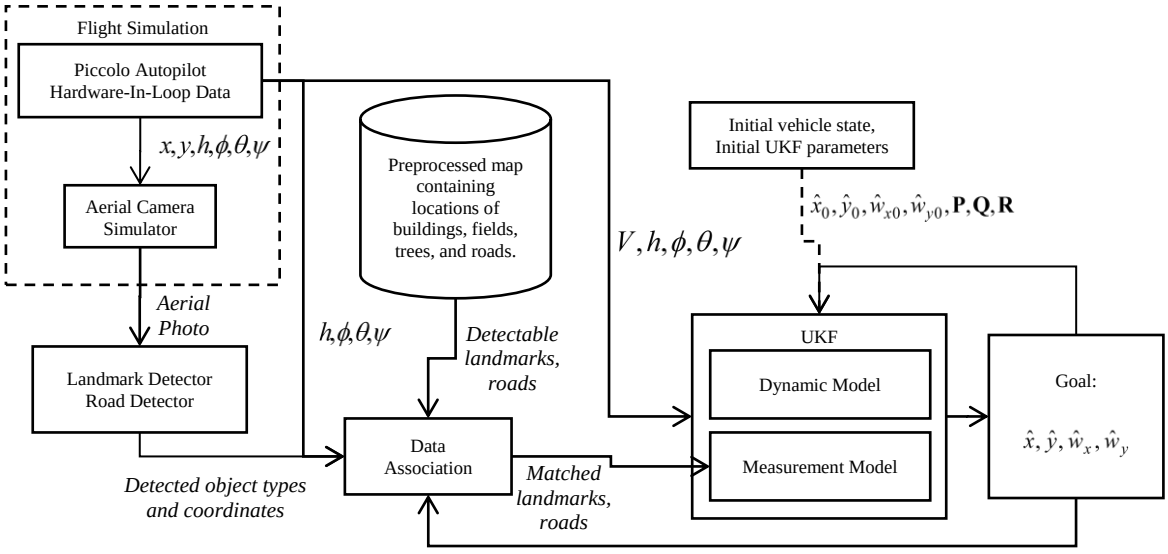


Figure 2-5. Flow of information used for estimating the navigation solution.

a 2.7 GB 22167 by 25100 pixel RGB map is reduced to a 110 KB 18389 by 3 element array of unsigned integers. Each row contains a single landmark's pixel coordinates and classification. The first column represents the landmark's x-coordinate (horizontally), and the second column represents the landmark's y-coordinate (vertically). The third column is an integer classification label where 1=buildings, 2=fields, and 3=trees.

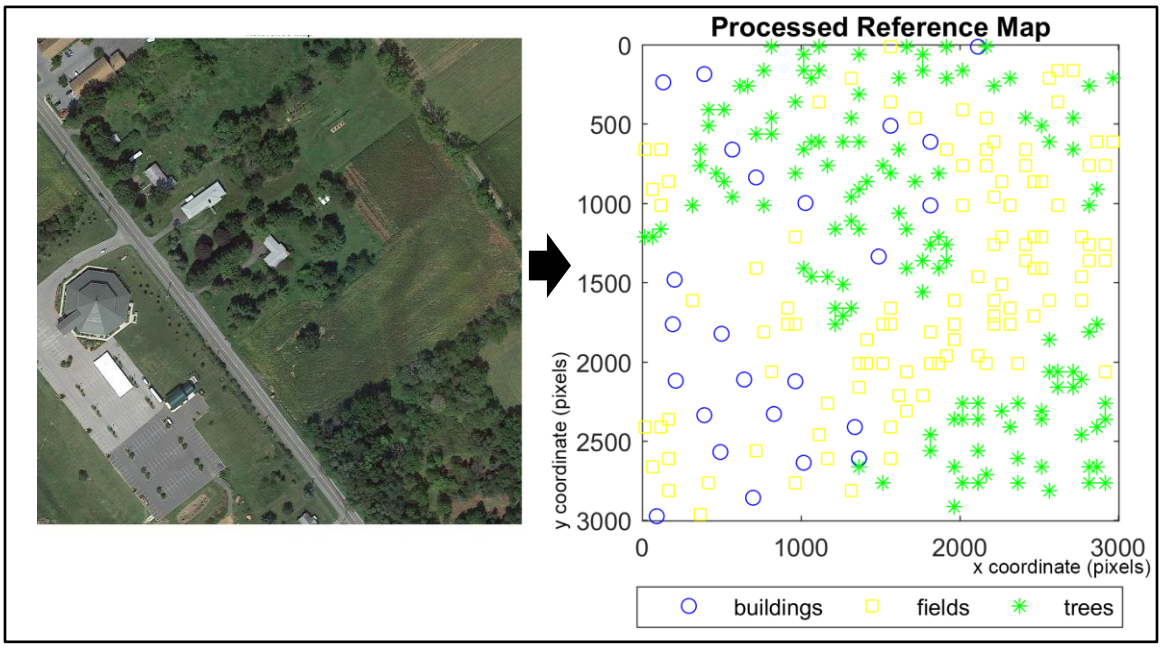


Figure 2-6. Object detectors scan the reference map and label the detected objects.

A similar landmark detection method carried over to process aerial photographs, which are then compared to the reference map. During a flight, the aerial photos are scanned with the detectors and the markers in the aerial photo are then associated with the objects in the map. Measurements are high-level features representative of the actual objects by a single attribute (building, field, or tree) rather than high-dimensional feature descriptors such as scale-invariant feature transform (SIFT) features or speeded-up robust features (SURF) traditionally used in past research. After processing, all imagery is reduced from arrays of RGB values to points labeled as building, field, or tree. A working example of a processed aerial photo is depicted in Figure 2-7.

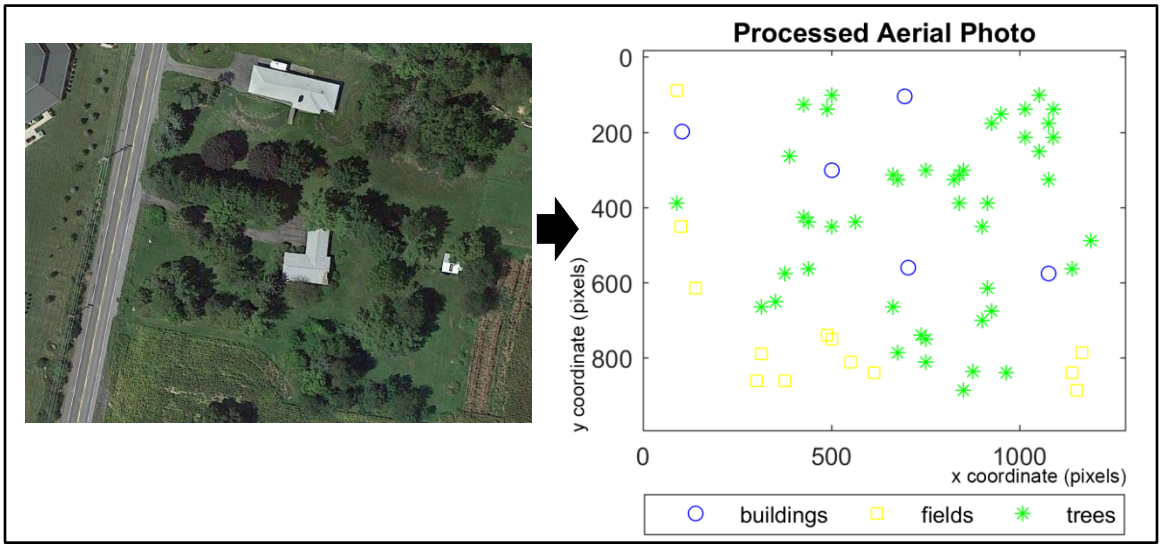


Figure 2-7. Object detectors scan the aerial photo and label the detected objects.

The next critical step is to associate the landmark markers from the aerial photo with the correct landmark markers in the map. Conceptually, this can be seen in Figure 2-8. The next chapter describes the method of matching landmarks observed in the aerial photo with the most plausible landmarks in the reference map. This step is the critical link for obtaining accurate visual measurements to not only correct INS drift, but also estimate wind velocity.

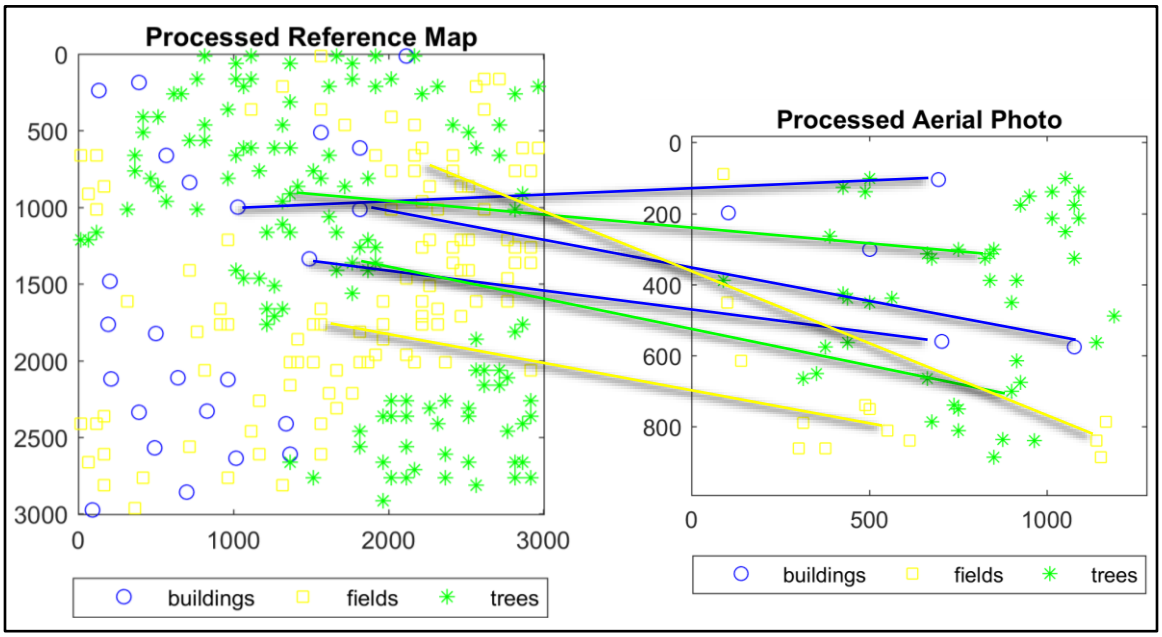


Figure 2-8. The landmarks must be correctly matched from the aerial photo to the map.

Chapter 3

Object Detection Techniques

The overall objective of this research is to investigate the feasibility of automating VFR navigation. Reliable computer vision is, however, critical for successful drift-free localization to meet this objective. Over time, INS information drifts and reliable measurements are necessary to correct the drift. Research on developing computer vision algorithms is a secondary objective, but of course is necessary to demonstrate the localization approach. This chapter introduces the theory of two existing computer vision object detection algorithms and explains how they could be applicable to landmark detection. Two fundamentally different object detection methods are applied to this research: 1) cascade object detector and 2) bag of visual words (BoW). Both methods are "learned" because they both require datasets to train detectors, which detect and label objects with similar appearances when queried. Both tools are modern and are now recently available in commercial computer vision software packages such as MATLAB's Computer Vision System Toolbox R2014b and later.

The cascade object detectors detect specific landmarks that match the shape and orientation of those contained in the training image dataset. For each landmark, a dataset of images of that specific landmark is necessary for successful detection. For example, a cascade object detector is trained to detect a specific building given multiple images of the same building.

The BoW object detectors detect and label generic objects that generally share similar appearances to others in the same category. Any category of detectors can be trained if objects of interest share similar visual elements. For example, the BoW detector detects buildings because it is trained with a dataset consisting of aerial views of many different buildings. In this research, the BoW object detectors must classify objects in three different categories – buildings, fields, and trees. Those three landmarks are appropriate categories because they are most abundant in the

chosen environments for the demonstrations. However, the categories of objects might vary depending on the specific mission or environment where the aircraft operates.

Object Detection Techniques: Cascade Object Detector

The cascade object detector uses the Viola-Jones algorithm [51] to detect landmarks, but the original motivation for the Viola-Jones algorithm was to detect faces. The algorithm is implemented in both OpenCV and MATLAB Computer Vision System Toolbox and can be used to train a custom classifier to detect objects other than faces. Some of the main advantages of the algorithm include:

- A small moving window quickly scans the image for the object of interest. The computation is fast for real time object detection.
- The method is general therefore it can be used to train other types of objects. In this case, it is trained to detect distinct landmarks.
- Object detection is computationally fast and intended for real-time operation.

Disadvantages include:

- The detector is sensitive to object rotations.
- Multiple detections of the same object occur due to overlapping detection windows.
- Objects with similar gradients may cause false positive detections.
- The detector may be sensitive to lighting conditions. Flight testing would be required to determine the severity of this sensitivity and possibly to develop algorithms to mitigate this issue.

The term “cascade” is used because it employs a set of stages to efficiently process image regions for the presence of a target object. *Each stage in the cascade applies increasingly more complex binary classifiers, which allows the algorithm to rapidly reject regions that do not*

contain the target. If the desired object is not found at any stage in the cascade, the detector immediately rejects the region and processing is terminated. By terminating, the object avoids invoking computation-intensive classifiers further down the cascade [52].

The cascade object detectors are trained with histograms of oriented gradients (HOG) features using the Cascade Object Detector in MATLAB. HOG features are obtained by first computing the gradient of an image. The image gradient is the computed change in intensity magnitude and direction of neighboring pixels. Areas with large gradient magnitude signify sharp changes in contrast such as edges. Areas with low gradient magnitudes indicate a more uniform, textureless area. Next, the image is divided into a grid of cells. Within each cell, a histogram of edge orientations is computed. The histograms can be normalized by the intensity across a larger region of the image or with respect to neighboring cells. The HOG descriptor is the concatenation of the normalized histograms. Figure 3-1 shows an image of a bicycle overlaid by 7×11 HOG cells.

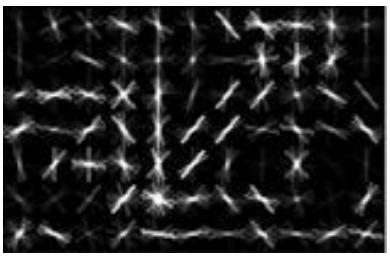


Figure 3-1. Visualization of HOG edge orientations. Figure reproduced from [53].

HOG features are used for their efficiency and demand less memory. Future work could investigate other descriptors such as local binary patterns (LBP), Haar features, and investigate object detection using rotation-invariant features such as Speeded Up Robust Features (SURF) or Scale-Invariant Feature Transform (SIFT) features. Using HOG, LBP, and Haar features require the object of interest in the image to be arranged in a similar orientation to the objects in the training set for successful detection. However, some researchers have investigated rotating the object detector as a work-around when scanning the image or they have used rotation invariant

template matching techniques [54], [55], [56]. In the method presented here, however, the photograph is simply rotated by the aircraft's yaw angle before running the object detection routine, thus allowing the aircraft to approach any landmark from any direction. The pair of pixel coordinates of the detected landmarks in the rotated image is then transformed to coordinates in the original aerial photograph using a direction cosine matrix for the yaw angle. The computed coordinates of the landmark in the aerial photograph represent a measurement to the navigation solution. The simulation experiments have shown that landmark detection occurs most reliably during straight-and-level flight. Since the camera is downward pointing, pitch and roll angles create out-of-plane image rotations leading to images unsuitable for the cascade object detector unless rectified.

Object Detection Techniques: Bag of Visual Words

Csurka et al introduced a "bag of keypoints" approach to visual categorization of everyday objects [57]. Note that "bag of visual words" (BoW), "bag of keypoints," and "bag of features" are terms used interchangeably. The goal was to identify object content within images with a computationally efficient, robust, and intrinsically invariant algorithm. The authors presented their "bag of keypoints" method and then generated final results with two different classifiers—Naïve Bayes and SVM—and compared them.

The algorithm works as follows: 1). Image patches are detected and described. 2). Patch descriptors are assigned to a set of predetermined clusters (analogous to a vocabulary). 3). The number of patches assigned to each cluster are counted (bag of keypoints) 4). A multi-class classifier determines which of the seven categories (face, building, trees, cars, phones, bikes, books) to assign to the image.

In step (1), the Harris affine detector is applied to determine an affine neighborhood. The affine region is then mapped to a circular region. Scale Invariant Feature Transform (SIFT) descriptors are computed on that region.

In step (2), feature vectors are clustered with a square-error partitioning method known as "k-means" [58]. Points are assigned to cluster centers and the cluster centers are iteratively recomputed. The authors run k-means several times and the final clustering giving the "lowest empirical risk in categorization" is selected [38].

In step (3), feature vectors are assigned to their closest cluster center and the number of occurrences of those cluster centers are counted in a histogram.

In step (4), the histograms are inputs to the Naïve Bayes and SVM classifiers, which are run separately and compared.

The results show that the Naïve Bayes error rate decreases as the number of clusters, k, increases. Naïve Bayes also handles multiple objects of the same category within images, even occluded objects at any orientation, and it has no problem handling background clutter. The Naïve Bayes is not without its problems, however, because it incorrectly ranks labels to some images. Next, the SVM results show that the SVM classifier performs better than the Naïve Bayes classifier. Finally, the authors run the experiments again using a different database of images. As claimed by the authors, advantages to the "bag of keypoints" method include simplicity, computational efficiency, robustness to occlusion, lighting, intra-class variations, and invariance to affine transformations.

A paper by Winn et al is an extension to this work and also points out some of the weaknesses of the method. The initial BoW method computes features on a sparse set of interest points over a fixed dictionary, whereas the method presented by [59] processes every pixel and automatically learns keypoints and dictionary size. Winn et al also presented a slightly different algorithm for the automatic recognition of object classes from images. The keypoints, referred as

"visual words," were learned from a manually segmented training set. The algorithm was built upon an appearance-based texton model rather than a shape-based model.

The authors goal was to present an object categorization method that is fast and robust to variations in object pose, luminance, and also handle multiple classes within the image. The method is applied to nine object classes—cow, grass, tree, sky, airplane, car, bike, building, face—in photographs with general lighting conditions and poses. The goal was to be able to quickly select objects in the image and have the algorithm automatically identify the object of interest. Examples can be seen in Figure 3-2.

Figure 3-2. Screenshots of the interactive object categorization demo. Figure reproduced from Figure 1 of [59].

The development of the method is summarized as follows:

1. Training and universal visual dictionary (UVD)

   a. 240 photographs were manually segmented and annotated for the training set. The objects in those photographs belong to the nine classes. Manual segmentation involved manually paining a color label over the object's area.

   b. Each training image was convolved with a filter-bank to generate filter responses over all of the images in the entire training set, independent of class labels.

   c. Clustering was done with K-means using Mahalanobis distance between cluster centers. The cluster centers and their associated covariances defined the UVD so that any image may be filtered to associate each pixel with the closest texton in the UVD.

   d. The histograms of textons were then computed on a region.

2. Object class modeling

a. An advantage to this technique is that the overall distribution of dictionary textons over the image is important. The paper also mentioned that the class modeling performance depended on dictionary size; therefore the method estimated an optimal dictionary size. Bayes' rule is used to make the histograms more similar within each class while making them more discriminative between class labels.

b. Next the method determined the mapping "which maximizes the conditional probability of the ground truth labels, given the texton histograms of all training regions." It grouped visual words in the original dictionary into a more compact dictionary.

c. After obtaining the compact UVD, the authors compared parametric (Gaussian class model) and non-parametric (nearest neighbor class model) classification methods.

With or without dictionary compression, the Gaussian and nearest neighbor classifications were both about 93% accurate. The Gaussian and compact (learned) UVD is much faster, though. The learned dictionary allows the Gaussian results to achieve accuracy greater than 90%. The method was tested on a different dataset of images and performed with an accuracy of about 74%. Objects such as bicycles and motorbikes caused higher confusion leading to poorer results. Furthermore the persons class had a high level of confusion due to the variability of people's clothing.

The BoW computer vision classification technique discussed above is applied to this research. First, sets of training images are used to train the classifier to discriminate buildings, trees, and fields. A small sample of training images from each category is depicted in Figure 3-3. SURF descriptors are extracted from the training images and constructed into vectors. The vectors are clustered with k-means, and the cluster centers become the "dictionary codewords."

Next, each extracted feature descriptor in the training set is matched and assigned to the closest codeword. Then the number of codeword occurances are counted thereby forming a histogam of codewords (“the bag of visual words”). Finally, the histograms from the training set are used to train an SVM classifier.

After the construction of the SVM classifier, a new image can be queried. From the query image, feature vectors are extracted with the same method (SURF) as dictionary generation, then the vectors are assigned to the closest dictionary codeword, and then number of codewords are counted. Finally, the “bag of visual words” from the query image is then evaluated with the trained SVM for classification as either a building, field, or tree.

The previous researchers have shown reliable classification for as many as nine different object categories. As the number of categories increases, the complexity and likelihood of incorrect classification increases. Fortunately in this research, object detectors are trained to classify only three different landmark categories. Once the landmark detector is constructed, a small scanning window is applied to the query imagery such as the photographic maps (which are processed before the flight) and aerial photographs to automatically classify landmarks in the scene. Therefore a pattern of different objects spatially arranged relative to one another provides useful information for matching observations with the map.

buildings

fields

trees

Figure 3-3. A sample set of training images of each category is shown.

Road Detection

Automatic road detection is employed for extracting roads from the reference map and also for extracting roads from the aerial photos obtained during the flight. Road detection serves three purposes in this research. First, road detection is used for obtaining road localization measurements, which are evaluated in the "Localization using Distinct Landmark Detection" simulations section of Chapter 5. Next, road detection is used for extracting the coordinates of intersections, which are evaluated in the "Localization using Generic Landmark Detection" section of Chapter 5. Road intersections are also treated as landmarks because they provide accurate 2-dimensional coordinates. Furthermore, road centerlines are used as a standard ground reference for the comparison of the generic landmarks. The perpendicular distance from the centerline to each landmark is one measure, but not the only measure, used in the data association task, which is explained in the next section - Data Association. Several commercial software packages are available to detect roads. These include *FeatureAnalyst* and *RoadTracker* from Visual Learning Systems, Inc. in Missoula, MT; *RoadMAP* from TerraSim in Pittsburgh, PA; and *Genie Pro* from Observera in Chantilly, VA. Sources such as [60], [61], [62], and [63] have presented road detection methods that could be useful for the purpose of this method. This section is not intended to reinvent viable road detection methods, but rather explain how roads were detected in this research so that sufficient data can be provided to the algorithms that depend on road information.

The road centerline is extracted with fundamental computer vision algorithms and simplified as straight line segments. The first step for identifying a road involves extracting the yellow pavement marking on the road by color segmentation. The yellow pavement line color closely matches color number 33538 (International Traffic Yellow), which is defined by SAE

International standard, AMS-STD-595™ *Colors Used in Government Procurement*, under daylight illumination (Illuminant C) with the following CIE tristimulus values [64]:

$$X = 0.5559 \quad Y = 0.5009 \quad Z = 0.0693 \tag{1}$$

Using the conversion methods defined in IEC 6199-2-1:1999, this color appears as [255 170 6] in the sRGB color space, which is the native color space for most digital cameras and displays [65]. Real world photographs of pavement markings show a variety of yellow colors, not just a single uniform color due to paint pigments errors, scene illumination changes, sensor noise, camera calibration, etc. Therefore a range of similar colors is assumed to represent the road markings. A range of similar yellow colors can be represented in the Hue, Saturation, Value (HSV) color space because a cylindrical slice (boundaries) can easily isolate a range of yellow hues inclusive of all ranges of saturation and brightness.

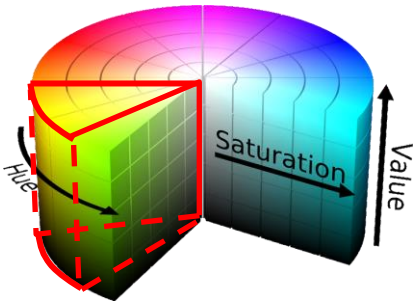


Figure 3-4. Due to cylindrical coordinates, a slice of yellow hues can be easily isolated with the HSV color space. Image source: M. Horvath [66]

Images represented in the sRGB color space are transformed to the HSV colorspace with the MATLAB function, rgb2hsv(). A range of yellow colors are isolated with the following constraints:

$$\begin{aligned} 0.099 \le \mathrm{H} \le 0.215 \\ 0.060 \le S \le 1.000 \\ 0.730 \le \mathrm{V} \le 1.000 \end{aligned} \tag{2}$$

All pixels with values outside of the constraints are labeled with zeros, and all pixels with values within the constraints are labeled with ones. An image dilation operation with a 5×5 kernel is applied to connect broken pavement markings. Then connected component analysis

filters out blobs and keeps only those with a major axis length in the range of $[350 \quad \infty]$ pixels and a minor axis length in the range of $[1 \quad 45]$ pixels. Lines are extracted based on the Hough transform with MATLAB's houghlines() function. If multiple lines are observed, the two longest lines are examined to determine whether they intersect. If the cosine of the angle between two lines is less than 1, then the lines are not parallel. For the purposes of road extraction in this research, the roads are assumed to intersect if the cosine of the angle between them is less than 0.8. Finally, the intersection point (junction) is computed using vector cross products in homogeneous coordinates. For example, to compute the intersecting point $\vec{p}$ in Figure 3-5, the cross products of homogeneous vectors $\vec{a}$ and $\vec{b}$ are cross multiplied with the cross products of homogeneous vectors $\vec{c}$ and $\vec{d}$ as seen in Eq. (3). The elements of a homogeneous vector $\vec{a} = [x \quad y \quad 1]$, for example, contains the Cartesian coordinates augmented with a 1. To obtain the intersecting point $\vec{p}$ in Cartesian coordinates, the first and second elements of $\vec{H}$ are divided by the last element of $\vec{H}$ as described by Eq. (4). The final output of the road detection algorithm returns the endpoints of the longest line segment and the intersection point (if any). Figure 3-6 illustrates the image processing steps to extract roads and intersections.

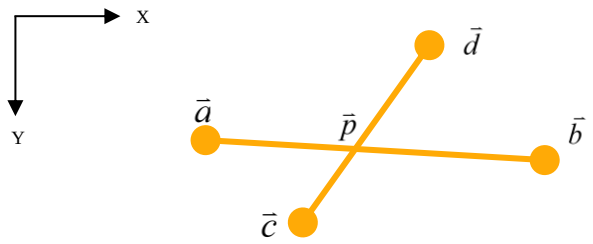


Figure 3-5. Example illustration of two intersecting roads in two dimenisonal coordinates for the purpose of computing the intersecting point.

$$\vec{H} = \left(\vec{a} \times \vec{b}\right) \times \left(\vec{c} \times \vec{d}\right) \tag{3}$$

$$\vec{p} = \begin{bmatrix} \vec{H}(1) & \vec{H}(2) \\ \vec{H}(3) & \vec{H}(3) \end{bmatrix} \tag{4}$$

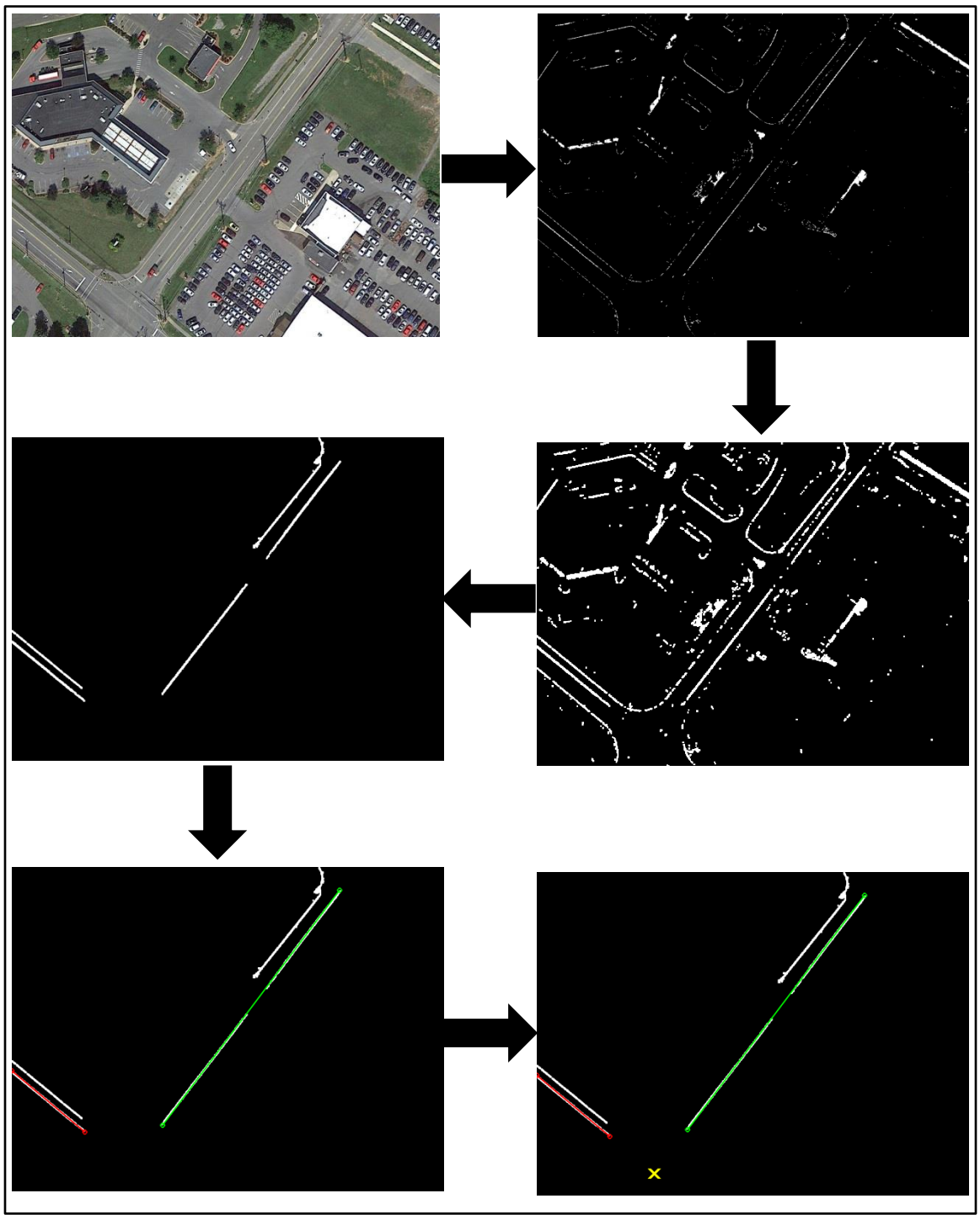


Figure 3-6. The result of each image proceesing step is illustrated to show the process of extracting the centerline of the road. The red and green lines show two intersecting roads. The yellow "X" shows the computed location of the intersection.

Data Association

As previously visualized in Figure 2-8, the data association task shall match the observed landmarks (intersections included) in the aerial photos with the corresponding landmarks recorded in the reference map. Since the initial aircraft location is known and all sequential estimates are updated at each time step, the data association task is confined to a small region in the reference map. Unique to this research, data association occurs in the aerial photo's image plane coordinate system rather than in the reference map coordinate system. First, the landmarks in the reference map are projected through the measurement model, which is reused for the UKF measurement update step and will be discussed in detail in Chapter 4. The measurement model predicts where the landmarks should appear in the image plane, given the aircraft location, orientation, and camera properties. This step determines which landmarks are possibly visible in the image frame. The current location of the aircraft is the estimated position, which is determined by the UKF prediction step, Eq. (14). The landmarks projected from the reference map into the image plane will be referred to as possible landmarks.

Next, several different landmark attributes are measured to determine the confidence in the match. A scoreboard is constructed so that each observed landmark is scored based on similar attributes to possible landmarks. A match is determined by selecting the highest scoring possible landmark for every observed landmark. Table 2 displays an example scoreboard that matches observed buildings to possible buildings. In the example, observed building 1 matches with possible building 1, observed building 2 matches with possible building 3, and observed building 3 matches with possible building 4. The same scoreboard concept is extended to matching observed fields to possible fields and observed trees to possible trees.

Table 2. Example scoreboard for data assocaition.

		Possible Buildings				
		Building 1	Building 2	Building 3	Building 4	• • •
Observed Buildings	Building 1	4.3	-0.5	0.63	-0.88	• • •
	Building 2	-0.25	-0.5	4.25	0.25	• • •
	Building 3	0	-0.75	0.5	3.9	• • •
	⋮	⋮	⋮	⋮	⋮	⋱

The data association task is summarized in the outline below and will be explained in the following sections.

1. Associate observed **buildings** to possible **buildings**
   1.1. Compare and match **buildings** based on their distance to road
   1.2. Compare and match the nearest **building**-to-building vectors, **building**-to-field vectors, and **building**-to-tree vectors
   1.3. Match **buildings** to their closest expected (possible) **buildings**.
   1.4. Associate **buildings** within a distance constraint
2. Return to step 1 and replace the bold text with a different landmark category such as fields and trees.

Find the distance to the road

The first attribute measured for each landmark is the distance from the road, (when a road is detected) because the road serves as a ground reference to compare each landmark. The distance from the landmark to the road is calculated as the shortest distance (perpendicular distance) with the following equations based on Figure 3-7. Eq. (5) computes the directional unit vector of the road. Eq. (6) computes the shortest vector from the road to point $\vec{p}$. Eq. (7) is simply the L^2 norm of $\vec{d}$. Eq. (8) computes $\S$, which is a numerical flag with the values of

either a 1 or -1 to indicate which side of the road the landmark lays. If the *x* component of $\vec{d}$ is negative, then S becomes -1, thus the landmark is on the left side of the road; if the *x* component of $\vec{d}$ is positive, then S becomes 1, thus the landmark is on the right side of the road. For each side of the road, every possible landmark's distance is compared to each observed landmark within its category. The possible landmark with the minimum difference between distances to the road increments its score by 1.

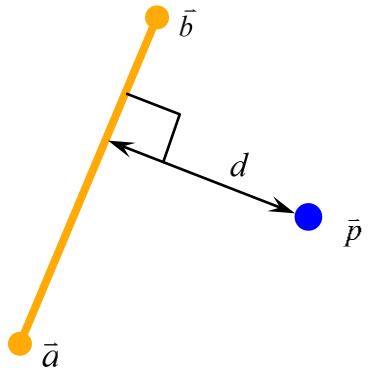


Figure 3-7. Shortest distance from point p to line segment a-b.

$$\hat{n} = \frac{\vec{b} - \vec{a}}{\left|\vec{b} - \vec{a}\right|} \tag{5}$$

$$\vec{d} = \left(\vec{p} - \vec{a}\right) - \left[\left(\vec{p} - \vec{a}\right) \cdot \hat{n}\right]\hat{n} \tag{6}$$

$$d = \left|\vec{d}\right| \tag{7}$$

$$s = \frac{\vec{d}_1}{\left|\vec{d}_1\right|} \tag{8}$$

Using Figure 3-8 as an example, the observed building, A, has a distance, d_A. This observed distance is subtracted from d_E and d_F. The absolute value of the differences show that $\left|d_A - d_E\right|$ is smaller than $\left|d_A - d_F\right|$, therefore the scoreboard table containing observed building

A and possible building E is incremented by 1. The above process is repeated for the other categories of fields and trees.

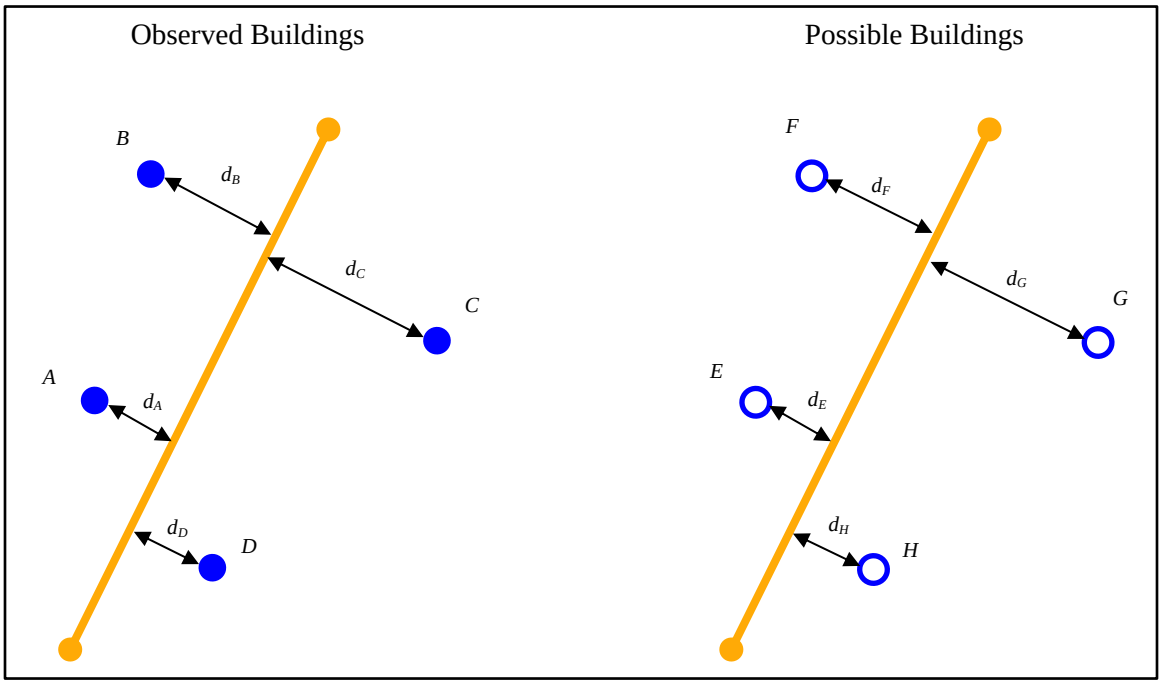


Figure 3-8. Example diagram of observed buildings and possible buildings.

Find the nearest buildings

The next attribute for associating observed landmarks to possible landmarks involves comparing the spatial relationships of different landmark categories. In particular, this step involves comparing vectors from each observed landmark of interest to the nearest buildings. The inspiration of matching vectors to other landmarks stems from high-dimensional SIFT or SURF feature vectors that describe the object of interest. In this case, the vectors simply describe a landmark's environment. Figure 3-9 illustrates an example of identifying an observed landmark of interest (building C) by matching its vectors to other landmarks with similar vectors drawn from the possible landmarks. Observed building C matches best with possible building G due to similar "spatial feature" vectors. Vectors drawn from every possible landmark are compared to

each observed landmark's descriptive vectors. Vector matching is evaluated by first normalizing to unit vectors and then computing the pairwise sum of squared distance between the observed building's set of feature vectors and the sets of feature vectors of all possible buildings. A threshold of 15% of the distance away from a perfect match is used to accept or reject the match. Multiple matches are allowed to occur, so if there are more observed buildings than possible buildings, then multiple observed buildings may match with a single possible building. For example, if a fourth building, D, is detected near building B, then both B and D is allowed to match with possible building F. Vector matching is computed with MATLAB's MatchFeatures() function, which was initially introduced in version R2011a. The amount of matched vectors is tallied and contributes to the score. A greater number of matching vectors contribute to a higher score. Points are awarded by adding the amount of matches divided by the number of possible buildings. Referring to Figure 3-9, two vectors $\vec{A}$ and $\vec{B}$ best match with the two vectors $\vec{E}$ and $\vec{F}$. Therefore the score that associates observed building C to possible building G increases by $\frac{2}{3}$. If vector $\vec{E}$ matches with vector $\vec{A}$, then the score that associates observed building C to possible building F increases by $\frac{1}{3}$.

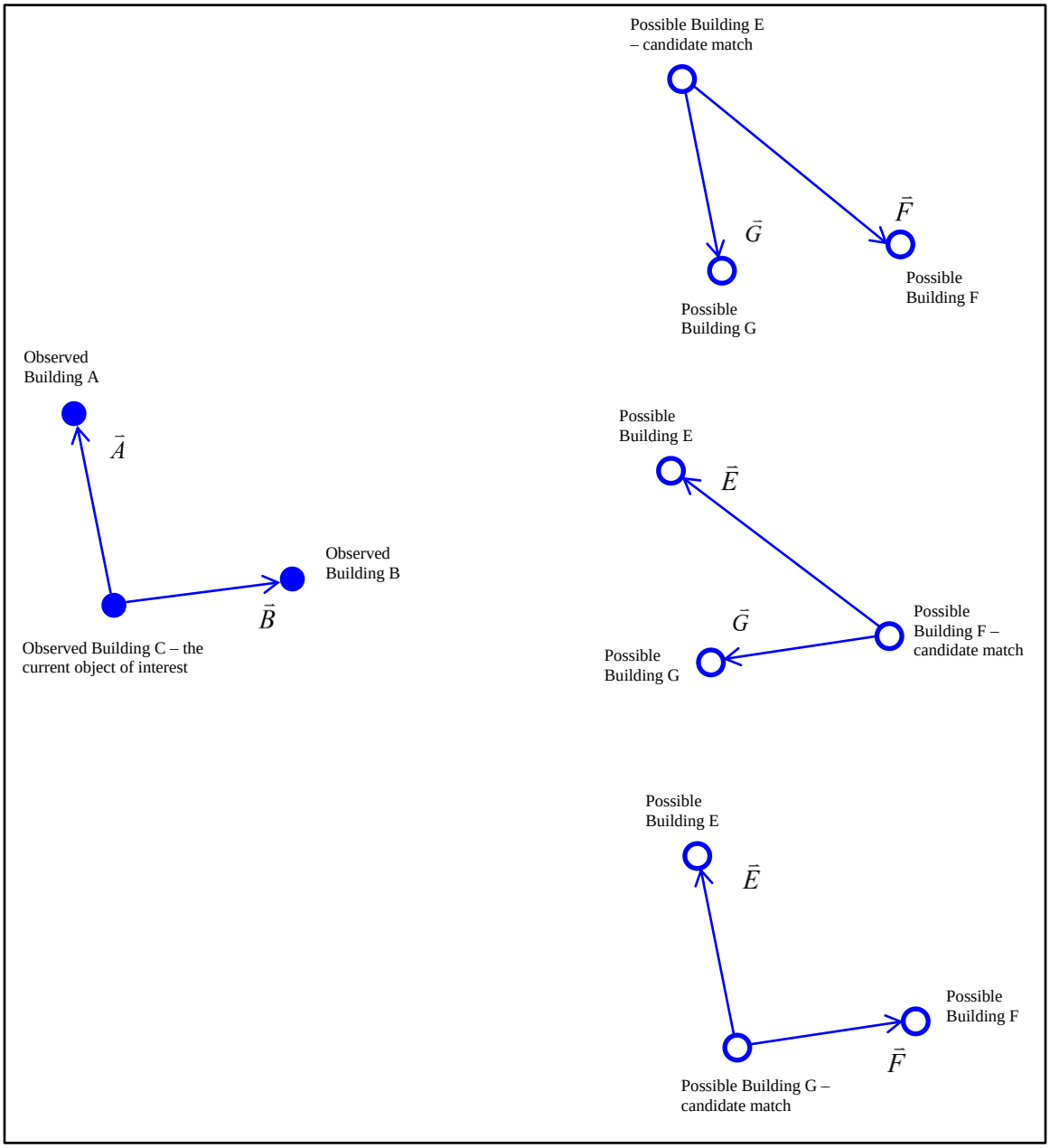


Figure 3-9. Example illustration comparing feature vectors from an object of interest (a builing in this case) to buildings. Observed building C is best matches with possible building G.

The same vector matching concept is extended to other categories of landmarks too. A building is also described by vectors to trees and fields as shown in Figure 3-10.

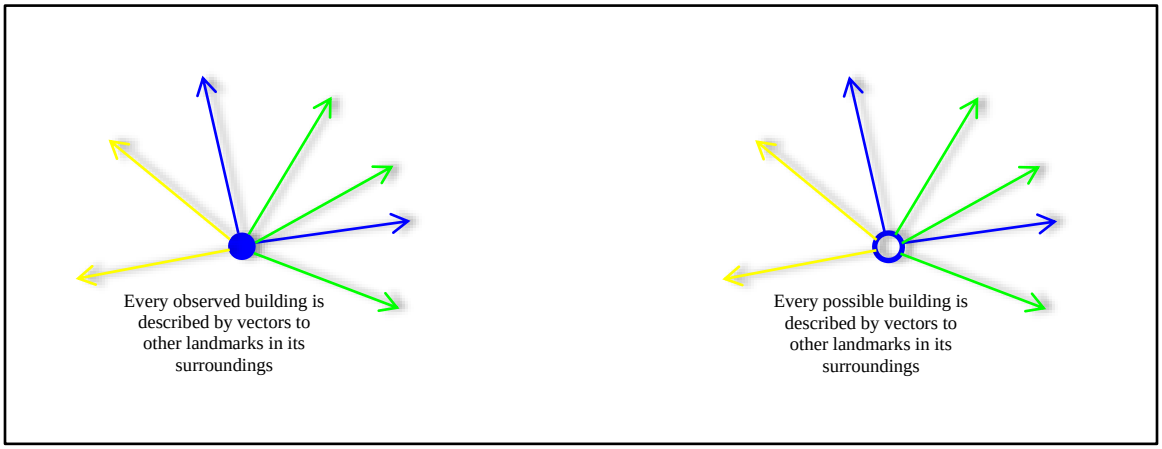


Figure 3-10. Each landmark is described by its spatial relationship to other landmarks.

Find the nearest fields

This step is similar to the step described in the previous section except that it compares vectors from each observed landmark of interest to the nearest fields (represented by yellow vectors in Figure 3-10). When flying over areas where fields are more abundant than buildings, the vector description of each landmark is limited to eight closest field neighbors. The limit reduces the influence of distant fields and confines the description to a local region.

Find the nearest trees

This step is similar to the step described in the previous section except that it compares vectors from each observed landmark of interest to the nearest trees (represented by green vectors in Figure 3-10). When flying over areas where trees are more abundant than buildings, the vector description of each landmark is limited to eight closest tree neighbors. Again, this limit reduces the influence of distant trees and confines the description to a local region.

Find the closest expected (possible) landmark of the same category

Matching observed landmarks to the nearest possible landmark of the same type is a naïve criterion that will not work without the other scoring qualifiers previously discussed. If used as the only data association criterion, observed landmarks will be assigned to the wrong landmarks in the reference map due to localization drift. Wrong associations cause inaccurate measurements and compound the localization error. When used with the other scoring criteria, this method is useful because it serves as an additional qualifier that can either accept or reject a match. If the distance to the closest matching landmark exceeds a threshold[*], then its score is demoted by –1, but if the distance is below the threshold, then its score increments by +1.

Associate the landmarks with the final score

After all above criteria have been evaluated, the observed landmarks are matched to possible landmarks based on their final scores. Referring back to the conceptual scoreboard example in Table 2, each row is matched to the column that contains the highest score.

The last qualifier for accepting or rejecting a match is a final distance threshold[†]. If the associated landmarks differ by a Euclidean distance that exceeds a threshold, then the match is rejected and will be excluded from the measurement matrix. Distance thresholds may need adjustments depending on the image resolution, focal length, and expected flight altitude. The measurement matrix has $n \times 4$ elements where n is the number of associated landmarks. Each row in the final measurement matrix contains the (x, y) coordinates of the observed landmark in aerial

[*] In the simulations later presented in Chapter 5, a distance threshold of 75 pixels is used for 640 by 480 pixel images and a threshold of 175 pixels is used for 1280 by 960 pixel images.

[†] A distance threshold of 125 pixels was chosen in the simulations when capturing photos with dimensions of 1280 by 960 pixels.

image coordinate frame and the (x, y) coordinates of the associated landmark in the reference map coordinate frame.

The above six subheadings are repeated for associating fields and trees. A working example of data association results is depicted in Figure 3-11. Blue circles indicate the location of observed buildings, yellow squares indicate the location of observed fields, and green asterisks indicate the location of observed trees. White circles indicate the possible location of buildings projected from the reference map, white squares indicate the possible location of fields projected from the reference map, and white asterisks indicate the location of trees projected from the reference map. The connecting lines show the final matching decisions based on the scoring criteria discussed above. Figure 3-12 displays a small region of the reference map preprocessed before the flight for comparison to the aerial photo in Figure 3-11.

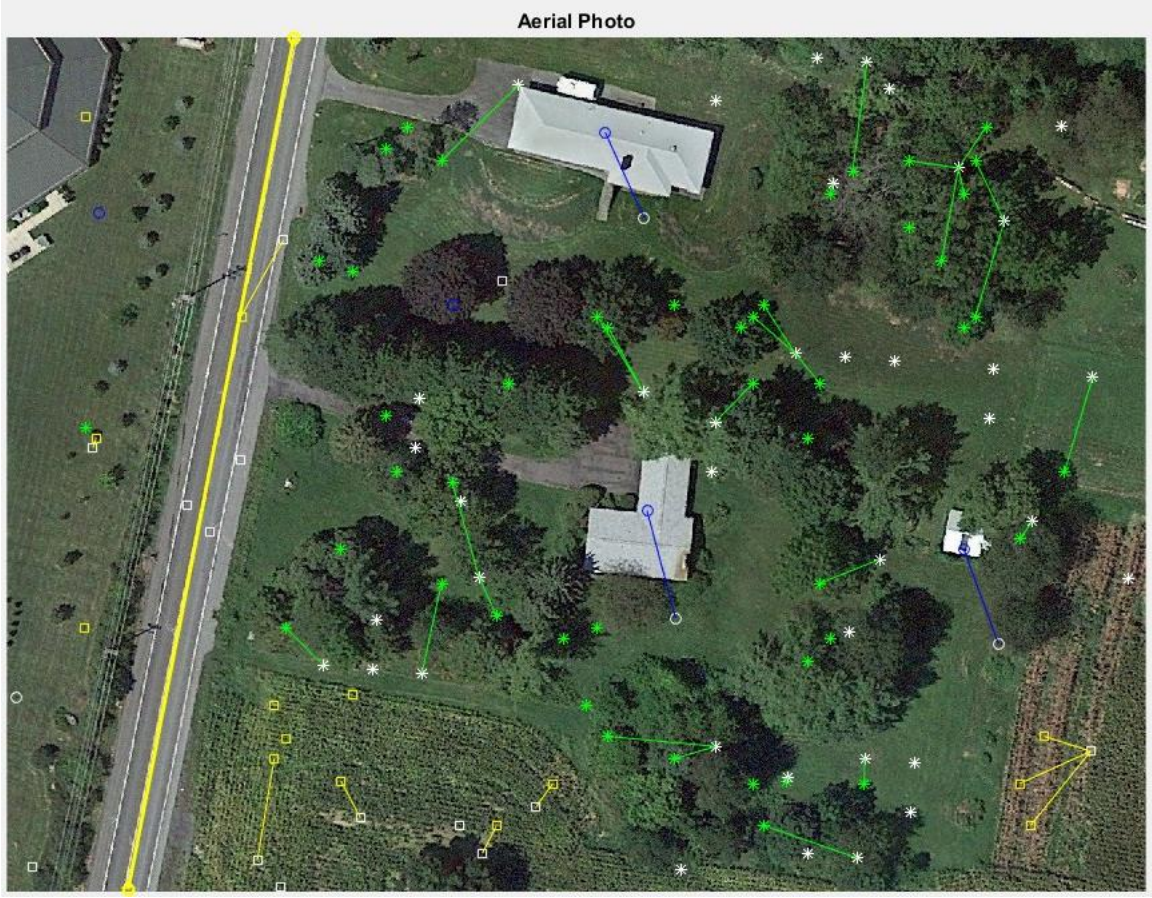


Figure 3-11. The connecting lines show which observed landmarks have been associated with possible landmarks (white markers).

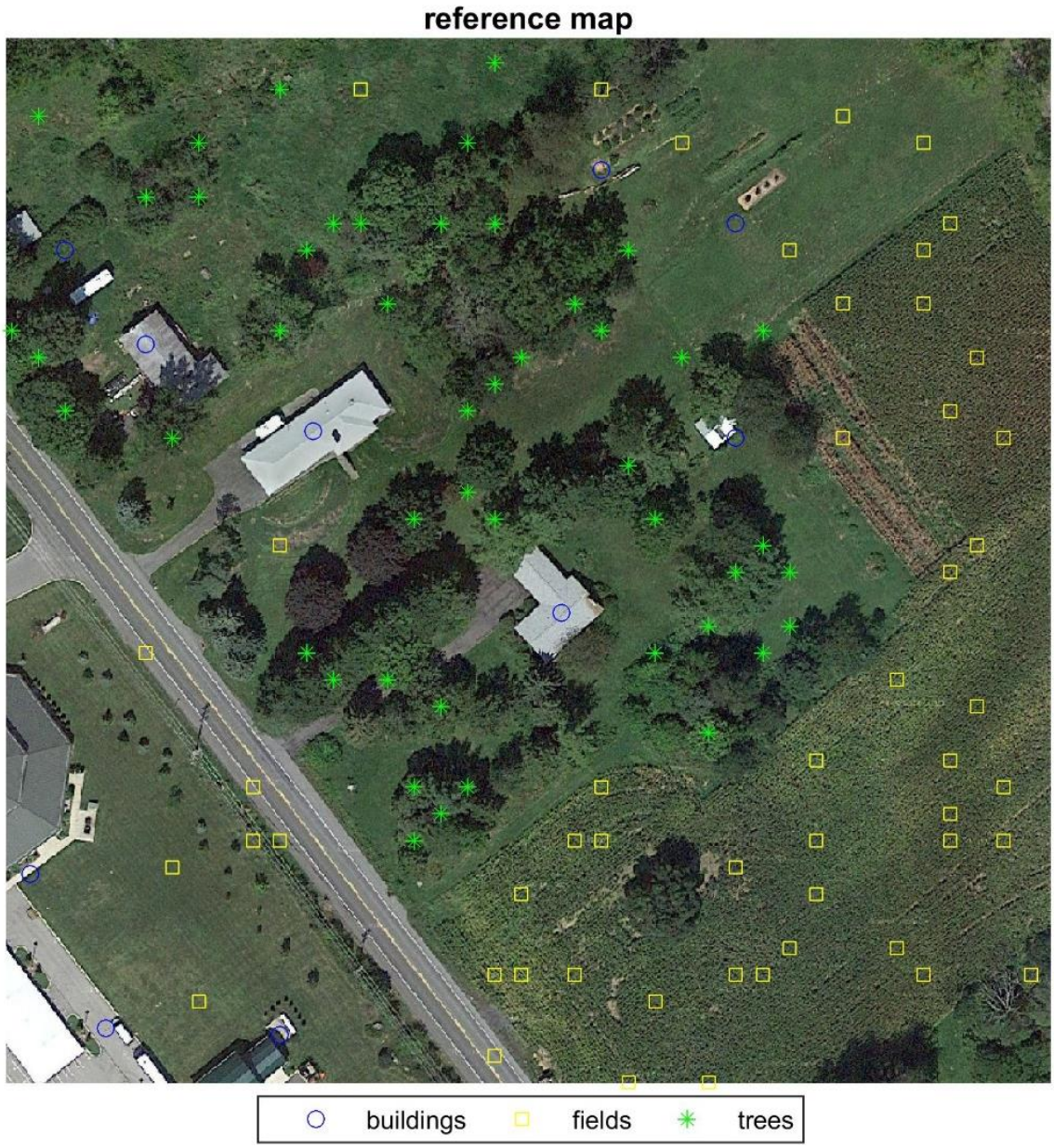


Figure 3-12. Processed refence map of detected landmarks overlaying the original imagery.

Chapter 4

State Estimation

State estimation is the mathematical prediction of a system state governed by a plant model, followed by a correction, which is governed by a measurement model. In the context of vehicle localization, the plant model is the kinematic equations that model the vehicle's motion. The measurement model is the set of equations that model the physical sensors. Both sets of system equations are highly nonlinear and contain trigonometric functions. Because of these nonlinearities, only state estimators designed to handle nonlinear system equations are considered. Langelaan [23] and Huster [67] compared the particle filter, extended Kalman filter (EKF) and Unscented Kalman Filter (UKF), which is also referred as a Sigma Point Filter, for a similar application to robotic vehicles. The particle filter is capable of modeling arbitrary probability distributions and more accurately account for uncertainties for nonlinear systems. As the number of particles used to represent the probability distribution increases, particle filter accuracy increases at the cost of increased computation. The EKF approximates the nonlinear system equations typically with a first-order Taylor series about the current best estimate. Langelaan states, "This linearization introduces errors in several ways: first, a potentially highly nonlinear system is approximated with a linear model; second, the linearization occurs about an uncertain state, hence the linearization itself may be incorrect" [23].

Instead of approximating nonlinear system equations, the UKF evaluates a reduced set of particles through the full nonlinear system equations and approximates the probability distribution of the estimated state. Unlike the particle filter, however, the reduced set of particles model a Gaussian probability distribution function. This reduced set of particles is referred as sigma points. In other words, sigma points, which are states that capture the mean and covariance of a random vector $X = N(\bar{x}, P)$, and are propagated through full nonlinear system equations

describing the motion of the vehicle to obtain an estimated mean and covariance [23]. That is the prediction step. When an actual sensor measurement arrives, the sigma points are then evaluated through a measurement model that predicts measurement values given the sigma points. The estimated measurements are then averaged (weighted) and subtracted from the actual noisy measurement. The difference is multiplied by the Kalman gain, *K*, and is added to the estimated states. The covariance is also updated during the update step. The algorithm for the UKF is reprinted in Table 3 from [23] and [68] below.

Table 3. UKF Algorithm

Initialize with $\hat{\boldsymbol{x}}_0$ and $\boldsymbol{P}_{0\|0}$
For t_k, $k \in (1,\ldots,\infty)$ compute sigma points:
$$\mathbf{X}_{k-1\|k-1} = \left[\hat{\mathbf{x}}_{k-1\|k-1} \quad \hat{\mathbf{x}}_{k-1\|k-1} + \eta\sqrt{\mathbf{P}_{k-1\|k-1}} \quad \hat{\mathbf{x}}_{k-1\|k-1} - \eta\sqrt{\mathbf{P}_{k-1\|k-1}}\right] \quad (9)$$ There are $2N+1$ sigma points, where N is the dimension of the state vector. In this algorithm, η is a weight factor, $\boldsymbol{w}_m$ is a vector of weights, $\boldsymbol{W}_c$ is a diagonal matrix of weights, $\boldsymbol{1}$ is a $(1\times 2N+1)$ matrix of ones, $\boldsymbol{Q}$ is process noise matrix, and $\boldsymbol{R}$ is measurement noise matrix. The weight factors are calculated as:
$$\eta = \alpha\sqrt{N} \quad (10)$$ The constant, α, is a parameter which determines the spread of the sigma points. Typically $10^{-4} \le \alpha \le 1$. The weight factor $\boldsymbol{w}_m$ and weight matrix $\boldsymbol{W}_c$ are:
$$\mathbf{w}_{m,1} = \frac{\alpha^2 - 1}{\alpha^2} \qquad \mathbf{w}_{m,i} = \frac{1}{2N\alpha^2} \quad (11)$$ $$\mathbf{W}_{c,1} = \frac{\alpha^2 - 1}{\alpha^2} + \left(1 - \alpha^2 + \beta\right) \qquad \mathbf{W}_{c,i} = \frac{1}{2N\alpha^2} \quad (12)$$ where $i = 2,\ldots,(2N+1)$. The parameter β incorporates prior knowledge of the distribution of the state vector. For Gaussian distributions, $\beta = 2$ is optimal.
Time update (prediction):
$$\mathbf{X}_{k\|k-1} = f\left(\mathbf{X}_{k-1\|k-1}, \mathbf{u}_p\right) \quad (13)$$ $f(\,)$ is the function that represents the kinematic aircraft model
$$\hat{\mathbf{x}}_{k\|k-1} = \mathbf{X}_{k\|k-1}\mathbf{w}_m \quad (14)$$

$$\mathbf{P}_{k|k-1} = \left[\mathbf{X}_{k|k-1} - \mathbf{x}_{k|k-1}\mathbf{1}\right]^T \mathbf{W}_c \left[\mathbf{X}_{k|k-1} - \mathbf{x}_{k|k-1}\mathbf{1}\right] + \mathbf{Q} \quad (15)$$

Measurement update (correction):

$$\mathbf{Z}_{k|k-1} = h\left(\mathbf{X}_{k|k-1}, \mathbf{u}_m\right) \quad (16)$$

$h(\ )$ is the function that represents the measurement model.

$$\hat{\mathbf{z}}_{k|k-1} = \mathbf{Z}_{k|k-1}\mathbf{w}_m \quad (17)$$

$$\mathbf{P}_{zz} = \left[\mathbf{Z}_{k|k-1} - \hat{\mathbf{z}}_{k|k-1}\mathbf{1}\right]^T \mathbf{W}_c \left[\mathbf{Z}_{k|k-1} - \hat{\mathbf{z}}_{k|k-1}\mathbf{1}\right] + \mathbf{R} \quad (18)$$

$$\mathbf{P}_{xz} = \left[\mathbf{X}_{k|k-1} - \mathbf{x}_{k|k-1}\mathbf{1}\right]^T \mathbf{W}_c \left[\mathbf{Z}_{k|k-1} - \hat{\mathbf{z}}_{k|k-1}\mathbf{1}\right] \quad (19)$$

$$\mathbf{K} = \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1} \quad (20)$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}\left(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}\right) \quad (21)$$

$\mathbf{z}_k$ is the vector of the actual measurement

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}\mathbf{P}_{zz}\mathbf{K}^T \quad (22)$$

The process noise covariance represented by the matrix, **Q**, adds uncertainty to the covariance matrix during the prediction step to account for motion modeling errors. Low values of **Q** elements indicate more confidence in the process model (prediction step). The measurement noise covariance represented by the matrix, **R**, adds uncertainty to the measurement covariance matrix during the correction step to account for measurement modeling errors. Low values of **R** elements indicate more confidence in the measurement model (correction step).

UKF Process Model (Prediction Step)

The UKF prediction step in Eq. (13) uses the following discrete-time nonlinear system given by Eq. (23) and is integrated forward in time with Euler numerical integration. $\mathbf{w}_k$ is a

vector whose elements represent the process noise terms, which are added to the aircraft velocity components and the wind components. $\mathbf{w}_k$ is assumed to have zero mean and **Q** covariance as shown in Eq. (24).

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, t_k) + \mathbf{w}_k \quad (23)$$

$$\mathbf{w}_k \sim (0, \mathbf{Q}) \quad (24)$$

$\mathbf{x}$ is the state vector with elements shown in Eq. (25) where x and y define the location of the aircraft in the local geographic reference frame, and W_w and W_n define the wind velocities from the west and north directions. $\mathbf{u}$ is the control input vector with elements shown in Eq. (26) and represent the true airspeed and yaw angle as measured from the autopilot.

$$\mathbf{x} = [x \quad y \quad W_w \quad W_n]^T \quad (25)$$

$$\mathbf{u} = [V \quad \psi]^T \quad (26)$$

For the following equations to make sense, a coordinate system is defined in Figure 4-1 to be consistent with maps where north is up and east is to the right. Heading angles are measured from north and increase clockwise. The positive Y axis points down because digital imaging software defines the origin at the top left corner of images.

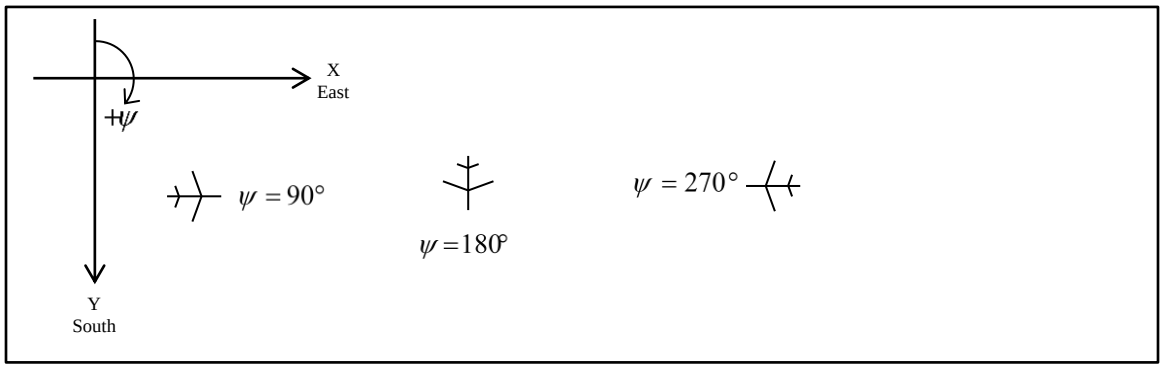


Figure 4-1. Global coordinate system

Eq. (23) is expanded to the simple kinematic model that describes the motion of an airplane flown by the autopilot. The noise terms, w_x, w_y, w_w, and w_n, are assumed to be uncorrelated with covariance represented by the diagonal elements of the **Q** matrix.

$$\dot{x} = V\sin(\psi) + W_w + w_x \quad (27)$$

$$\dot{y} = -V\cos(\psi) + W_n + w_y \quad (28)$$

$$\dot{W}_w = w_w \quad (29)$$

$$\dot{W}_n = w_n \quad (30)$$

UKF Measurement Model (Correction Step)

The measurement model predicts the estimated position of the where the landmark (building, field, tree, road intersection) should appear in the image frame by transforming the landmark's actual 3D world coordinates to 2D image frame coordinates given the aircraft's position, attitude, altitude, and camera intrinsics. The input vector, $\mathbf{u_m}$, contains additional elements such as the aircraft altitude above ground level (AGL), roll angle, pitch angle, yaw angle, lens focal length, and the actual coordinates of the landmarks in the world reference map.

$$\mathbf{u_m} = \begin{bmatrix} h & \phi & \theta & \psi & \lambda & \vec{L}_i^W \end{bmatrix} \quad (31)$$

Eq. (32) uses transformation matrices in homogeneous form to transform landmark positions in the 3D world frame to the 3D camera frame. Conversion of homogeneous coordinates in the camera frame to 2D image plane coordinates is accomplished by dividing the first two elements of the 4×1 column vector, $\vec{\mathbf{L}}_\mathbf{i}^\mathbf{C}$, by its fourth element. The predicted measurement is the 2D image plane coordinates. More information about image transformations in homogeneous coordinates can be found in [69] and each transformation matrix will be discussed in detail below. Multiple landmarks may appear in a single image frame, therefore the subscript, *i*, is shown to index each landmark. $\mathbf{T}^\mathbf{A}$ simply transforms the origin from the center of the camera frame to the upper left corner, which is the standard image coordinate system for plotting images.

$$\vec{\mathbf{L}}_\mathbf{i}^\mathbf{C} = \mathbf{T}^A\mathbf{T}^P\mathbf{T}^R\mathbf{T}^\phi\mathbf{T}^\theta\mathbf{T}^\psi\mathbf{T}^O\vec{\mathbf{L}}_i^W \quad (32)$$

Figure 4-2 illustrates an aircraft flying over landmarks. The $\vec{\mathbf{L}}_i^W$ vectors define the coordinate location of the landmarks which must be transformed from world coordinates to camera coordinates with the Eq. (32) above.

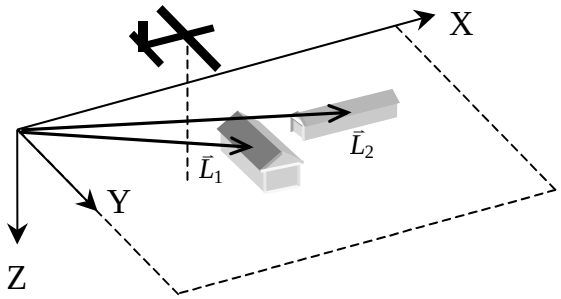


Figure 4-2. When the detected landmarks are in the field-of-view of the camera, their world coordinates must be transformed into image plane coordiates.

The first transformation matrix is defined by Eq. (33) and translates the aircraft to the origin of the world coordinate frame with zero rotation angles before proceeding with rotations. Figure 4-3 graphically shows the translations.

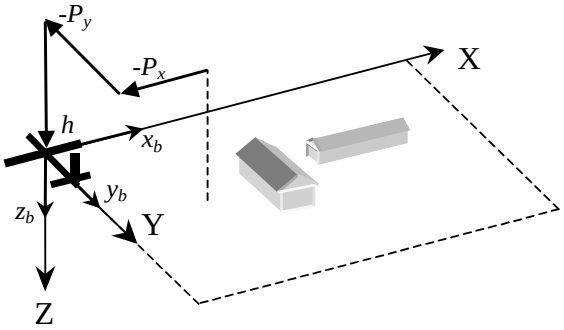


Figure 4-3. The aircraft is translated back to the origin of the world coordinate system.

$$T^O = \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{33}$$

Next, the aircraft body coordinates undergo a series of three rotations. The first rotation is along the *Z* axis by angle ψ , which is depicted in Figure 4-4 and Eq. (34).

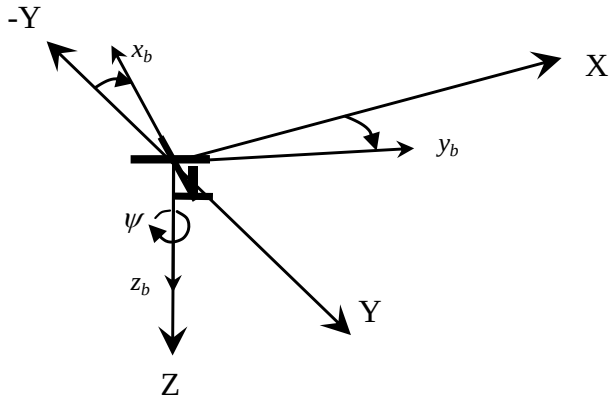


Figure 4-4. Aircraft body rotation about the *Z*-axis by amount ψ .

$$T^{\psi} = \begin{bmatrix} \sin(\psi) & -\cos(\psi) & 0 & 0 \\ \cos(\psi) & \sin(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

Then the body coordinates are rotated long y_b by amount θ to pitch the nose of the aircraft as seen in Figure 4-5 and Eq. (35).

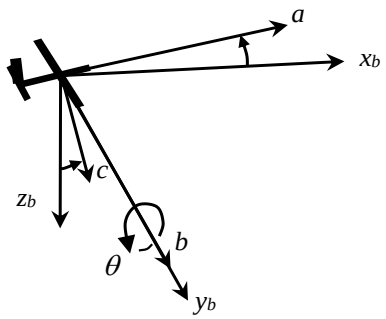


Figure 4-5. Aircraft body rotation about the y_b-axis by amount θ.

$$T^{\theta} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{35}$$

The final aircraft body rotation is along the intermediate axis, *a*, and is rotated by amount ϕ to roll the aircraft as seen in Figure 4-6 and Eq. (36).

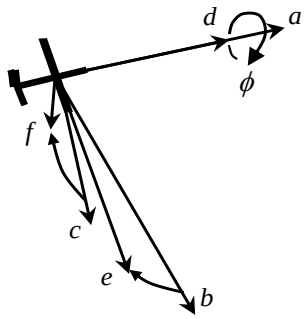


Figure 4-6. Aircraft body rotation about the *a*-axis by amount ϕ.

$$T^{\phi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) & 0 \\ 0 & -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{36}$$

Next, the camera undergoes a $90°$ rotation so that the image frame's x-axis is parallel to the aircraft body y-axis, and the image frame's y-axis is parallel to the aircraft body –x-axis. The z-axis of the camera frame and the aircraft body frame coincide. The camera rotation is depicted in Figure 4-7 and Eq. (37). This rotation effectively defines North along the –Y axis and East (90°) along the +X axis in Figure 4-3 so that heading angles are measured from north. To model a forward-looking camera, an additional rotation matrix can be inserted between $\mathbf{T}^{P}$ and $\mathbf{T}^{R}$ in Eq. (32) for a rotation along the x_c axis. A positive rotation along the x_c axis tilts the camera's z-axis forward.

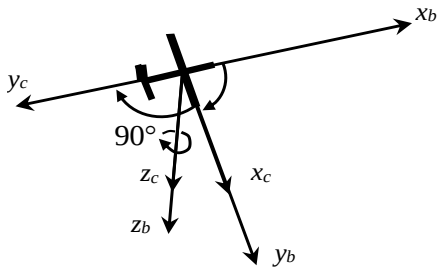


Figure 4-7. The camera coordinate system is rotated about the z_b-axis by 90 degrees.

$$T^R = \begin{bmatrix} \cos(90°) & \sin(90°) & 0 & 0 \\ -\sin(90°) & \cos(90°) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{37}$$

The following rotation is the perspective transformation, which is also known as the imaging transformation. The perspective transform projects 3D points onto a plane, effectively scaling the field-of-view of the scene with the focal length, λ, as modeled by Figure 4-8.

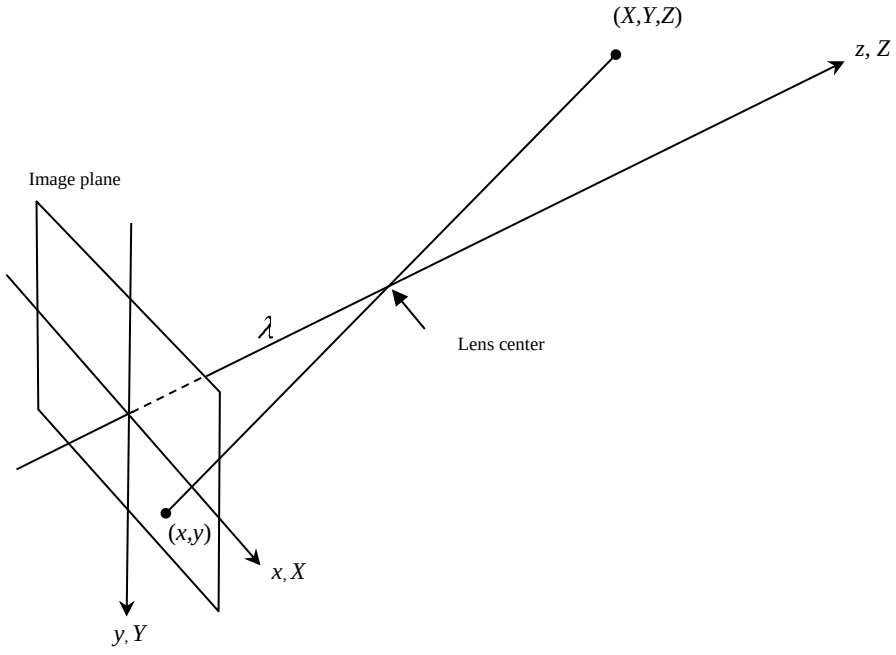


Figure 4-8. Simplified model of the imaging process.

Assuming $Z > \lambda$ and all points of interest lie in front of the lens, 3D points in the world become 2D points in the image plane by similar triangles:

$$\frac{x}{\lambda} = \frac{-X}{Z-\lambda} \tag{38}$$

$$\frac{y}{\lambda} = \frac{-Y}{Z-\lambda} \tag{39}$$

Since objects would appear inverted in the image plane, the *x* and *y* coordinates are multiplied by -1.

$$x = \frac{\lambda X}{Z-\lambda} \tag{40}$$

$$y = \frac{\lambda Y}{Z-\lambda} \tag{41}$$

Eq. (42) defines the perspective transformation matrix, which models Figure 4-8, in homogeneous coordinates.

$$T^P = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/\lambda & 1 \end{bmatrix} \tag{42}$$

To prove how Eq. (42) transforms 3D world coordinates into a 2D image plane as modeled in Figure 4-8, consider a world point in Cartesian coordinates (*X*,*Y*,*Z*) defined in homogeneous coordinates as (*kX*,*kY*,*kZ*,*k*), where *k* is an arbitrary, nonzero constant. After multiplying the perspective transformation matrix and the homogenous coordinates together in Eq. (43), the first three rows of the product must be divided by the fourth row to obtain image plane coordinates in Cartesian form. As a result, Eq. (44) and Eq. (45) are respectively equivalent to Eq. (40) and Eq. (41), and therefore show how the perspective transformation matches the model in Figure 4-8.

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/\lambda & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} = \begin{bmatrix} -kX \\ -kY \\ -kZ \\ \frac{-kZ}{\lambda} + k \end{bmatrix} \tag{43}$$

To obtain 2D image plane coordinates in Cartesian form, divide the right side of Eq. (43) by its fourth row.

$$x = \frac{-kX}{\frac{-kZ}{\lambda} + k} = \frac{\lambda X}{Z - \lambda} \tag{44}$$

$$y = \frac{-kY}{\frac{-kZ}{\lambda} + k} = \frac{\lambda Y}{Z - \lambda} \tag{45}$$

Next, the origin of the image frame must be shifted from the center to the upper left corner to match the image coordinate system imaging software such as MATLAB (illustrated in Figure 4-9).

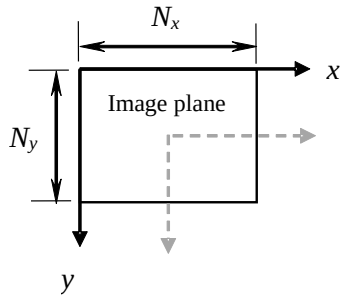


Figure 4-9. Image plane coordinate translation

The coordinate is shifted by Eq. (46) where N_x and N_y represent the image resolution in the horizontal and vertical directions, respectively. For a 640 by 480 pixel image, $N_x=640$, and $N_y=480$.

$$T^A = \begin{bmatrix} 1 & 0 & 0 & N_x/2 \\ 0 & 1 & 0 & N_y/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{46}$$

Finally, after computing the $\bar{\mathbf{L}}_\mathbf{i}^\mathbf{C}$ vector, its first two rows must be divided by the fourth row to convert from homogenous coordinates to Cartesian coordinates. The final Cartesian

coordinate form is the predicted *x*-coordinate and *y*-coordinate of the landmark in the 2D image plane. These predicted 2D landmark coordinates are the outputs of the function $h(\ )$ in Eq. (16), and are then eventually compared to actual imaging sensor measurements in Eq. (21).

$\mathbf{z} = \begin{bmatrix} {}^{I}x & {}^{I}y \end{bmatrix}^{T}$ (pixels) is the measurement vector representation for landmarks and intersections.

The measurement model for the road localization is similarly constructed but models the diagram previously seen in Figure 2-3. Instead of transforming one (*x,y)* map coordinate per landmark, two (*x,y)* coordinates, which define the endpoints of the road segment in the map, are transformed with Eq. (32) from map coordinates to image frame coordinates. Next, the direction (unit vector, $\hat{\mathbf{n}}$) of the road and the shortest Euclidian distance, d (in pixels), from a point $\vec{\mathbf{p}}$ are computed by Eq. (47) and Eq. (49). In this case, point $\vec{\mathbf{p}}$ is the origin of the 2-D image frame.

$$\hat{\mathbf{n}} = \frac{\vec{\mathbf{L}}_2^C - \vec{\mathbf{L}}_1^C}{\left\| \vec{\mathbf{L}}_2^C - \vec{\mathbf{L}}_1^C \right\|} \tag{47}$$

$$d = \left\| \left(\vec{\mathbf{p}} - \vec{\mathbf{L}}_1^C \right) - \left[\left(\vec{\mathbf{p}} - \vec{\mathbf{L}}_1^C \right) \bullet \hat{\mathbf{n}} \right] \hat{\mathbf{n}} \right\| \tag{48}$$

To summarize, the road measurement model predicts the direction and location of the where the road line segment should appear in the image frame given the current aircraft orientation and aircraft estimated position. This prediction is compared to an actual measurement during the measurement update step in the UKF. $\mathbf{z} = \begin{bmatrix} d & \vec{n} \end{bmatrix}$ (pixels, 2×1 unit vector) is the measurement vector representation for road localization. When distinct landmarks are not detectable (e.g. flight between distinct landmarks), the distance to and the direction of the road are the measurements obtained from the aerial images.

Hardware-in-the-loop simulations discussed in the next chapter incorporate GPS measurements, which are regarded as truth data. Wind velocity estimates are computed within the Piccolo Plus autopilot, but the algorithm is unknown and not documented (proprietary). Therefore a separate UKF is constructed in this research to estimate the wind velocity based on GPS

measurements. The measurement model for GPS measurements is simply the time-updated (predicted) position generated from the process model. The goal is to compare wind velocity estimates based on GPS measurements against wind velocity estimates based on landmark measurements. Both of those estimates are also compared to the wind velocity computed in the Piccolo autopilot.

Acceptable values for **Q** and **R** are displayed in Table 4 and were obtained experimentally with the simulations. The terms along the diagonal represent notional noise covariances. The actual noise is not readily determined and the matrices should be tuned through simulation more thoroughly. Simulations have shown that lower values lead to smoother wind estimate solutions that are less susceptible to abrupt transient changes when measurements are obtained.

Table 4. Acceptable values for Q and R for the simulations demonstrated in this research.

For Distinct Landmark Detection	$\mathbf{Q}=\begin{bmatrix} 9\,\text{ft}^2 & 0 & 0 & 0 \\ 0 & 9\,\text{ft}^2 & 0 & 0 \\ 0 & 0 & 0.2\left(\text{ft}/2\right)^2 & 0 \\ 0 & 0 & 0 & 0.2\left(\text{ft}/2\right)^2 \end{bmatrix}$	$\mathbf{R}=\begin{bmatrix} 1600\,\text{px}^2 & 0 \\ 0 & 1600\,\text{px}^2 \end{bmatrix}$
For Road Localization	$\mathbf{Q}=\begin{bmatrix} 9\,\text{ft}^2 & 0 & 0 & 0 \\ 0 & 9\,\text{ft}^2 & 0 & 0 \\ 0 & 0 & 4\left(\text{ft}/2\right)^2 & 0 \\ 0 & 0 & 0 & 4\left(\text{ft}/2\right)^2 \end{bmatrix}$	$\mathbf{R}=\begin{bmatrix} 1600\,\text{px}^2 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$
For Generic Landmark Detection	$\mathbf{Q}=\begin{bmatrix} 400\,\text{ft}^2 & 0 & 0 & 0 \\ 0 & 400\,\text{ft}^2 & 0 & 0 \\ 0 & 0 & 1\left(\text{ft}/2\right)^2 & 0 \\ 0 & 0 & 0 & 1\left(\text{ft}/2\right)^2 \end{bmatrix}$	$\mathbf{R}=\begin{bmatrix} 2500\,\text{px}^2 & 0 \\ 0 & 2500\,\text{px}^2 \end{bmatrix}$
For Wind Velocity Estimates based on GPS	$\mathbf{Q}=\begin{bmatrix} 400\,\text{ft}^2 & 0 & 0 & 0 \\ 0 & 400\,\text{ft}^2 & 0 & 0 \\ 0 & 0 & 0.4\left(\text{ft}/2\right)^2 & 0 \\ 0 & 0 & 0 & 0.4\left(\text{ft}/2\right)^2 \end{bmatrix}$	$\mathbf{R}=\begin{bmatrix} 25\,\text{ft}^2 & 0 \\ 0 & 25\,\text{ft}^2 \end{bmatrix}$

Chapter 5

Simulation

Flight simulations are conducted on the Cloud Cap Technologies Piccolo Simulator (Figure 5-1) and model a Sig Kadet Senior single-engine airplane (Figure 5-2) flying over various regions of Pennsylvania. Hardware-in-the-loop flight simulation sensor data is recorded from a Piccolo Plus autopilot, (Figure 5-3). Various flight maneuvers such as straight-and-level flight, and S-turns are conducted while Piccolo sensor telemetry data is recorded during the flight simulation. The position information recorded from the flight simulator is considered truth data when compared to the localization methods presented in this research. Wind can be simulated by entering nonzero values for each principle direction. Flight progress can be monitored in the Piccolo Command Center, which is depicted in Figure 5-4. A benefit of generating flight simulation data from the Piccolo autopilot is that the method could be later flight tested using the same hardware.

After the flight simulations are conducted, the recorded Piccolo telemetry data is used as input data for post processing in MATLAB. MATLAB code generates realistic aerial photographs based on the true location and orientation of the airplane with a downward-pointing camera at each time step. A more thorough discussion about how the images are generated is included in the next section called *Aerial Camera Simulator*. Separate image processing functions then scan each photograph for landmarks and roads. Landmark coordinates, orientation information, airspeed, and altitude are then sent to a separate UKF function that estimates the aircraft position and wind velocity.

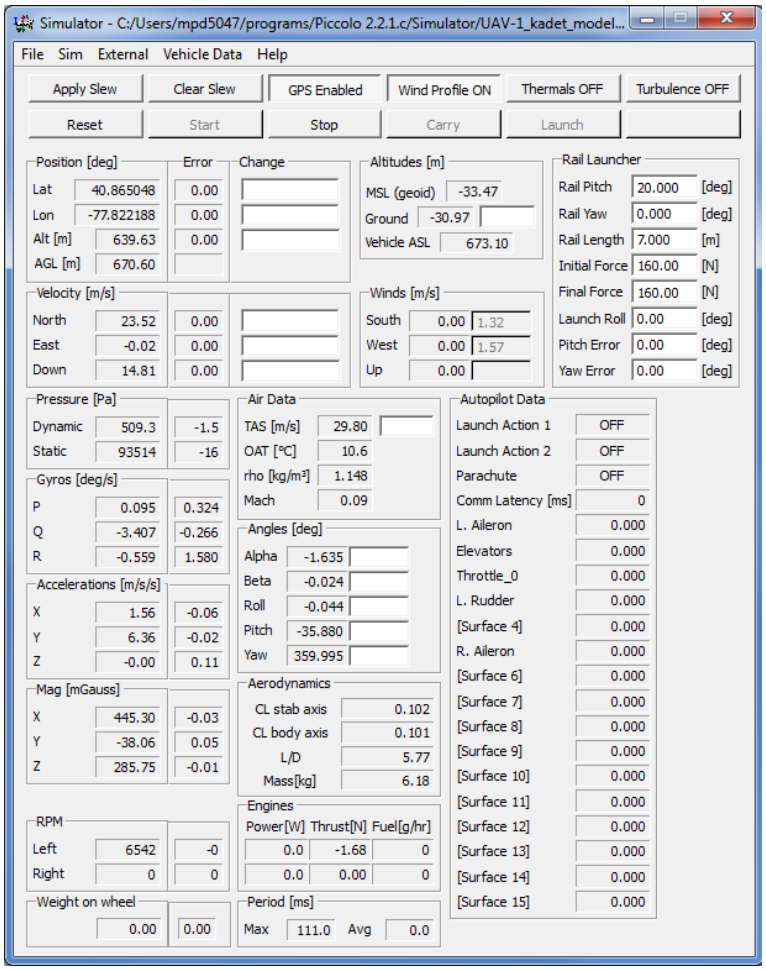


Figure 5-1. Screenshot of the Piccolo Simulator.

Figure 5-2. Photograph of the Sig Kadet Senior airplane.

Figure 5-3. Piccolo Plus Autopilot

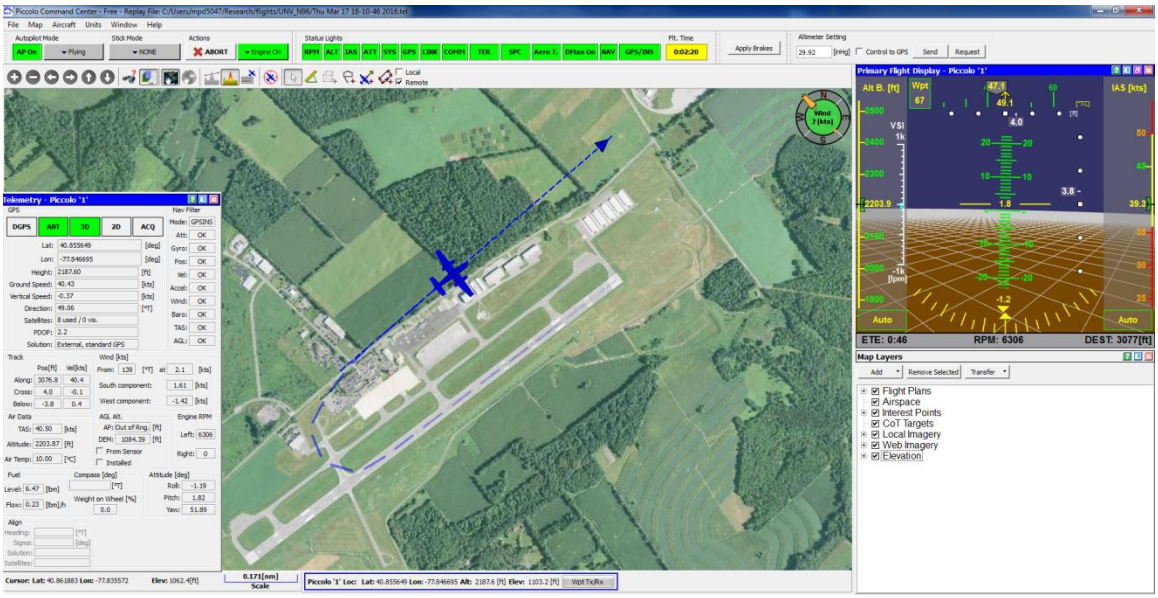


Figure 5-4. Screenshot of the Piccolo Command Center

Aerial Camera Simulator

A camera is modeled to take aerial photos as the aircraft flies its route. In the simulation, the camera is fixed to the origin of the aircraft body axes and returns a digital image of the scene subjected to $x, y, h, \phi, \theta, \psi, \lambda$, and resolution. The scenery is modeled in MATLAB by plotting a

surface plot of the ground with a large high resolution satellite image as the texture. Imagery is simulated from Google Maps and high resolution othoimagery obtained from the United States Geological Survey (USGS). To determine the Google Maps resolution, 54 pixels were counted along a displayed scale indicating a length of 20 feet. Therefore, the Google Maps imagery equates to a resolution of 0.37 feet. The resolution of the USGS orthoimagery is reported as 1 foot.

MATLAB's graphics engine translates and rotates the camera to match the aircraft state. The aerial photographs take into account the roll, pitch, and yaw of the aircraft to display a realistic perspective of the ground in the image plane. Table 5 lists the necessary MATLAB graphics engine functions that render an aerial perspective of the high resolution surface plot. The functions translate and rotate the graphics engine camera to render images consistent with the location and orientation of the airplane.

Table 5. MATLAB functions used for rendering aerial images from a downward-pointing camera.

MATLAB function	**Explanation**
camproj('perspective')	Sets a perspective projection
view(180,90)	Sets the initial azimuth and elevation angles of the camera to initialize an aerial downward-pointing camera.
camtarget([0 0 0])	Points camera to the origin
campos([0 0 *h*])	Sets the camera positon above the origin at altitude *h*
camzoom(0.63)	Sets the camera zoom
campan(-90,0)	Rotates the camera so that the northern direction points up
camdolly(x,y,0,'movetarget', 'data')	Translates the camera to the aircraft position
campan(ϕ,θ,'data',[0 1 0])	Rotate the camera to match the aircraft's bank and pitch angles
campan(ψ,0,'data',[0 0 1])	Rotate the camera to match the aircraft's yaw angle

In practice, altitude and focal length of a real camera will be known. Shorter lens focal lengths tend to render images that appear as though the aircraft is flying at a higher altitude and vice versa. In the MATLAB graphics engine, the relationship between altitude and camera zoom is arbitrary and specifying the focal length is not an option. After choosing an altitude and focal length, the camera zoom parameter in the MATLAB graphics engine must be adjusted so that the measurement model accurately transforms 3D points to 2D image plane points. The calibration

process requires the following information: a map of the scenery, landmark positions in the map (world coordinates), the vehicle position, altitude, and orientation, and a simulated aerial photograph that approximates a particular altitude and focal length. The camera zoom is tuned to render an image so that landmarks appear as close as possible (within a few pixels) to the measurement model prediction for a chosen altitude and focal length.

To further test the both localization method's capabilities, the camera angles mounted with respect to the airframe were simulated with noise to evaluate the effects of imperfect camera alignment to the aircraft body axes. The noise was normally distributed with a zero mean and standard deviation of 1 degree. The noise slightly changed the roll, pitch, and yaw angles of the camera for every time step, thereby introducing measurement error to the aerial photos. In other words, the camera vibrates.

The following section in this chapter first demonstrates airplane localization using distinct landmarks followed by demonstrations of airplane localization using generic landmarks. The distinct landmark simulations demonstrate flights at 400 feet without wind, 400 feet with wind, and finally 1000 feet flying for a longer duration with a variety of maneuvers. The generic landmark simulations demonstrate a direct comparison to the distinct landmark simulation operating at 1000 feet. Two other distinct landmark demonstrations include flights over different Pennsylvania locations and are simulated over longer distances with different scenery.

Localization using Distinct Landmark Detection

The first simulation evaluates a flight over a diverse region of unique buildings, fields and trees, and the route is depicted in Figure 5-5. The imagery is photographed to simulate flight altitudes of 400 feet and 1000 feet above ground level (AGL) with a wide angle lens.

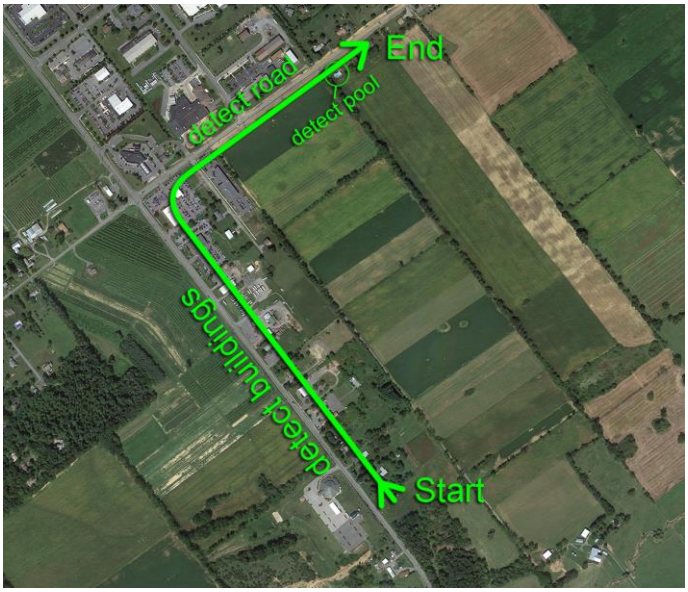


Figure 5-5. Flight path overlaying the simulation environment of approximately 1 square mile.

Nine buildings and one swimming pool are chosen as landmarks for the simulation (see Figure 5-6). The nine buildings are arranged along the northwest-southeast leg of the trajectory and the swimming pool is located near the end of the northeast trajectory. The northeast-southwest road is chosen to evaluate road navigation separately from landmark navigation.

Landmark 1 Landmark 2 Landmark 3 Landmark 4

Landmark 5 Landmark 6 Landmark 7

Landmark 8 Landmark 9 Landmark 10

Figure 5-6. Landmarks are chosen prior to the flight.

Data Preparation and Landmark Detector Training

For image training purposes, on average, approximately 100 images of each landmark are labeled as positives, and 1200 images of each landmark's nearby surroundings are labeled as negatives. Positive images are labeled with MATLAB's built-in Training Image Labeler App as depicted in Figure 5-7. The Labeler App is a user-friendly way to create bounding boxes around a region(s) of interest (ROI). The Labeler App exports the ROI as a structure variable containing the path to every image file and the coordinates of the bounding boxes.

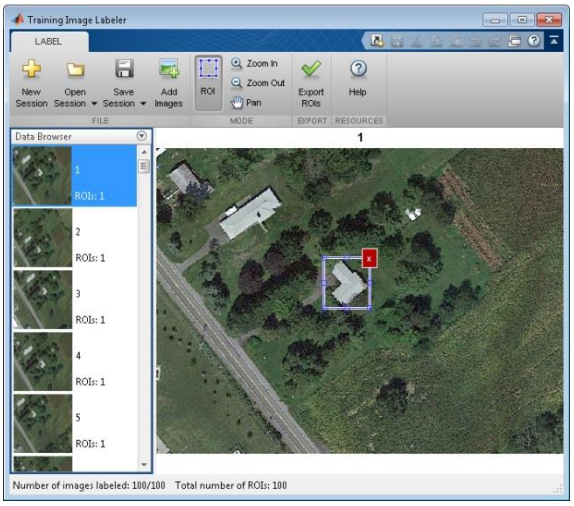


Figure 5-7. The MATLAB Training Image Lableler App is used to manually select the landmarks to delevlop image training data. Each landmark is selected 100 times within images of slightly different perspectives and altitudes.

Each training photograph differs randomly either by altitude, attitude, position, or a combination thereof within a reasonable flight envelope to obtain various observation perspectives. A variation of flight variables also creates a practical set of negative training images that allows the aircraft to approach the landmarks at multiple altitudes, attitudes, and headings. The positive training set, however, must contain landmarks in a similar orientation because the cascade object detector is not rotation-invariant. Training a single detector to handle all landmark orientations will not work [70]. In practice, the aircraft may overfly the landmark

from any direction; therefore the aerial photographs taken from the aircraft must be rotated to the orientation of the positive training set before running the object detector.

Since ten landmarks are chosen, ten separate landmark detectors are trained. First, MATLAB's Computer Vision System Toolbox function called trainCascadeObjectDetector() is used to train custom object detectors. In this case, the custom objects are the 10 chosen landmarks. Inputs include the output filename, the positive training data, the negative training folder, the false alarm rate, the number of cascade stages, the object training size, and the feature type (e.g. HOG). The output of trainCascadeObjectDetector() is an XML file. Next, a detector system object is constructed from that XML file with the vision.CascadeObjectDetector() function, which is also included in the MATLAB Computer Vision System Toolbox (R2012a and later). Inputs include the XML file, which contains the training results, the minimum detectable object size, the maximum detectable object size, and the merge threshold. The minimum and maximum detection window sizes are two-element [height, width] vectors. The merge threshold is a tunable property that "defines the criteria needed to declare a final detection in an area where there are multiple detections around an object. Groups of colocated detections that meet the threshold are merged to produce one bounding box around the target object" [52]. During a flight, the region manager chooses appropriate detectors system objects based on the most current position estimate. A chosen detector system object and aerial photograph become inputs to the MATLAB step() function, which is used to apply a detector system object to an aerial photograph. Note that the MATLAB step() function has other uses which depend on system inputs such as simulating a step response to a dynamic system. The output is the [x y width height] coordinates of a bounding box that surrounds the detected object. If multiple detector system objects are chosen by the region manager, the step() function is evaluated separately for each object.

Simulation: 400 ft. altitude

In this section, two simulations are compared—one scenario without wind and another with constant winds from the 307° radial at 10 knots. When the aircraft is launched in the windy scenario, only the initial position estimate is given. The initial wind estimate is assumed to be unknown and is set to zero to test if the estimator can successfully estimate the wind after receiving landmark measurements. The first landmark is depicted in Figure 5-8, and the last landmark is depicted in Figure 5-9. For a visual comparison of the final results, the estimated position coordinates are plotted with the actual aircraft position coordinates in Figure 5-10 and Figure 5-11. The position errors are shown in Figure 5-12 and Figure 5-13. As the aircraft flies the first leg of the trajectory, only buildings are used to localize the aircraft. Position errors for that leg are seen within the first 40 seconds of the flight (zero winds) and the first 55 seconds of flight (headwind). When the aircraft turns to the right, the road enters the field-of-view and the aircraft obtains lateral position estimates relative to the road until the swimming pool is detected. The road measurements correct the perpendicular (lateral) distance relative to the road as seen in Figure 5-14 and Figure 5-15, but they cannot resolve the parallel position relative to the road, therefore the position estimate drifts from about $t=40\ s$ until $t=64\ s$ or from approximately $t=55\ s$ until $t=80\ s$ when simulating winds. If the aircraft does not use the road estimator (dead reckons until the swimming pool landmark), then the position error is worse.

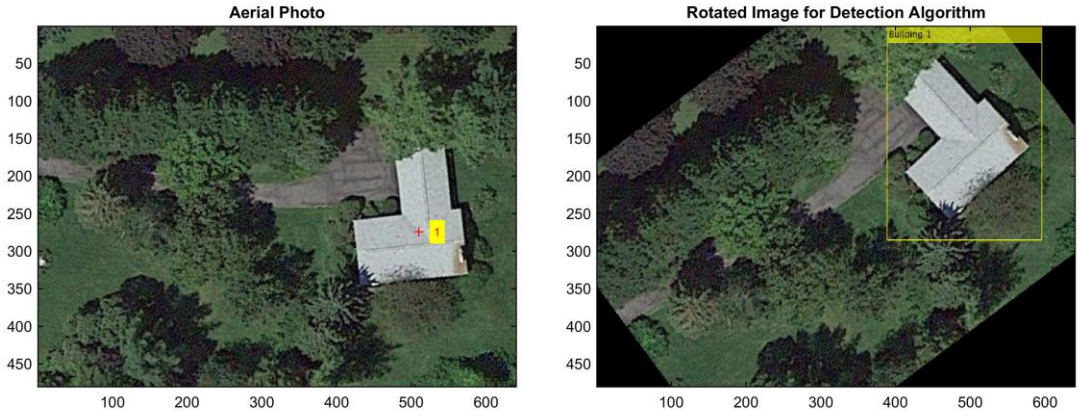


Figure 5-8. The image on the left shows an aerial photograph of the first landmark. The red "+" indicates the location of the building determined by the cascade object detector, which was performed on the right image.

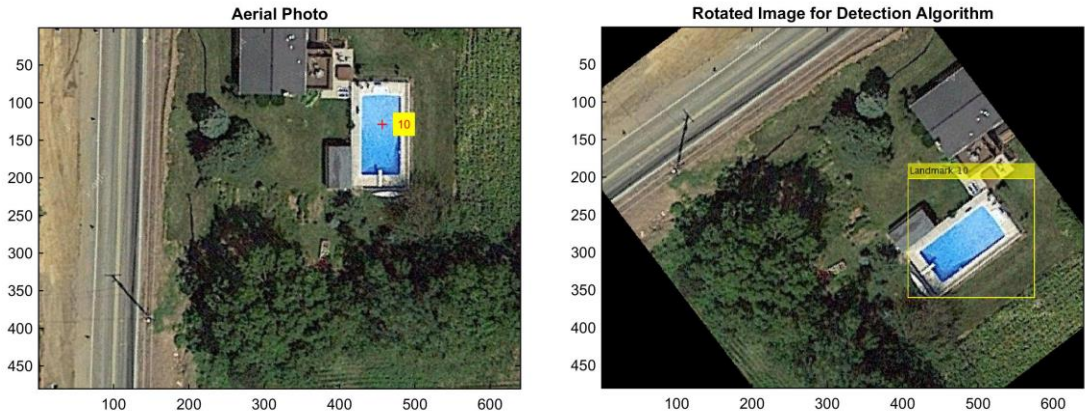


Figure 5-9. The image on the left shows an aerial photograph of the last landmark. The red "+" indicates the location of the swimming pool determined by the cascade object detector, which was performed on the right image.

Error accrues due to the imprecision of the object detector, small errors in the measurement models, and imperfect camera alignment relative to the aircraft body axes. The object detector draws a box around the landmark upon detection as it scans the image. The center of the box is treated as the location of the landmark in the image. Therefore measurement error occurs when the detector box is not perfectly centered over the landmark. Next, the measurement models attempt to model the camera by mathematically transforming a world points to the 2D

image plane. The measurement modeling errors will persist due to errors in the aircraft orientation, camera orientation and camera intrinsics.

Furthermore, position errors have an effect on the wind velocity estimate. For example, if the measured position trails the expected position, the estimator will adjust by reporting a headwind. Even while zero winds are simulated, the estimator reported wind speeds as high as 4 knots as shown in Figure 5-16. For most of the time, however, the estimated winds were on the order of 2 knots, which is reasonably low. In the second scenario, the wind estimate error is as high as 4.5 knots as seen in Figure 5-17.

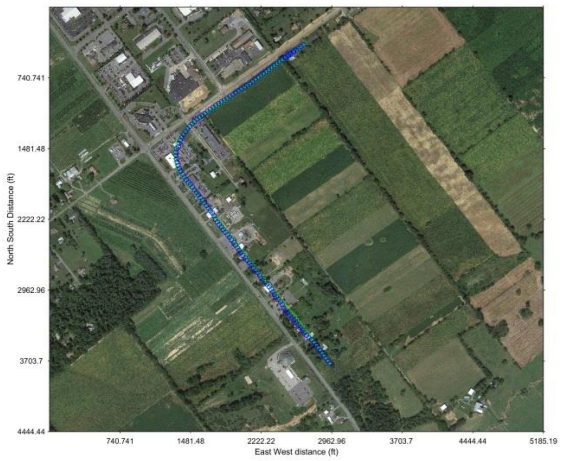


Figure 5-10. Comparison of the estimated position and the actual position of the aircraft. *h*=400 ft, no wind.

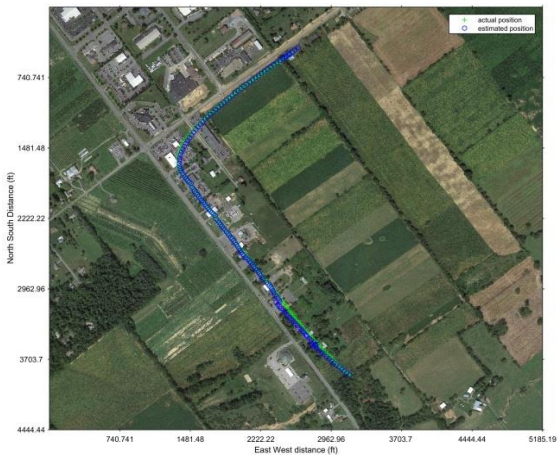


Figure 5-11. Comparison of the estimated position and the actual position of the aircraft. *h*=400 ft, wind 307° at 10 kts.

Figure 5-18 and Figure 5-19 demonstrate the expected behavior of how measurements affect the spread of sigma points over each subsequent time step. When measurements are obtained, the position uncertainty decreases and the sigma points contract. When no measurements are available, the airplane dead reckons, the position uncertainty increases, and the sigma points expand.

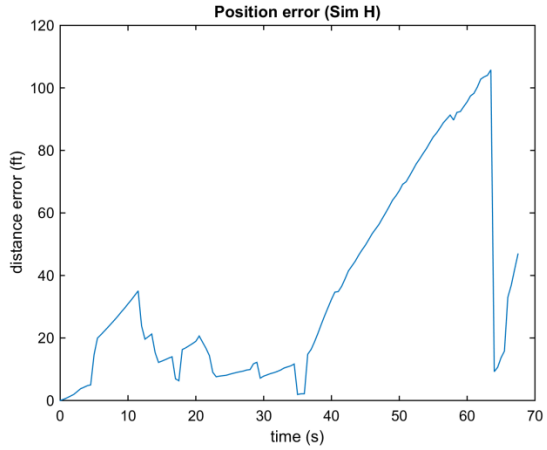


Figure 5-12. Position error vs time, h=400 ft, no wind

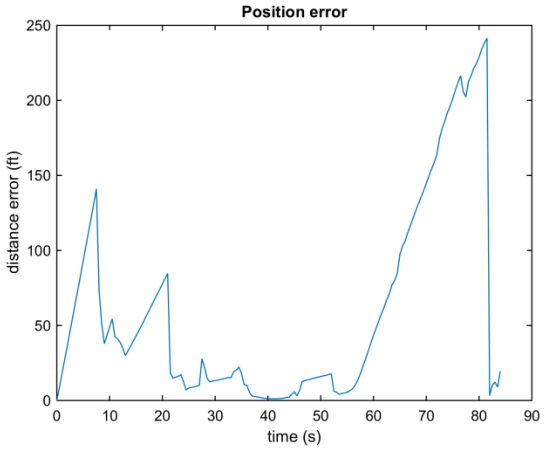


Figure 5-13. Position error vs time, *h*=400 ft, wind 307° at 10 kts

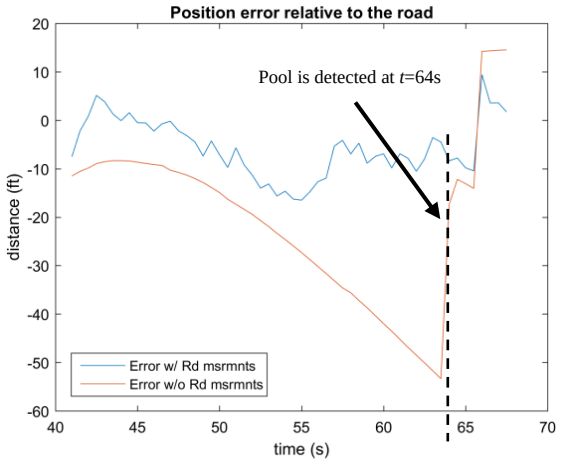


Figure 5-14. Position error relative to the road improves significantly when using road measurments. From *t*=64 s through *t*=65 s, the swimming pool landmark is detected. The detection of the swimming pool confirms that the road detection is useful. *h*=400 ft, no wind.

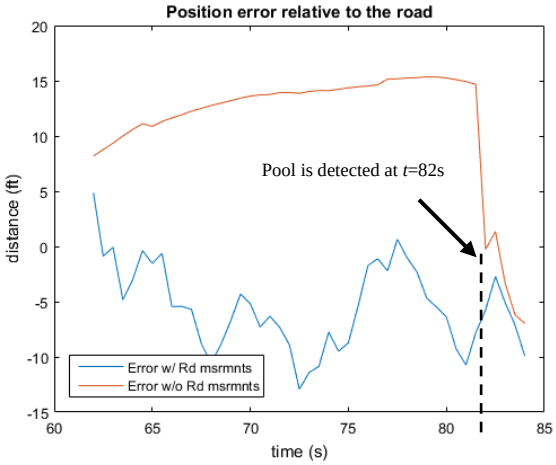


Figure 5-15. Position error relative to the road improves significantly when using road measurments. From *t*=82 s through *t*=83.5 s, the swimming pool landmark is detected. The detection of the swimming pool confirms that the road detection is useful. *h*=400 ft, wind 307° at 10 kts.

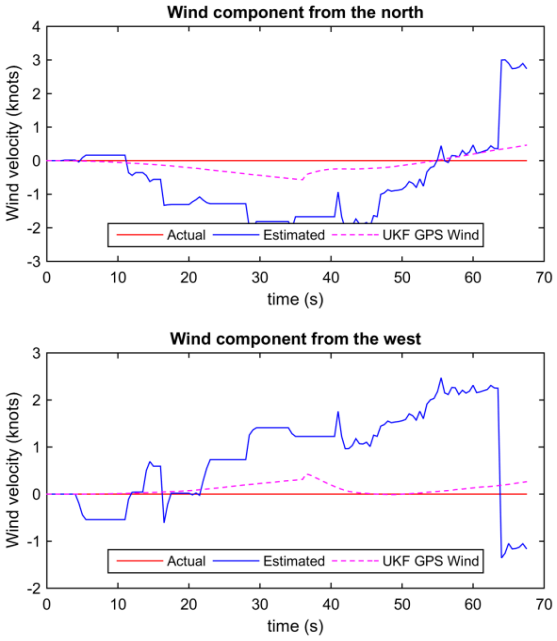


Figure 5-16. Wind velocity estimate, *h*=400ft, simulated no wind

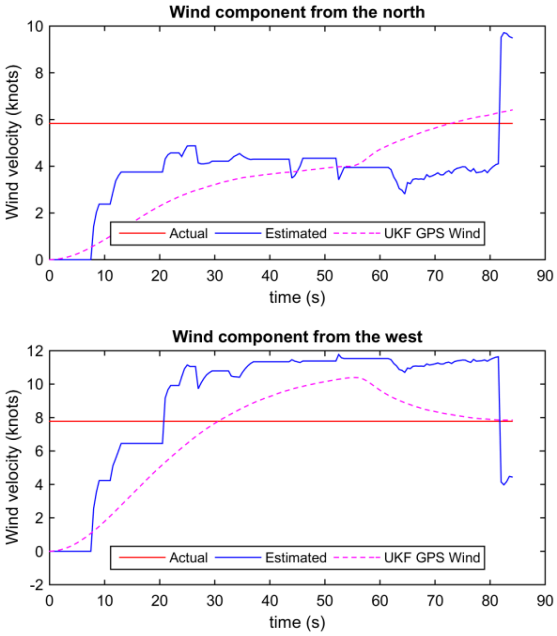


Figure 5-17. Wind velocity estimate, *h*=400ft, simulated wind 307° at 10 kts

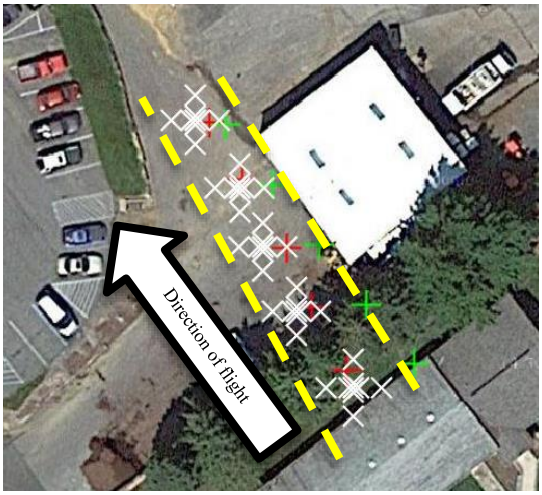


Figure 5-18. Sigma points (white markers) contract when measurements are obtained. Red markers indicate the estimated airplane position. Green markers indicate the actual position.

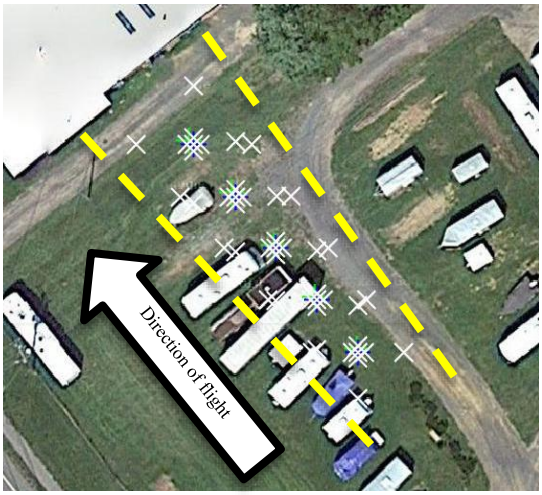


Figure 5-19. Sigma points (white markers) expand when measurements are not obtained. Blue markers indicate the estimated airplane position. Green markers indicate the actual position.

Simulation: 1000 ft. altitude, no wind, extended flight

To evaluate the robustness of the landmark detection method, a third simulation ran over a longer time span with multiple maneuvers such as S-turns in multiple directions to cover a more random set of flights over the test area.

Figure 5-20 displays the estimated position results against the actual position, Figure 5-21 displays a plot of the position error over time, and Figure 5-22 compares the estimated wind velocity components using vision measurements, against the actual wind components and estimated wind components using GPS-incorporated measurements.

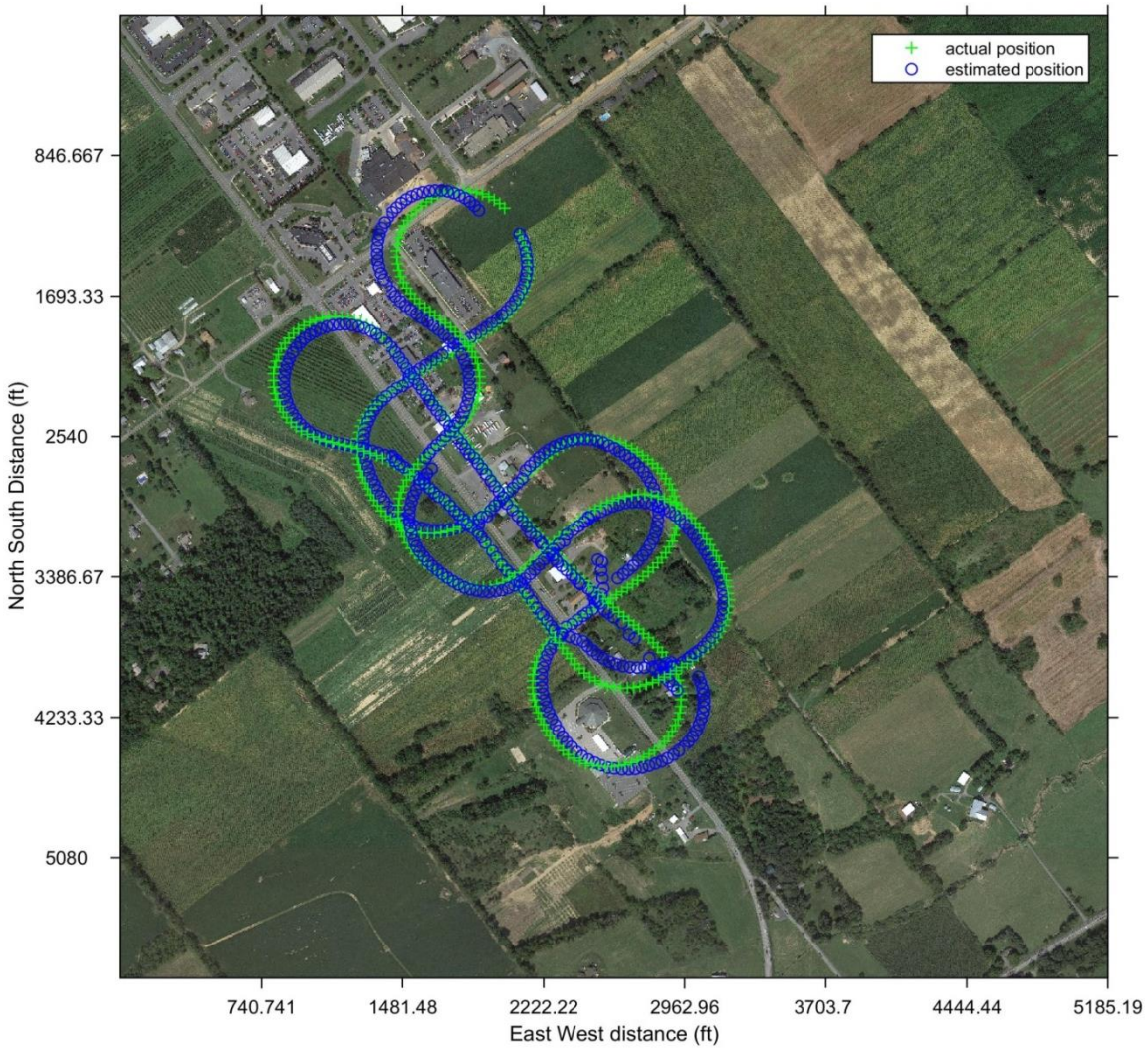


Figure 5-20. Estimated vs actual position, winds calm *h*=1000 ft

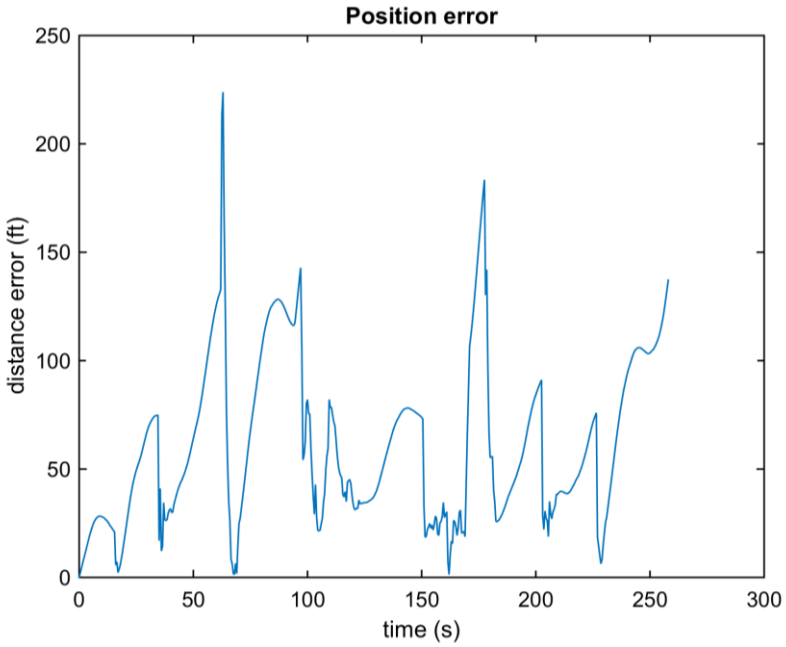


Figure 5-21. position error vs time, winds calm, *h*=1000 ft, no wind

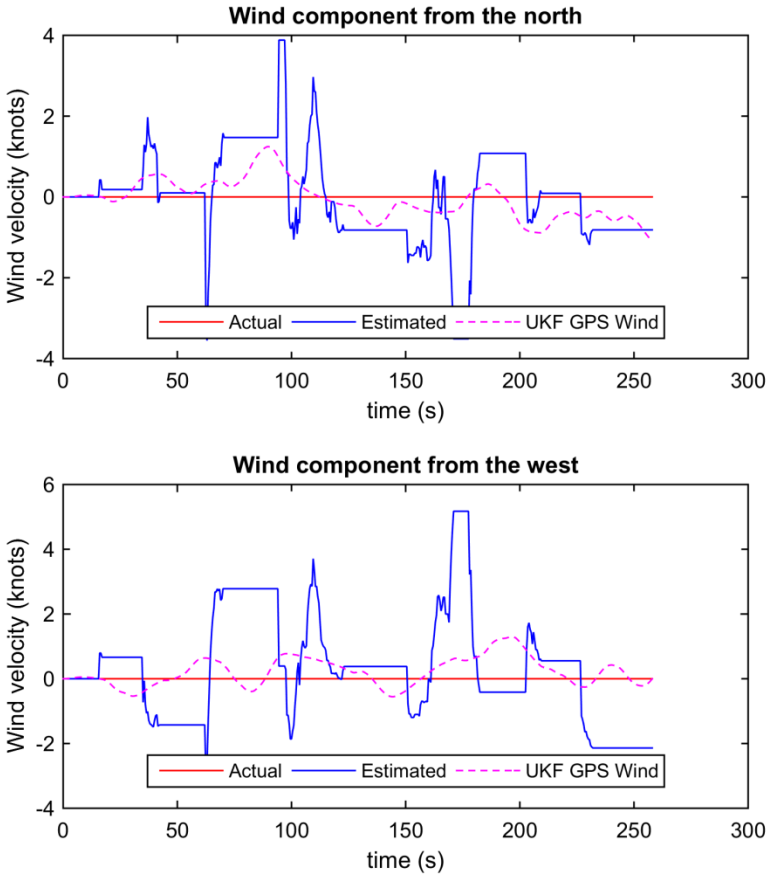


Figure 5-22. Simulated wind vs estimated wind, h=1000 ft

Localization using Generic Landmark Detection

The following simulations demonstrate airplane localization using generic landmarks. A set of training images categorized by buildings, fields, and trees are used to generate a histograms of BoW codewords as described in Chapter 3. Then an SVM classifier is trained to classify landmarks. Unlike the previous method, this second method does not require the selection of unique landmarks. Three separate simulations demonstrate flights at different altitudes over different regions of Pennsylvania during different seasons. An advantage to this method is the ability to easily retrain the landmark classifier with new training images to account for changes in seasons.

Data Preparation and Landmark Detector Training

High resolution imagery is broken into small tiles and manually separated by category. For the simulations presented in this section, at least 2000 images of each category are used to train the classifier. The MATLAB Computer Vision Toolbox functions, bagOfFeatures() and trainImageCategoryClassifier()are used to construct the classifier with a dictionary size of 500 cluster centers. To test the accuracy of the classifier, 70% of the training set is partitioned for training and the remainder is partitioned for testing. The testing results are displayed in the confusion matrices below in Table 6 and Table 7.

Table 6. Confusion matrix of classification accuracy using Google Maps Imagery

Google Maps Imagery (0.37 ft. resolution)				
	Predicted			
		Buildings	Fields	Trees
Known	Buildings	0.9866	0.0033	0.0100
	Fields	0.0145	0.9437	0.0418
	Trees	0.0015	0.0610	0.9375

Table 7. Confusion matrix of classification accuracy using USGS Orthoimagery

2006 USGS Orthoimagery (1 ft. resolution)				
	Predicted			
		Buildings	Fields	Trees
Known	Buildings	0.9600	0.0317	0.0083
	Fields	0.0257	0.9057	0.0686
	Trees	0.0081	0.0396	0.9523

Next, a flight region is selected and a photographic map is processed with the classifier and road detection algorithm to extract the landmarks and roads. Landmark coordinates are automatically computed by scanning the imagery with a "scanning window." The small region (tile) within the scanning window is evaluated with the landmark classifier. The classifier returns the label and the score of the classification result. To improve the accuracy of the classification, the window is shifted by 50 pixels around each landmark's local 8-neighborhood region until a maximum score is obtained. At the location with a maximum score, the coordinates of the center of the window and the landmark classification label is recorded in an array. The entire image is scanned again with a larger window size and the process is repeated. Experiments by trial and error have indicated that scanning windows of at least 150 by 150 pixels or larger should be evaluated with the BoW classifier. Smaller images contain too few features for accurate classification. However, larger images beyond 300 by 300 pixels may contain multiple categories at the resolutions used in this research. Incorrect classification is not problematic as long as the object is consistently classified incorrectly in both the reference map and aerial photos. A notable occurrence in this application is the classification of parking lots as buildings. As long as the markers are correctly associated from the aerial photos to the reference map, the navigation solution does not degrade. The reference maps are scanned twice with scanning windows sizes of 200 by 200 pixels and 300 by 300 pixels. The aerial photos are scanned three times with window sizes of 175 by 175 pixels, 200 by 200 pixels, and 250 by 250 pixels.

Multiple detections of the same landmark may occur when the landmark exceeds the boundaries of the scanning window. If building markers are too close to one another, they are assumed to represent the same building therefore nearby building coordinates (markers) are averaged together. For example, if building markers are within 200 pixels of each other, then their coordinate locations are averaged together to form a single landmark coordinate. The chosen distance threshold depends on image resolution, focal length, aircraft altitude, and the general size of buildings in the vicinity.

Simulation: 1000 ft. altitude, no wind, extended flight

Figure 5-23 displays the processed reference map of automatically detected landmarks overlaying the orthoimagery. The map is processed before the flight simulation, and only the blue, yellow, and green markers are stored in the aircraft's map database.

Figure 5-23. Processed reference map overlaying original orthoimagery.

The first simulation result is shown in Figure 5-24 and is intended for comparison to the previous landmark-aided localization method in Figure 5-20. In the previous method that uses distinct landmarks, measurement updates occur when predetermined landmarks are detectable in the

camera's field-of-view. As a result, the RMS error for the simulation using distinct landmarks is 72.7 feet. In the second method presented here, however, generic landmarks are used and measurement updates are obtained more frequently. The best measurements are obtained when all three categories of landmarks are observed, and the measurement vector accuracy is further improved when road intersections are detected also. The RMS error in this simulation is 32.4 feet, which is calculated from the position error results displayed in Figure 5-25.

Figure 5-26 displays the wind components. In this simulation, no wind is simulated as indicated by the green line in the plots. The solid red curve shows the wind estimate using the localization technique using generic landmarks. The magenta curve displays the Piccolo autopilot wind estimate. For further comparison, the black curve shows the wind estimate using GPS position together with a separate UKF. The wind error associated with this simulation is on the order of 2 knots, and if GPS is used, the wind error is less, but follows a similar trend with the new method.

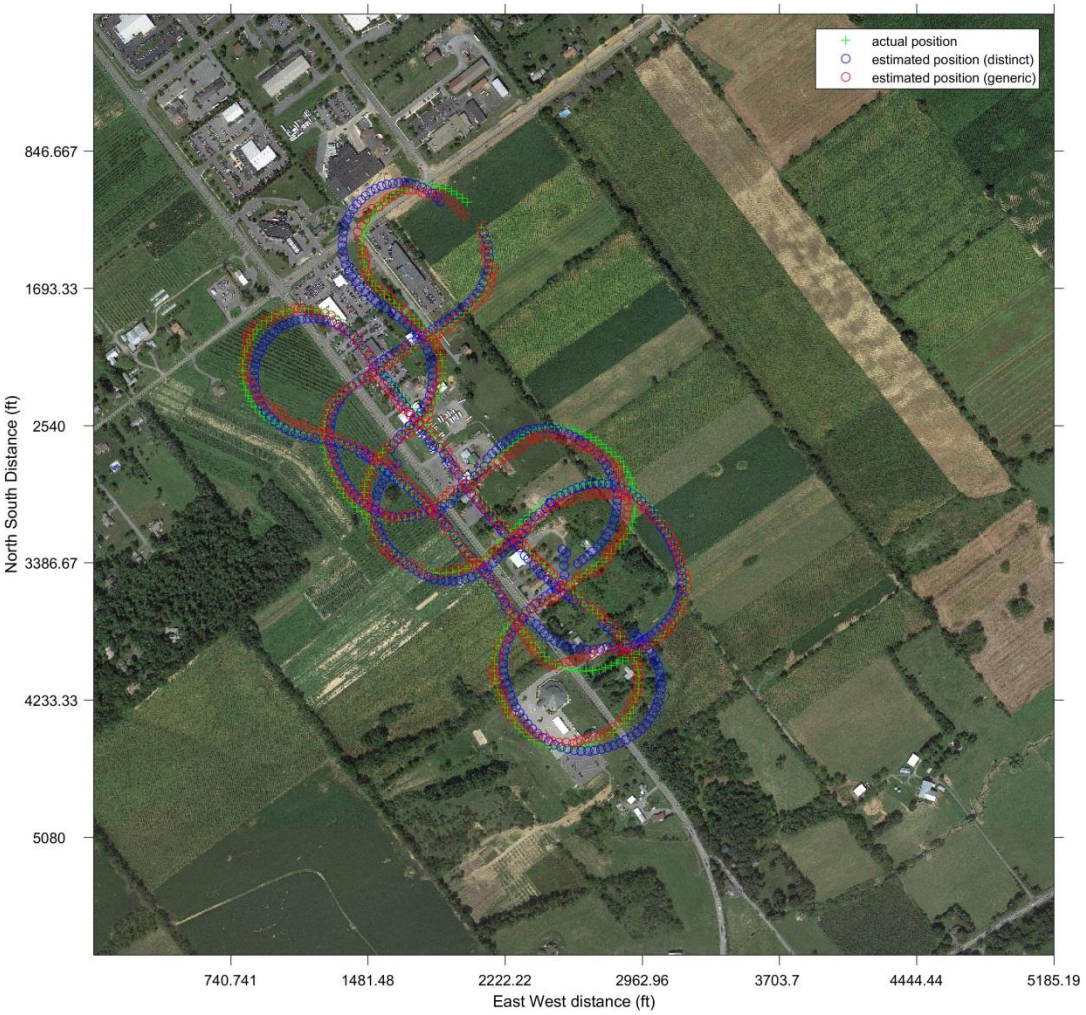


Figure 5-24. Comparison of distinct landmark localization (blue), BoW generic landmark localization (red), and actual location (green). 1000 ft altitude, no wind.

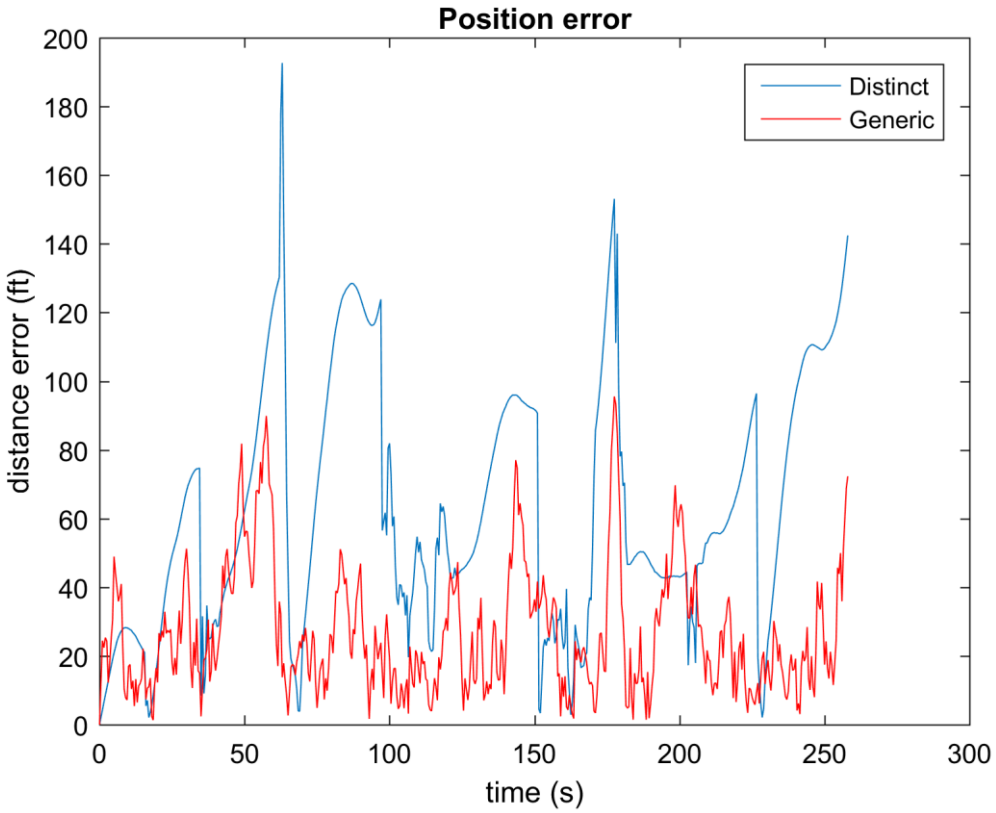


Figure 5-25. Comparison of position error for distinct landmark localization and BoW generic landmark localization. 1000 ft altitude, no wind.

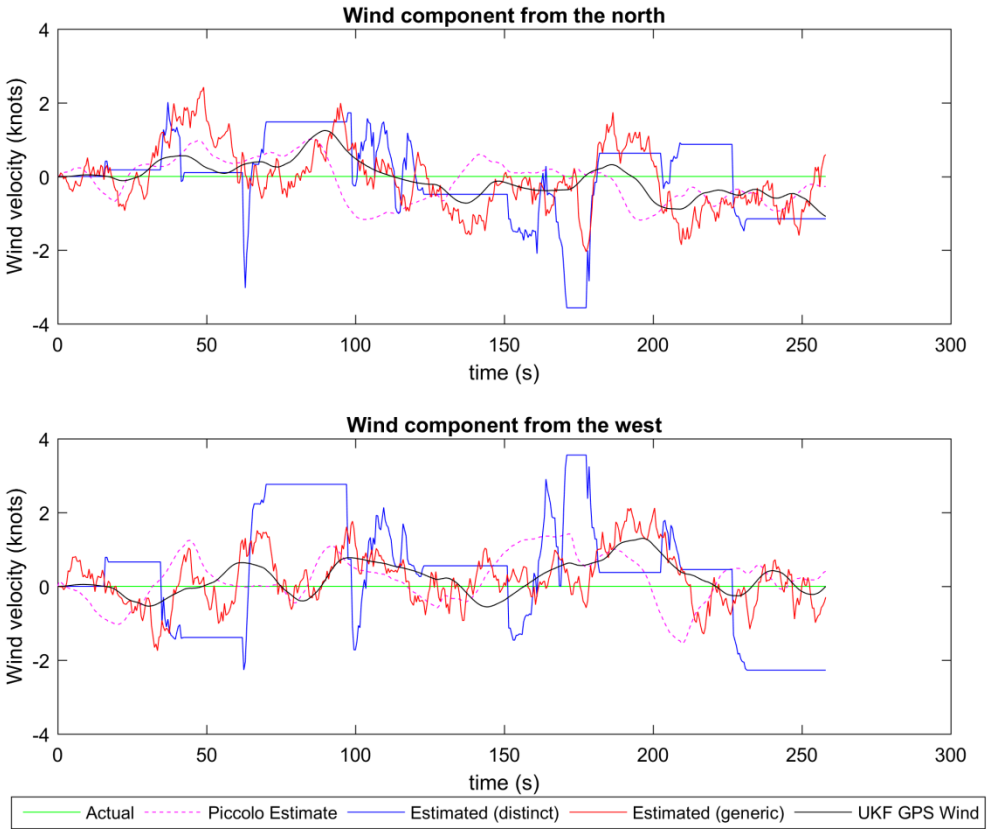


Figure 5-26. Comparison of wind error for distinct landmark localization and BoW generic landmark localization. 1000 ft altitude, no wind.

Simulation: Late winter, 2500 ft. altitude, wind 180° at 5 knots

This simulation occurs over a region in western Pennsylvania and uses high resolution orthoimagery (1 pixel = 1 foot) dated March 7, 2006 obtained from the USGS. The flight begins from the east, flies west along a road, performs a series of turns and returns to the east without the road in view. Figure 5-27 displays the processed reference map of automatically detected landmarks overlaying the orthoimagery. The map is processed before the flight simulation, and only the blue, yellow, and green markers are stored in the aircraft's map database. Figure 5-28 displays the actual and estimated aircraft position markers overlaying the reference map, Figure 5-29 displays the position error verses time, and Figure 5-30 displays the wind estimate

comparisons. In this scenario, a crosswind of 5 knots from the south is simulated and the altitude is increased to approximately 2500 feet above ground level.

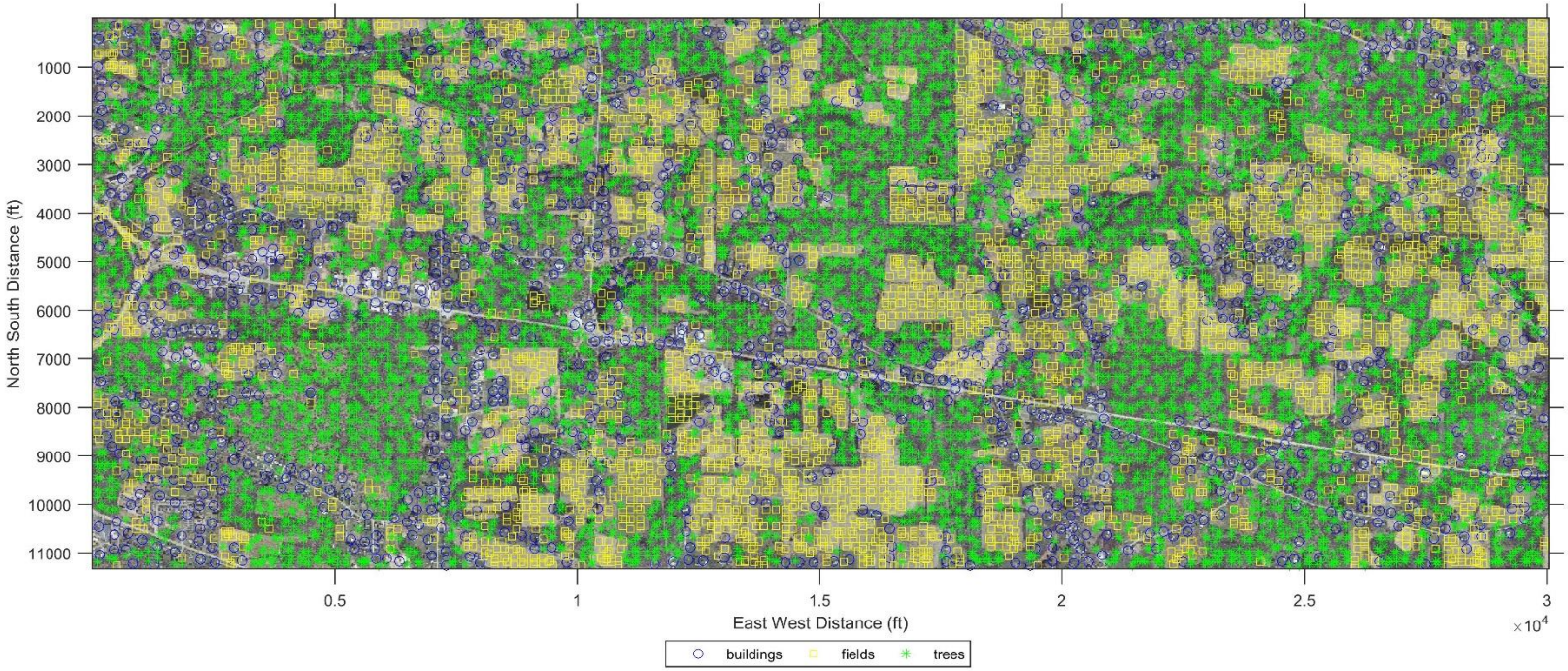


Figure 5-27. Processed reference map overlaying original orthoimagery

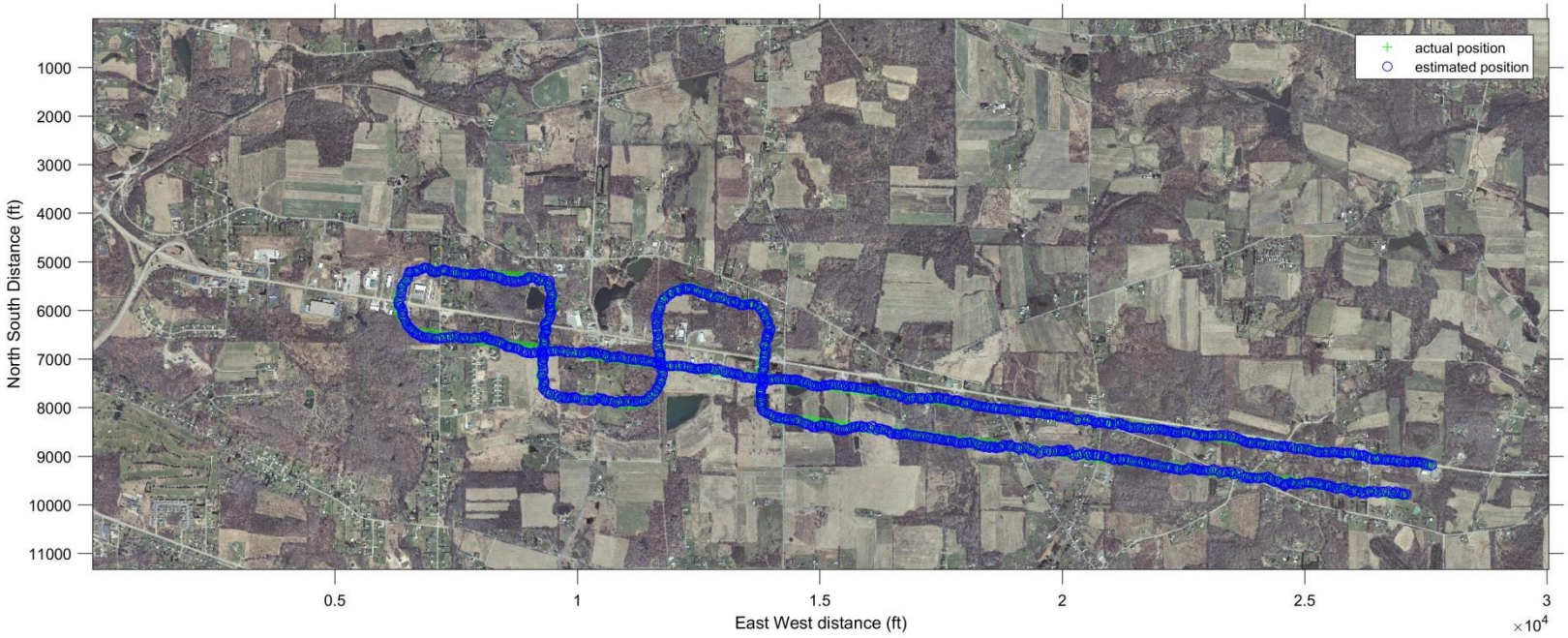


Figure 5-28. Actual position compared to estimated position using BoW gerneric landmark detection.

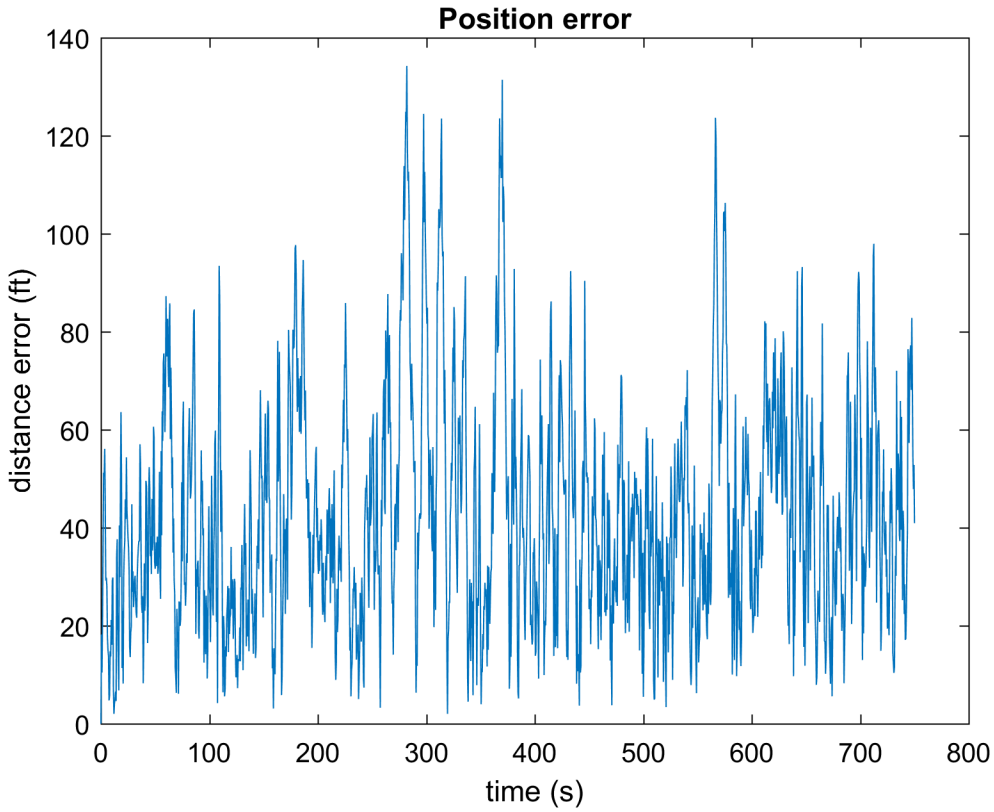


Figure 5-29. Position error using BoW generic landmark detection.

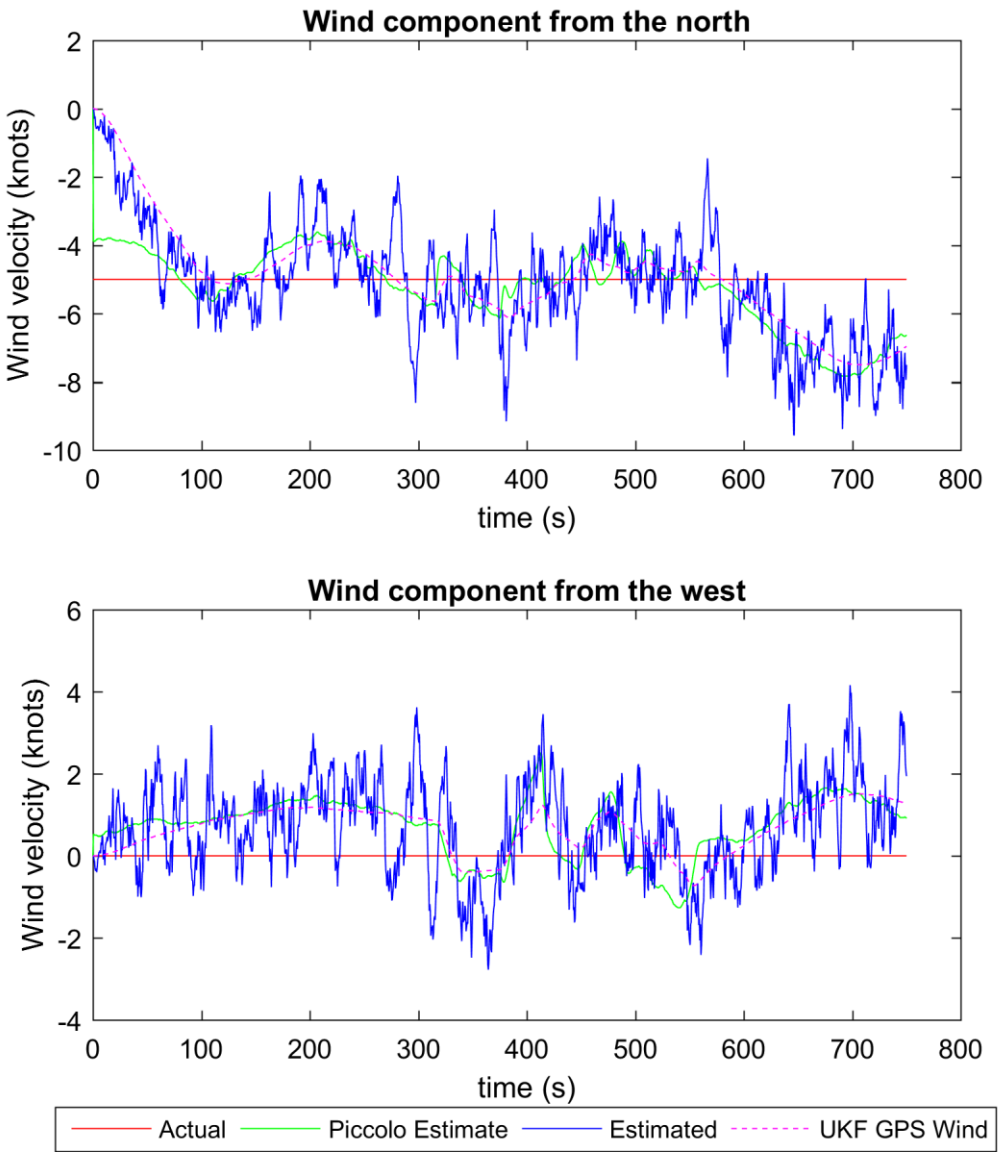


Figure 5-30. Wind estimate comparisons.

Simulation: Early spring, 3500 ft. altitude, wind 230° at 4 knots

This simulation occurs over a region in central Pennsylvania. The altitude was increased to 3500 feet to ensure observation of a diverse set of landmarks. The aircraft begins above the threshold of runway 24 at University Park Airport, flies the runway heading, enters a right pattern

and then departs to Bellefonte Airport to the northeast. The aircraft enters a downwind leg, base leg, final approach leg of runway 25, and then returns to the starting location near runway 24 of University Park Airport. High resolution orthoimagery (1 pixel = 1 foot) dated April 9, 2006 is obtained from the USGS. For added realism at the time of this simulation, actual winds reported from the University Park (KUNV) METAR were 230° at 04 knots on March 13, 2016.

Figure 5-31 displays the processed reference map of automatically detected landmarks overlaying the orthoimagery. The map is processed before the flight simulation, and only the blue, yellow, and green markers are stored in the aircraft's map database.

Figure 5-32 displays the actual and estimated aircraft position markers overlaying the reference map. Small callboxes show an expanded view at the time of the four greatest position error peaks corresponding with Figure 5-33. Error increases due to a lack of landmark diversity (when flying strictly over a uniform category of objects) or due to wrong landmark associations in the map. As expected, the position error increases due to flying at higher altitudes. Figure 5-34 displays the noisy camera orientation compared to the actual aircraft orientation. The noise introduces alignment errors in the camera mounts, effectively simulating a vibrating camera along all three axes. Therefore the camera's perspective from the airplane is not perfectly aligned with the airplane body axes due to the noise. Black circles are plotted at locations consistent with the callboxes at *t=197 s*, *t=384 s*, *t=619.5 s*, and *t=703 s* on Figure 5-34. Finally, Figure 5-35 displays the wind estimates.

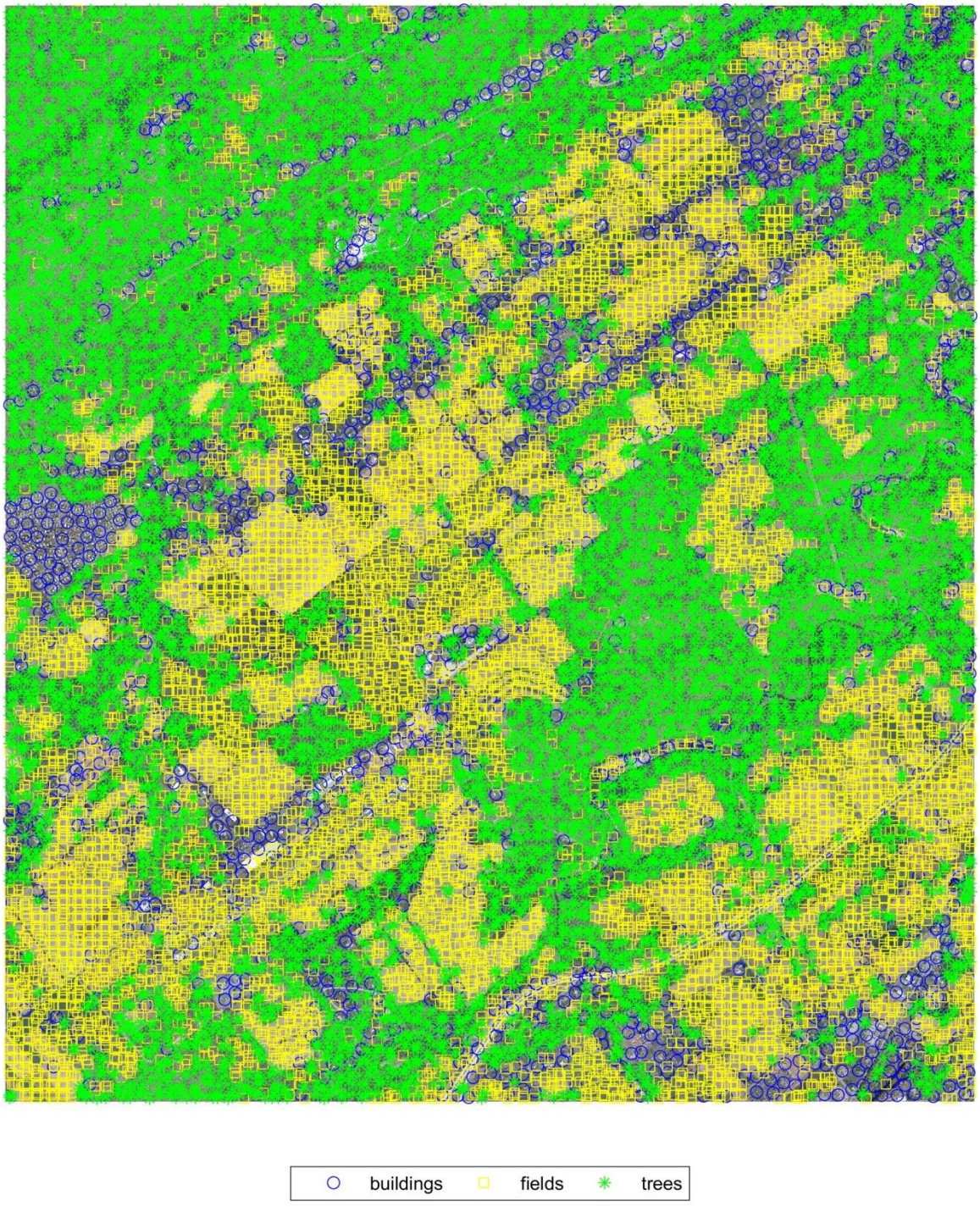


Figure 5-31. Processed reference map overlaying original orthoimagery of central PA.

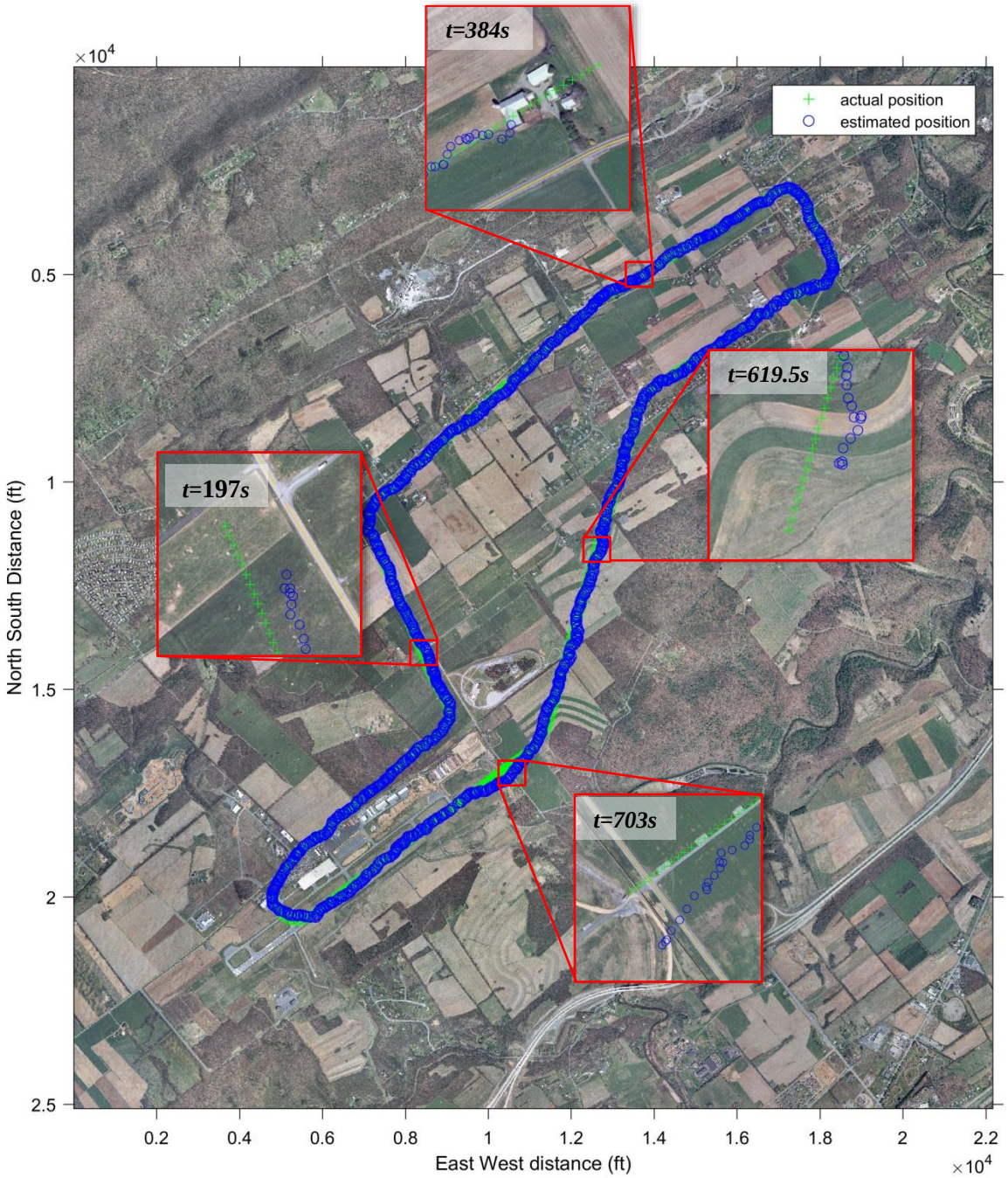


Figure 5-32. Actual position compared to estimated postion using the BoW generic landmark detection.

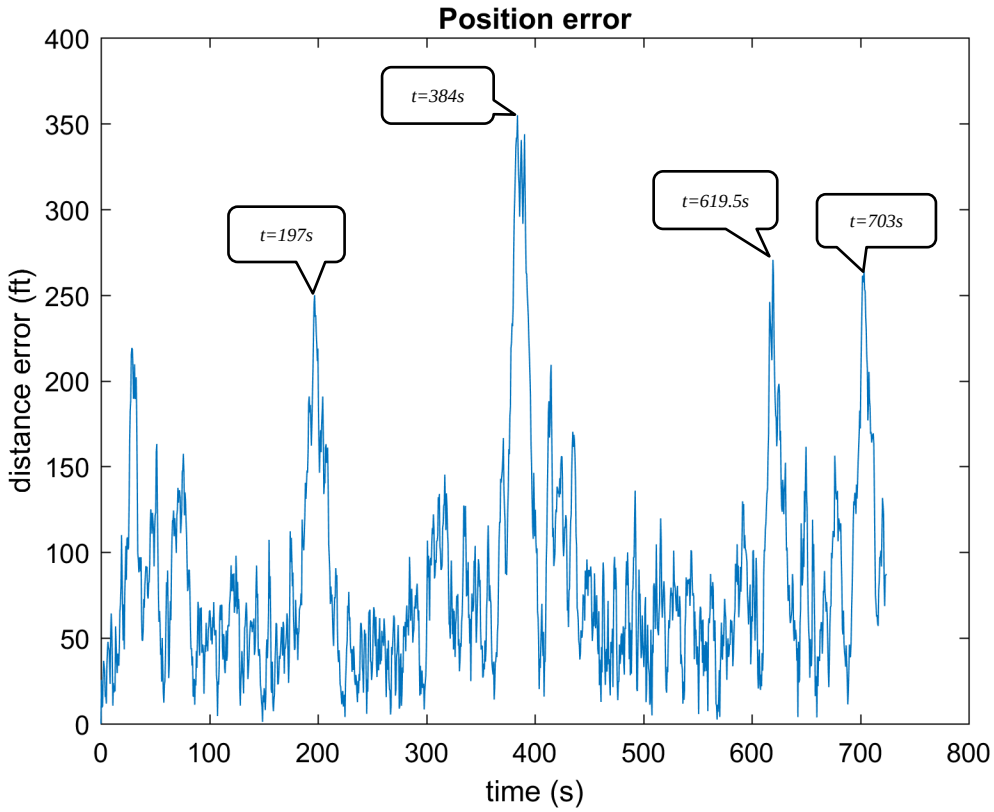


Figure 5-33. Position error using the BoW generic landmark detection.

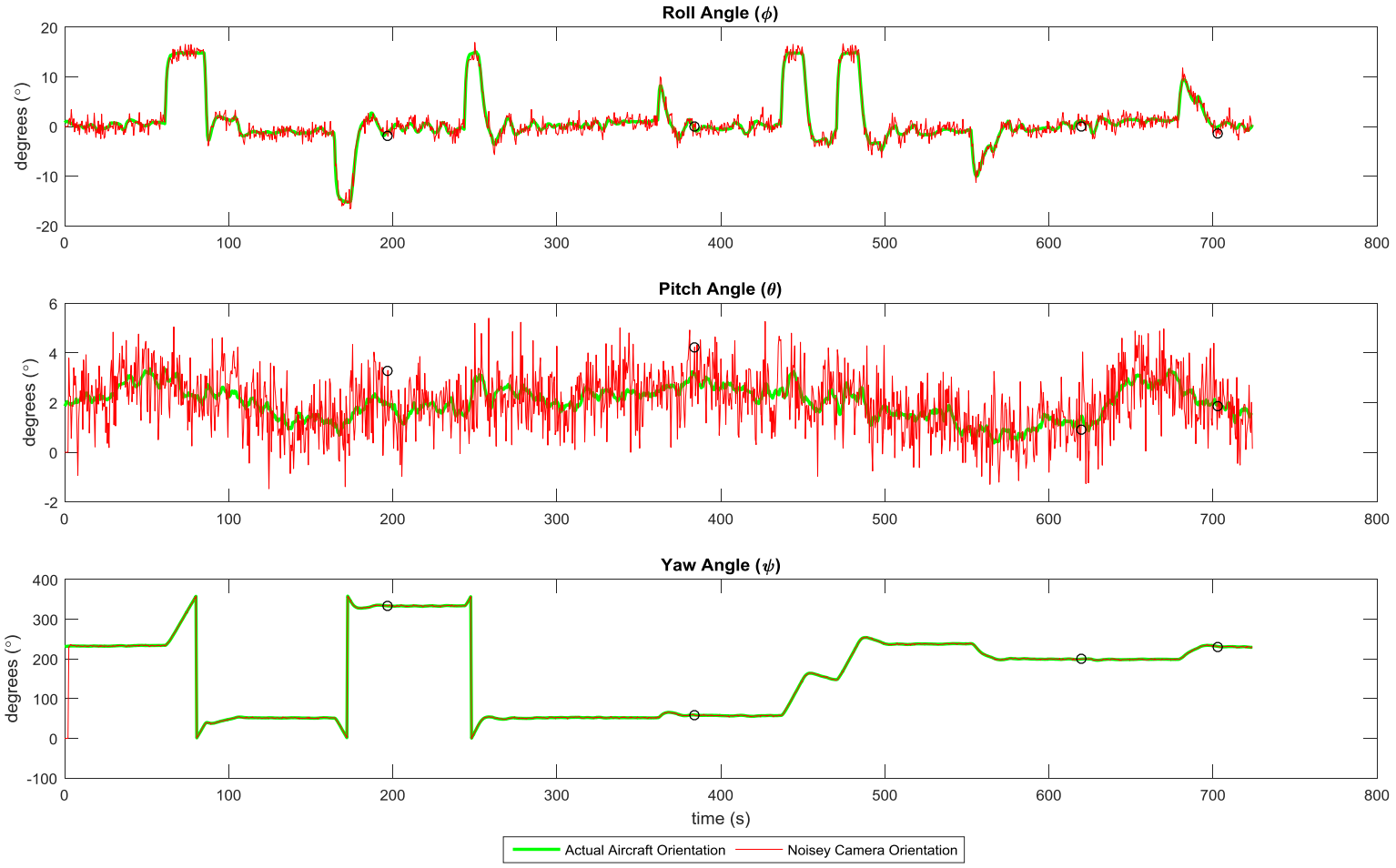


Figure 5-34. Noise *N~(0,1°)* corrupts the camera orientation angles (red curve) compared to the actual aircraft orientation (green curve).

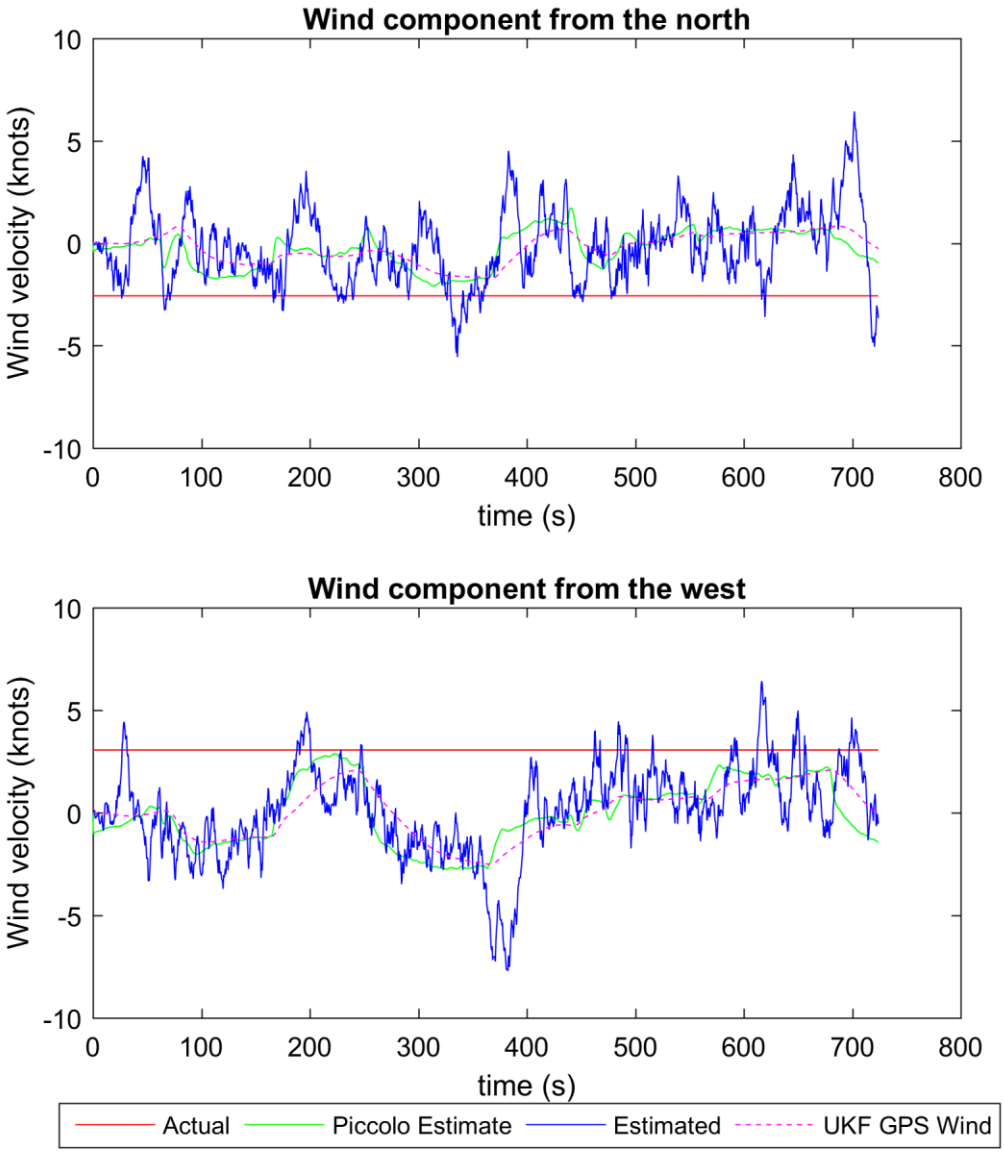


Figure 5-35. Wind estimate comparisons.

Sensitivity Analysis

A sensitivity analysis is performed on the localization method that uses generic landmarks to evaluate its robustness to sensor noise. Airspeed and yaw measurements are inputs to the UKF process model, which predicts the location of the airplane. Roll, pitch, and yaw

measurements are inputs to the UKF measurement model, which predicts the 2D image plane coordinates of landmarks. The sensitivity analysis shows how poor sensor measurements affect localization error. When the position error increases, the likelihood of wrong data association increases. Wrong data association further compounds the error because it generates incorrect measurement model predictions during UKF correction step. In other words, the objective of the sensitivity analysis should answer, "How bad can each sensor reading become until position error becomes unacceptable?"

Noisy random walk signals in airspeed, roll, pitch, and yaw measurements are separately imposed on the sensor measurements, which are fed to the UKF. A total of 280 separate simulations consist of 7 different standard deviations (SD) evaluated 10 times for each sensor measurement. The flight path repeats the first simulation in this chapter but simulates an altitude of 1000 feet and 10 knots of wind. After each simulation run, the RMS position error is computed. Unlike the previous simulations, the camera mounts are not corrupted by noise but are perfectly aligned with the aircraft body axes during the sensitivity analysis.

The first set of simulations analyzes sensitivity to airspeed measurement errors. SD in airspeed include: 0.2 ft/s, 0.4 ft/s, 0.6 ft/s, 1 ft/s, 1.5 ft/s, 2 ft/s, and 2.5 ft/s. This range of standard deviations has allowed the error magnitude to range from 0 ft/s to 40 ft/s (24 knots). The subplots in Figure 5-36 display the random walk error signals which are added to airspeed measurements. Figure 5-37 displays the RMS position error verses SD of noisy airspeed measurements. The position error remains consistent until the SD becomes 2 ft/s and allows the airspeed error to approach a magnitude 20 ft/s.

The second set of simulations analyzes sensitivity to roll (bank) angle measurement errors. SD in bank angle include: 0.2°, 0.4°, 0.6°, 1°, 1.5°, 2°, and 2.5°. The subplots in Figure 5-38 display the random walk error signals which are added to bank angle measurements. Figure 5-39 displays the RMS position error verses SD of noisy roll angle measurements. SD of 1°

allows the roll angle error to exceed a magnitude of 10°. When the roll angle error exceeds 10° the position error is likely to become unacceptable because the RMS position error more than doubles from the 0.2° SD simulations. Roll angle error statistics with SD of 1.5 and higher further degrade localization accuracy. Errors on the order of 20° and higher cause localization divergence as indicated by the large RMS position error spikes at SD of 2 in Figure 5-39. This trend also appears consistent for pitch angle and yaw angle measurements seen in Figure 5-41 and Figure 5-43. Fortunately, even low cost inertial navigation equipment is expected to measure orientation with less error.

The third set of simulations analyzes sensitivity to pitch angle measurement errors. Like the roll angle analysis, standard deviations of 0.2°, 0.4°, 0.6°, 1°, 1.5°, 2°, and 2.5° were chosen. Figure 5-40 displays the random walk error signals which are added to pitch angle measurements. Figure 5-41 displays the RMS position error verses SD of noisy pitch angle measurements. Figure 5-39 and Figure 5-41 appear very similarly and may suggest that errors in both roll and pitch angle measurements cause similar localization degradation. Future work could investigate errors in roll and pitch angles compare when different aerial photo aspect ratios are used.

The fourth set of simulations analyzes sensitivity to yaw angle measurement errors. Again, standard deviations of 0.2°, 0.4°, 0.6°, 1°, 1.5°, 2°, and 2.5° were chosen. Figure 5-42 displays the random walk error signals which are added to yaw angle measurements. Figure 5-43 displays the RMS position error verses SD of noisy yaw angle measurements. The RMS positon error appears consistent from SD=0.2° to SD=1°. The results also show that errors in yaw angle are slightly less detrimental than errors in roll and pitch angle measurements. One possible reason is that landmarks expected to appear near the center of the image frame will remain near the center of the image frame regardless of yaw angle for the data association task. At SD=1.5° the yaw angle errors exceed 10° and the RMS position error more than doubles in comparison to the lower SD runs.

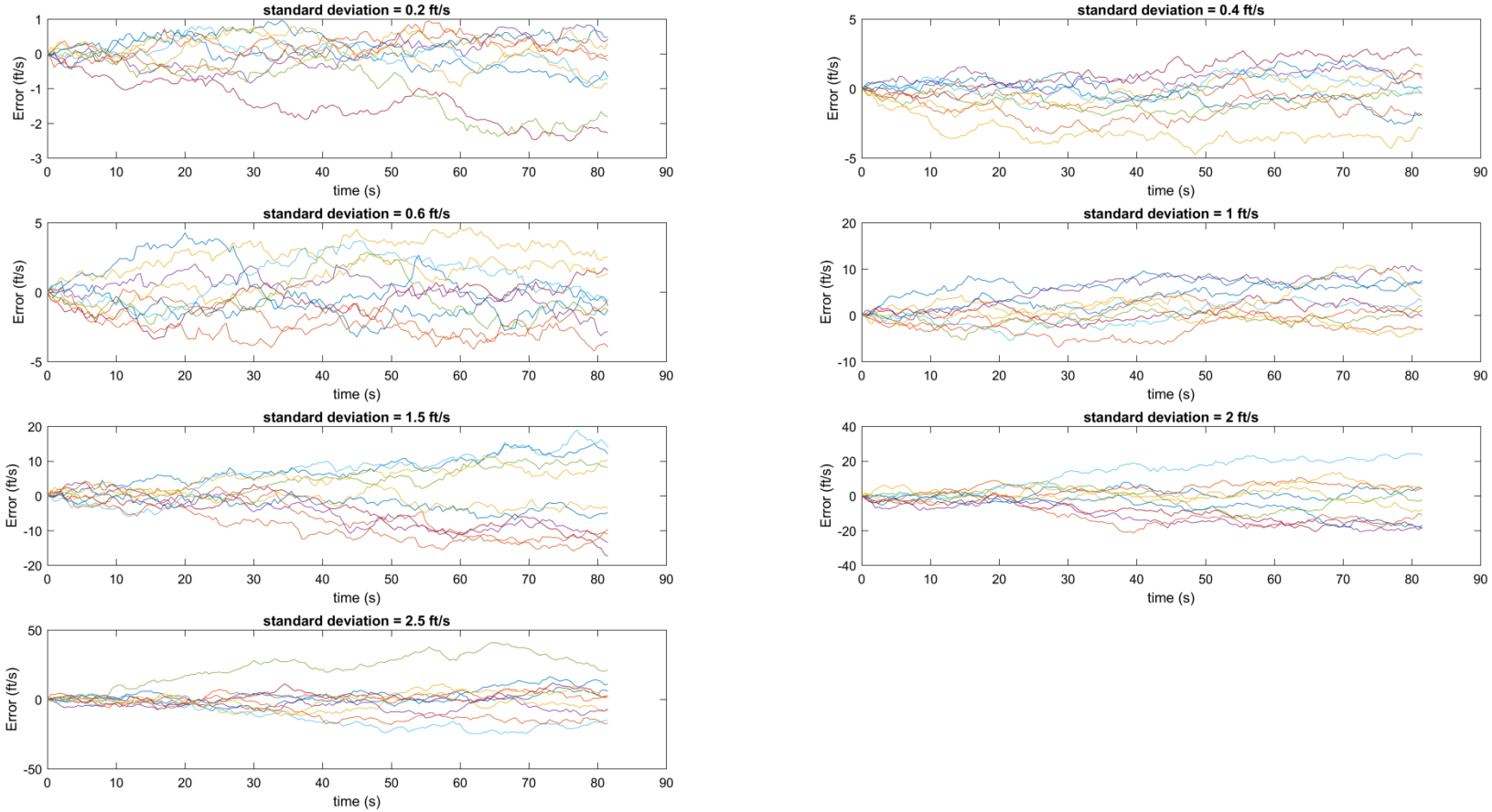


Figure 5-36. Noise signals applied to the airspeed measurements

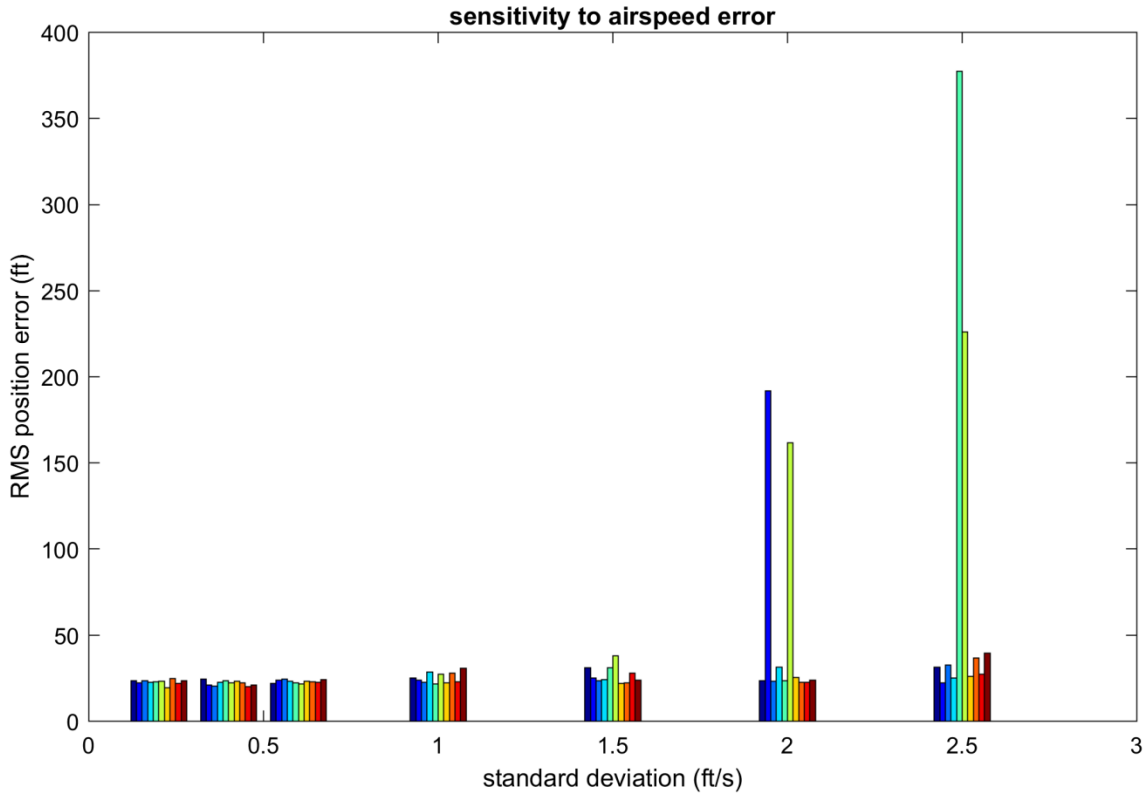


Figure 5-37. RMS position error vs standard deviation of noisy airspeed measurements

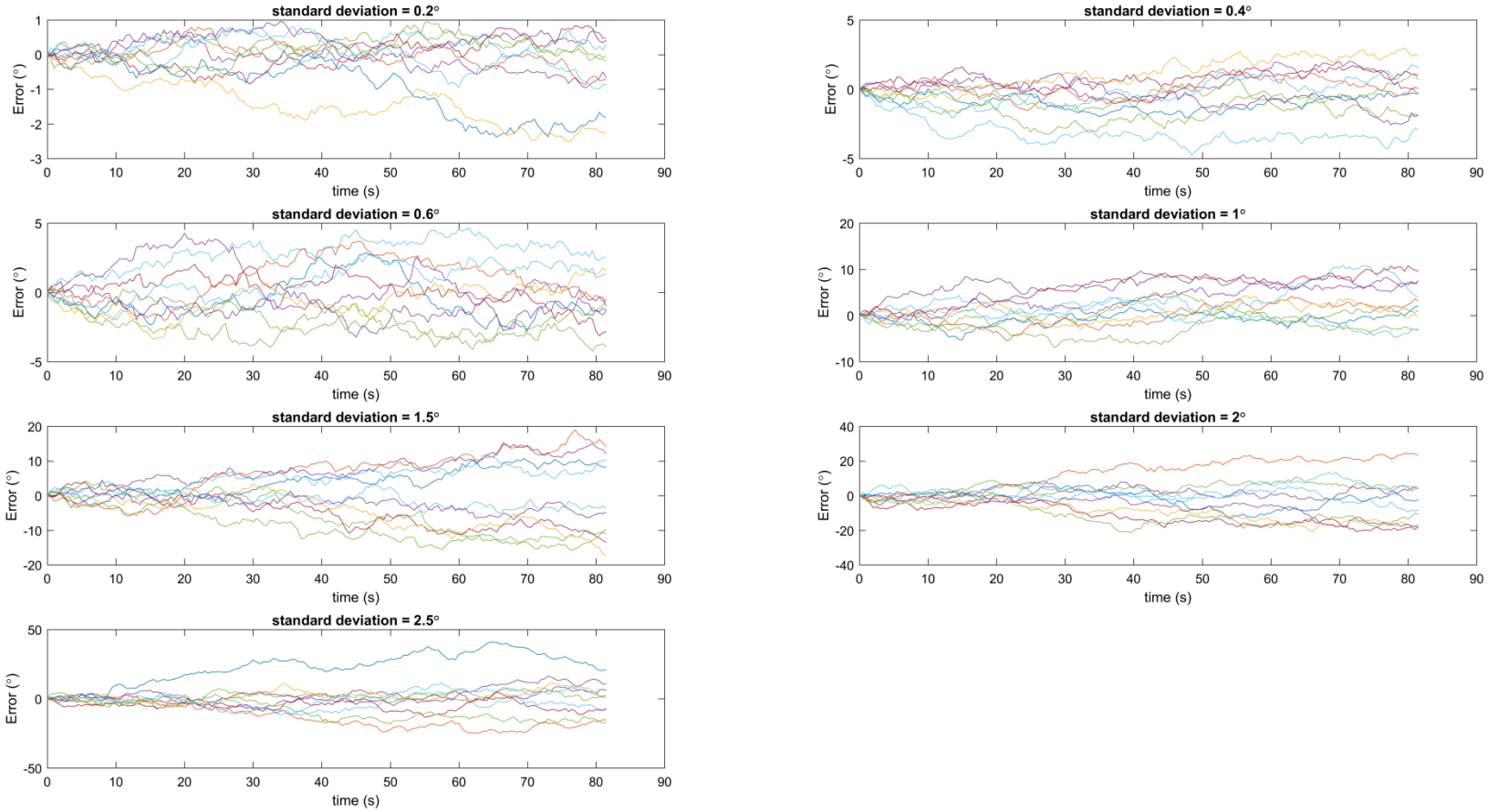


Figure 5-38. Noise signals applied to roll angle measurements

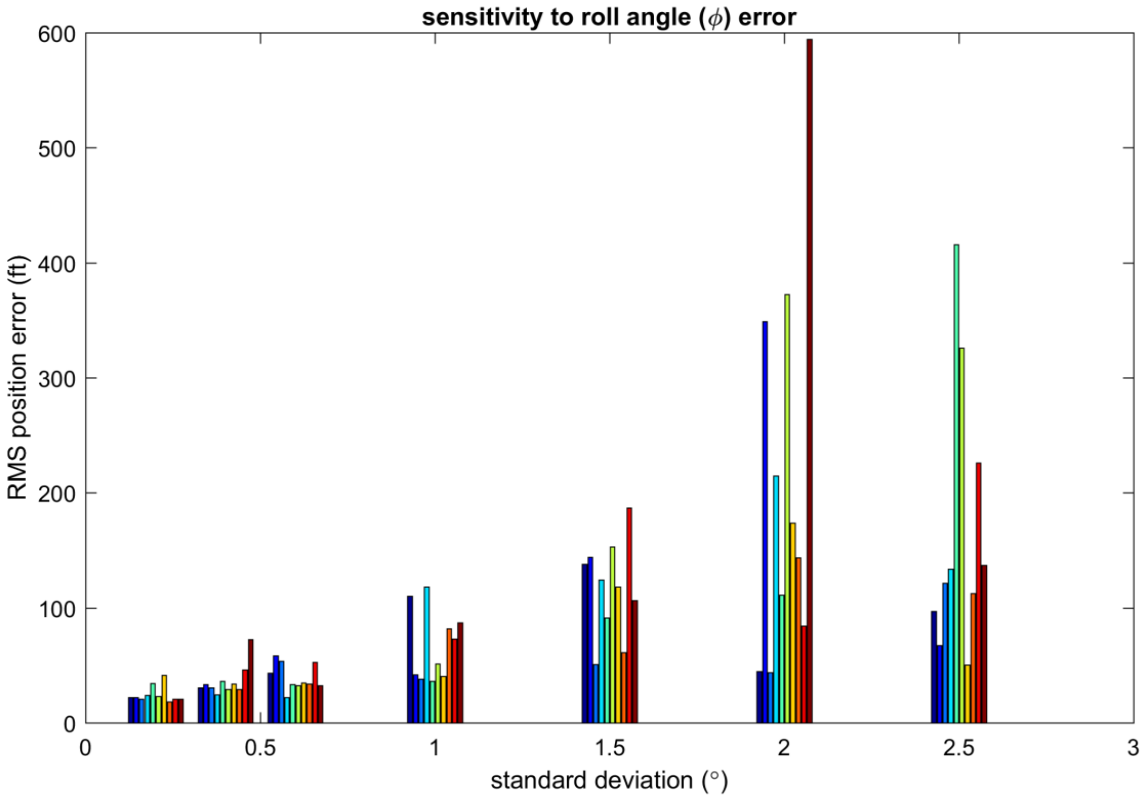


Figure 5-39. RMS position error vs standard deviation of noisy roll angle measurements

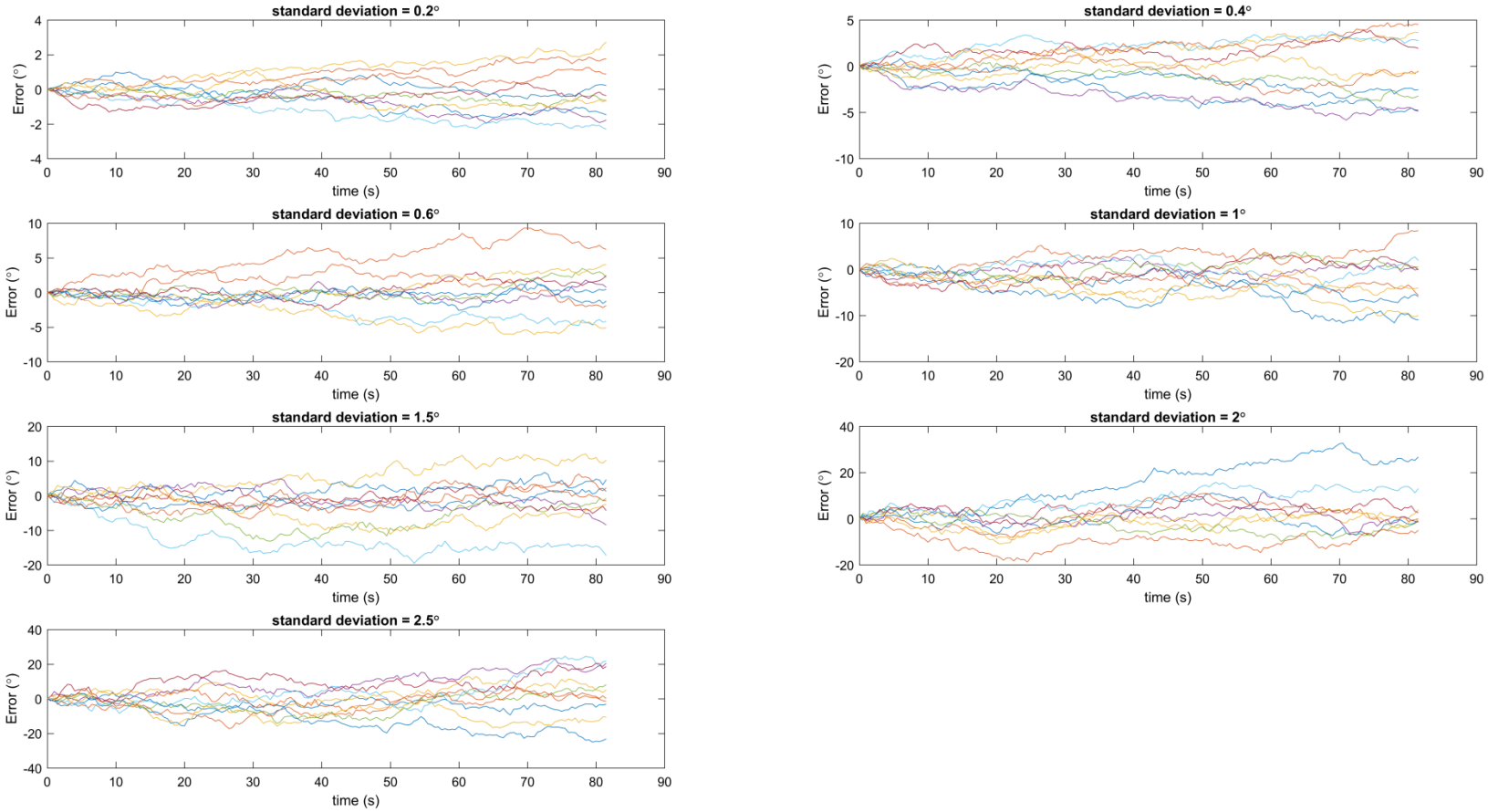


Figure 5-40. Noise signals applied to pitch angle measurements

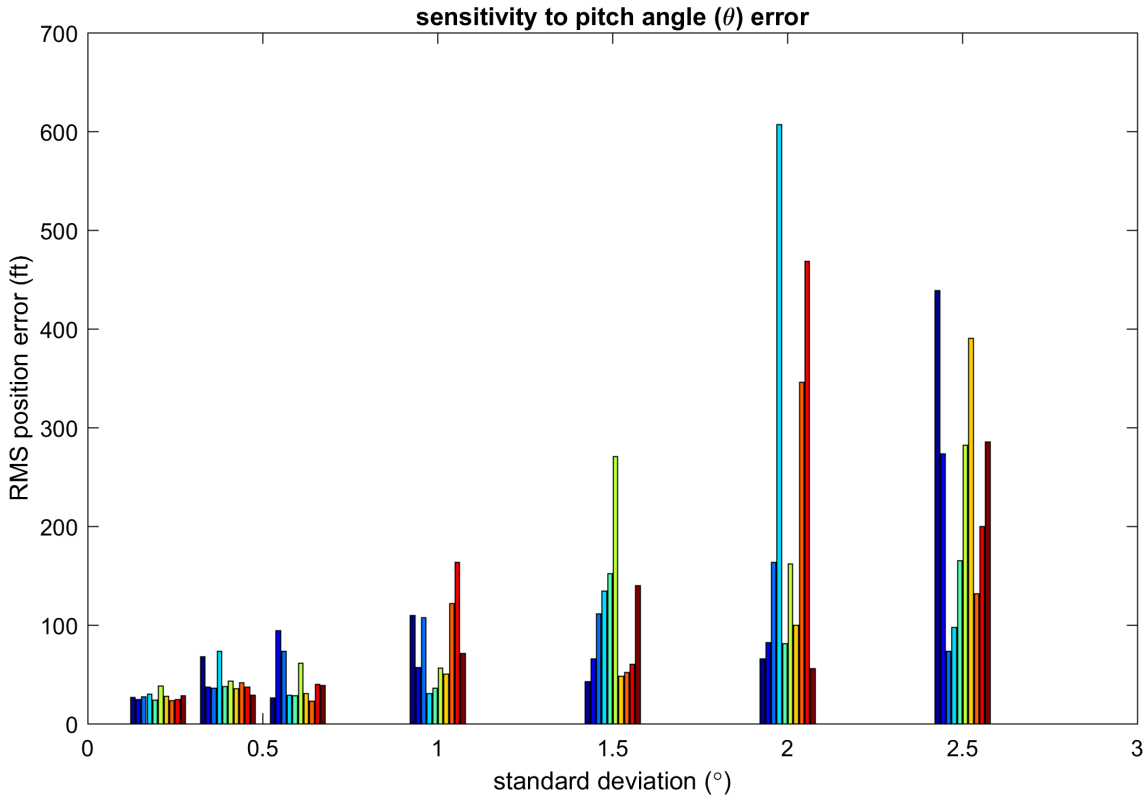


Figure 5-41. RMS error vs standard deviation of noisy pitch angle measurements

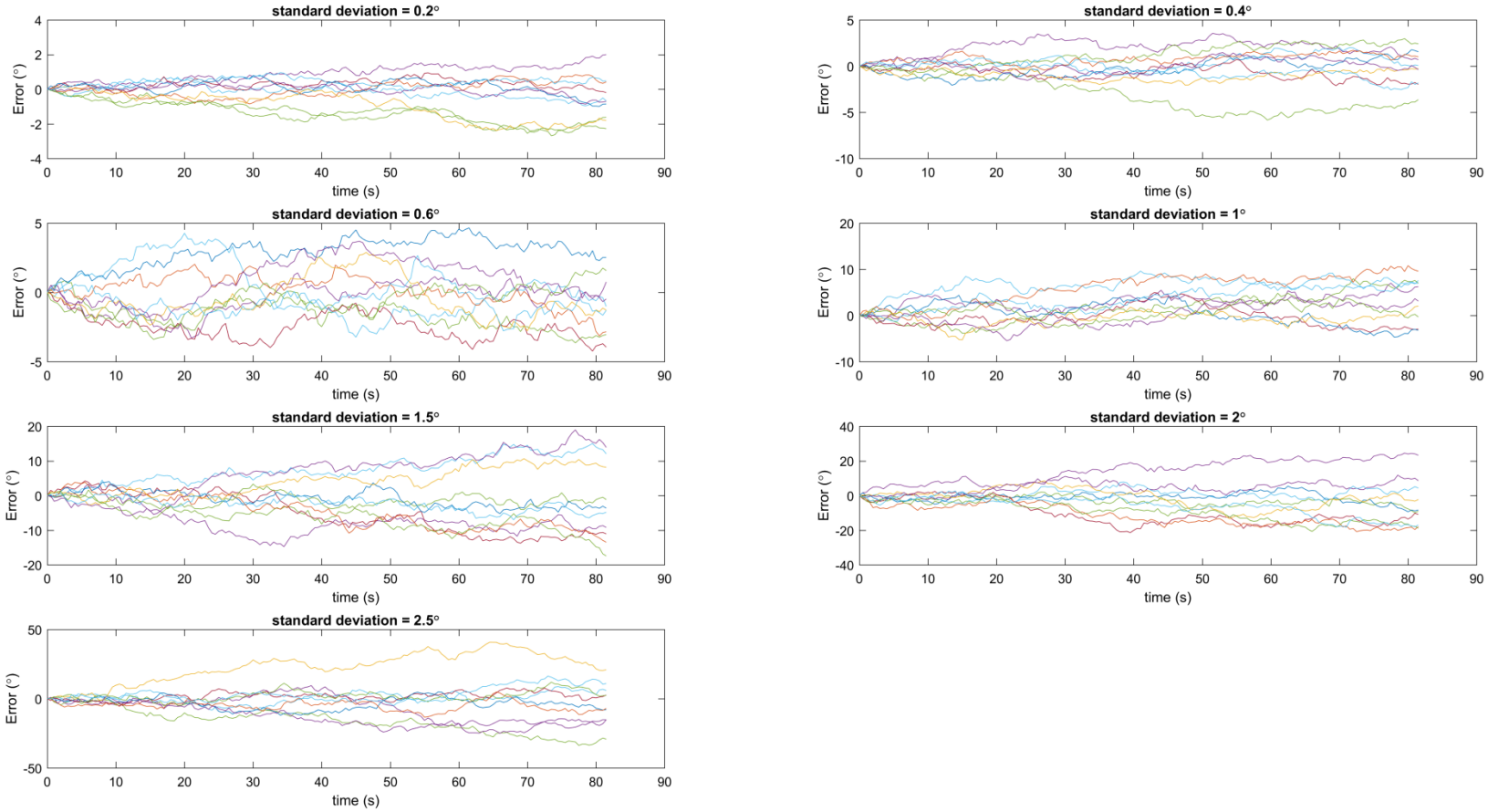


Figure 5-42. Noise signals applied to yaw angle measurements

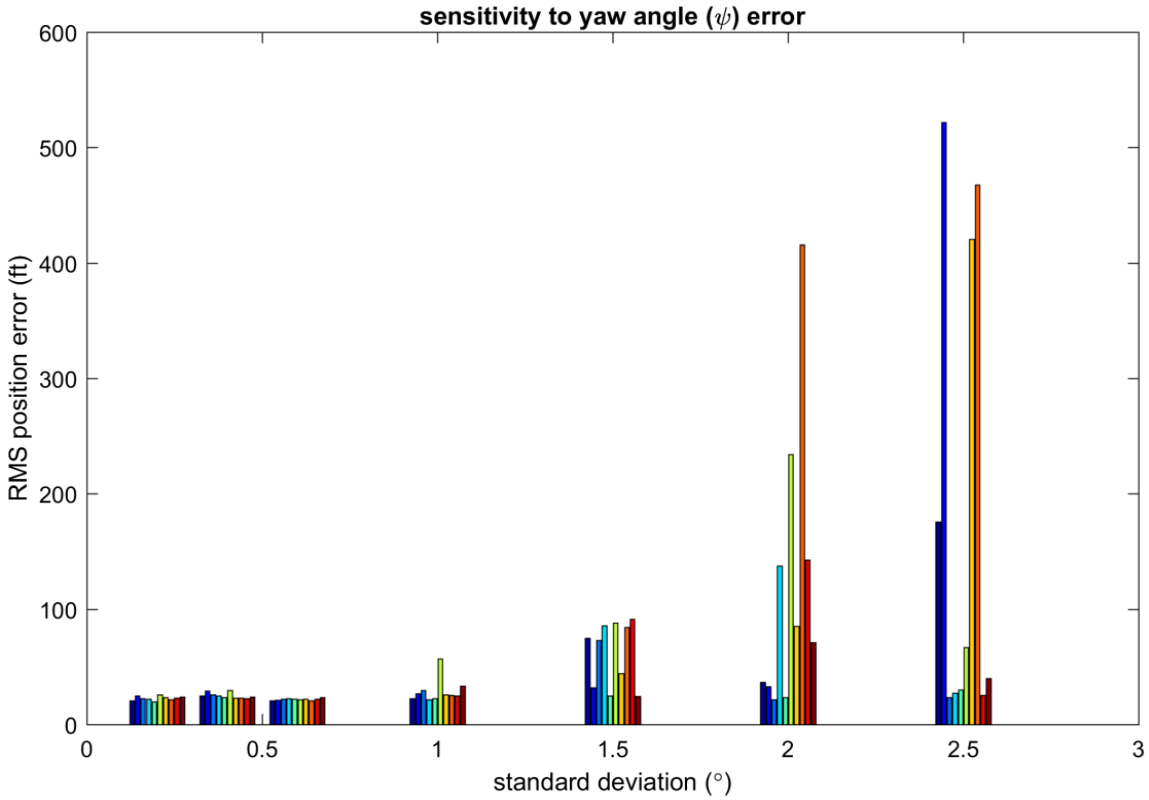


Figure 5-43. RMS error vs standard deviation of noisy yaw angle measurements

Performance

The results show that successful navigation is achievable, and the first method is capable of running in real-time with images captured at a rate of 2 Hz. The average computation time for each loop in MATLAB R2015b is 0.16 seconds on an Intel Core i7-2600 CPU operating at 3.40 GHz. Each loop contains functions to capture a photo, choose appropriate landmark detectors, scan the photo, execute the UKF, and compare wind velocity estimates to those obtained with GPS information.

Computation time for the localization method using generic landmarks was significantly higher. Localization estimation results were obtained by post processing the flights demonstrated in Chapter 5 and Chapter 6. Aerial photos are scanned three times with scanning windows with dimensions of 175 by 175, 200 by 200, and 250 by 250 pixels. As a result, image processing time for each photo requires a maximum of 14 seconds on the same computer described above. The computation time is expected to decrease if the scans are executed in parallel. Furthermore, many of the MATLAB R2015b image processing functions can be executed on graphics processing units (GPUs), which are designed for parallel processing operations. The data association task requires an average of 10 seconds for each image due to numerous "if" statements and 40 "for loops" that cycle through each type of observed landmark and possible landmark. Fast and low flights will further demand shorter computation time.

Due to memory limitations and to increase computation efficiency, the high resolution scenery is automatically divided into a smaller local region in the proximity of the aircraft when capturing aerial photographs in the simulations. For example, if the aircraft flies within a global region of 5 square miles (22000 by 25000 pixels), a local region of 3000 by 3000 pixels is loaded into memory for the camera simulator. The center point of the local region is the current location of the aircraft. Therefore the camera points to a scene that is 3000 by 3000 pixels instead of

22000 by 25000 pixels since the remainder of the scene is not within the camera's field-of-view. For higher altitude flights and steeper turns, the dimensions of the local region must be increased because the camera is able to observe larger areas and farther ground distances.

Chapter 6

Flight Tests, Results

The purpose of the flight test is to validate successful aircraft localization using generic landmarks and limited hardware. A synthetic indoor environment is constructed in an indoor motion capture laboratory with floor space of approximately 625 ft^2 and is depicted in Figure 6-1. A VICON motion capture system accurately measures the location of a flight vehicle. The VICON motion capture system uses retroreflective markers adhered to a subject (vehicle). Infrared cameras arranged around the room track the markers at a rate of 100Hz. The VICON location measurements are compared to the localization method using generic landmarks presented in this research. For the purposes of this demonstration, the VICON position measurements are treated as the actual vehicle locations (truth data) and can be regarded as a high fidelity GPS system. The diagram of the layout is precisely depicted in Figure 6-2 and serves as the reference map of landmarks.

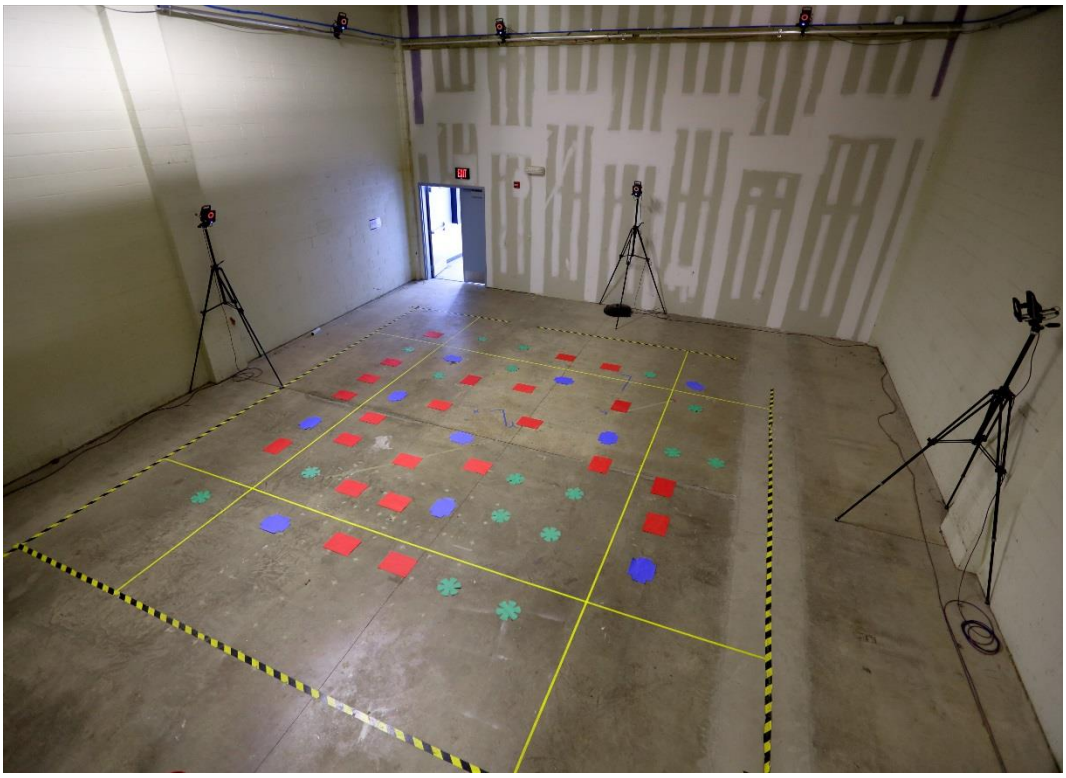


Figure 6-1. Overview photo of the motion capture laboratory with roads and landmarks on the floor.

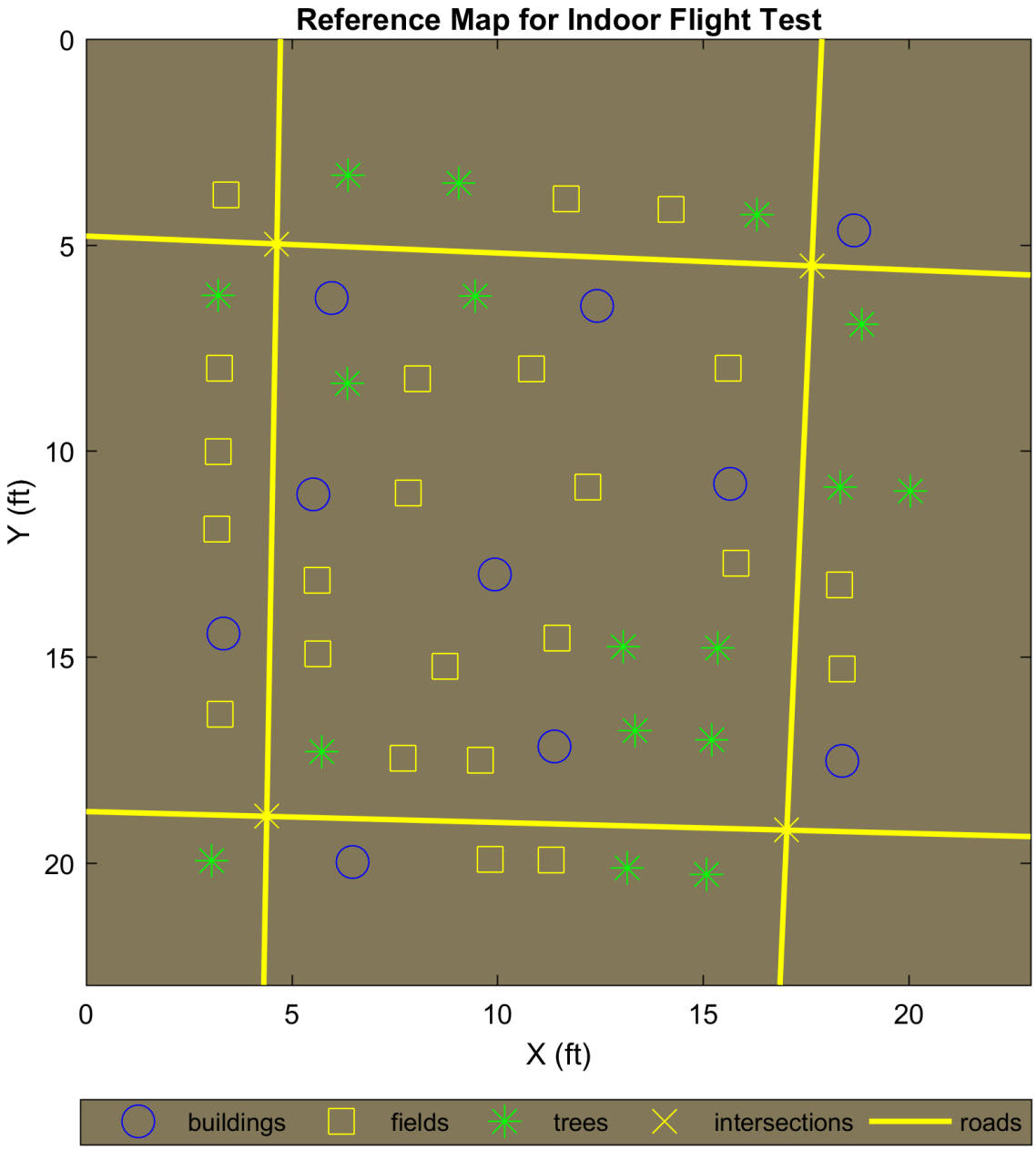


Figure 6-2. Layout of indoor environment for the hardware demonstration.

The flight vehicle is an IRIS+ quadcopter (depicted in Figure 6-3), which is manufactured by 3D Robotics. The IRIS+ is 21.6 in. (550mm) from motor to motor, weighs 2.8 lbs. (1282g) with the 3S 5.8 Amp-hr. lithium polymer battery, and can carry a maximum payload of 0.8 lbs. (400g). Orientation information is obtained with onboard gyro sensors.

Figure 6-3. The IRIS+ quadcopter is used for hardware demonstrations inside the motion capture laboratory.

The flight begins near the lower left corner of the room and ends near the starting position after flying counter-clockwise around the room. The altitude is approximately 10 feet for the duration of the flight. The vehicle dynamics – orientation, speed, and altitude – are updated in the UKF at 20 Hz.

The simulations in the previous chapter modeled the trajectory of an airplane with simple kinematic equations. A quadcopter, however, is used in the laboratory flight demonstration. Therefore the dynamic equations are changed to model the motion of a quadcopter in the UKF Process Model (Prediction Step). Velocity is difficult to measure on a quadcopter with onboard sensors therefore it is approximated by integrating accelerations induced by nonzero roll and pitch angles assuming constant altitude. When the quadcopter rolls or pitches, the thrust vector tilts and causes lateral motion as depicted in Figure 6-4, and additional thrust is necessary to maintain constant altitude.

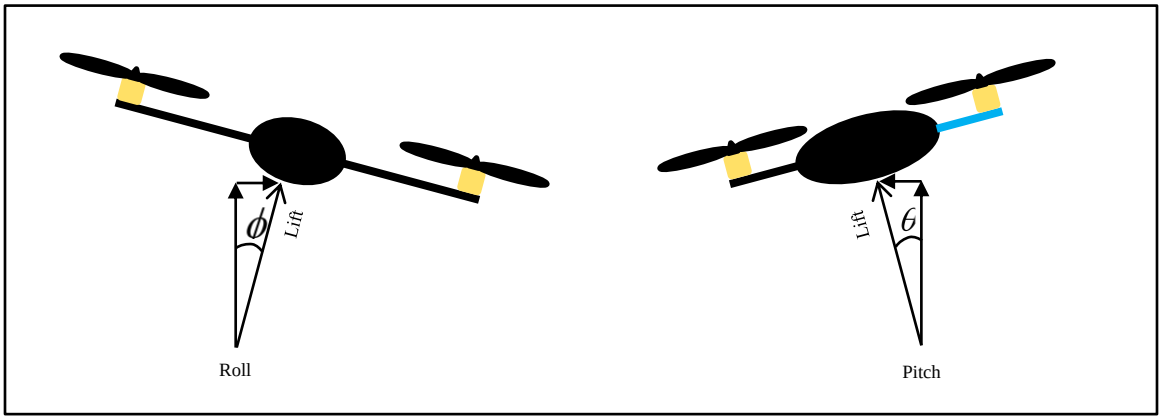


Figure 6-4. Diagram of lift compoenents for side-to-side motion and forward-or-backward motion.

The state vector is expanded in Eq. (49) to allow the UKF to estimate two additional states, u and v, which are the longitudinal and lateral velocity components in the quadcopter body coordinate frame. The control input vector is expanded in Eq. (50) to include the roll, pitch yaw, and altitude.

$$\mathbf{x} = \begin{bmatrix} x & y & W_w & W_n & u & v \end{bmatrix}^{\mathrm{T}} \tag{49}$$

$$\mathbf{u} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^{\mathrm{T}} \tag{50}$$

The following dynamic equations are integrated forward in time to describe the motion of the quadcopter and the predicted position prior to receiving a measurement update.

$$\dot{u} = -g\tan(\theta) \tag{51}$$

$$\dot{v} = g\tan(\phi) \tag{52}$$

$$\dot{x} = u\sin(\psi) + v\cos(\psi) \tag{53}$$

$$\dot{y} = -u\cos(\psi) + v\sin(\psi) \tag{54}$$

Figure 6-5 displays the localization results of the quadcopter flying in the motion capture laboratory. Figure 6-6 displays the position error plotted over time, Figure 6-7 displays altitude plotted over time, and Figure 6-8 displays the wind velocity estimates plotted over time. A sample aerial photograph taken during the flight is displayed in Figure 6-9.

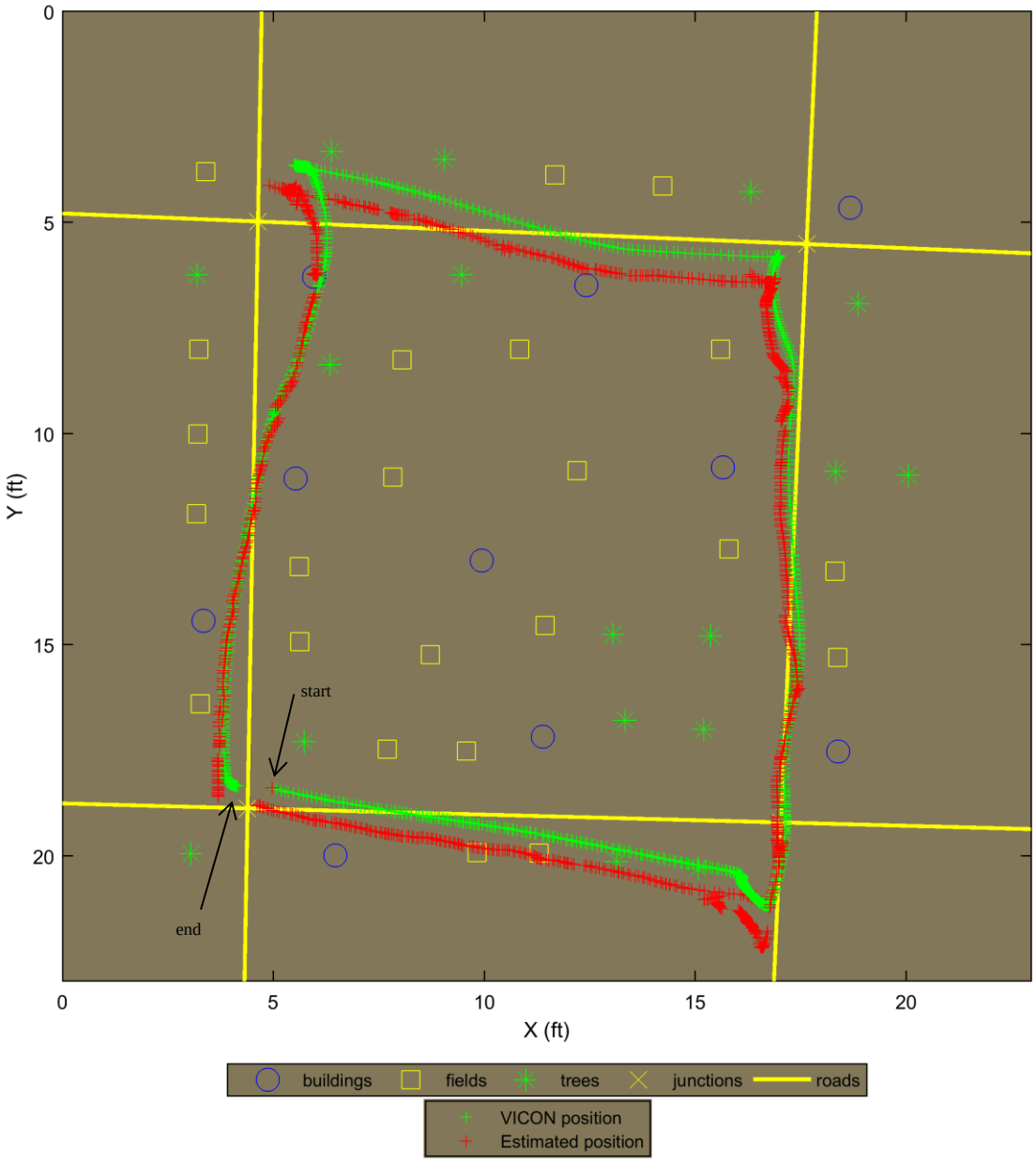


Figure 6-5. Estimated position and VICON position overlaying the reference map of the laboratory. Velocity is estimated by integrating accelerations due to roll and pitch.

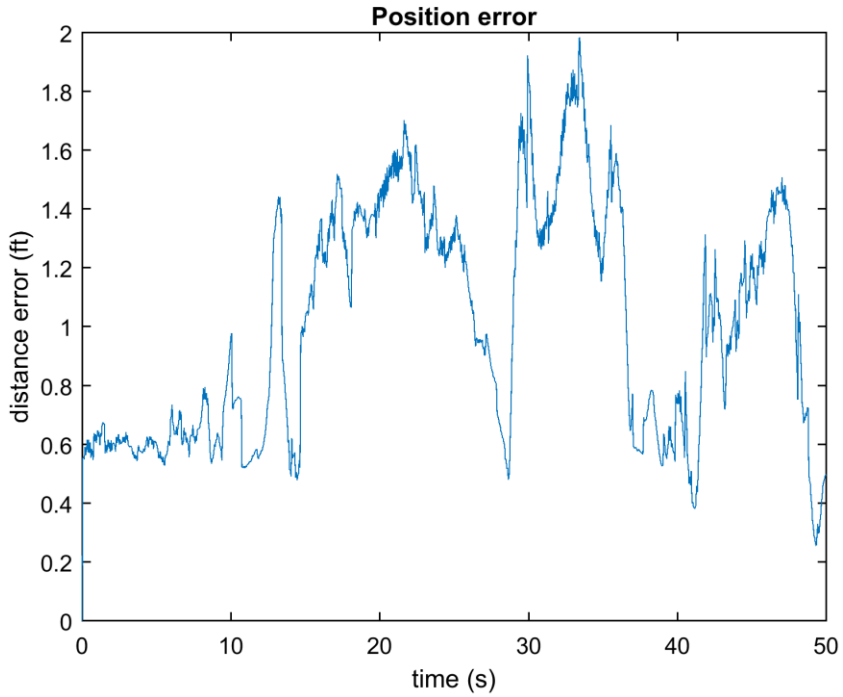


Figure 6-6. Position error

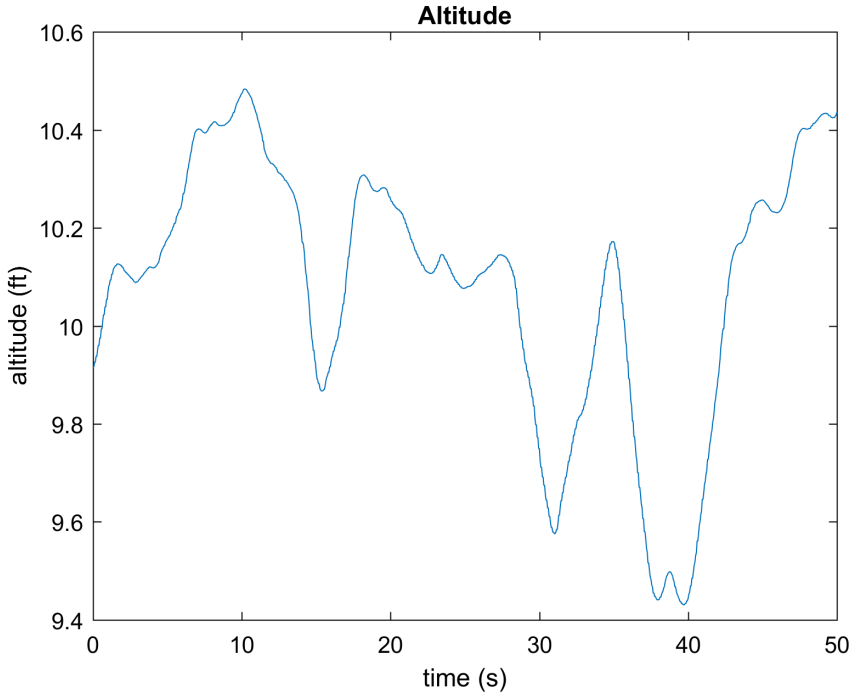


Figure 6-7. Altitude

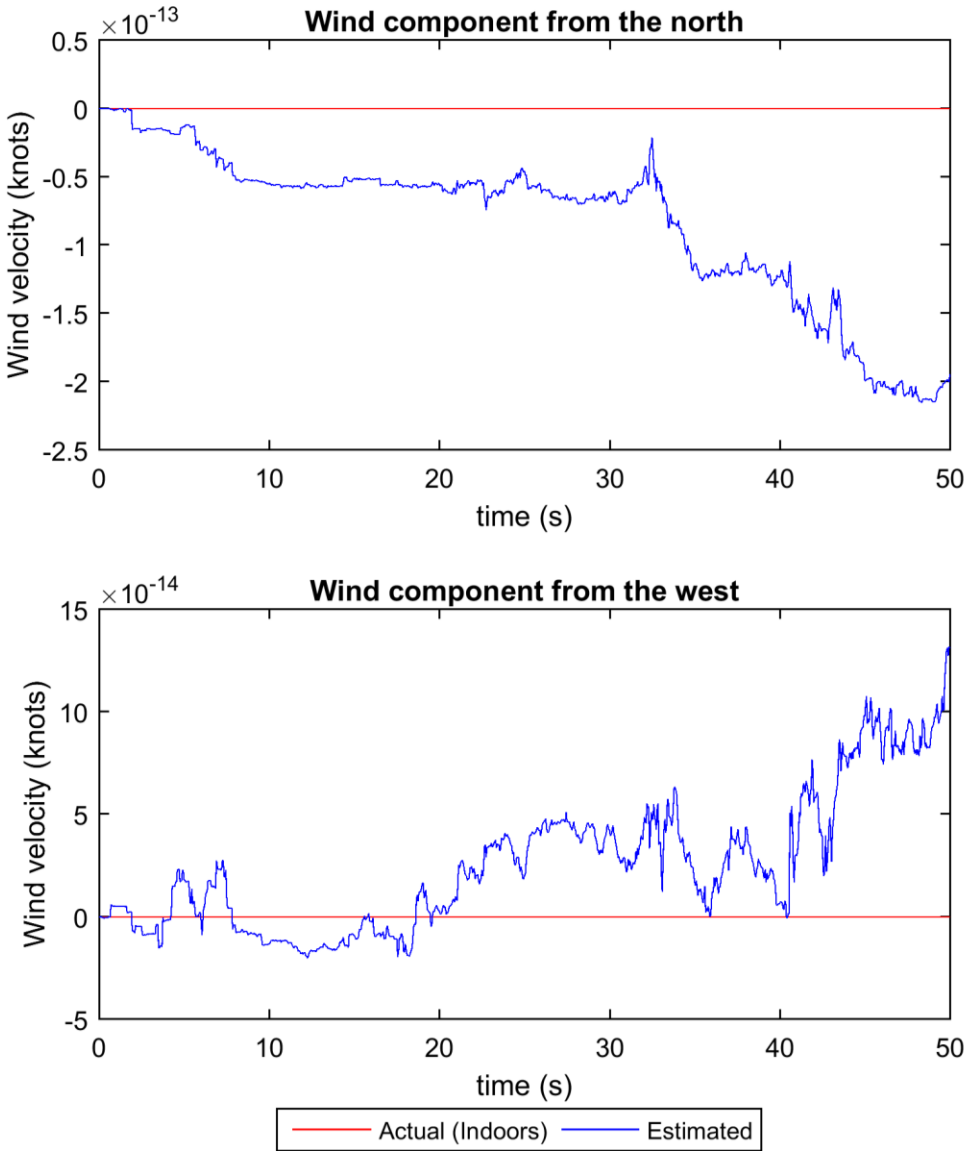


Figure 6-8. Wind estimate. The flight is indoors therefore the wind estimate should be near zero.

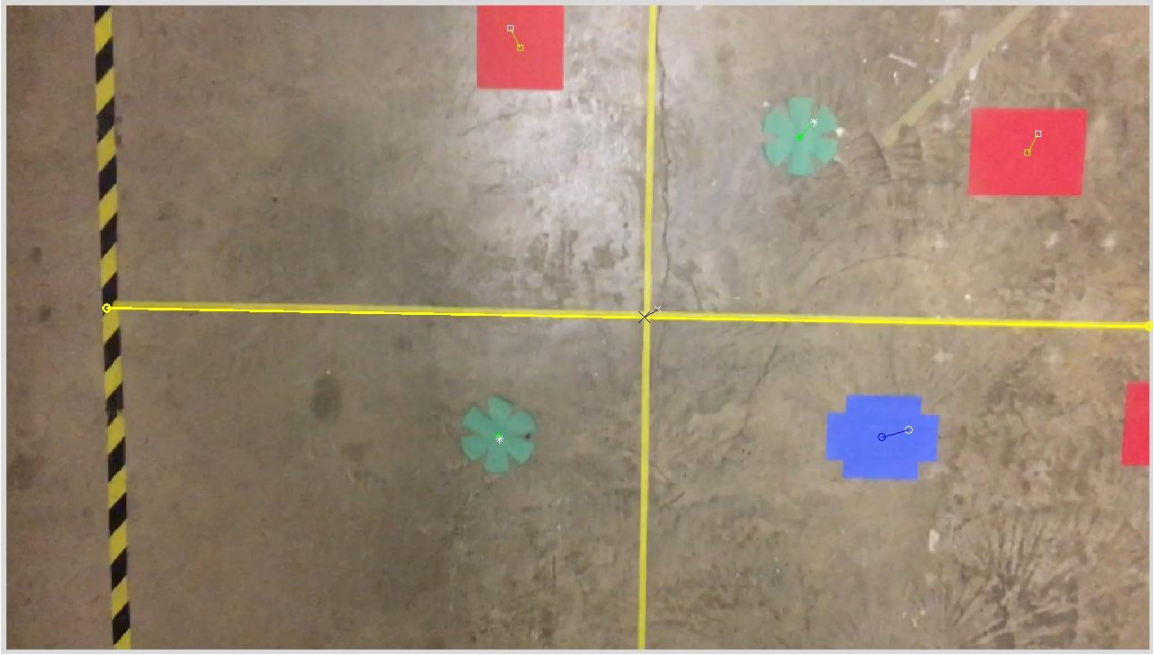


Figure 6-9. A sample aerial photograph captured at the end of the flight is displayed. The white markers indicate the predicted location of the landmarks, and the connecting lines indicate correct data association with the observed landmarks. Resolution: 1920×1080

The photographic measurements ensure vehicle localization within 2 feet. Without measurement updates, the estimated location of the quadcopter quickly diverges. Using only the quadcopter dynamic equations and orientation measurements *but no photographic measurements*, poor localization results are displayed in Figure 6-10. The position error is plotted in Figure 6-11. These poor results signify the importance of obtaining photographic measurements to correct the drift. Therefore the measurement model and the data association algorithm presented in this research are essential elements to successful aircraft localization.

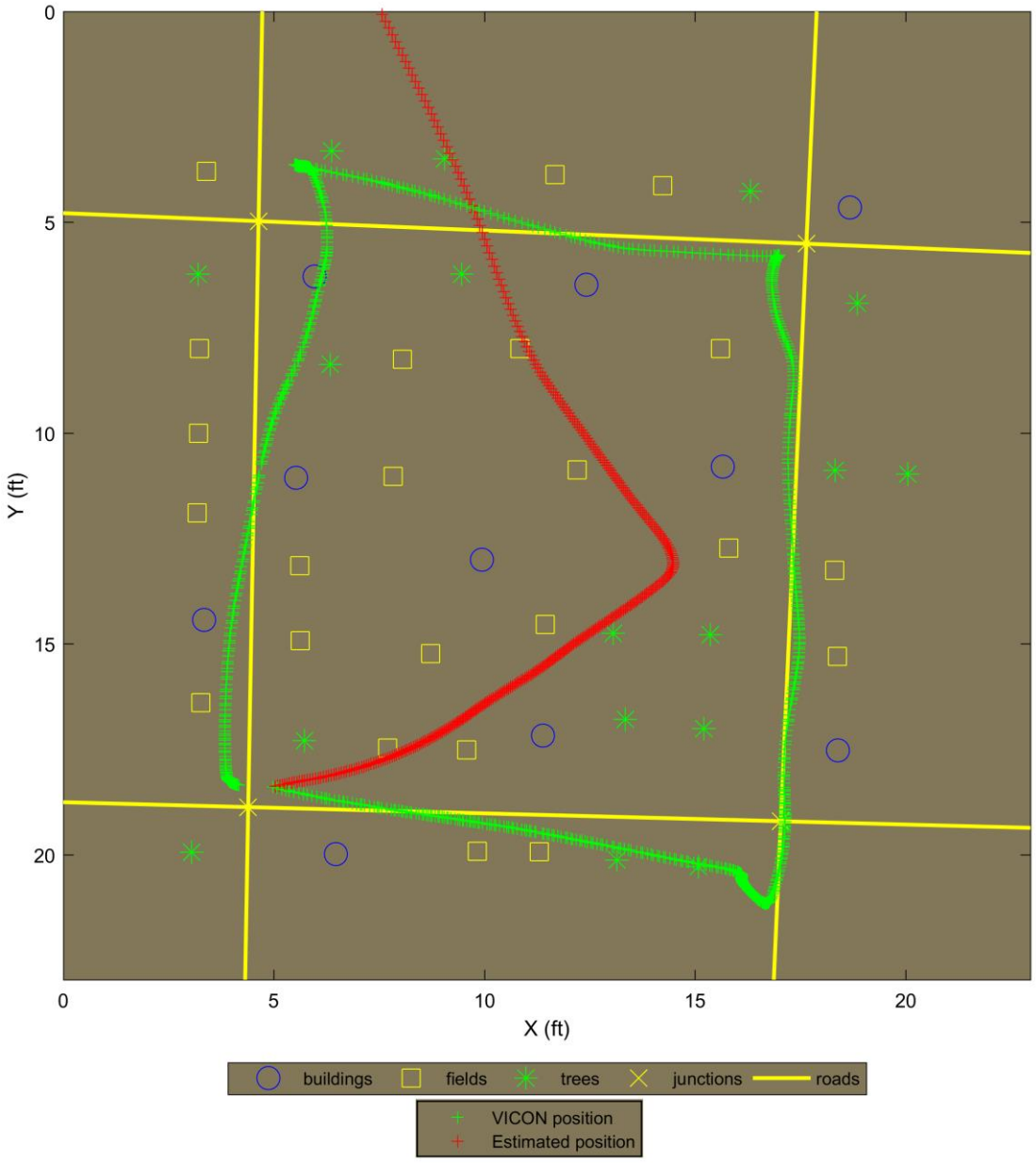


Figure 6-10. Estimated position results without aerial photographs are compared to the VICON (actual) position.

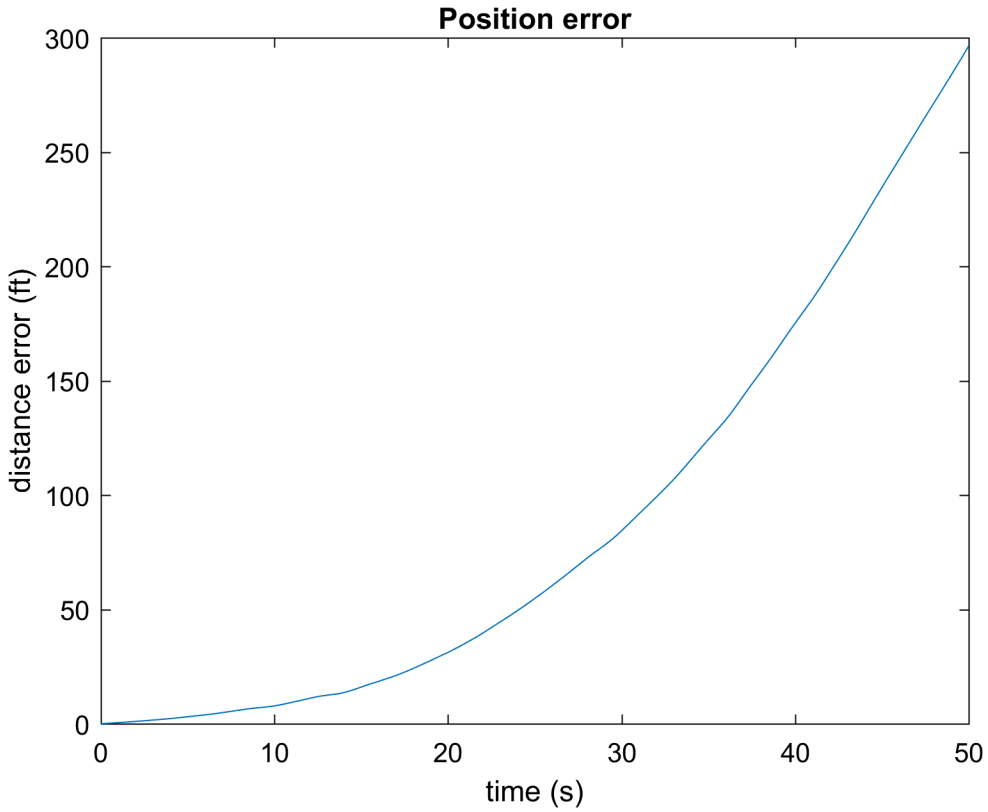


Figure 6-11. Position error without aerial photographs

Integrating pitch and yaw induced accelerations approximate the quadcopter's velocity whereas the VICON motion capture system obtains a very accurate vehicle velocity by differentiating frequent position measurements over small time steps. Since accurate quadcopter velocity *is available in the laboratory environment*, the localization results are processed again to show how an accurate measurement of vehicle velocity changes the position estimation. The state vector and input vector is reset to Eq. (25) and Eq. (26), respectively. Instead of using Eq. (51) through Eq. (54), the equations of motion for the UKF Process Model (Prediction Step) become:

$$\dot{x} = V_x + w_w \tag{55}$$

$$\dot{y} = V_y + w_n \tag{56}$$

When using accurate velocity information, the localization results appear almost identical to the results that use calculated velocities from orientation as seen in Figure 6-12. This indicates that the localization method relies primarily on aerial photographic measurements.

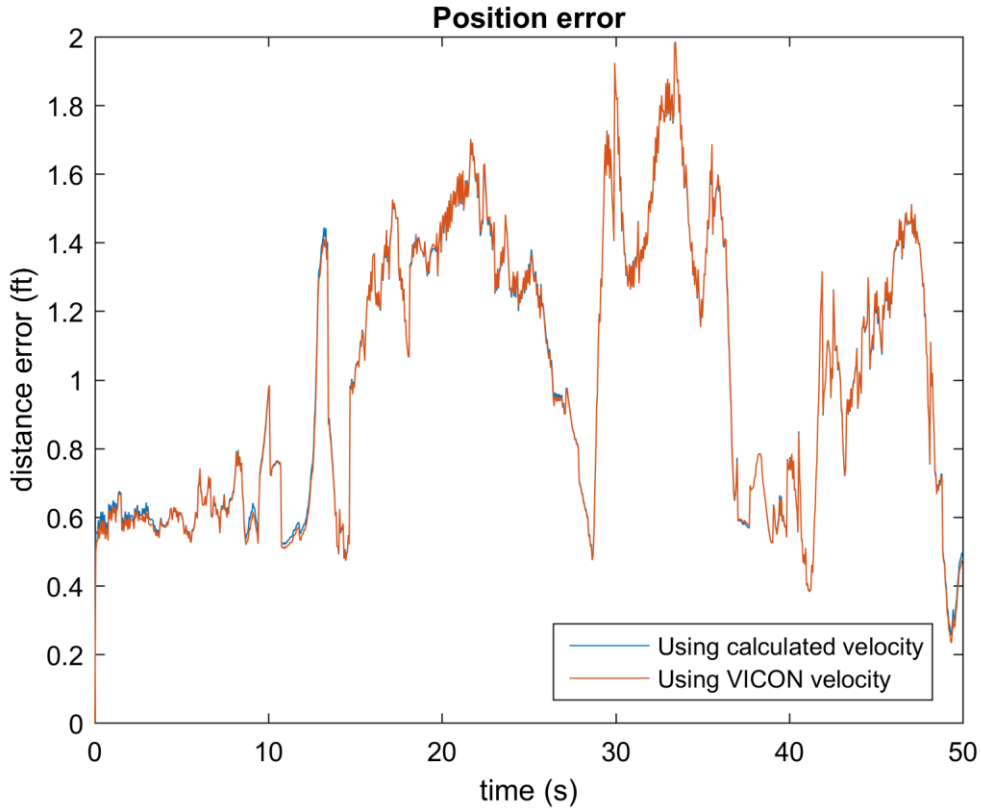


Figure 6-12. Using accurate VICON velocity information does not improve localization results due to the reliance on photographic measurements.

Even though the VICON velocity measurements are accurate, they are not sufficient for vehicle localization without the aerial photograph measurement updates. If no aerial photographs are used as measurement updates, the position error diverges over time as seen in Figure 6-13 and Figure 6-14.

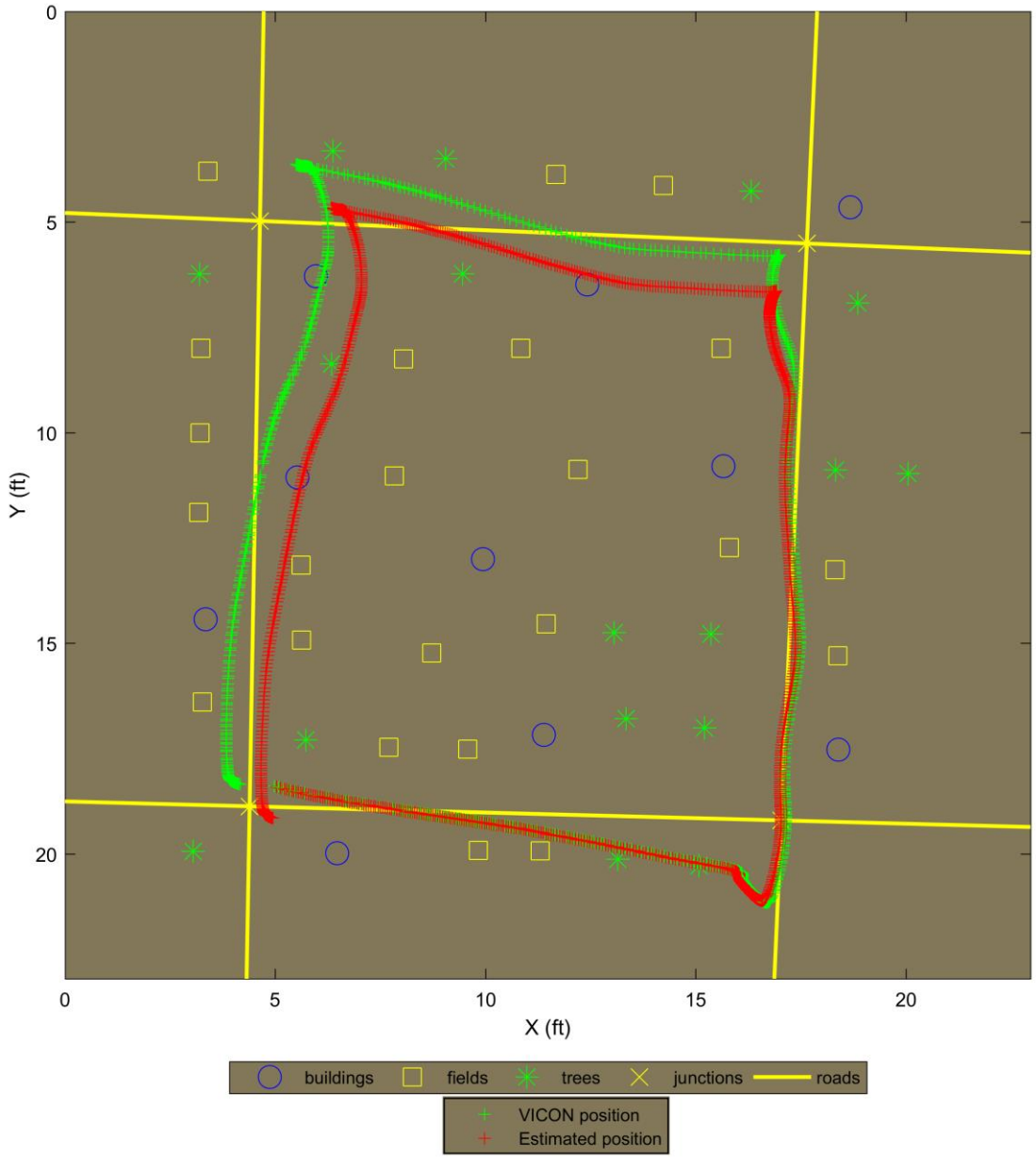


Figure 6-13. Estimated position using VICON velocity measurements but no aerial photo measurements is plotted with VICON position.

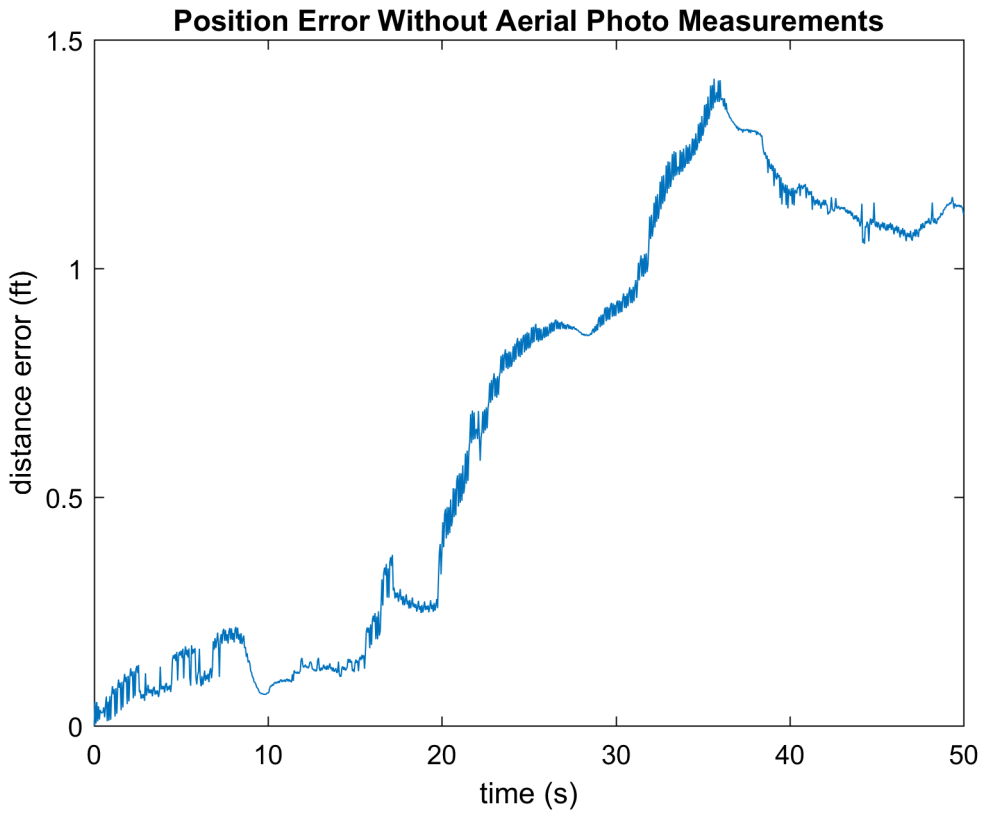


Figure 6-14. Position error using VICON velocity measurements but no aerial photograph measurements

Chapter 7

Conclusion

Summary

This research investigates aerial localization using fixed landmarks on the ground without the help of an external network such as GPS during the flight. Inspiration for this work derives from the way human pilots visually compare the scene features to maps. Assuming the initial location of the aircraft is known during launch, two different localization methods are presented and demonstrate practical localization free from an external network. The first method investigates aircraft localization using predetermined distinct landmarks. Landmarks—buildings, roads, and even a swimming pool—are chosen along a flight path and cascade object detectors are trained to detect them. A photo-realistic flight simulation environment is constructed with actual high resolution imagery and the aircraft dynamics are recorded by a Piccolo Plus autopilot. As the autopilot flies the aircraft, photographs of the ground are captured from a fixed, downward pointing camera. Each photograph is scanned with appropriate detectors as determined by the region manager. The first method uses roads as another important ground reference for localization where landmarks are unobservable. Information such as airspeed, altitude, heading, attitude, and the landmark coordinates are combined into a UKF to obtain an estimate of aircraft position and wind velocities.

This work also shows feasible aircraft localization using generic landmarks such as buildings, fields, trees, and road intersections. Both the aerial images and the reference map uses the same object detection technique – bag of visual words – to automatically classify locations of buildings, fields, and trees. Association between the aerial images and the map employs spatial relationships between categories of abundant ground landmarks. As expected, simulations have

shown position divergence when overflying a scene lacking landmark diversity. In other words, if the aerial images contain a single, uniform category only (such as a dense forest), localization error increases until other categories "break-up" the scene. One solution is to climb higher until multiple landmark categories are detectable in the image frame. An advantage of this method is automation because distinct landmarks are not chosen prior to the flight. Furthermore, image data is simplified to three categories of human-understandable objects, which can be easily verified. Another advantage is the method's ability to overcome incorrect object classification especially when the classification results are consistently incorrect for both the reference map and the aerial photograph. Many more measurements are evaluated using the generic landmark points compared to the distinct landmark localization method, which uses fewer (and sometimes a single point) distinct landmark coordinates. When using distinct landmarks, a false detection is far more detrimental to the localization solution.

Future Work

The hardware demonstration shows feasible aircraft localization on a small scale without reliance on an external network. The VICON motion capture system is used to verify the position estimation results. Ideally, future work would involve an airplane flight demonstration at locations similar to the simulations for a direct comparison of real world results to simulation results.

Error prediction and quantification is complex due to several factors. In the case of generic landmark detection, landmark locations are recorded as 2D coordinates of the center of a scanning window. The object classifier simply reports the presence of either buildings, fields, or trees within a queried area (the texture contained within the scanning window), but the UKF implementation requires measurements to be reported as a pair of numerical coordinates. The

small scanning window may initially contain regions of multiple categories (such as trees and fields), but the classifier can report a single category for the texture only. Therefore the window is rotated around a local neighborhood until a maximum score of the initial classification is obtained. Since the scale of the landmarks in the reference map differs from the scale of landmarks in the aerial photos due to focal length and altitude, the coordinates reported in the aerial photo will not exactly match the respective location of the landmark in the reference map. A comparison of Figure 3-11 and Figure 3-12 illustrates the discrepancy between the placement of blue markers, yellow markers, and green markers in the aerial photo and reference map. In other words, the placement of the landmark markers in the reference map are close to the placement of markers in the aerial photos for the same landmark categories, but the precise location differs. The aircraft localization error would decrease if markers placed in the reference map can be placed over the exact respective location in the aerial photo. Future work could involve studies to quantify how the localization error is affected by inconsistent location of markers on the reference map and the aerial photos.

Correct data association is a critical step for successful measurement update corrections. Future work could further develop methods to more intelligently match observed landmarks to reference landmarks. Alternative ways of scoring observed landmarks to possible landmarks can also be explored. When localization error diverges, the observed landmarks are matched to the wrong reference landmarks. In that case, a method to reinitialize the measurement update with a good measurement is needed. Therefore, future work can also blend both distinct landmark detection and generic landmark detection methods together. The method will also break down if subsequent aerial photographs contain uniform categories of landmarks. Example include flying over a vast, dense forest or large fields at low altitudes. A way of quantifying regions with sufficient landmark diversity could help generate feasible flight regions and altitudes to ensure satisfactory localization estimates.

The measurement model is another factor that contributes to localization errors. The measurement model predicts where landmarks should appear in the image. If the model does not accurately model the camera, the localization estimate degrades due to the discrepancy between predicted landmark location and the actual landmark location in the aerial photograph. In the simulations, the measurement model can accurately model the simulated camera within 20 pixels. However, at higher altitudes, each pixel represents areas that span longer distances, and therefore localization error increases at higher altitudes. Future work could involve the development of advanced measurement models or predict localization error as a function of altitude.

Future work could also investigate localization with a forward-looking camera. The current measurement model provides a framework to allow changes in the camera angles by the multiplication of coordinate transformation matrices. The coordinate transformations realistically depict a 3D scene into a 2D image plane and accounts for aircraft attitude and camera orientation.

The localization with distinct landmarks method could benefit from advanced object detection techniques. The cascade object detector is sensitive to aspect ratios because it discriminates shapes but not color. Therefore landmarks with unique shapes should be chosen. Otherwise, if multiple landmarks have similar shapes, the detector will identify all of them as the same landmark. In that case, an entire industrial site or housing plan could be used as a single landmark. The cascade object detection is also sensitive to image rotations therefore real world flight tests are needed to verify its robustness to errors in heading angles. Future work could explore better object detection techniques that incorporate both color and shape information.

The computer vision road detection methods use fundamental image processing techniques that can be fooled by various environmental factors. The intention of road detection in this research is to extract necessary data to complete the road localization and data association tasks. Advanced road detection algorithms may be more practical for real-world applications.

Furthermore, straight roads are simplified as line segments, but many roads have curves, which can be exploited as unique features in the scene.

One criterion of this research is the use of low cost sensors such as digital cameras that detect visible light. Low cost sensors and low complexity allows for hardware validation with available resources. Future work could investigate the use of multispectral sensing of the environment. Passive sensors such as infrared cameras or active sensors such as light detection and ranging (LiDAR) may introduce new attributes for landmark detection and discrimination. Further research could also explore applications to underwater vehicles equipped with a sidescan sonar.

Appendix A

Alternative Measurement Model

An alternative measurement model is included in this appendix. The model presented here assumes a simple pinhole camera model and does not require knowledge of the lens focal length. Instead, it uses the camera's field-of-view angles, which are easy to approximate by measuring the horizontal and vertical distances of a flat surface depicted in the image plane and distance from the camera to the flat surface. The model is less accurate than the model presented in Chapter 4, especially near the edges of the frame.

The transformation matrices follow the same rotations depicted previously in Figure 4-3, Figure 4-4, Figure 4-5, Figure 4-6, and Figure 4-7, but are defined in Cartesian coordinates instead of homogeneous coordinates. The matrix multiplication is shown in Eq. (62).

$$T^{O} = \begin{bmatrix} L_x^W - P_x \\ L_y^W - P_y \\ L_z^W + h \end{bmatrix} \tag{57}$$

$$T_{\psi}^{b} = \begin{bmatrix} \sin(\psi) & -\cos(\psi) & 0 \\ \cos(\psi) & \sin(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{58}$$

$$T_{\theta}^{b} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{59}$$

$$T_{\phi}^{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{60}$$

$$T^{R} = \begin{bmatrix} \cos(90°) & \sin(90°) & 0 \\ -\sin(90°) & \cos(90°) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{61}$$

$$\vec{\mathbf{L}}_i^C = \mathbf{T}^R \mathbf{T}^{\phi} \mathbf{T}^{\theta} \mathbf{T}^{\psi} \mathbf{T}^O \vec{\mathbf{L}}_i^W \tag{62}$$

To describe all $\bar{\mathbf{L}}_{\mathbf{i}}^{\mathbf{C}}$ vectors in the image plane, a modified pinhole model projects the $\bar{\mathbf{L}}_{\mathbf{i}}^{\mathbf{C}}$ coordinates with azimuth (γ_x) and depression (γ_y) angles through the image plane [23]. As depicted in Figure A- 1, γ_x is $\arctan\left(L_x^C / L_z^C\right)$, where L_x^C and L_z^C are the *x* component and *z* component, respectively, of the vector between the ground point and the camera. γ_y is $\arctan\left(L_y^C / L_z^C\right)$, where L_y^C is the *y* component of the vector between the ground point and the camera.

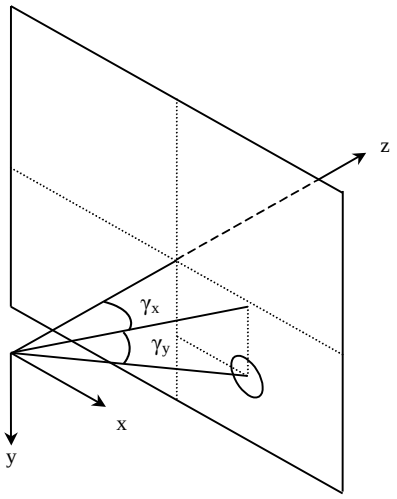


Figure A- 1. Azimuth and depression angles projected to the image plane define the location of each ground point in the camera coordinates.

Both angles associated with each ground point are converted into pixel locations on the image plane with knowledge of the camera properties such as field-of-view and image resolution [71]. The conversation factors N_x/α and N_y/β are multiplied with γ_x and γ_y, respectively to portray the *x* and *y* pixel coordinates.

$$I_x^C = \gamma_x\left(\frac{N_x}{\alpha}\right) \tag{63}$$

$$I_y^C = \gamma_y\left(\frac{N_y}{\beta}\right) \tag{64}$$

Figure A- 2 shows the image plane dimensioned with appropriate labels. N_y is the number of horizontal pixels, and N_z is the number of vertical pixels. α is the horizontal field-of-view angle, and β is the vertical field-of-view angle.

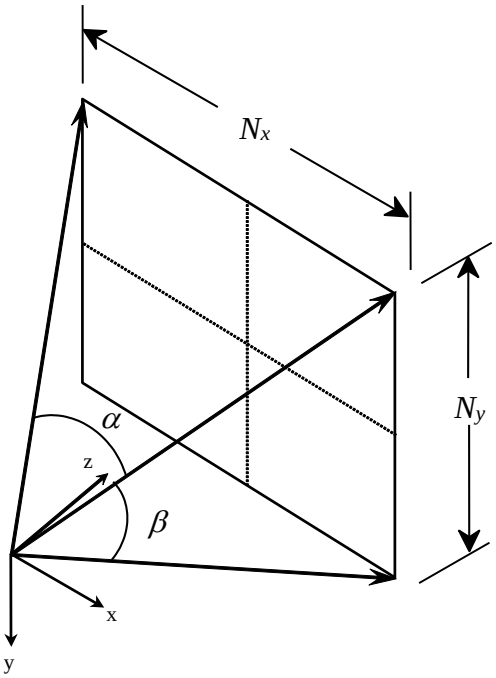


Figure A- 2. The image plane is defined by the camera's y and z coordinates.

Finally, the origin of the image plane is shifted to the upper left corner for consistency with image software. Therefore the location of a landmark in 2-D image plane coordinates is defined by Eq. (65) and Eq. (66).

$$I_x = I_x^C + \frac{N_x}{2} \tag{65}$$

$$I_y = I_y^C + \frac{N_y}{2} \tag{66}$$

Bibliography

1 Leonard, J. J., A. A. Bennet, C. M. Smith, and H. J. S. Feder. "Autonomous Underwater Vehicle Navigation." *MIT Marine Robotics Laboratory Technical Memorandum 98-1* (1998).

2 United States of America. Department of the Navy. *The Navy Unmanned Undersea Vehicle (UUV) Master Plan*. 3, 2004. Web. <www.navy.mil/navydata/technology/uuvmp.pdf>.

3 RAND Corporation. *A Survey of Missions for Unmanned Undersea Vehicles*. By R. W. Button, J. Kamp, T. B. Curtin, and J. Dryden. U.S. Navy, 2009. Web. <http://www.rand.org/pubs/monographs/MG808.html>.

4 "Federal Register, Volume 77 Issue 47." U.S. Government Printing Office, 9 Mar. 2012. Web. 03 Apr. 2012. <http://www.gpo.gov/fdsys/pkg/FR-2012-03-09/html/2012-5735.htm>.

5 UAV Global. July 21, 2016 Web. < http://www.uavglobal.com/list-of-manufacturers/>

6 United States of America. Department of Defense. *Unmanned Systems Integrated Roadmap FY2011-2036* 2013. Web. <http://www.defense.gov/pubs/DOD-USRM-2013.pdf>

7 Conte, G and P. Doherty. "Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information." *EURASIP Journal of Advances in Signal Processing* 2009.387308 (2009): 18.

8 Prazenica, R.J., A. Watkins, A. J. Kurdila, Q. F. Ke, and T. Kanade. "Vision-based Kalman Filtering for Aircraft State Estimation and Structure from Motion." *AIAA Guidance, Navigation, and Control Conference Proceedings* (2007).

9 Call, B., R. Beard, and C. Taylor. "Obstacle Avoidance For Unmanned Air Vehicles

Using Image Feature Tracking" *AIAA Guidance, Navigation, and Control Conference* (2006).

10 Robertson, D. and R. Cipolla. "An Image-Based System for Urban Navigation," 2004 Web. <ftp://svr-ftp.eng.cam.ac.uk/pub/reports/cipolla_bmvc04.pdf>

11 Wu, Allen D. *Vision-based Navigation and Mapping for Flight in GPS-Denied Environments*. Diss. Georgia Institute of Technology, 2010. Web. <https://smartech.gatech.edu/handle/1853/37281>.

12 Kim, J. and S. Sukkarieh. "Autonomous Airborne Navigation in Unknown Terrain Environments," *IEEE Transactions on Aerospace and Electronic Systems*, 40.3 (2004).

13 Caballero, F., L. Merino, J. Ferruz, and A. Ollero. "Vision-based Odometry and SLAM for Medium and High Altitude Flying UAVs." *Journal of Intelligent and Robotic Systems* 54 (2008): 137-167.

14 Watanabe, Y.., E. N. Johnson, and A. J. Calise. "Stochastic Guidance Design for UAV Vision-based Control Applications." *AIAA Guidance, Navigation and Control Conference* (2008).

15 Steder, B. G. Gristetti, C. Stachniss, and W. Burgard. "Visual SLAM for Flying Vehicles." *IEEE Transactions on Robotics and Automation* 24.5 (2008): 1088-1093.

16 Jensfelt, P, D. Kragic, J. Folkesson, and M. Björkman. "A Framework for Vision Based Bearing Only 3D SLAM." *Proceedings of the IEEE International Conference on Robotics and Automation* (2006).

17 Piniés, T. Lupton, S. Sukkarieh, and J. D. Tardós. "Inertial Aiding of Inverse Depth SLAM Using a Monocular Camera." *Proceedings of the IEEE International Conference on Robotics and Automation* (2007).

18 Kümmerle, R., B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. "Large Scale Graph-based SLAM Using Aerial Images as Prior Information." *Autonomous Robots* 30.1 (2011): 25-39.

19 Saripalli, S., J.F. Montgomery, and G. S. Sukhatme. "Vision-based Autonomous Landing of an Unmanned Aerial Vehicle." *Proceedings of the 2002 IEEE International Conference on Robotics 8 Automation* (2002): 2799-2804.

20 Meingast, M., C. Geyer, and S. Sastry. "Vision Based Terrain Recovery for Landing Unmanned Aerial Vehicles." *43rd IEEE Conference on Decision and Control* (2004): 1670-1675.

21 Merz, T., S. Duranti, and G. Conte. "Autonomous Landing of an Unmanned Helicopter based on Vision and Inertial Sensing." *Experimental Robotics IX, Springer Tracts in Advanced Robotics* 21 (2006): 343-352.

22 Frietsch, N., O. Meister, C. Schlaile, J. Seibold, and G.F. Trommer. "Vision Based Hovering and Landing System for a VTOL-MAV with Geo-Localization Capabilities." *AIAA Guidance, Navigation and Control Conference* (2008).

23 Langelaan, Jacob W. *State Estimation for Autonomous Flight in Cluttered Environments*. Ph.D. Dissertation. Dept. of Aeronautics and Astronautics, Stanford University, Stanford, CA, 2006. Print.

24 Celik, K., S.J. Chung, M. Clausman, and A. K. Somani. "Biologically Inspired Monocular Vision Based Navigation and Mapping in GPS-Denied Environments." *AIAA Infotech@Aerospace Conference* (2009).

25 Madison, R. W., G. L. Andrews, P. A. DeBitetto, S. A. Rasmussen, and M. S. Bottkol. "Vision-Aided Navigation for Small UAVs in GPS-Challenged Environments." *The Charles Stark Draper Laboratory, Inc.*, Presented at *AIAA Infotech at Aerospace Conference* (2007).

26 Dogruer, C. U., B. Koku, and M. Dolen. "A Novel Soft-Computing Technique to Segment Satellite Images for Mobile Robot Localization and Navigation." *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2007): 2077-2082.

27 Dogruer, C. U., B. Koku, and M. Dolen. "Global Urban Localization of Outdoor Mobile Robots Using Satellite Images." *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008): 3927-3932.

28 Viswanathan, A., B. R. Pires, and D. Huber. "Vision Based Robot Localization by Ground to Satellite Matching in GPS-denied Situations," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* (2014): 192-198.

29 van Dalen, G. J. J., D. P. Magree, and E. N. Johnson. "Absolute Localization using Image Alignment and Particle Filtering." *AIAA SciTech Proceedings* (2016).

30 Elfes, A. "Sonar-based Real-world Mapping and Navigation." *IEEE Journal on Robotics and Automation* 3.3 (1987): 249-265.

31 Lane, D. M., and J. P. Stoner. "Automatic Interpretation of Sonar Imagery Using Qualitative Feature Matching." *IEEE Journal of Oceanic Engineering* 19.3 (1994): 391-405.

32 Lane, D.M., M. J. Chantler, and Dongyong Dai. "Robust Tracking of Multiple Objects in Sector-scan Sonar Image Sequences Using Optical Flow Motion Estimation." *IEEE Journal of Oceanic Engineering* 23.1 (1998): 31-46.

33 Trimble, G. M. "Underwater Object Recognition and Automatic Positioning to Support Dynamic Positioning." *Proc. UUST-7,* Univ. of New Hampshire, (1991).

34 Cuschieri, J., and S. Negahdaripour. "Use of Forward Scan Sonar Images for Positioning and Navigation by an AUV." *OCEANS '98 Conference Proceedings* 2 (1998): 752-756.

35 Ruiz, I. T., S. de Raucourt, Y. Petillot, and D.M. Lane. "Concurrent Mapping and Localization Using Sidescan Sonar." *IEEE Journal of Oceanic Engineering* 29.2 (2004): 442-456.

36 Zerr, B., G. Mailfert, A. Bertholom, and H. Ayreault. "Sidescan Sonar Image Processing for AUV Navigation." *Oceans 2005 - Europe* 1 (2005): 20-23.

37 Kerneis, D., B. Zerr. "Comparison of Multisensor Fusion Methods for Seabed Classification." *Oceans 2005 - Europe* 2 (2005): 878-882.

38 Vapnik, V. N. *Statistical Learning Theory*. New York: John Wiley and Sons, 1998.

39 Lanaaya, H., A. Martin, D. Aboutajdine, A.H. Khenchaf. "A New Dimensionality Reduction Method for Seabed Characterization: Supervised Curvilinear Component Analysis." *Oceans 2005 - Europe* 1 (2005): 339-344.

40 Lanaaya, H., A. Martin, A. Khenchaf, and D. Aboutajdine. "Dimensionality Reduction of Sonar Images for Sediments Classification" *Colloque Carctérisation in-situ des Fonds Marins* (2004).

41 Lanaaya, H., A. Martin, D. Aboutajdine, A.H. Khenchaf. "Feature Selection Using Genetic Algorithm for Sonar Images Classification with Support Vector Machines" *European Conference on Propagation and Systems* 24.11 (2005).

42 Tao, W., J. Zhao, J. Liu, H. Zhang. "Study on the Sidescan Sonar Image Matching Navigation Based on SURF." *2010 International Conference on Electrical and Control Engineering (ICECE)* (2010): 2181-2184.

43 Nad, D., N. Miskovic, V. Djapic, Z. Vukic. "Sonar Aided Navigation and Control of Small UUVs." *2011 19th Mediterranean Conference on Control & Automation (MED)* (2011): 418-423.

44 Groen, J., E. Coiras, J. Del Rio Vera, B. Evans. "Model-based Sea Mine Classification with Synthetic Aperture Sonar." *Radar, Sonar & Navigation, IET* , 4.1 (2010): 62-73.

45 Vaganay, J., M. Elkins, D. Esposito, W. O'Halloran, F. Hover, M. Kokko. "Ship Hull Inspection with the HAUV: US Navy and NATO Demonstrations Results." *OCEANS 2006* (2006): 1-6.

46 Johannsson, H., M. Kaess, B. Englot, F. Hover, J. Leonard. "Imaging Sonar-aided Navigation for Autonomous Underwater Harbor Surveillance." *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2010): 4396-4403.

47 Montseny, E., P. Sobrevilla, S. Romani, A. Montferre. "Fuzzy-Based Automatic Approach for Underwater Docks' Anomalies Detection." *Annual meeting of the North American Fuzzy Information Processing Society* (2006): 547-552.

48 Kim, A, R. Eustice. "Pose-graph Visual SLAM with Geometric Model Selection for Autonomous Underwater Ship Hull Inspection." *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2009): 1559-1565.

49 Englot, B., F. Hover. "Inspection Planning for Sensor Coverage of 3D Marine Structures." *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010): 4412-4417.

50 Hollinger, G.A., B. Englot, F. Hover, U. Mitra, G. "Uncertainty-driven View Planning for Underwater Inspection." *2012 IEEE International Conference on Robotics and Automation* (2012): 4884-4891.

51 Viola, P., Jones, M., "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Computer Vision and Pattern Recognition*, 2001, pp. I-511,I-518.

52 MATLAB, Ver R2014b, "Detect Objects Using the Viola-Jones Algorithm," MathWorks Documentation, URL: http://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html [cited 19 July 2015].

53 Girshick, R. B., "Discriminatively Trained Deformable Part Models," 5 September 2012. http://www.cs.berkeley.edu/~rbg/latent/index.html

54 Brar, S.B., and Kaur, A., "Analysis of Rotation Invariant Template Matching Techniques for Trademarks," *International Journal of Engineering Research and Applications (IJERA),* Vol. 2, No. 5, Sept-Oct 2012, pp. 1742-1747

55 Stefanou, S., Argyros, A.A., "Efficient Scale and Rotation Invariant Object Detection Based on HOGs and Evolutionary Optimization Techniques," *Advances in Visual Computing, Springer Berlin Heidelberg,* Vol.7431, 2012, pp. 220-229.

56 Villamizar, M., Moreno-Noguer, F., Andrade-Cetto, J., & Sanfeliu, A., "Efficient Rotation Invariant Object Detection Using Boosted Random Ferns," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, IEEE, 13-18 June 2010, pp. 1038-1045.

57 Csurka, G., C.R. Dance, L. Fan, J. Willamowski, C. Bray. "Visual Categorization with Bags of Keypoints." *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague. 2004.

58 O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, John Wiley & Sons, 2000.

59 Winn, J., A. Criminisi, T. Minka. "Object Categorization by Learned Universal Visual Dictionary." *Tenth IEEE International Conference on Computer Vision* 2 (2005):1800-1807.

60 Frew, E., McGee, T., Kim, Z. W., Xiao, X., Jackson, S., Morimoto, M., Rathinam, S., Padial, J., and Sengupta, R., "Vision-Based Road-Following Using a Small Autonomous Aircraft," *IEEE Aerospace Conference*, IEEE , Vol. 5, 6-13 March 2004, pp.3006-3015.

61 Koutaki, G., Uchimura, K., and Hu, Z., "Road Updating from High Resolution Aerial Imagery using Road Intersection Model," *WG V/6 Processing and Visualization using High-Resolution Images*, Vol. XXVI-5/W1, ISPRS, Pitsanulok, Thailand, 18-20 Nov. 2004.

62 Talal, T. M., Dessouky, M. I., El-Sayed, A., Hebaishy, M., and El-Samie, F. A., "Road Extraction from High Resolution Satellite Images by Morphological Direction Filtering and Length Filtering," *24th International Conference on Computer Theory and Applications,* ICCTA, 11-13 Oct. 2008, pp.137-141.

63 Mena, J. B., “State of the Art on Automatic Road Extraction for GIS Update: A Novel Classification,” *Pattern Recognition Letters,* Vol. 24, No. 16, Dec. 2013, pp. 3037-3058.

64 SAE International, “AMS-STD-595™ *Colors Used in Government Procurement,”* Oct 2015.

65 International Electrotechnical Commission, “IEC 61966-2-1:1999 *Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space – sRGB,*” Oct 1999.

66 Horvath, Michael. Digital graphic, Dec 2013, https://commons.wikimedia.org/wiki/File:HSV_color_solid_cylinder.png

67 Huster, Andreas. *Relative Position Sensing by Fusing Monocular Vision and Inertial Rate Sensors*. Ph.D. Dissertation. Dept. of Electrical Engineering, Stanford University, Stanford, CA, 2003.

68 Van Der Merwe, R., and Wan, E. A., “The Square-Root Unscented Kalman Filter for State and Parameter-Estimation,” *Proc. of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2001. ICASSP, Salt Lake City, UT. Print.

69 Gonzalez, R., and Woods, R., *Digital Image Processing*, Addison-Wesley, 1992, pp. 52-68.

70 MATLAB, Ver R2014b, “Train a Cascade Object Detector,” MathWorks Documentation, URL: http://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html [cited 10 May 2015].

71 DeLullo, A. M., "Computer Vision-Based Tracking Using Triangulation for Coordinated, Autonomous Unmanned Aerial Vehicles," Master’s Thesis, Dept. of Aerospace Engineering, The Pennsylvania State University, University Park, PA, 2006.

VITA

Mark P. DeAngelo

Mark DeAngelo was born and raised in western Pennsylvania. He became interested in transport airplanes during childhood and eventually learned to fly during his teenage years. He earned his private pilot certificate in 2006. Later that year, he attended The Pennsylvania State University where he pursued Aerospace Engineering, completing the Bachelor of Science (B.S.) degree in 2010, the Master of Science (M.S.) degree in 2012 and PhD degree in 2016. Favorite hobbies include photography, hunting, and fishing.