Location-Based Sensor Fusion for UAS Urban Navigation

by

Justin R. Rufa

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2014

Doctoral Committee:

Associate Professor Ella M. Atkins, Chair
Professor Dennis S. Bernstein
Associate Professor Ryan M. Eustice
Assistant Professor James R. Forbes

The only thing necessary for the triumph of evil is for good men to do nothing.

Based on multiple quotes, most notably, Edmund Burke (1770) and John Stuart Mill (1867)

2014

To my amazing wife Natalie, your patience is eternal, to my mother Glory, who taught me to never stop learning, and to my son Drew, who brought warmth to a long cold Michigan winter

ACKNOWLEDGMENTS

I would like to thank the United States Air Force Academy Department of Mathematical Sciences and their department head Colonel John Andrew for the opportunity to take three years of my career to earn a doctorate. It was the privilege of returning to their department to teach our nation's future leaders that served as a constant reminder of why I undertook this endeavor. Their faith in me, their timely advice, and words of reassurance have gotten me through many tough times during this program. I am also grateful to the University of Michigan Department of Aerospace Engineering Graduate Committee for making the commitment to fund my tuition over the three year course of study in a somewhat unique circumstance. I would also like to thank my advisor Associate Professor Ella Atkins for her guidance, support, and wisdom as these constants kept me along the three year graduation path. The passion she shows for eliminating the nonsensical roadblocks to the safe use of small unmanned aircraft is an example to be followed by all in our respective fields of interest. I would finally like to thank my three additional dissertation committee members for their support and time in helping me produce a quality document to hopefully be used by students in the future. This includes Professor Associate Ryan Eustice, who spent at least a few hours teaching me the finer points of stochastic cloning, Professor Dennis Bernstein, and Assistant Professor James Forbes. Last, but not least, a huge thanks to Doctor Justin Bradley for giving my dissertation one final review before submittal.

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ACRONYMS

ADS air data system

A-GNSS assisted-Global navigation Satellite System

ANEES average normalized estimation error squared

ANIS average normalized innovation squared

CID cellular ID

DOP dilution of precision

EnKF Ensemble Kalman Filter

EKF Extended Kalman Filter

FAA Federal Aviation Administration

GNC guidance, navigation, and control

GPS Global Positioning System

HDOP horizontal dilution of precision

IMU inertial measurement unit

INS inertial navigation system

KF Kalman Filter

LTE Long Term Evolution

LQR linear quadratic controller

NAS National Airspace System

NEES normalized estimation error squared

NIS normalized innovation squared

OTDOA observed time difference of arrival

RISS reduced inertial sensor system

RMSE root-mean-square error

SAM sensor accuracy mode

TDOA time difference of arrival

UAS unmanned aircraft system

UERE user equivalent range error

VDOP vertical dilution of precision

ABSTRACT

Location-Based Sensor Fusion for UAS Urban Navigation

by

Justin R. Rufa

Chair: Associate Professor Ella M. Atkins

For unmanned aircraft systems (UAS) to effectively conduct missions in urban environments, a multi-sensor navigation scheme must be developed that can operate in areas with degraded Global Positioning System (GPS) signals. This thesis proposes a sensor fusion plug and play capability for UAS navigation in urban environments to test combinations of sensors. Measurements are fused using both the Extended Kalman Filter (EKF) and Ensemble Kalman Filter (EnKF), a type of Particle Filter. A Long Term Evolution (LTE) transceiver and computer vision sensor each augment the traditional GPS receiver, inertial sensors, and air data system. Availability and accuracy information for each sensor is extracted from the literature. LTE positioning is motivated by a perpetually expanding network that can provide persistent measurements in the urban environment. A location-based logic model is proposed to predict sensor availability and accuracy for a given type of urban environment based on a map database as well as real-time sensor inputs and filter outputs. The simulation is executed in MATLAB where the vehicle dynamics, environment, sensors, and filters are user-customizable. Results indicate that UAS horizontal position accuracy

is most dependent on availability of high sampling rate position measurements along with GPS measurement availability. Since the simulation is able to accept LTE sensor specifications, it will be able to show how the UAS position accuracy can be improved in the future with this persistent measurement, even though the accuracy is not improved using current LTE state-of-the-art. In the unmatched true propagation and filter dynamics model scenario, filter tuning proves to be difficult as GPS availability varies from urban canyon to urban canyon. The main contribution of this thesis is the generation of accuracy data for different sensor suites in both a homogeneous urban environment (solid walls) using matched dynamics models and a heterogeneous urban environment layout using unmatched models that necessitate filter tuning. Future work should explore the use of downward facing VISION sensors and LiDAR, integrate real-time map information into sensor availability and measurement weighting decisions, including the use of LTE for approximate localization, and more finely represent expected measurement accuracies in the GPS and LTE networks.

CHAPTER 1

Introduction

1.1 Expanding Urban-Focused UAS Missions

As unmanned aircraft system (UAS) technology continues to mature, numerous visions for UAS applications in rural and urban areas have emerged. The Federal Aviation Administration (FAA) will oversee the integration of UAS into the National Airspace System (NAS). This process, as outlined by the FAA Modernization and Reform Act of 2012, has begun with the identification of six test sites, and will provide support for integration of UAS into the NAS with a target date of no later than September 30, 2015 [1]. With UAS flying in the NAS, many non-federal government entities will have the opportunity to operate UAS for missions including urban law enforcement, anti-terrorism, riot control, traffic surveillance, natural disaster monitoring, emergency medicine, agriculture, and communications relay among others [2]. Many of these missions require operations in urban environments, characterized by dense canyons formed between high-rise buildings as well as monuments, pedestrian bridges, parks, overpasses/bridges, billboards, antennas, spires, construction equipment, and possibly other aircraft. In addition to these potential obstructions, the large volumes of vehicle and pedestrian traffic on the ground must not be placed at higher risk due to the introduction of UAS. Despite the difficulties in safely operating UAS in urban environments, law enforcement organizations in the United States, Canada, and Europe are giving consideration to these missions and the issues they will face.

As early as 2007, the United Kingdom investigated requirements necessary for UAS to augment manned aircraft conducting urban law enforcement missions. One of the challenges specifically cited was sensor obscuration, which further shows the need for navigation and control systems independent of any one data source [3]. In Canada, the Ontario Provincial Police received a Special Flight Operations Certificate to fly UAS in a law enforcement role across the province, including urban areas [4]. The first use of a UAS under this certificate was by the Kenora, Ontario Forensic Identification Services in mid- 2007 for forensic evidence gathering during a homicide investigation [5]. In the United States, the Seattle, Washington Police Department and the Arlington, Texas Police Department both received FAA permission to operate UAS under certain conditions for missions such as traffic accident surveillance, possible hostage situations in and around buildings [6], crime scene photography, and missing person searches [7].

The importance of this class of missions, especially in time-sensitive law enforcement and emergency situations, require navigation systems be sufficiently robust to handle sensor anomalies common in urban canyons, including degradation/denial/spoofing of Global Positioning System (GPS) signals. This is necessary to ensure the safety of people on the ground (pedestrians and vehicle traffic), and to avoid damage to public infrastructure as well as private property.

1.2 GPS Degradation, Loss, or Denial in Urban Canyons

UAS normally depend on reliable equipment and reliable GPS signals for accurate and precise navigation. However, GPS signals can become degraded due to natural phenomena such as free-space loss and refraction/absorption in the atmosphere. Signals can also degrade or become unavailable due to man-made reflection and masking from urban structures and foliage [8]. In urban areas where GPS signals are available, multipath and masking also contribute to an increased geometric dilution of precision of the measurements [9].

Lu *et al*. [10] conducted a GPS accuracy trial in the Wanchai area of Hong Kong, seen in Figure 1.1, known to have one of the densest high-rise building cores on the island. It showed that 50% of the area studied did not have adequate GPS reception. When a GPS solution was available, the accuracy was worse than 20 meters for 40% of the points and worse than 100 meters for 9% of the points.

Figure 1.1: Wanchai District of Hong Kong from Victoria Harbour courtesy of WiNG [11].

There are also intentional, yet inadvertent reasons for GPS signal unavailability, including the use of GPS signal jammers, which was the case in the Newark Airport GPS outage event from November 2009 through April 2010 [12]. When the GPS signal is unavailable, either due to the environment or a jamming device, other instruments are required for navigation.

In 2007, the Defense Advanced Research Projects Agency (DARPA), concerned by the possibility of losing GPS navigation capabilities, funded a Robust Surface Navigation program that studied how navigation could continue without using GPS signals. It examined signals of opportunity, including cellular network signals, television signals, and even signals emanating from other satellites, to determine if any would be viable for navigation purposes [13]. In a similar effort, BAE Systems developed "NAVSOP" (navigation via signals of opportunity) that uses Wi-Fi, television, and cellular network signals among others to provide navigation capabilities independent of GPS [14]. With two large research

and development organizations, from government and private industry coming to the same conclusion, a multiple sensor approach that includes pre-existing signals is critical to investigate and mature.

An initial step towards multi-sensor navigation is to identify possible data sources. One such source with rapidly expanding infrastructure is the cellular phone network. Smart phones on the Long Term Evolution (LTE) network provide a means for producing sporadic position estimates using one of many cellular-network based geolocation techniques. The perpetually increasing accuracy of these techniques can be attributed to the Federal Communications Commission's Enhanced-911 device location requirements that enforce network-calculated accuracy within 300 meters for 90% of the requested position fixes, with more stringent requirements on device-calculated accuracy. Table 1.1 summarizes the E-911 accuracy requirements, with mandatory compliance by September 11, 2012 [15].

Table 1.1: FCC E-911 accuracy requirements [15].

	67%	90%
Network-Based	100 meters	300 meters
Handset-Based	50 meters	150 meters

As LTE network position estimate accuracy increases, it can be used for navigation, roadside assistance, landmark locators, billing aid for toll roads, fleet management, and even location-sensitive sales alerts [16]. While the state-of-the-art accuracy is largely guarded by the companies developing this technology, Polaris Wireless (http://www.polariswireless.com), set a goal to eventually reach an accuracy of 10 meters with a 1 second time-to-fix using its Polaris Wireless Locating Signatures technology.

Given that GPS is unreliable in the urban environment, can a UAS with an LTE-network capable transceiver, GPS receiver, combined with a compact computer vision system, inertial measurement unit (IMU), and an air data system (ADS) combination allow a UAS to safely and reliably navigate autonomously while flying critical urban missions around

the world? A review of existing urban canyon navigation techniques will show the types of navigation solutions that already exist for vehicles in the urban environment, and our simulations will investigate sensor suite accuracy and availability.

1.3 Existing Urban Canyon Navigation Techniques

Since the early 1990's, researchers have studied urban canyon vehicle state estimation in the presence of degraded and sometimes unavailable GPS sensor data. The least complex state estimation technique uses only a single sensor, while more advanced techniques integrate data from sources such as onboard environment-sensing, inertial sensors, urban map databases, and ground-based navigation transmitters. Multi-source techniques generally use Bayesian filtering techniques such as the Kalman Filter, Extended Kalman Filter, or Particle Filter to integrate measurements from the different sources. Each of these different urban canyon navigation techniques for both ground vehicles and UAS are summarized below.

1.3.1 GPS-Only Navigation

In terms of onboard systems integration complexity, the most basic solution is to use only a single data source. This is normally a high sensitivity GPS receiver with high signal availability. References [17], [18], and [19] used a high sensitivity GPS receiver as a baseline solution in their research, while [20] used standard receivers.

MacGougan *et al*. [17] used four receivers during their driving trials in the two Canadian cities, with two standard receivers and two high-sensitivity receivers. In Vancouver, British Columbia (as seen in Figure 1.2), the two-dimensional root-mean-square position error ranged from 10.8 meters to 23.1 meters once outliers were removed with the root-mean-square horizontal dilution of precision (HDOP) ranging from 4.0-7.0. The root-mean-square error (RMSE) for height was less accurate ranging from 11.9 meters to 62.4

meters. When the test was repeated in Calgary, Alberta, despite better than expected signal availability, the root-mean-square HDOP for the four receivers ranged from 2.8-12.5. [18] the authors also conducted a field test in Calgary, reporting trajectory errors in the range of several hundred meters to almost a kilometer using an epoch-by-epoch least-square approach.

Figure 1.2: Vancouver urban street-level image courtesy of MacGougan *et al.* [17].

Hide *et al.* [19], in their driving trial in Nottingham, United Kingdom, reported high signal availability, but horizontal position RMSE of roughly 5 meters, with vertical RMSE of almost 39 meters, and heading RMSE of almost 29 meters at speeds below 5 meters/second. This study also reported horizontal error position spikes of 30 meters and 70 meters, despite at least seven satellites being available, demonstrating the multipath issues inherent to GPS signals in urban canyons.

Modsching *et al.* [20] conducted a walking trial of Görlitz, focusing on GPS accuracy in an environment with less urban build-up, shown in Figure 1.3(a) and an environment with more urban build-up, shown in Figure 1.3(b). For an environment with less urban build-up environment, the mean two-dimensional error was 2.42 meters while with significant urban buildup, the mean two-dimensional error was 15.43 meters.

(a) Less urban build-up

(b) More urban build-up

Figure 1.3: Görlitz Urban Build-up Images courtesy of Modsching *et al.* [20].

1.3.2 Navigation by GPS with Inertial Sensors

A popular method of augmenting GPS data is through the use of inertial sensors. Inertial sensors generate acceleration and angular rate measurements with respect to an inertial reference frame without using an outside reference. Examples include gyroscopes and accelerometers. Mezentsev *et al.* [18] tested a GPS/gyroscope sensor combination, implementing zero velocity updates to minimize gyro drift. Using a three-state Kalman Filter, the vehicle's position was estimated during a driving trial in Calgary with less than 20 meters average error, but spiking to $60-75$ meters. GPS availability of only 30% demonstrated the blocking effect of the urban environment on GPS signals.

References [19] and [21] described results when GPS measurements are combined with with a gyroscope/accelerometer unit, traditionally known as an inertial navigation system (INS). Hide *et al.* [19] implemented a Kalman Filter to calculate position error estimates

during a driving trial in Nottingham, England. The horizontal position RMSE was roughly 2.5 meters, with vertical RMSE under 10 meters. The maximum horizontal RMSE was only 11 meters, even with only 48.4% GPS solution availability. Davidson *et al.* [21] used an Extended Kalman Filter to produce state estimates during driving tests in both Tampere (Finland) and Portland (Oregon). The position error was less than 20 meters, even when the GPS error was up to 35 meters in some locations.

1.3.3 Navigation by GPS with other Sensors

Mar and Leu [22] proposed the use of a compass sensor and odometer for dead reckoning to augment differential GPS (DGPS). The two position sources were integrated using both an eight-state and ten-state Kalman Filter. Simulation results showed that when differential GPS was available, the filter had 3.083/2.075 meter North/East mean error with 7.510/4.837 meter standard deviation for the eight-state/ten-state Kalman Filter, respectively. However, when differential GPS was not available, the dead reckoning error jumped to 7.506/7.83 meter mean error with 57.52/22.62 meter standard deviation. Even with reasonable error means, high standard deviations indicate low confidence in the estimates.

References [23] and [24] describe errors resulting from GPS measurements augmented with a gyroscope/odometer combination to provide measurements via dead reckoning. Viscek *et al.* [23] used a filter with stationary and moving modes to estimate the position of a vehicle during a driving test in San Francisco, California. Their results showed that dead reckoning only accumulates error on the order of 2% of the distance traveled when using GPS. Georgy *et al.* [24] quantified the accuracy of the gyroscope/odometer combination using both a Kalman Filter and a mixture Particle Filter separately during a driving test in southeastern Ontario. In each instance of GPS outage, the Particle Filter outperformed the Kalman Filter substantially, with 6.89 meter Particle Filter error and 27.86 meter Kalman Filter error in one case as well as 8.9 meter Particle Filter error and 41.45 meter Kalman Filter error in another case. These results show how using 100 particles with the nonlinear

dynamics of a system can more accurately estimate the state than when using the linearized model necessary for the Kalman filter.

1.3.4 Navigation by GPS with Inertial Sensors and Beacons

A further enhancement to the GPS/inertial sensor navigation solution is the use of ground-based navigation beacons. Wei *et al*. [25] used GPS receiver-compatible pseudolites as their ground-based navigation beacons to augment GPS when only two satellites were available. This GPS/pseudolite position data as well as the INS range estimates were double differenced to eliminate clock bias error and then used in the measurement model of an Extended Kalman Filter. Although this technique extended the availability of GPS to urban canyons [26], the error reached almost 20 meters in both horizontal directions.

Lu *et al*. [27] used an INS with Bluetooth navigation beacons to position vehicles when GPS was degraded or unavailable. A Kalman Filter was used to integrate the INS measurements and GPS when it was available and accurate. In cases where GPS was not available, Bluetooth beacons would send their location to the vehicle as a correction to eliminate INS-induced drift. Driving trials in a densely urban part of Hong Kong showed the effectiveness of the beacons with position error dropping from over 50 meters using only GPS/INS to just over 7 meters when the beacons were also activated.

1.3.5 Navigation by GPS with Map Matching

Urban street maps can be used to augment GPS-based navigation systems in urban canyon navigation. Cui and Ge [28] proposed a technique to model a vehicle's path as concatenated curves to represent the roads on a map, lowering the required number of GPS satellites to two. These curves consisted of straight lines, arcs, polynomials, and other piecewise continuous curves, making the map a series of curves connected by nodes or junctions. Their work used a state augmentation method and an Extended Kalman Filter to provide estimates of both the actual path and positioning information for the vehicle. Results showed

mean distance error of 0.436 meters when using path modeling, compared to a 1.205 meter error without path modeling. When only one satellite was available the mean error jumped to only 2.73 meters, demonstrating robustness of the technique to satellite loss.

Drevelle and Bonnifait [29] created the Interval Global Positioning with road Surface by tightly-coupling three-dimensional maps with GPS pseudorange measurements to position the vehicle. The technique was presented as a constraint satisfaction problem that enforced the GPS measurements and drivable space on the map as geometric constraints. The position confidence domain was then calculated using interval analysis and contractors. Results from a driving trial in Paris showed less than 6.5 meter error 95% of the time with the radius of the confidence domain less than 16 meters 95% of the time, both with two or more satellites in view. The trial also demonstrated the lack of satellite availability in urban environments with three or fewer satellites available 77% of the time and two or fewer satellites available 40% of the time. GPS error reached 10 meters in the horizontal plane and 35 meters in the vertical plane during these periods.

1.3.6 Navigation by GPS with Map Matching and Inertial Sensors

In addition to GPS and urban street maps, the more traditional inertial sensors can also be integrated into a vehicle's navigation system to provide more accurate state estimates. Syed's [30] navigation technique used a gyroscope and digital map as well as a high-sensitivity GPS receiver to locate a vehicle in an urban setting. The foundation of this algorithm was the identification of the current vehicle's two-dimensional road link and position on the link. Once this information was determined for one epoch, it was used to discard bad GPS pseudoranges for the next epoch. A driving test in Calgary showed this solution provided correct fixes up to 89% and 93% of the time with no false position fixes for the two different GPS receivers. These high sensitivity receivers also provided 96% and almost 99% real-time solution availability.

References [31] and [32] describe tests where GPS is augmented with odometer and

map data to estimate the position of a vehicle in dense urban settings using different Particle Filter variants. Boucher and Noyer's [31] solution implemented a hybrid filter that took advantage of separating the states within the system so that the linear states could be estimated using a Kalman Filter within the Particle Filter, known as a Rao-Blackwellized Particle Filter. Map data was composed of two-dimensional piecewise arcs connected by nodes. Position and heading along these nodes was calculated through map measurement equations to give an independent set of measurements. Results from a driving trial in Calais (France) showed a 16 meter error when using dead reckoning navigation during the GPS outage while the proposed solution with the addition of map measurements reduced the error to 2.6 meters.

Georgy *et al*. [32] used a mixture Particle Filter based on the sampling importance resampling. A gyroscope and two accelerometers were included along with the odometer to form a three-dimensional reduced inertial sensor system (RISS). The technique also used map measurements tightly integrating the three measurement sources. Map measurements were computed from a four-step process that matched a predicted two-dimensional solution to $1,000$ foot candidate segments until the segment with the minimum distance to the predicted solution was found. During two driving trials in downtown Toronto, Ontario, the mixture Particle Filter was used for three-dimensional RISS/GPS integration and the three-dimensional RISS/GPS/Map integration. The three-dimensional RISS/GPS mixture Particle Filter had an RMSE of 10.73 meters for the first trial and 9.19 meters for the second trials with a 0.0014 seconds per iteration execution time. The three-dimensional RISS/GPS/Map mixture Particle Filter had a 6.72 meter and 6.22 meter RMSE for the two trials, with a 0.0180 seconds per iteration execution time. .

1.3.7 Navigation by Environment-Based Sensors: Computer Vision and Laser

Computer vision and laser navigation techniques differ from GPS and inertial sensing techniques in that they provide information about the position, speed, and orientation of the vehicle with respect to its local environment rather than with respect to a global inertial reference frame. Muratet *et al.* [33] combined an IMU with a video camera as the UAS's primary sensors. Optic flow, the apparent motion of pixels from frame to frame in an image, was calculated using real-time quantized region matching. Their results showed the tendency of a vehicle to slow down in the presence of obstacles, center itself in a canyon, and complete U-turns when necessary to avoid obstacles in front of the vehicle while avoiding collisions. These tendencies were driven by the relative value of a time-to-contact parameter compared to a preset minimum acceptable value. Wu *et al.* [34] also used vision to augment inertial sensors in estimating the UAS's position and attitude. In their solution, vision-based target position measurements, from a pinhole camera model, were fed into an Extended Kalman Filter providing state estimates to correct the inertial position and attitude estimates. In simulation, the maximum position errors were 2.804 meters, 1.432 meters, 0.58 meters in the x,y,z directions respectively. In an experimental test, these errors rose slightly to 2.926 meters, 2.53 meters, 1.22 meters in the x, y, z directions, verifying the feasibility of a vision/inertial navigation system for urban environments. Graham *et al.* [35] integrated visual odometry with GPS for accurate urban environment navigation. Their algorithm detected unmodeled state propagation and measurement errors due to visual odometry drift/biases and used filtering techniques to improve the estimate while preventing GPS multipath errors from causing the estimate to further deteriorate. They were able to reduce two-dimensional position error means from 15.10 meters using an Extended Kalman Filter to 12 meters using an l_1 filter, while lowering the 1σ deviation from 10.16 meters to 6.87 meters.

Hrabar and Sukhatme's [36] vision based solution combined side-facing optic flow

cameras with a front-facing stereo vision camera on a UAS and ground vehicle. Optic flow data was calculated using the Lucas and Kanade feature tracker. This work demonstrated the ability of a vehicle to detect obstacles using stereo vision and navigate around them using vision based turn rate commands. It also demonstrated how optic flow can be used to center a vehicle within a canyon. When combining data from both cameras the test vehicle was able to successfully navigate through T-junctions and 90 degree bends in an urban canyon.

Shim *et al*. [37] showed that range and angle data from a laser scanner could be used to build an obstacle map as well as a model predictive control (MPC) obstacle avoidance algorithm. Using these tools, their vehicle was able to safely navigate a simulated urban canyon.

Soloviev [38] used an integrated GPS, INS, and laser scanner system for navigation in urban canyons. This work used the laser as a complementary sensor to GPS such that when GPS was unavailable due to obstructions, the laser was able to use local environment features to navigate. Local range and angle measurements taken at consecutive time steps determine change in position and orientation of the vehicle. The INS was used for angular motion computation and to fill in the gaps when neither GPS and nor laser were able to produce measurements. The INS data was also used in algorithms for feature matching, laser tilt compensation, and coasting. The standard deviation for user position residuals errors over two driving trials in Athens, Ohio was roughly 0.03 meters in the East-West direction and 0.02 meters in the North-South direction.

Tomić *et al*. [39] proposed a UAS navigation solution using laser odometry or correlation-based vision odometry to augment inertial sensor data in an Extended Kalman Filter. This allowed full six degree of freedom state estimation with one filter. The selection between the two different sensors was made based on which had the smaller covariance at a particular time step, and drift error was corrected using frames from visual odometry or landmark recognition. Experimental results from both indoor and outdoor environments

showed less than 0.2 meter error in each of the three directions over a minute-long flight test. A longer flight test would be needed to determine if the error would stay steady at the reported value, increase, or decrease.

Overall trends show that GPS augmented with other sensors and data sources perform significantly better than GPS-only. Additionally, the use of a Particle Filter to produce a state estimate from multiple measurement sources shows better accuracy than the Kalman Filter/Extended Kalman Filter implementation for the same set of sensors at the cost of higher computational load per iteration. However, using the Particle Filter to its full potential requires the use of a particle weighting algorithm, determining a reasonable number of particles, and understanding the distribution of measurement noise.

1.4 Problem Statement

The goal of this research is to investigate the accuracy of combinations of navigation sensors and filtering techniques for different urban environments in which UAS might operate. To achieve this goal, a user-configurable urban UAS navigation simulation was developed using filtered sensor state accuracy data from traditional and nontraditional aircraft sensors including an IMU, air data system, LTE network, GPS receiver, and computer vision sensors. In each urban environment scenario, the fixed-wing UAS uses available sensors to estimate its 12 states including the inertial position coordinates, airspeed, wind frame angles, attitude, and angular rates. This six degree of freedom model assumes no wind so that the inertial velocities can be represented by airspeed, angle of attack, and angle of sideslip using basic transformations. Considerations must be taken to account for inaccuracies in low-cost sensors, delayed measurement reporting, and availability issues. Ultimately, a fielded UAS must appropriately fuse measurements from each of these sensors on-board and in real-time to safely guide the UAS through a narrow urban environment, i.e., flight below building roofs on both sides of a street, without persistent dependence on GPS.

1.5 Research Objectives

The four primary research objectives of this dissertation are listed below:

- The first objective is to design a MATLAB®-based UAS guidance, navigation, and control simulation, including a basic guidance scheme, controller, sensor models, and a state estimation filtering structure using an existing UAS dynamics model. The software must be customizable to facilitate simulations over any suite of sensors and filter parameters.

- The second objective is to quantify the relative location of the UAS within an urban environment at all times as it correlates to sensor availability, sensor measurement error, and state estimation filter properties.

- The third objective is to use the simulation to study the effects of measurement error for the various sensors on the accuracy of the state estimate within a homogeneous urban environment using matching dynamics models and process noise model.

- The fourth objective is to use the simulation to study the effects of incorporating sensor availability as a function of the environment and adding model uncertainty. Accuracy of the state estimate is characterized during simulated flights through a heterogeneous urban environment.

1.5.1 Approach

To achieve the first research objective, the MATLAB® environment is used to develop a simulation using a series of functions and subfunctions for each process within the UAS guidance, navigation, and control loop. Customized data structures are used to ensure the smooth flow of data between functions. An important feature of the simulation is the ability to easily customize the vehicle, sensor, environment, and filter models to meet user needs. This includes the six degree of freedom UAS rigid-body dynamics model, literature-based

sensor models, and a simulated urban environment model including buildings and obstacles such as skybridges and antennas.

To achieve the second research objective, vertical and horizontal relative location partitions are defined along with the algorithms to categorize the UAS location within each. These algorithms use UAS position with respect to the buildings within the environment to determine sensor availability and measurement noise covariances. If available these measurement noise covariance values are then used to generate realistic measurements for the simulation.

To achieve the third research objective, a straight-and-level trajectory is flown at multiple altitudes both above and within a homogeneous urban canyon using multiple sensor combinations. This set of simulations determines baseline navigation accuracy for different sensor combinations as well as determining what effect, if any, the novel fusion of LTE measurements with traditional navigation data has on the state estimate.

To achieve the fourth research objective, both straight-and-level as well as sinusoidal trajectories are flown above and in a heterogeneous urban environment, including intersections. Sensor availability is varied and an unmatched model is used where the UAS plant dynamics propagation model coefficient and constant values do not match their counterparts in the filter dynamics model. This set of simulations demonstrates how performance of a more realistic UAS model is affected by increased process noise, dynamically changing sensor availability, and measurement-only sensors that must estimate their own noise values.

1.6 Contributions

1. A modular end-to-end UAS simulation is developed within the MATLAB® environment to include UAS dynamics, urban environments, and sensor models to account for the environment and delays using object-oriented data structures to emphasize

reconfigurability.

2. A compilation of measurement availability and accuracy statistical information for a suite of current and proposed UAS sensors is presented based on a thorough literature review.

3. Bayesian filter UAS urban navigation consistency and accuracy statistics are developed as a function of urban environment characteristics.

4. Quantification of UAS urban navigation performance using LTE network observed time difference of arrival (OTDOA) position measurements to augment sensors in areas of poor GPS availability is given as are results using traditional sensor options including GPS, vision, and IMU.

1.7 Innovations

1. Sensor availability, sampling rate, measurement delay, and filtered state noise covariance values are used for the first time in a rigorous location-dependent simulation study synthesizing results from current literature and existing commercial off the shelf sensor options.

2. An intuitive discretization of the urban environment is used for the first time in the generation of GPS and LTE location-dependent availability and noise values that are then incorporated into the state estimation filter.

3. The LTE network OTDOA positioning capability is exploited to provide independent and persistent time-delayed position measurements which are incorporated into the state estimation process.

4. UAS navigation performance is compared for matching UAS plant dynamics and

filter dynamics models as well as for unmatched UAS plant dynamics and filter dynamics models.

1.8 Dissertation Overview

The remainder of the dissertation is divided into six chapters. Chapter 2 discusses the technical background including linear quadratic controller (LQR) regulator design, nonlinear fixed-wing aircraft equations of motion, including their linearization and equilibrium (trim) states. Chapter 2 also presents background in possible UAS sensors and state estimation filtering techniques, including the handling of delayed measurements. Chapter 3 describes the development of the UAS urban canyon simulation in terms of the guidance, navigation, and control loop. The main topics include the simulation data classes, the specific linearized UAS model, and the development of basic guidance algorithms along with UAS controller design. Chapter 4 introduces the relative location categorization process which splits the urban environment into vertical and horizontal planes to aid in determining accuracy for GPS and LTE. Additionally, sensor availability and accuracy models are discussed for GPS and vision sensors. Chapter 5 presents accuracy results for homogeneous urban canyon simulations using matched models for the UAS plant dynamics propagation and state estimation filter dynamics. Chapter 6 studies UAS navigation in a realistic urban canyon with varying building heights and intersections using both a matched model and an unmatched model for the UAS plant dynamics propagation model versus the filter dynamics model. Chapter 7 concludes the dissertation with a summary of key results and discussion of future work needed to realize robust urban canyon navigation.

CHAPTER 2

Background

This chapter summarizes modeling and control techniques necessary for UAS guidance, navigation, and control. Next, detailed models of potential UAS sensors, including their accuracy and availability, are introduced. The chapter concludes with a review of two non-linear Bayesian state estimation filters, including the Extended Kalman Filter and the Ensemble Kalman Filter as well as a description of metrics to quantify their consistency and accuracy. In this dissertation, scalar-valued variables are represented in a lowercase un-bolded form such as p, vector valued variables are represented in lowercase bolded form such as $\mathbf{x}$, and matrix valued variables are represented in uppercase un-bolded form such as P.

2.1 UAS Control using a Linear Quadratic Regulator (LQR)

An LQR controller [40] is commonly used for UAS due to its straightforward implementation and constant gains when the UAS dynamics are linearized about a trim (steady flight) condition. The control law is

$$\mathbf{u}_{k-1} = \bar{K}(\mathbf{x}_{cmd_{k-1}} - \mathbf{x}_{k-1}) + \mathbf{u}_{trim} \tag{2.1}$$

where $\mathbf{u}$ is the control vector, $\mathbf{x}_{cmd}$ is the commanded state vector, $\mathbf{x}$ is the state vector, and $\mathbf{u}_{trim}$ is the trim control vector.

The gain matrix $\bar{K}$ is calculated using

$$\bar{K} = (\bar{R} + B^T \bar{P} B)^{-1} B^T \bar{P} A. \tag{2.2}$$

A and B are the state transition and control matrices calculated during the linearization process, and the $\bar{Q}$ and $\bar{R}$ matrices are user-defined values selected to minimize

$$J = \sum_{0}^{\infty} (\mathbf{x}^T \bar{Q} \mathbf{x} + \mathbf{u}^T \bar{\rho} \bar{R} \mathbf{u}). \tag{2.3}$$

$\bar{Q}$ and $\bar{R}$ can be set to the identity matrix with dimensions equal to those of $\mathbf{x}$ and $\mathbf{u}$, respectively, and can be tuned by adjusting the scalar value of $\bar{\rho}$. Bryson's Rule [41], another well-known technique, initially sets the diagonal elements of $\bar{Q}$ and $\bar{R}$ to the inverse of the square of the maximum allowable deviation of each state and control from trim, respectively.

$\bar{P}$ is calculated using

$$\bar{P} = \bar{Q} + A^T (\bar{P} - \bar{P} B (\bar{R} + B^T \bar{P} B)^{-1} B^T \bar{P}) A \tag{2.4}$$

the discrete algebraic Ricatti equation [40].

2.2 Fixed-Wing Rigid-Body Aircraft Equations of Motion

The fixed-wing rigid-body aircraft equations of motion used in this research are in the form $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, where $\mathbf{x}$ the state vector is

$$\mathbf{x} = [x_N \; x_E \; h \; V_T \; \alpha \; \beta \; \phi \; \theta \; \psi \; p \; q \; r]^T \tag{2.5}$$

consisting of the inertial position (North position, x_N, East position, x_E, and altitude, h), wind frame parameters (assuming zero wind) (airspeed, V_T, angle of attack, α, angle of sideslip, β), Euler angles (attitude) (roll angle, ϕ, pitch angle, θ, and yaw angle, ψ), and body-fixed angular velocities (roll rate, p, pitch rate, q, and yaw rate, r).

The control vector $\mathbf{u}$ is

$$\mathbf{u} = [\delta_a \; \delta_e \; \delta_r \; F_T]^T \tag{2.6}$$

consisting of the aileron deflection, δ_a, elevator deflection, δ_e, rudder deflection, δ_r, and thrust, F_T.

The full set of position dynamics equations with a zero wind assumption are given by

$$\begin{gathered}
\begin{bmatrix} \dot{x}_N \\ \dot{x}_E \\ \dot{h} \end{bmatrix} = V_T \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ \sin\theta & -\sin\phi\cos\theta & -\cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix} \\
\dot{V}_T = \tfrac{1}{m}(X^w + F_T\cos\alpha\cos\beta + mg_x^w) \\
\dot{\alpha} = \tfrac{1}{\cos\beta}(\tfrac{1}{mV_T}(Z^w - F_T\sin\alpha + mg_z^w) + q^w) \\
\dot{\beta} = \tfrac{1}{mV_T}(Y^w - F_T\cos\alpha\sin\beta + mg_y^w) - r^w \\
\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan\theta\sin\phi & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\
\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{b^{-1}} \left(\begin{bmatrix} L^b \\ M^b \\ N^b \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I^b \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right).
\end{gathered} \tag{2.7}$$

They are available with full derivation in Chapter 3 of [42] with aircraft model properties available in the appendix of [42]. Analogous equations are available in Chapter 5 of [43].

These differential equations contain aerodynamic forces acting on the UAS in the wind frame, $\{X^w, Y^w, Z^w\}$, gravity force in the wind frame, $\{g_x^w, g_y^w, g_z^w\}$, and two roll rates in the wind frame, q^w, r^w as well as the constant physical inertia matrix of the UAS, I^b, and the aerodynamic torques on the UAS, $\{L^b, M^b, N^b\}$.

Since the engine cannot instantaneously switch from its current throttle setting to a

different commanded throttle setting, the equation of motion governing the rate at which the engine speed can change is

$$\dot{\bar{n}} = -\frac{1}{\tau_{\bar{n}}}\bar{n} + \frac{1}{\tau_{\bar{n}}}\bar{n}_{cmd} \tag{2.8}$$

with $\bar{n}$ representing the engine revolutions per second, $\bar{n}_{cmd}$ representing the commanded engine revolutions per second, and $\tau_{\bar{n}}$ as the engine constant. The total thrust force is calculated as

$$F_T = \rho \bar{n}^2 D^4 C_{F_T}(J) \tag{2.9}$$

where D is the propeller diameter and $C_{F_T}(J)$ is the coefficient of thrust as a function of advance ratio J [42].

2.2.1 Rewriting UAS Equations of Motion in Terms of System States

The aerodynamic force wind frame components, gravity force wind frame components, aerodynamic torque components, roll rate wind frame components are defined in terms of the states of the system/control inputs in (2.10)-(2.20).

The aerodynamic force components are defined as

$$X^w = \frac{\rho V_T^2}{2} S (C_{X1} + C_{X\alpha}\alpha + C_{X\alpha 2}\alpha^2 + C_{X\beta 2}\beta^2) \tag{2.10}$$

$$Y^w = \frac{\rho V_T^2}{2} S C_{Y1}\beta \tag{2.11}$$

$$Z^w = \frac{\rho V_T^2}{2} S (C_{Z1} + C_{Z\alpha}\alpha). \tag{2.12}$$

The gravity force components are defined as

$$g_x^w = g(-\sin\theta\cos\alpha\cos\beta + \cos\theta\sin\phi\sin\beta + \cos\theta\cos\phi\sin\alpha\cos\beta) \tag{2.13}$$

$$g_y^w = g(\cos\alpha\,\sin\beta\,\sin\theta + \cos\beta\,\cos\theta\,\sin\phi - \sin\alpha\,\sin\beta\,\cos\theta\,\cos\phi) \tag{2.14}$$

$$g_z^w = g(\sin\alpha\,\sin\theta + \cos\alpha\,\cos\theta\,\cos\phi). \tag{2.15}$$

The aerodynamic torque components are defined as

$$L^b = \frac{\rho V_T^2}{2} S\,b\left(C_{La}\delta_a + C_{L\beta}\beta + C_{L\tilde{p}}\frac{b\,p}{2V_T} + C_{L\tilde{r}}\frac{b\,r}{2V_T}\right) \tag{2.16}$$

$$M^b = \frac{\rho V_T^2}{2} S\,\bar{c}\left(C_{M1} + C_{Me}\delta_e + C_{M\alpha}\alpha + C_{M\tilde{q}}\frac{\bar{c}\,q}{2V_T}\right) \tag{2.17}$$

$$N^b = \frac{\rho V_T^2}{2} S\,b(C_{N\delta_r}\delta_r + C_{N\tilde{r}}\frac{b\,r}{2V_t} + C_{N\beta}\beta). \tag{2.18}$$

The roll rate wind frame components are defined as

$$q^w = q\,\cos\beta - p\,\sin\beta\,\cos\alpha - r\,\sin\alpha\,\sin\beta \tag{2.19}$$

$$r^w = -p\,\sin\alpha - r\,\cos\alpha. \tag{2.20}$$

2.3 Aircraft Equation of Motion Linearization and Trim State Calculation

For a UAS to achieve trimmed flight, it must maintain control inputs at or near values that maintain this trim or equilibrium state. To accomplish this, the equations of motions must be linearized about a trim state vector $\mathbf{x}_{trim}$ and control vector $\mathbf{u}_{trim}$ using a first-order Taylor Series Expansion.

2.3.1 Linearization about a Trim Condition

The desired form of the linearized equations of motion is

$$\dot{\bar{\mathbf{x}}} = A\bar{\mathbf{x}} + B\bar{\mathbf{u}} \tag{2.21}$$

where $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}_{trim}$, and $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}_{trim}$ are the deviations from the trim conditions, trim state vector, and trim control vector, respectively. $A = \frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}_{trim},\mathbf{u}_{trim}}$ is the state transition Jacobian and $B = \frac{\partial f}{\partial \mathbf{u}}|_{\mathbf{x}_{trim},\mathbf{u}_{trim}}$ is the control Jacobian, both linearized about the trim state vector and the trim control vector [43].

The symbolic Jacobian for the state transition matrix is

$$A = \begin{pmatrix}
0 & 0 & 0 & \frac{\partial \dot{x}_N}{\partial V_T} & \frac{\partial \dot{x}_N}{\partial \alpha} & \frac{\partial \dot{x}_N}{\partial \beta} & \frac{\partial \dot{x}_N}{\partial \phi} & \frac{\partial \dot{x}_N}{\partial \theta} & \frac{\partial \dot{x}_N}{\partial \psi} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial \dot{x}_E}{\partial V_T} & \frac{\partial \dot{x}_E}{\partial \alpha} & \frac{\partial \dot{x}_E}{\partial \beta} & \frac{\partial \dot{x}_E}{\partial \phi} & \frac{\partial \dot{x}_E}{\partial \theta} & \frac{\partial \dot{x}_E}{\partial \psi} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial \dot{h}}{\partial V_T} & \frac{\partial \dot{h}}{\partial \alpha} & \frac{\partial \dot{h}}{\partial \beta} & \frac{\partial \dot{h}}{\partial \phi} & \frac{\partial \dot{h}}{\partial \theta} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial \dot{V}_T}{\partial V_T} & \frac{\partial \dot{V}_t}{\partial \alpha} & \frac{\partial \dot{V}_t}{\partial \beta} & \frac{\partial \dot{V}_T}{\partial \phi} & \frac{\partial \dot{V}_t}{\partial \theta} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial \dot{\alpha}}{\partial V_T} & \frac{\partial \dot{\alpha}}{\partial \alpha} & \frac{\partial \dot{\alpha}}{\partial \beta} & \frac{\partial \dot{\alpha}}{\partial \phi} & \frac{\partial \dot{\alpha}}{\partial \theta} & 0 & \frac{\partial \dot{\alpha}}{\partial p} & \frac{\partial \dot{\alpha}}{\partial q} & \frac{\partial \dot{\alpha}}{\partial r} \\
0 & 0 & 0 & \frac{\partial \dot{\beta}}{\partial V_T} & \frac{\partial \dot{\beta}}{\partial \alpha} & \frac{\partial \dot{\beta}}{\partial \beta} & \frac{\partial \dot{\beta}}{\partial \phi} & \frac{\partial \dot{\beta}}{\partial \theta} & 0 & \frac{\partial \dot{\beta}}{\partial p} & 0 & \frac{\partial \dot{\beta}}{\partial r} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \dot{\phi}}{\partial \phi} & \frac{\partial \dot{\phi}}{\partial \theta} & 0 & \frac{\partial \dot{\phi}}{\partial p} & \frac{\partial \dot{\phi}}{\partial q} & \frac{\partial \dot{\phi}}{\partial r} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \dot{\theta}}{\partial \phi} & 0 & 0 & 0 & \frac{\partial \dot{\theta}}{\partial q} & \frac{\partial \dot{\theta}}{\partial r} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \dot{\psi}}{\partial \phi} & \frac{\partial \dot{\psi}}{\partial \theta} & 0 & 0 & \frac{\partial \dot{\psi}}{\partial q} & \frac{\partial \dot{\psi}}{\partial r} \\
0 & 0 & 0 & \frac{\partial \dot{p}}{\partial V_T} & 0 & \frac{\partial \dot{p}}{\partial \beta} & 0 & 0 & 0 & \frac{\partial \dot{p}}{\partial p} & \frac{\partial \dot{p}}{\partial q} & \frac{\partial \dot{p}}{\partial r} \\
0 & 0 & 0 & \frac{\partial \dot{q}}{\partial V_T} & \frac{\partial \dot{q}}{\partial \alpha} & 0 & 0 & 0 & 0 & \frac{\partial \dot{q}}{\partial p} & \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial r} \\
0 & 0 & 0 & \frac{\partial \dot{r}}{\partial V_T} & 0 & \frac{\partial \dot{r}}{\partial \beta} & 0 & 0 & 0 & \frac{\partial \dot{r}}{\partial p} & \frac{\partial \dot{r}}{\partial q} & \frac{\partial \dot{r}}{\partial r}
\end{pmatrix}. \tag{2.22}$$

Traditionally, the states are divided into longitudinal and lateral-directional categories. The longitudinal states are $[x_N, h, V_T, \alpha, \theta, q]$ and the lateral-directional states are $[x_E, \beta, \phi, \psi, p, r]$. As seen in (2.22), many of the elements in the state transition matrix are equal to zero. However, there are many elements that show coupling between the lon-

gitudinal and lateral-directional states as evidenced by the non-zero partial derivatives of longitudinal states with respect to a lateral state and vice-versa. In certain steady flight conditions, many of these coupled partial derivatives take on values near zero, but others do not. A specific example is shown in Chapter 3 to demonstrate this non-trivial coupling.

The symbolic Jacobian for the control matrix is

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial \dot{V}_T}{\partial F_T} \\ 0 & 0 & 0 & \frac{\partial \dot{\alpha}}{\partial F_T} \\ 0 & 0 & 0 & \frac{\partial \dot{\beta}}{\partial F_T} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\partial \dot{p}}{\partial \delta_a} & 0 & \frac{\partial \dot{p}}{\partial \delta_r} & 0 \\ 0 & \frac{\partial \dot{q}}{\partial \delta_e} & 0 & 0 \\ \frac{\partial \dot{r}}{\partial \delta_a} & 0 & \frac{\partial \dot{r}}{\partial \delta_r} & 0 \end{pmatrix}. \tag{2.23}$$

Traditionally, the controls are divided into longitudinal and lateral-directional categories as well. The longitudinal controls are $[\delta_e, F_T]$ and the lateral-directional controls are $[\delta_a, \delta_r]$. As seen in (2.23), most of the elements in the control matrix are equal to zero. The only coupling between the longitudinal and lateral-directional states is the partial derivative of the sideslip time rate of change with respect to thrust.

2.3.2 Calculating a Trim Condition

Trim conditions can be calculated by solving

$$[\dot{x}_E \, \dot{h} \, \dot{V}_T \, \dot{\alpha} \dot{\beta} \dot{\phi} \dot{\theta} \dot{\psi} \, \dot{p} \dot{q} \dot{r}]^T = f(\mathbf{x}_{trim}, \mathbf{u}_{trim}) \tag{2.24}$$

for $\mathbf{x}_{trim}$ and $\mathbf{u}_{trim}$ where the components of left hand side are all equal to zero with the exception of $\dot{x}_E$ (for turning flight) and $\dot{h}$.

The value for $\dot{x}_E$ is calculated as

$$\dot{x}_{E_{trim}} = \frac{V_T}{R_{Turn}} \cos\gamma \tag{2.25}$$

and the value for $\dot{h}$ is calculated as

$$\dot{h}_{trim} = V_T \sin\gamma \tag{2.26}$$

respectively, where γ is the flight path angle, R_{Turn} is the turn radius, and V_T is the desired airspeed.

The set of nonlinear equations in 2.24 can be solved using one of several algorithms, including Levenberg-Marquardt, Gauss-Newton, and Steepest Descent [44]. The relationship between pitch angle, flight path angle, and angle of attack ($\theta = \gamma + \alpha$) can be used when setting the initial state vector. In the case of constant altitude flight, the pitch angle is equal to the angle of attack ($\theta = \alpha$), eliminating one of the unknowns [43].

2.4 UAS Sensors

Small UAS have the capability to carry a variety of onboard sensors for aircraft pose measurements. These are the pre-GPS legacy sensors and the modern odometry/localization sensors. The first category includes both the IMU [45] and the ADS [46], [47], [48]. The

second category includes computer vision sensors that can be processed with a variety of techniques [49], [36], [50], [35], a laser scanner [37], [38], [39], a GPS receiver [51], [47], [52], and a proposed LTE transceiver using the OTDOA technique [53].

2.4.1 Inertial Measurement Unit (IMU)

The IMU for a small UAS generally consists of 3-axis gyroscopes, 3-axis accelerometers, and a 3-axis magnetometer and is installed at the center of gravity of the aircraft aligned with the body frame. In this case where the sensor frame and body frame are aligned and the angles between the body and inertial frames are small, such as in steady level flight, the noise on each individual accelerometer and gyroscope signal is assumed to be white and Gaussian. However, this is generally not the case for magnetometers, which are biased by the effect of buildings on local magnetic fields. The individual sensors output raw data that is filtered to provide measurements of the aircraft's angular rates and accelerations. An attitude and heading reference system (AHRS) uses filters such as an Extended Kalman Filter [51], a Complementary Filter [54], [55], and an Unscented Kalman Filter [56] to generate roll, pitch, and heading measurements using angular rates, acceleration (gravity vector), and magnetometer outputs. The gravity vector can only be estimated accurately under the assumption that accelerations due to maneuvers are close to zero, as is the case in trimmed flight [57].

Typical filtered sensor state angular rate 1σ noise values are generally in the 0.5 degrees/second range [58], [47], [59]. Filtered Euler angle 1σ noise values are generally similar for pitch and roll in the range of $0.6 - 3$ degrees [60], [54], [56]. However, yaw is generally measured using an inertial navigation system (INS), which uses GPS measurements to correct the IMU yaw to roughly 8 degree 1σ noise value [45]. This noise value is used for the yaw measurement throughout this research.

2.4.2 Air Data System (ADS)

In addition to the IMU, another legacy aircraft sensor is the ADS. Most ADS include, as a minimum, a pitot tube that uses static and dynamic (stagnation) ports to generate static pressure which in turn can be used to generate airspeed measurements. Airspeed measurements are almost always available on fixed-wing UAS [61]. A 5-hole pitot probe is capable of providing airspeed, angle of attack, and angle of sideslip [48]. There are also customized ADS solutions that can augment an autopilot to provide angle of attack and angle of sideslip measurements [62]. Airspeed 1σ accuracy is between $1-1.5$ meters/second [48], [46]. Angle of attack and angle of sideslip 1σ accuracies, using a differential pressure probe, are both roughly 1 degree. Since the static port measures ambient pressure, the ADS can also produce altitude measurements. Typical 1σ altitude accuracy ranges from $1.5-3$ meters [63], [46].

2.4.3 Computer Vision-Based Sensors

An area of active research in UAS navigation is the use of computer vision-based sensors to provide navigation information for position, airspeed, and attitude. These sensors use a variety of techniques including optical flow [33], [64], [36], feature detection [50], and vanishing points [65] to provide measurements to the filter. One of the largest advantages of using this type of sensor is that it does not depend on any type of electromagnetic signal to work properly, making it a complementary sensor to the GPS.

2.4.3.1 Optical Flow

Optical flow is defined in [66] as the distribution of apparent velocities of brightness pattern movements in an image. It is generally calculated by comparing pixels in sequential images to determine the local velocity of the camera that is capturing the images. This concept can be applied to a UAS operating in an urban canyon by attaching a camera to the vehicle and

calculating the apparent local velocities of adjacent buildings or the street below. Accuracy is typically measured in pixels per frame, with a scaling process necessary to convert to meters per second.

An ideal optical flow application to the urban environment is the 'centering response', with biological inspiration from bees. Reference [67] explains that bees are able to hold this centerline trajectory by equalizing the apparent motion images on their retinas. This phenomenon has been demonstrated in biological tests and on UAS operating in urban canyons, in simulation [33], [68] and in experiments [69], [70]. In addition to maintaining a centerline trajectory, further experiments have shown that vehicles equipped with side facing optical flow sensors and a pair of front facing stereo sensors for obstacle detection can also navigate 90° turns in a simulated urban canyon [36].

Standardized accuracy metrics have been established to compare performance of the different optical flow calculation methods. The two main metrics throughout the literature are 1) average angular error and 2) endpoint error. Average angular error is described in [71], [72], [73], [74] and Middlebury Dataset [75]. It is measured as the angle between the true velocity vector $\vec{v}_c$ and the estimated velocity vector $\vec{v}_e$ in the image coordinate plane using

$$\psi_E = \cos^{-1}(v_c \bullet v_e). \tag{2.27}$$

Endpoint error [73] defined in image plane coordinates as

$$\begin{aligned} E_x &= |u_c - u_e| \\ E_y &= |v_c - v_e| \end{aligned} \tag{2.28}$$

can also be used to quantify the absolute magnitude of differences of components.

The Middlebury Dataset is a vast resource providing optical flow accuracy characterization information for both metrics over 91 different methods using well-known standardized image sequences such as Urban, Translating Tree, and Yosemite [75]. More detailed optical

flow information is found in Appendix A.

2.4.3.2 Line Detection and Vanishing Points

Another computer vision-based technique for UAS navigation uses line detection and vanishing points to generate roll and pitch measurements. A brief overview of the process to measure attitude angles from vanishing points is discussed here with further details available in [65]. The first step in the algorithm is to detect parallel lines in a two-dimensional image. These lines may represent vertical edges of buildings parallel to the direction of gravity or horizontal edges of buildings at the street level, orthogonal to the direction of gravity. Once the lines have been detected, the second step in the algorithm is to follow the lines to points of intersection as shown in Figure 2.1. These points of intersection are known as vanishing points, categorized as either vertical or horizontal based on the direction of the parallel lines.

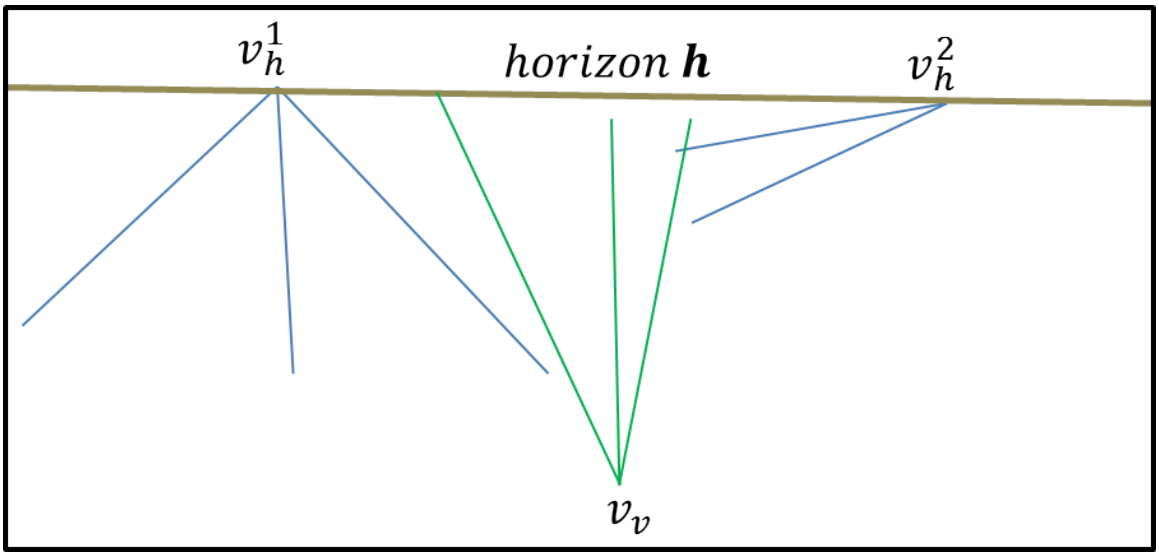


Figure 2.1: Example of parallel lines and vanishing points using an urban scene courtesy of Hwangbo and Kanade [65].

The third step in the algorithm is to use the vanishing points to calculate roll and pitch. A single vertical vanishing point v_v^* with coordinates (v_x, v_y) can be used to calculate both angles

$$\phi = atan2(v_x, v_y) \tag{2.29}$$

and

$$\theta = atan\frac{1}{\sqrt{v_x^2 + v_y^2}} \tag{2.30}$$

where the coordinates are generated from the projection of the world z-axis onto the two-dimensional camera plane.

Horizontal plane vanishing point coordinates, v_h^*, are calculated as

$$v_h^* = \left[\frac{\cos\phi\sin\psi - \sin\phi\sin\theta\cos\psi}{\cos\theta\cos\psi}, \frac{-\sin\phi\sin\psi - \cos\phi\sin\theta\cos\psi}{\cos\theta\cos\psi}\right]^T \tag{2.31}$$

using the projection of the direction of travel axis onto the unit vector in the direction of each horizontal vanishing point. These points can also be used for the calculation of roll and pitch if at least two horizontal vanishing points are present in an image as equation is not decoupled in the roll and pitch directions. Once all vanishing points are calculated from an image, they can be used in a Kalman Filter to reset the error in the IMU-based attitude angle estimates.

2.4.3.3 Feature Detection

In [50], a scale-invariant feature transform technique is used to match features from one image to the next in order to accurately correct position, airspeed, and attitude angles. Use of feature matching for state estimation has several steps. The first step is to analyze the initial image, capturing each scale-invariant descriptor (or feature). When the next image is received, the Euclidean distance is calculated between a descriptor in the first image and its nearest neighbors in the next image. Once the images are matched, the homography (or relationship) between the two images is calculated using a scale factor defined as the ratio

of the camera coordinate out of the bottom of the aircraft before and after transformation. Reference [50] gives details on how to use the homography matrix to solve for the rotation matrix and the translation vector. This rotation matrix and translation vector can then be used to correct IMU and altimeter measurements.

2.4.3.4 Urban Cues

In the urban environment, images from computer vision systems can be used to detect cues such as roads, lane markings, crosswalks, and stop lines. While primarily used for ground vehicle applications, UAS applications are possible when operating at sufficiently low altitudes to detect and isolate these cues in successive images. Many works have studied this problem including [76], [77], [78]. Paetzhold and Franke [76] focused on both unknown and known road situations to locate these cues. In the unknown situations they made assumptions about the characteristics of the cues in order to extract them from images as polygons. These assumptions include orientation of cues with respect to each other and the vehicle trajectory as well as constant brightness and linear shape among others. With a map of cues, frame to frame feature matching was used to generate cue-based state measurements. Stereo vision was then used to separate vertical and horizontal shapes and motion as well as identify image clutter. He *et al*. [77] used an intensity (grayscale) image to detect the road boundaries, including curvature, and then used the full color image to find the road area within the boundaries. Newman *et al*. [78] created a navigation system to both generate three-dimensional maps and pose estimates using vision and lasers. Their system analyzed the generated maps to provide labels within the maps to classify the different parts of the image as walls, foliage, grass, etc.

2.4.4 GPS Receiver

As previously discussed, GPS navigation suffers performance degradation in the urban environment with availability rates range from 27% to 50% [18], [10], [19], [29]. When

GPS measurements are available some of the specific factors that contribute to the error include the atmosphere, satellite geometry, satellite clock drift, multipath, and measurement noise [47]. These error sources generally consist of a bias error term and random error that can be combined using a root sum square to calculate total error from each source. Nominal total error values for each source are shown in Table 2.1, with the root sum square total user equivalent range error (UERE) in the last row. Filtered UERE values range from 4 meters [79] to of 5.1 meters [43]

Table 2.1: UERE typical error standard deviation (meters) courtesy of u-blox.com [79].

Source	Total
Ephemeris Data	1.5 meters
Satellite Clock	1.5 meters
Ionosphere	3.0 meters
Troposphere monitoring	0.7 meters
Multipath	1.0 meters
Receiver measurement	0.5 meters
Filtered UERE (RMS)	4.0 meters

The dilution of precision (DOP), the second component of the GPS error, is a measure of the effect of satellite geometry on the accuracy of the position calculation [80]. Nominal HDOP and vertical dilution of precision (VDOP) are 1.3 and 1.8, respectively [43]. Using UERE and DOP, the position error 1σ RMSE is calculated for the horizontal position using

$$E_{1\sigma_{n-e}} = HDOP * UERE \tag{2.32}$$

and the vertical position using

$$E_{1\sigma_h} = VDOP * UERE. \tag{2.33}$$

From this model, GPS 1σ position error in both horizontal dimensions is roughly 3.67 meters and roughly 7.2 meters in the vertical direction. The horizontal values are similar to the 3 meter error values used in [46] and [58]. The horizontal values are much lower

than those published from urban canyon experiments that ranged from 30 meters to almost a kilometer [18], [19].

If time-varying error is assumed, the standard deviation is calculated as

$$v_{k+1} = exp\{-k_{GPS} * T_s\}v_k + \eta_{GPS_k} \tag{2.34}$$

using the Gauss-Markov error model [81], consisting of both the slowly varying zero-mean bias and a random noise component where v_k is the position error standard deviation at time step k, initialized using 2.32 or 2.33. The remaining terms include $-k_{GPS}$ as the process time constant, T_s as the GPS measurement sampling time, and η_{GPS} as the zero-mean Gaussian random noise component. Table 2.2 shows typical values for these parameters.

Table 2.2: GPS measurement error parameter values [43].

Direction	$1/k_{GPS}$ (seconds)	T_s (seconds)	η_{GPS} (meters)
North	1100	1	0.21
East	1100	1	0.21
Down	1100	1	0.40

2.4.5 LTE

As shown in Figure 2.2, there are several different geolocation techniques currently available to pinpoint the location of a smart phone with varying levels of quality of service. They fall into cellular ID (CID), time difference of arrival (TDOA), and assisted-Global navigation Satellite System (A-GNSS) categories. The main characteristic of the CID techniques is that they use the location of the current Base Transceiver Station (cellular network tower) as an estimate of the location of the receiver. The TDOA techniques use the time difference between reference signals sent from different multiple towers to the device to determine its location. The A-GNSS techniques use satellite signals to determine the device location, but are hampered by longer time-to-first-fix and shorter battery life for the phone as compared

to the other techniques [82]. Enhanced CID, OTDOA, and A-GNSS were first defined in the 3GPP TS 36.355 Release 9 as the LTE Positioning Protocol techniques [83].

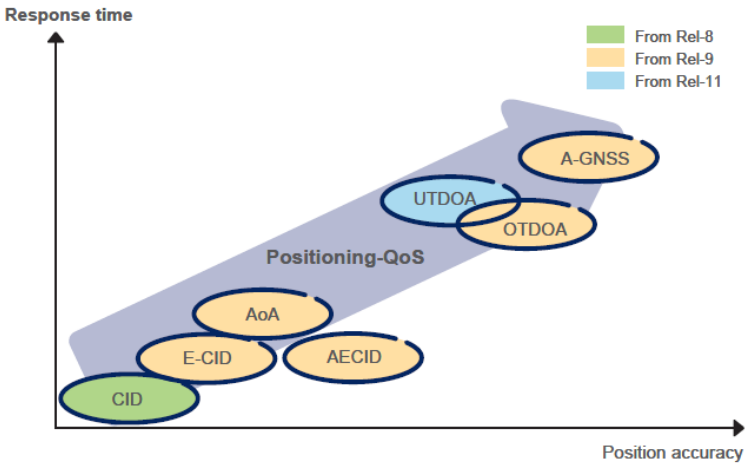


Figure 2.2: Smart phone geolocation techniques courtesy of Ericsson [84].

Geolocation within an LTE network is a multilayered three-step process involving several different nodes in the network. The first step provides information and assistance to complete the location estimation. The second step is signal measurement and reporting of results. The third is actual location estimation using the measurements [85]. More information on how geolocation data is routed through the cellular network can be found in [86], [87] and [88].

2.4.5.1 LTE Standardized Geolocation Techniques

Enhanced Cell ID The enhanced-CID method is a rapid network-based geolocating technique that requires the device to send information, as listed below, to allow the network to calculate an estimated location of the phone [89] based on its current cell.

- Cell Tower (known as eNodeB or eNB in LTE network) and Serving CID
- Received Signal Strength
- Neighboring cell IDs and signal strengths

- Timing advance
- Angle of Arrival

Enhanced CID is not a primary geolocation method when a high level of accuracy is needed. However, this technique can quickly provide an accuracy check for the OTDOA and A-GNSS methods by identifying the device's current serving tower.

Assisted-Global Navigation Satellite Systems The A-GNSS geolocation method is a device-assisted technique that is the primary and most accurate method, in general, used in LTE networks [90]. Here, the device uses network-based assistance data to speed up the satellite signal acquisition process from either the GPS or GLONASS constellations or a combination of both. The assistance data comes from stationary network GPS receivers and is passed to a location server and then to the device. Examples of the two different assistance data types are below [89].

- Position calculation assisting data can include reference time, reference position, satellite ephemeris, and clock corrections.
- Measurement-assisting data can include reference time, visible satellite list, satellite signal Doppler, code phase, and Doppler and code phase search windows.

Once the first fix has been achieved, either the device or network can calculate the estimated location with GPS-level accuracy. This technique allows for more accurate location estimates and drains the battery more slowly than if GPS is used by itself. However, its performance still suffers from the urban environment [82].

Observed Time Difference of Arrival The OTDOA method, shown in Figure 2.3, uses multilateration (hyperbolic lateration) to geolocate the device. This is similar to the technique used by GPS receivers but with shorter baselines (thus less accuracy). Either the

device or the network can execute the geolocation calculations, but in order for the device-based mode to be accurate, the following conditions must be met.

- The device must have a priori knowledge of the two-dimensional coordinates of all cell towers in its operating area.
- All cell towers in the operating area must be synchronized with each other, but do not need to be synchronized with the device.

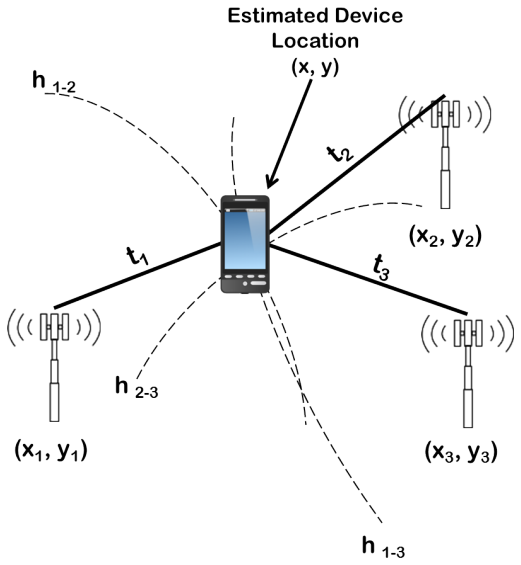


Figure 2.3: OTDOA geolocation technique.

The process is initiated when a position reference signal is sent from hearable cell towers n_h to the device, where $n_h \geq 3$. Once the device timestamps the arrival times of the n_h signals, the difference between each pair of arrival times $(t_j - t_i)$ is calculated. It is converted to a difference in distance traveled

$$d_j - d_i = c * (t_j - t_i) \tag{2.35}$$

by multiplying the time difference by the speed of light, c.

These constant differences form a hyperbola (h_{j-i}) between the two towers that contain the estimated position of the device. To narrow down the device location to a maximum of two points along h_{j-i}, a second hyperbola is drawn (h_{i-k}) [89]. If the two hyperbolas only intersect in one location, then the estimated device location is that point of intersection. However, as Figure 2.3 shows for three hearable towers, sometimes h_{j-i} and h_{i-k} intersect in two points, so a third hyperbola, h_{j-k} is needed to find the unique location of the device. For three hearable towers, the geolocation process requires solving the following three non-linear equations simultaneously

$$\begin{aligned} d_2 - d_1 &= \sqrt{(x_2 - x)^2 + (y_2 - y)^2} - \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \\ d_3 - d_1 &= \sqrt{(x_3 - x)^2 + (y_3 - y)^2} - \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \\ d_3 - d_2 &= \sqrt{(x_3 - x)^2 + (y_3 - y)^2} - \sqrt{(x_2 - x)^2 + (y_2 - y)^2}. \end{aligned} \tag{2.36}$$

Normally numerical methods are used to solve for the location of the receiver (x, y). These include algorithms such as Levenberg-Marquardt (LM), Gauss-Newton, and Steepest Descent, all discussed in [44]. Another iterative algorithm that specifically addresses a moving source is the Constrained Weighted Least Squares algorithm [91]. While this method is able to accurately geolocate a UAS using only three towers, as a numerical solver, it may not be able to reach a solution as quickly as other methods.

When $n_h \geq 4$, the system of equations can be linearized and solved using matrix operations for most tower layouts. The two-dimensional geolocation equations for this technique are

$$A_{m_h} x + B_{m_h} y + D_{m_h} = 0 \tag{2.37}$$

$$A_{m_h} = \frac{2x_{m_h}}{d_{m_h} - d_1} - \frac{2x_2}{d_2 - d_1} \tag{2.38}$$

$$B_{m_h} = \frac{2y_{m_h}}{d_{m_h} - d_1} - \frac{2y_2}{d_2 - d_1} \tag{2.39}$$

$$D_{m_h} = d_{m_h} - d_1 - (d_2 - d_1) - \frac{x_{m_h}^2 + y_{m_h}^2}{d_{m_h} - d_1} - \frac{x_2^2 + y_2^2}{d_2 - d_1} \tag{2.40}$$

where $m_h = \{3, ..., n_h\}$. $\{x, y\}$ are calculated using

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} A_3 & B_3 \\ \vdots & \vdots \\ A_{m_h} & B_{m_h} \end{bmatrix}^{\dagger} - \begin{bmatrix} D_3 \\ \vdots \\ D_{m_h} \end{bmatrix} \tag{2.41}$$

where a pseudo-inverse is used since the matrix may not always be square. The complete three-dimensional derivation is available in [92]. Reference [93] shows that in situations where $n_h > 4$, the accuracy of the TDOA method increases as the number of towers (and therefore number of equations) increases.

A similar technique in the TDOA family is Uplink-Time Difference of Arrival (UTDOA), set to be included in the 3GPP Release 11. It will not be discussed further in this research due its limited use. UTDOA is essentially the fully network-based version of OTDOA currently used for the E-911 mission [84], [93].

2.4.5.2 Current OTDOA Accuracy

OTDOA accuracy data is difficult to collect due to the sensitive nature of different carrier network architectures and tower layouts. As a result most available data is simulation-based. Table 2.3 [94], [95] shows early results of two simulation studies that vary substantially. This disparity shows the difficulty in determining the true accuracy of the OTDOA geolocation technique.

Table 2.3: OTDOA two-dimensional measurement accuracy statistics.

Sensor	Neuland *et al.* [94]	Tao [95]
Mean, μ (meters)	16.3	45.54
Standard Deviation, σ (meters)	31.4	2.65

More accurate LTE geolocation data was generated in [53] as a function of the number

of hearable towers. Although this data has yet to be reproduced, in this research it will be assumed that the LTE network may or will eventually have 1σ accuracy ranging from 3 meters with 20 hearable towers to 7.5 meters with 7 hearable towers. More LTE accuracy data is shown in Appendix E.4.

2.4.6 LiDAR

LiDAR is another navigation sensor that could possibly be used on small UAS in the urban environment. LiDAR sensors have been shown to increase urban canyon navigation accuracy by over an order of magnitude over the traditional GPS/IMU/odometry solution [96]. LiDAR has also been shown to have sub-meter user position accuracy in urban environments when tightly coupled with GPS/INS [38]. Since it can operate in both bad weather and GPS degraded/denied conditions it is a good candidate to consider for urban navigation. However, LiDAR systems can cost at least $2K$ dollars [97] and have a typical range of only 30 meters [98], making them very expensive for a low-cost small UAS. They also are only useful when flying trajectories that are within sensor range of buildings to be useful.

For the remainder of this dissertation each sensor will be referenced as follows: inertial measurement unit as IMU, air data system as ADS, LTE device/transceiver as LTE, GPS receiver as GPS, computer vision using feature detection as VISION/IMU and computer vision optical flow as VISION-OF.

2.5 Bayesian State Estimation Filters

One application of the state estimation filter is to provide an estimate of the state of a system that can be used in feedback control of real-world systems. Bayesian state estimation filters accomplish this task through use of a prediction-correction structure to calculate both a state estimate of the system and an error covariance matrix. Kalman Filter variants generally assume a Gaussian distribution of the state with a known initial mean and covariance, while

the particle filter makes no assumptions about the distribution of the state and allows the user to quantify its results as necessary.

The general form of the n dimensional dynamics of a system is

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \tag{2.42}$$

where $\mathbf{x}_k$ is the state of the system at a given time-step, $\mathbf{u}_k$ is the control at a given time, and the unknown process (plant) noise is $\mathbf{w}_k \sim \mathcal{N}(0, Q_k)$. The general form of the discrete measurement model is

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k) \tag{2.43}$$

with measurement noise $\mathbf{v}_k \sim (0, R_k)$, where R_k, the measurement noise covariance matrix, is empirically determined.

2.5.1 Kalman Filter Derivation

To understand how the Kalman Filter [99], [100] generates a state estimate vector and accompanying covariance, the filter equations are derived using an adaptation of [101]. The first step is to write the both (2.42) and (2.43) as linear equations

$$\mathbf{x}_k = A_{k-1}\mathbf{x}_{k-1} + B_{k-1}\mathbf{u}_{k-1} + L_{k-1}\mathbf{w}_{k-1} \tag{2.44}$$

$$\mathbf{z}_k = H_k\mathbf{x}_k + M_k\mathbf{v}_k \tag{2.45}$$

with the eventual goal of finding an expression for the state estimate vector, $\hat{\mathbf{x}}_k$, that minimizes the error, $\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$, between the state vector and its estimate as well as the covariance $P_k = \mathcal{E}[\mathbf{e}_k\mathbf{e}_k^T]$ of this estimate.

2.5.1.1 Prediction

At each time-step the dynamics of the system are recursively propagated forward to generate the predicted (a priori) state estimate, $\hat{\mathbf{x}}_k^-$

$$\hat{\mathbf{x}}_k^- = A_{k-1}\mathbf{x}_{k-1} + B_{k-1}\mathbf{u}_{k-1} \tag{2.46}$$

and the predicted covariance matrix, P_k^-.

$$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1} \tag{2.47}$$

where $Q_k = E[\mathbf{w}_k\mathbf{w}_k^T]$.

Although (2.46) is the linear version of (2.42), with the noise term removed, (2.47) does not have an equivalent expression from the dynamics and therefore must be derived. From (2.44) and (2.46) the predicted error vector is

$$\begin{aligned}
\mathbf{e}_k^- =& \mathbf{x}_k - \hat{\mathbf{x}}_k^- \\
=& A_{k-1}\mathbf{x}_{k-1} + B_{k-1}\mathbf{u}_{k-1} + L_{k-1}\mathbf{w}_{k-1} - A_{k-1}\hat{\mathbf{x}}_{k-1} + B_{k-1}\mathbf{u}_{k-1} \\
=& A_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}^-) + L_{k-1}\mathbf{w}_{k-1} \\
=& A_{k-1}\mathbf{e}_{k-1}^- + L_{k-1}\mathbf{w}_{k-1}.
\end{aligned}$$

The predicted covariance as the expectation of the square of the predicted error is rewritten as

$$\begin{aligned}
P_k^- =& \mathcal{E}[\mathbf{e}_k^-\mathbf{e}_k^{-T}] \\
=& \mathcal{E}[(A_{k-1}\mathbf{e}_{k-1}^- + L_{k-1}\mathbf{w}_{k-1})(A_{k-1}\mathbf{e}_{k-1}^- + L_{k-1}\mathbf{w}_{k-1})^T] \\
=& \mathcal{E}[A_{k-1}\mathbf{e}_{k-1}^-\mathbf{e}_{k-1}^{-T}A_{k-1}^T + 2L_{k-1}\mathbf{w}_{k-1}\mathbf{e}_{k-1}^{-T}A_{k-1}^T + L_{k-1}\mathbf{w}_{k-1}\mathbf{w}_{k-1}^T L_{k-1}^T] \\
=& A_{k-1}\mathcal{E}[\mathbf{e}_{k-1}^-\mathbf{e}_{k-1}^{-T}]A_{k-1}^T + 2L_{k-1}\mathcal{E}[\mathbf{w}_{k-1}\mathbf{e}_{k-1}^{-T}] + L_{k-1}\mathcal{E}[\mathbf{w}_{k-1}\mathbf{w}_{k-1}^T]L_{k-1}^T.
\end{aligned}$$

Assuming that process noise is uncorrelated with state error allows $\mathcal{E}[\mathbf{w}_k\mathbf{e}_k^{-T}] = 0$. Making the necessary substitutions into the above equation, P_k^- simplifies to

$$P_k^- = A_{k-1}P_{k-1}^-A_{k-1}^T + L_{k-1}Q_{k-1}L_{k-1}^T. \tag{2.48}$$

2.5.1.2 Correction

The correction step of the Kalman Filter uses available measurements to refine both the predicted state estimate and covariance matrix, generating a corrected (a posteriori) state estimate, $\hat{\mathbf{x}}_k$, and a corrected covariance matrix, P_k. The corrected state estimate is calculated using

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + K_k(\mathbf{z}_k - \hat{\mathbf{z}}_k^-) \tag{2.49}$$

where K_k is the Kalman gain, $\mathbf{z}_k$ is available measurement vector, and $\hat{\mathbf{z}}_k^- = H_k\hat{\mathbf{x}}_k^-$ is the prediction of the measurement.

The corrected covariance matrix, defined as $P_k = \mathcal{E}[\mathbf{e}_k\mathbf{e}_k^T]$, is derived by finding an expression for $\mathbf{e}_k$ and then calculating the expectation value. First, (2.45) is substituted into

$$\begin{aligned}
\hat{\mathbf{x}}_k =& \hat{\mathbf{x}}_k^- + K_k(H_k\mathbf{x}_k + M_k\mathbf{v}_k - H_k\hat{\mathbf{x}}_k^-) \\
=& \hat{\mathbf{x}}_k^- + K_kH_k\mathbf{x}_k + K_kM_k\mathbf{v}_k - K_kH_k\hat{\mathbf{x}}_k^- \\
=& (I - K_kH_k)\hat{\mathbf{x}}_k^- + K_kH_k\mathbf{x}_k + K_kM_k\mathbf{v}_k
\end{aligned}$$

and simplified. Using the corrected state estimate error equation, $\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$, and the above

result, an expression for $\mathbf{e}_k$ is written as

$$\begin{aligned}
\mathbf{e}_k &= \mathbf{x}_k - \hat{\mathbf{x}}_k \\
&= \mathbf{x}_k - (I - K_k H_k)\hat{\mathbf{x}}_k^- - K_k H_k \mathbf{x}_k - K_k M_k \mathbf{v}_k \\
&= (I - K_k H_k)\mathbf{x}_k - (I - K_k H_k)\hat{\mathbf{x}}_k^- - K_k M_k \mathbf{v}_k \\
&= (I - K_k H_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) - K_k M_k \mathbf{v}_k \\
&= (I - K_k H_k)\mathbf{e}_k^- - K_k M_k \mathbf{v}_k.
\end{aligned}$$

The corrected covariance matrix is written using the above result

$$\begin{aligned}
P_k &= \mathcal{E}[((I - K_k H_k)\mathbf{e}_k^- - K_k M_k \mathbf{v}_k)((I - K_k H_k)\mathbf{e}_k^- - K_k M_k \mathbf{v}_k)^T] \\
&= \mathcal{E}[(I - K_k H_k)\mathbf{e}_k^- \mathbf{e}_k^{-T}(I - K_k H_k)^T - 2(I - K_k H_k)\mathbf{e}_k^- \mathbf{v}_k^T M_k^T K_k^T + K_k M_k \mathbf{v}_k \mathbf{v}_k^T M_k^T K_k^T] \\
&= (I - K_k H_k)\mathcal{E}[\mathbf{e}_k^- \mathbf{e}_k^{-T}](I - K_k H_k)^T - 2(I - K_k H_k)\mathcal{E}[\mathbf{e}_k^- \mathbf{v}_k^T]M_k^T K_k^T + K_k M_k \mathcal{E}[\mathbf{v}_k \mathbf{v}_k^T]M_k^T K_k^T.
\end{aligned}$$

Recalling $P_k^- = \mathcal{E}[\mathbf{e}_k^- \mathbf{e}_k^{-T}]$, letting $R_k = \mathcal{E}[\mathbf{v}_k \mathbf{v}_k^T]$ and assuming there is no correlation between $\mathbf{v}_k$and $\mathbf{e}_k^-$, the above expression can be simplified to

$$P_k = (I - K_k H_k)P_k^-(I - K_k H_k)^T + K_k M_k R_k M_k^T K_k^T \tag{2.50}$$

the Joseph Formula.

2.5.1.3 Calculating the Optimal Kalman Gain

The optimal Kalman gain matrix is the solution of an optimization problem minimizing the trace of the corrected covariance matrix, P_k.

$$\begin{aligned} tr[P_k] =& tr[\tilde{Q}\mathbf{\Lambda}\tilde{Q}^{-1}] \\ =& tr[\mathbf{\Lambda}QQ^{-1}] \\ =& tr[\Lambda] \\ =& \sum_{i=1}^{n} \lambda_{P_k,i}. \end{aligned}$$

Minimizing the trace of the matrix is equivalent to minimizing the sum of the eigenvalues of the matrix, which is in turn equivalent to minimizing the uncertainty of the corrected state estimate as shown below, where $\tilde{Q}$ is the matrix formed by the columnwise eigenvectors of P_k, Λ is the diagonal matrix of eigenvalues, and λ_i is the i^{th} eigenvalue of P_k.

The formal problem is now posed as

$$\begin{aligned} Minimize\, J_k(K_k) =& tr[P_k] \\ =& tr[(I-K_kH_k)P_k^-(I-K_kH_k)^T + K_kM_kR_kM_k^TK_k^T] \\ =& tr[P_k^- - K_kH_kP_k^- - P_k^-H_k^TK_k^T + K_kH_kP_k^-H_k^TK_k^T + K_kM_kR_kM_k^TK_k^T] \\ =& tr[P_k^- - K_kH_kP_k^- - P_k^-H_k^TK_k^T + K_kW_kK_k^T] \end{aligned}$$

where

$$W_k = HP_k^-H_k^T + M_kR_kM_k^T.$$

To find the minimum value of $J_k(K_k)$, its derivative with respect to K_k is calculated as

$$\frac{\partial J_k(K_k)}{K_k} = 2(-P_k^-H_k^T + K_kW_k) = \mathbf{0}$$

and set equal to the zero matrix. Solving for K_k yields the optimal gain

$$K_k = P_k^- H_k^T W_k^{-1}.$$

Right multiplying this expression by $W_k K_k^T$ gives

$$\begin{aligned} K_k W_k K_k^T &= P_k^- H_k^T W_k^{-1} W_k K_k^T \\ &= P_k^- H_k^T K_k^T. \end{aligned}$$

Substituting the above result into the expanded version of (2.50) gives

$$\begin{aligned} P_k &= P_k^- - K_k H_k P_k^- - P_k^- H_k^T K_k^T + P_k^- H_k^T K_k^T \\ &= P_k^- - K_k H_k P_k^- \\ &= (I - K_k H_k) P_k^-. \end{aligned}$$

When using the optimal Kalman gain, the corrected covariance becomes

$$P_k = (I - K_k H_k) P_k^- \tag{2.51}$$

completing the derivation.

2.5.2 Extended Kalman Filter

When (2.42) and/or (2.43) are nonlinear functions, the posterior distribution loses its Gaussian properties after the first nonlinear state propagation/measurement generation. The Extended Kalman Filter allows the nonlinear dynamics to be propagated and provides a predicated covariance, corrected estimated state vector, and corrected covariance approximation by linearizing (2.42) and/or (2.43) at each instance in time around the most recent estimated state vector [100]. This approximates the posterior Gaussian distribution, while

allowing the filter to generate corrected values using the Kalman Filter equations. The Jacobians A_k, L_k, H_k, and M_k are calculated as

$$A_k = \frac{\partial f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)}{\partial \mathbf{x}_k}|_{\hat{\mathbf{x}}_k, \mathbf{u}_k} \tag{2.52}$$

$$L_k = \frac{\partial f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)}{\partial \mathbf{w}_k}|_{\hat{\mathbf{x}}_k, \mathbf{u}_k} \tag{2.53}$$

$$H_k = \frac{\partial h(\mathbf{x}_k, \mathbf{v}_k)}{\partial \mathbf{x}_k}|_{\hat{\mathbf{x}}_k} \tag{2.54}$$

$$M_k = \frac{\partial h(\mathbf{x}(t), \mathbf{v}_k)}{\partial \mathbf{v}_k}|_{\hat{\mathbf{x}}_k} \tag{2.55}$$

A disadvantage of the Extended Kalman Filter is that (2.52) - (2.55) must be recalculated at every time step since they are linearized about the previous estimated state vector. This causes the predicted covariance matrix, Kalman gain, and corrected covariance matrix to also be recalculated at each time step, whereas these can be calculated offline for the Kalman Filter (KF). It may also result in an unstable filter if the local linear approximation is not valid. Since the Extended Kalman Filter (EKF) is not optimal, the calculated covariances are not the true values, but are instead approximations [102]. These disadvantages have led to the development of the Unscented Kalman Filter (UKF) as a more computationally efficient and stable estimator for non-linear systems, generating state estimates and covariances without the need to calculate any Jacobians [103]. However, this research will use the Extended Kalman Filter due to its more straightforward implementation as the UAS will fly in accordance with linearized system dynamics.

2.5.3 Ensemble Kalman Filter

In contrast to the Extended Kalman Filter, which uses a recursive calculation of the estimated state vector mean and covariance to represent the posterior belief distribution of each unobservable state, particle filters use an ensemble of N_p samples or particles to represent

the distribution [104], where each particle is drawn using

$$\mathbf{x}_k^i \sim p(\mathbf{x}_k|\mathbf{z}_{1:k}, \mathbf{u}_{1:k}). \tag{2.56}$$

In this type of filter only the ensemble is calculated recursively with the corrected mean and covariance being calculated empirically at each time-step if desired.

The Ensemble Kalman Filter (EnKF), introduced in [105], is a variant of the particle filter in which all distributions are assumed to be Gaussian. The ensemble is formed as $\mathbf{X}_k = \{\mathbf{x}_k^1, \mathbf{x}_k^2, ... \mathbf{x}_k^{N_p}\}$ with increasing accuracy as $N_p \to \infty$. Similar to the Extended Kalman Filter, the Ensemble Kalman Filter includes both prediction and correction steps, generally known as the forecast step and the analysis step.

The EnKF filtering process [106] is initialized by drawing N_p particles from $\mathcal{N}(\mathbf{x}_0, P_0)$ to form the initial ensemble. Each of these particles is propagated during the forecast step

$$\mathbf{x}_k^{i-} = f(\mathbf{x}_{k-1}^{i-}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}^i \tag{2.57}$$

to form X_k^{i-} where $w_k^i \sim \mathcal{N}(0, Q_k)$.

The forecasted estimated state vector is created according to

$$\hat{\mathbf{x}}_k^- = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{x}_k^{i-} \tag{2.58}$$

and the state error vector ensemble is calculated as

$$E_{\mathbf{x}_k^-} = [\mathbf{x}_k^{1-} - \hat{\mathbf{x}}_k^- \dots \mathbf{x}_k^{N-} - \hat{\mathbf{x}}_k^-] \tag{2.59}$$

and the forecasted estimated state vector covariance is calculated as

$$P_k^- = \frac{E_{\mathbf{x}_k^-} E_{\mathbf{x}_k^-}^T}{N_p - 1} \tag{2.60}$$

as the unbiased mean square of the error.

The analysis step of the Ensemble Kalman Filter is very similar to the correction step of the Extended Kalman Filter, with a change in the calculation of the innovation vector. Since there are N_p particles in the ensemble, the innovation vector becomes an innovation ensemble. It is formed by first replicating the available measurement vector $\mathbf{z}_k$ to match the number of particles and adding zero mean Gaussian noise $\mathbf{v}_k^i \sim \mathcal{N}(0, R_k)$ to each measurement vector creating the measurement ensemble Z_k

$$\mathbf{Z}_k = [\mathbf{z}_k + \mathbf{v}_k^1 \dots \mathbf{z}_k + \mathbf{v}_k^N] \tag{2.61}$$

The predicted measurement ensemble is formed using

$$\hat{\mathbf{Z}}_k = H_k \hat{\mathbf{X}}_k^{i-} \tag{2.62}$$

with the innovation ensemble N_k calculated as

$$\mathbf{Z}_k - \hat{\mathbf{Z}}_k. \tag{2.63}$$

The Ensemble Kalman Filter then directly calculates the Kalman Gain

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}. \tag{2.64}$$

The elements of the corrected estimated state ensemble X_k^i are calculated as

$$\mathbf{x}_k^i = \mathbf{x}_k^{i-} + K_k N_k^i \tag{2.65}$$

with the estimated state vector mean calculated as

$$\hat{\mathbf{x}}_k = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{x}_k^i. \tag{2.66}$$

2.5.4 Multi-Sensor Fusion Techniques

In a system with multiple sensors measuring different states or sets of states, measurements must be fused properly to ensure the best filter performance. One method to fuse these disparate measurements is discussed and analyzed in [107]. This method converts the multiple individual measurements into a single augmented measurement vector

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{z}_k^1 \\ \mathbf{z}_k^2 \\ \vdots \\ \mathbf{z}_k^q \end{bmatrix} \tag{2.67}$$

where q is the number of active sensors at time k.

The measurement sensitivity and measurement error covariance matrices, H_k and R_k, are also augmented as

$$H_k = \begin{bmatrix} H_k^1 \\ H_k^2 \\ \vdots \\ H_k^q \end{bmatrix} \tag{2.68}$$

and

$$R_k = \begin{bmatrix} R_k^1 & 0 & \cdots & 0 \\ 0 & R_k^2 & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & R_k^q \end{bmatrix} \tag{2.69}$$

respectively.

The measurement sensitivity matrix H_k is sized $n_z \times n$, where n_z is the total number of available measurements at time k and n is the the total number of states in the system. The measurement covariance matrix R_k is sized $n_z \times n_z$. The number of rows will vary over time if using more than one sensor, assuming sensors have no delay or the lag is not

compensated. Since this method accounts for all measurements at a time step, it gives the most information to the state estimation filter. With all measurements the state estimation filter should yield the most realistic estimates, but do so at an increasing computational cost as the number of sensors increases.

2.5.5 Delayed Measurement Compensation

A common technique to properly account for delayed measurements is known as state augmentation or stochastic cloning [108], [109]. For a measurement with a known delay of m time steps that becomes available at time step k, the state augmentation process keeps a copy of the propagated UAS plant dynamics state vector (in simulation) and estimated state vector at the conclusion of the correction step at time step $k-m$ and appends it to the bottom of the state estimate vector, while also expanding the covariance matrix, state transition matrix, process noise matrix, and measurement sensitivity matrix. The augmented estimated state vector $\breve{\hat{\mathbf{x}}}$ is formed as

$$\breve{\hat{\mathbf{x}}}_{k-m} = \begin{bmatrix} \hat{\mathbf{x}}_{k-m} \\ \hat{\mathbf{x}}_{k-m} \end{bmatrix}. \tag{2.70}$$

In order to properly account for the effect of the augmented states on the covariance matrix, it is augmented and becomes $\breve{P}$

$$\breve{P}_{k-m} = \begin{bmatrix} P_{k-m} & P_{k-m} \\ P_{k-m} & P_{k-m} \end{bmatrix} \tag{2.71}$$

while the state transition matrix is augmented as $\breve{A}$

$$\breve{A}_{k-m} = \begin{bmatrix} A_{k-m} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{2.72}$$

so that the evolving states propagate according to the dynamics of the system while the augmented states are not affected by the prediction step. Since the process noise covariance matrix only affects the evolving states of the system, it is augmented to become $\breve{Q}$

$$\breve{Q}_{k-m} = \begin{bmatrix} Q_{k-m} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{2.73}$$

The measurement sensitivity matrix $\breve{H}$ is also augmented to account for the increased size of the augmented estimated state vector

$$\breve{H}_{k-m} = \begin{bmatrix} H_{k-m} & \mathbf{0} \end{bmatrix}. \tag{2.74}$$

The filter operates normally from time-step $k-m+1$ to $k-1$ using the augmented matrices shown above. During this time, the non-evolving elements of the augmented estimate state vector are corrected using non-delayed measurements, due to the correlation between the evolving states and the augmented states in the covariance matrix. $\breve{H}_k$ is used to calculate the augmented predicted measurement vector $\breve{\mathbf{z}}$. The remainder of the correction step is unchanged. Once the evolving estimated state vector and covariance matrix have been corrected, the augmented states and their associated covariance matrix entries are marginalized out of the system and the process is repeated for the next measurement from the delayed sensor.

2.5.6 Filter Consistency and Accuracy Metrics

Several metrics exist to determine the accuracy and precision of the estimates and covariances generated by state estimation filters. From Bar-Shalom *et al*. [110], these include the average normalized estimation error squared (ANEES), the average normalized innovation squared (ANIS), the sample autocorrelation statistic, and the RMSE. The ANEES is a filter consistency metric that measures the square of the state error as compared to the reported

filter covariance. The ANIS is a second filter consistency metric that shows if the innovations match the calculated innovation covariance. The final filter consistency metric is the sample autocorrelation statistic, which determines if the innovations are white (uncorrelated). The RMSE is used to determine filter accuracy over time by measuring the average absolute relative error at each time step over the complete set of Monte Carlo trials.

2.5.6.1 Average Normalized Estimation Error Squared

The normalized estimation error squared (NEES), ϵ, is

$$\epsilon_k = \mathbf{e}_k^T P_k^{-1} \mathbf{e}_k \tag{2.75}$$

a squared Mahalanobis distance calculated at each time step as a measure of the accuracy of the calculated filter error covariance matrix as compared to the state error vector. To calculate the state error vector, the propagated UAS plant dynamics states of the system or measured ground truth data must be known.

When the NEES value is chi-squared distributed with n degrees of freedom, the filter is said to be consistent. Since the filter may be consistent for one Monte Carlo trial, but not consistent for a large number of trials, the average NEES (ANEES) value is calculated as

$$\bar{\epsilon}_k = \frac{1}{MC} \sum_{mc=1}^{MC} (\epsilon_k^{mc}). \tag{2.76}$$

In (2.76), MC is the number of Monte Carlo trials. The (2.76) distribution must be chi-squared with $n * MC$ degrees of freedom for the filter to be consistent. If the ANEES value exceeds the upper chi-squared distribution limit for more than 5% of the trajectory, this indicates that the estimation error is increasing without the expected increase in estimated covariance. This trend is known as an optimistic filter. If the ANEES value shrinks below the lower chi squared distribution limit, the filter is over predicting the error, which is known as a pessimistic filter. In either case, in a simulation environment, the process noise

can be tuned to bring the ANEES value back within the chi squared bounds.

2.5.6.2 Average Normalized Innovation Squared

Similar to the NEES, the normalized innovation squared (NIS) ϵ_ν is

$$\epsilon_{\nu_k} = \nu_k^T W_k^{-1} \nu_k \tag{2.77}$$

a squared Mahalanobis distance calculated at each time step as a measure of the accuracy of the filter-calculated innovation covariance matrix as compared to the actual innovations, generated using both the sensors and filter. Having an accurate innovation covariance matrix ensures that the correction step in the filter weights the measurements appropriately. Since it does not require knowledge of the propagated UAS plant dynamics states of the system or ground truth measurements, it can be calculated in real-time using incoming measurements and the filter.

The consistency criteria for the NIS is that it is chi-squared with n_z degrees of freedom. The number of degrees of freedom will change at each time-step for systems with sensors that have different sampling rates so the consistency bounds will also change as the number of available measurements changes. If a large number of Monte Carlo trials is used, the average NIS (ANIS) value is calculated as

$$\bar{\epsilon}_{\nu_k} = \frac{1}{MC} \sum_{mc=1}^{MC} (\epsilon_{\nu_k}^{mc}). \tag{2.78}$$

As with the ANEES, $\bar{\epsilon}_{\nu_k}$ must be chi-squared with $n_z * MC$ degrees of freedom for the filter to be consistent. If the ANIS is less than the lower acceptance region bound or higher than the upper acceptance region bound more than 5% of the values, the filter is inconsistent. The filter can be tuned in the real environment using process noise and measurement noise to bring the ANIS trajectory back within the bounds.

2.5.6.3 Sample Autocorrelation Statistic

The final test for filter consistency is the sample autocorrelation statistic, $\hat{\rho}_{k,j}$, calculated as

$$\hat{\rho}_{(k,j)_q} = \frac{1}{\sqrt{n_{z_q}}} \sum_{mc=1}^{MC} \nu_{k_q}^{mc^T} \left[\sum_{mc=1}^{MC} \nu_{k_q}^{mc} \nu_{k_q}^{mc^T} \right]^{-\frac{1}{2}} \left[\sum_{mc=1}^{MC} \nu_{j_q}^{mc} \nu_{j_q}^{mc^T} \right]^{-\frac{1}{2}} \nu_{j_q}^{mc} \tag{2.79}$$

to verify that the innovation sequences are white. This test can be done per measurement or for the entire innovation vector per sensor. Here the statistic is calculated for the entire innovation vector per sensor where n_{z_q} is the number of measurements for sensor q. It is calculated using the innovations at consecutive time steps with available measurements per sensor, meaning that the difference in time-steps k and j is generally the sampling period.

To meet the filter consistency criteria of the innovation sequences being uncorrelated, the sample autocorrelation statistic values, where $k = j$, should have a mean of zero with a variance equal to $\frac{1}{MC}$. This requires 95% of the values to fall between $-r$ and r where $r = \frac{1.96}{\sqrt{MC}}$.

2.5.6.4 Root-Mean-Square Error

The RMSE is

$$RMSE_k = \sqrt{\frac{1}{MC} \sum_{mc=1}^{MC} e_k^2} \tag{2.80}$$

an $n \times 1$ vector at each time step quantifying the absolute accuracy of the estimated value for each state [111]. It can increase or decrease based on several factors including the amount of process noise in the system as well as the availability and accuracy of measurements.

CHAPTER 3

Development of the UAS GNC Simulation

This chapter describes the UAS guidance, navigation, and control (GNC) simulation used in this dissertation, shown in Figure 3.1. The systems-level simulation includes both the GNC object-oriented class structures and the urban environment generation software. The fixed-wing UAS model from Chapter 2 is linearized about multiple trim states to facilitate the design of an LQR controller. A real-time guidance strategy of obstacle avoidance is presented in Appendix G. While alone not a particularly novel research contribution, this work was a necessary building block for all the simulation-based analysis.

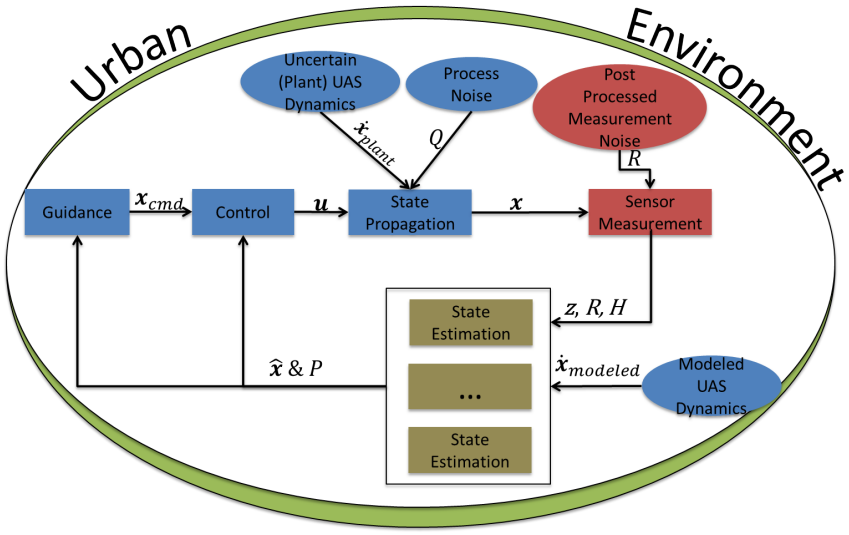


Figure 3.1: UAS GNC simulation system diagram.

3.1 Simulation Flow

The simulation shown in Figure 3.1 is comprised of guidance, control, state propagation, sensor measurement, and state estimation blocks with an overarching urban environment affecting sensor accuracy and availability. Each has inputs from the previous block in the simulation as a minimum. The guidance command vector $\mathbf{x}_{cmd}$ is generated by the guidance block based on the estimated state vector $\hat{\mathbf{x}}$ of the UAS. This command vector is then passed into the control block along with the estimated state vector to generate the control vector $\mathbf{u}$. The control vector and the UAS plant model $\dot{\mathbf{x}}_{plant}$ are passed into the state propagation block as an input, where the model can be exact or have uncertainty in specified coefficients and constants.

The unknown state vector $\mathbf{x}$ from the state propagation block is passed into the sensor measurement block along with post-processed sensor measurement noise values to form the measurement noise covariance matrix R. From the sensor measurement block, the measurement vector $\mathbf{z}$, the measurement sensitivity matrix H, and the measurement noise covariance matrix are passed into the state estimation block. The exact UAS model $\dot{\mathbf{x}}_{modeled}$ is also passed into the state estimation block. The estimated state vector is then calculated in the state estimation block as well as the estimated state covariance matrix P completing the simulation loop.

3.2 Simulation Code Outline

Creating a realistic UAS GNC simulation requires a methodical system-level design with intuitive data structures. The goal is to produce a framework that is easy to use and to customize. Object-oriented programming practices are used to organize the data necessary to execute the simulation and output results into structures/objects. All data classes and subclasses are shown in italics when referenced for the remainder of the dissertation. The five data classes defined are *simulation*, *urban environment*, *UAS dynamics*, *sensor*, and

estimator. As shown in Figure 3.1, each of the ovals represents user-defined or simulation-generated data, while each block represents a process or function that uses data from one or more of these classes as inputs to produce the specified outputs. Each is color-coded to show when each data class is called. Orange represents *simulation* class data, green represents *urban environment* class data, blue represents *UAS dynamics* class data, red represents *sensor* class data, and brown represents *estimator* class data. Multiple *estimator* blocks are shown to highlight customization available to the user when selecting the appropriate filter or filters. The *urban environment* structure encapsulates the entire cycle to represent its effects on all aspects of the simulation.

Figure 3.2 summarizes the five data classes using the above color-coding along with their second-tier subclasses. Each subclass is designated into one of three types: user-specified initialization values (darkest colors), randomly-generated parameters (medium colors), and data updated at each time step (light colors).

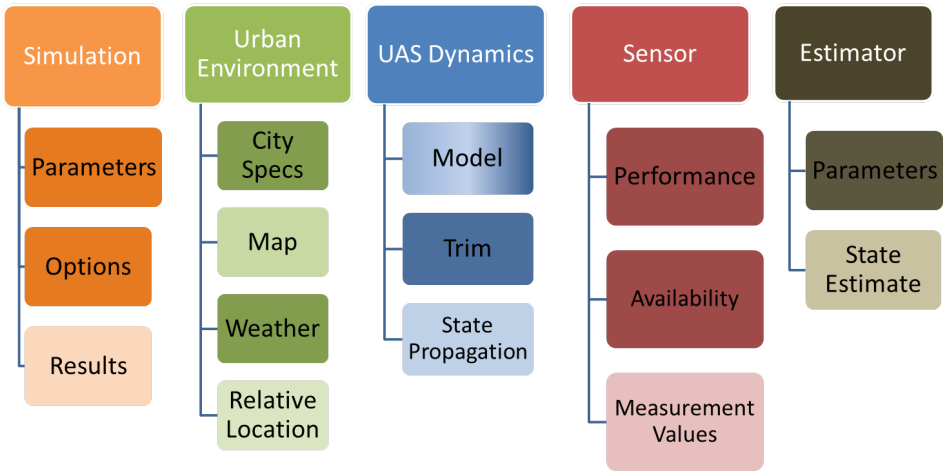


Figure 3.2: Simulation data classes with first level of subclasses.

Each class is described below along with a diagram and description of all lower-level subclasses. Selected properties are included in the urban environment class diagram in italics to illustrate how the properties fit into the overall class/subclass structure. Full prop-

erty details for all subclasses including names, allowable values, dimensions, and units are available in Appendix B.

3.2.1 Simulation Data Class

The *simulation* class, shown in Figure 3.3, defines overall parameters, flags, and storage for details needed to generate simulation outputs. It consists of three second-level subclasses: *parameters*, *options*, and *results*.

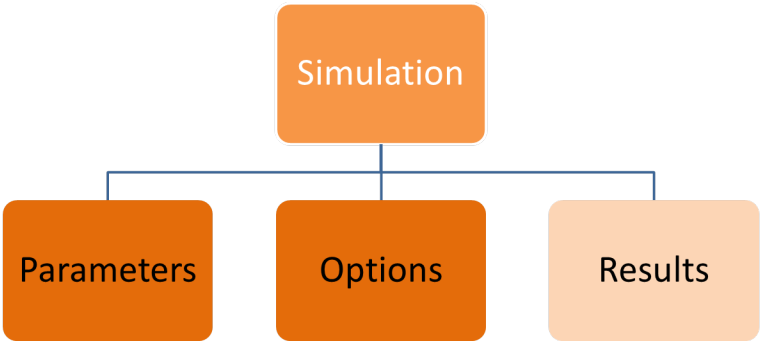


Figure 3.3: Simulation class diagram with subclasses.

Simulation – Parameters The *parameters* subclass contains mostly user-defined initialization properties. Its properties store the temporal execution values for the simulation including the length of a time step, the length of the simulation, number of time steps, and the CPU time to execute each run. It also includes the initial position of the UAS and the number of Monte Carlo trials.

Simulation – Options The *options* subclass contains user-defined initialization properties that set flags to trigger different algorithms and on/off flags during the simulation execution. The algorithm flags are used to select the environment type, the feedback type, and the estimation filter type. There are on/off flags for saving different data sets, sensor measurement delay compensation, and sensor self-accuracy determination.

Simulation – Results The *results* subclass contains a lower-level auto-correlation subclass and properties updated at each time step that are used in the post-simulation analysis software. These properties serve as storage for the mean value and standard deviation values of several simulation outputs averaged over all Monte Carlo trials for each time step. They include the state vector, control vector, estimated state vector, estimated state covariance matrix, state error vector, relative location classification value vector, the average normalized estimation error squared value, the average normalized innovations squared, and the root mean squared error vector. This subclass also includes a property for the mean cycle time average over all Monte Carlo trials.

Simulation – Results – Autocorrelation The *autocorrelation* subclass contains properties to store the autocorrelation statistic for each sensor each time it is used for a measurement as well as the time-steps which the measurements were taken.

3.2.2 Urban Environment Data Class

The *urbanenvironment* class, shown in Figure 3.4, defines the area of operations for the UAS including local environment infrastructure, weather information, and provides storage for a novel position categorization method. It is comprised of four second-level subclassses including *cityspecifications*, *map*, *weather*, and *relativelocation*. The *map* subclass has three levels of subclasses under it, including *canyon*, *building*, *obstacles*, *antennas*, and *skybridges* (last two not shown), but does not have any properties within its own subclass. The obstacle subclass also does not have any properties within its own subclass.

Urban Environment – City Specifications The *cityspecifications* subclass contains user-defined initialization properties to generate the urban landscape. These properties fall under three general categories within the subclass. The first category defines the size and spacing of the grid-based city. These properties are the number of blocks, the number

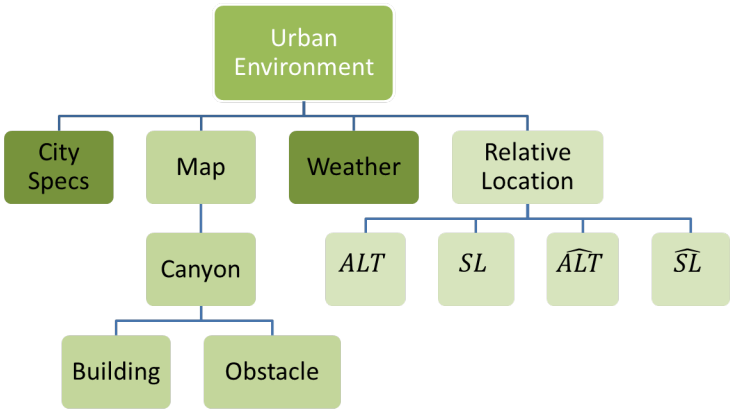


Figure 3.4: Urban environment class diagram with subclasses.

of buildings per block, the spacing between buildings, the width of the street separating the buildings, and any blocks that may contain open space. The second allows the user to customize building sizes by specifying minimum and maximum rectangular footprint dimensions and heights. The third category allows the user to choose obstacle sizes and quantities. The antenna obstacle properties include antenna length and radius minimum and maximum sizes, and the minimum height of a building to have an antenna on its roof. The skybridge obstacle properties allow the user to specify the number of skybridges along with their altitudes, heights from top to bottom, and the buildings which anchor them on either side of the street.

Urban Environment – Map – Canyon The *canyon* subclass properties are randomly generated using inputs from the city specifications subclass to define each canyon within the city. The number of canyons (thus instances of the canyon subclass) is equal to the number of blocks in the city. Its properties are the two-dimensional coordinates of each of the canyon's four corners and the height of the shortest and tallest buildings within the canyon.

Urban Environment – Map – Canyon – Building The *building* subclass properties are randomly generated using building inputs from the *city specifications* subclass. The

number of buildings (and instances of this class) is double the product of the number of blocks and number of buildings per block to account for buildings on both sides of the street. Its properties are the two-dimensional coordinates of each building's four corners as well as their heights and a unique identification number.

Urban Environment – Map – Canyon – Obstacle – Antenna *antenna* subclass properties are randomly generated using inputs from the city specifications subclass. The number of antennas (and instances of this class) is equal to the number of blocks since it is assumed there is one antenna placed on the tallest building on each block (where the tallest building is above the minimum height for antenna). Its properties are the two-dimensional coordinates of the antenna, its base height, top height, and radius.

Urban Environment – Map – Canyon – Obstacle – Skybridge *skybridge* subclass properties are also generated using the skybridge inputs from the city specifications subclass. The number of skybridges (and instances of this class) is defined in the city specifications subclass. Its properties are the four two-dimensional corner coordinates, base altitude, and top altitude.

Urban Environment – Weather The *weather* subclass contains user-defined initialization properties representing current weather conditions. These properties include a current wind vector, temperature, and visibility. The utilization of these properties for UAS navigation is left to future work.

Urban Environment – Relative Location The *relativelocation* subclass properties are updated at every time step. This subclass is instantiated twice, once for propagated UAS plant dynamics three-dimensional inertial position and again for estimated inertial position. Separate properties define relative altitude and street-level position of the UAS as well as the number of the current canyon in which it is located.

3.2.3 UAS Dynamics Data Class

The *UAS dynamics* class, shown in Figure 3.5, defines the parameters necessary to trim and simulate flight for the six degree of freedom UAS model. It is divided into three second-level subclasses: *model*, *trim*, and *state propagation*. The *model* subclass has properties of its own. The model subclass contains properties, and also one subclass: *parameters*. The *statepropagation* subclass contains properties and one subclass: *guidance*. The trim subclass contains no properties, but two subclasses: *parameters* and *truestates*.

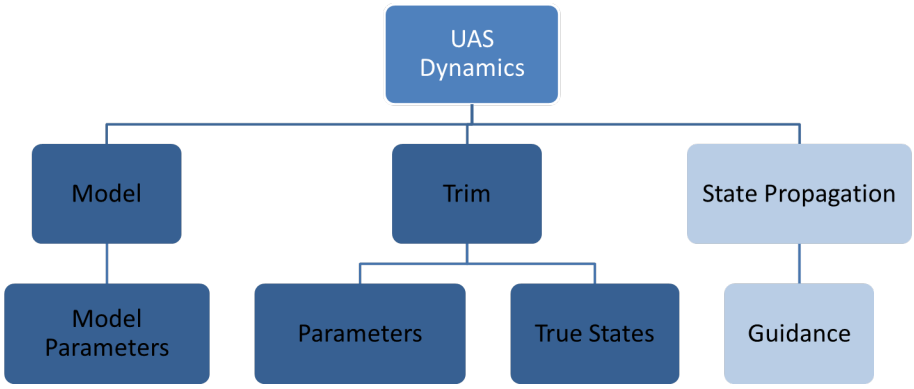


Figure 3.5: UAS dynamics and control class diagram with subclasses.

UAS Dynamics – Model The *model* subclass contains user-defined initialization properties. These properties include the dimensions of both the state vector and the control vector, the process noise matrix, the process noise matrix scaling factor, and physical limits or saturation constraints on control inputs.

UAS Dynamics – Model – Parameters The *parameters* subclass contains specific UAS aerodynamic coefficients, mass and inertia properties, and engine parameters. It is instantiated once for the UAS plant dynamics propagation model and once for the state estimation filter dynamics model.

UAS Dynamics – Trim – Parameters The *parameters* subclass contains user-defined initialization properties necessary to calculate UAS trim conditions. These properties include the desired trim airspeed, altitude, flight path angle, and heading angle, as well as the trim conditions such as in (2.24).

UAS Dynamics – Trim – True States The *truestates* subclass contains indirect user-defined initialization properties specifying UAS trim or equilibrium conditions. These properties include the calculated trim state vector as well as the resulting squared two-norm value of the residual. This residual shows the amount of error in the actual trim conditions relative to the desired trim conditions.

UAS Dynamics – State Propagation The *state propagation* subclass contains the propgated UAS plant dynamics state vector and control vector updated at every time step.

UAS Dynamics – State Propagation – Guidance The *guidance* subclass object is also updated at every time step. The first subclass property is the current three-dimensional obstacle avoidance waypoint. The second is the user-desired vertical clearance between the UAS and any obstacle in its current path. The final property is the UAS commanded state vector to be output per Figure 3.1.

3.2.4 Sensor Data Class

The *sensor* class, shown in Figure 3.6, defines performance characteristics of all modeled sensors as well as providing storage for the measurement quantities. It is split into three second-level subclasses: *performance*, *availability*, and *measurementvalues*.

Sensor – Performance The *performance* subclass contains user-defined initialization properties characterizing each modeled sensor. The subclass is instantiated once for each sensor. Properties includes two main categories. The first is the sensor temporal properties

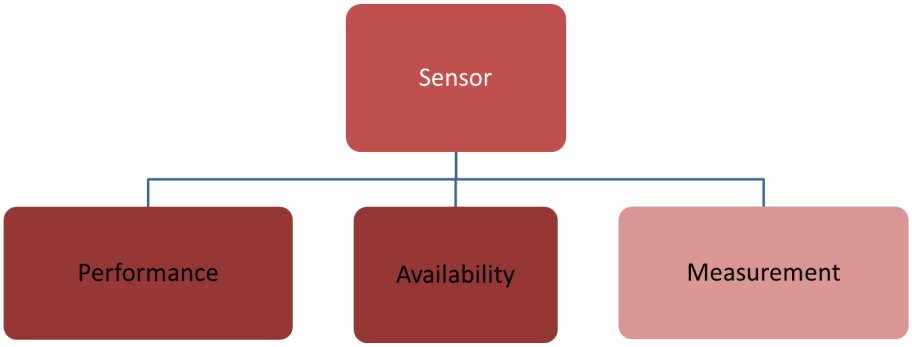


Figure 3.6: Sensor class diagram with subclasses.

category, including how frequently the sensor can take measurements, any delay in the measurement becoming available, and when the first measurement can be taken. The second is the sensor measurement generation properties categories, including a flag to determine if the sensor is in use, the availability of the sensor based on the environment, the sensor's orientation with respect to the buildings, the states that can be measured, the accuracy of these measurements, and a measurement sensitivity vector for each state being measured.

Sensor – Availability The *availability* subclass contains a list of sensors that are in use for a given simulation and a second list of sensors that provide delayed measurements. The availability of these sensors during the simulation is subject to sensor performance constraints described in the previous paragraph and urban environment geometry, to be described in future chapters.

Sensor – Measurement Values The *meaurementvalues* subclass contains properties that are updated for each sensor when a measurement becomes available for that particular sensor, based on sensor performance constraints and the urban environment geometry. It is instantiated for each sensor in use. The properties include the time stamp of when a measurement was taken for each sensor and the innovation value for the measurement to be used when calculating filter consistency metrics.

3.2.5 Estimator Data Class

The *estimator* class, shown in Figure 3.7, provides storage for quantities used for the state estimation filters. It has two second-level subclasses: *parameters* and *stateestimate*. The *stateestimate* subclass contains properties as well as two second level subclasses: *ExtendedKalmanFilter* and *ParticleFilter*.

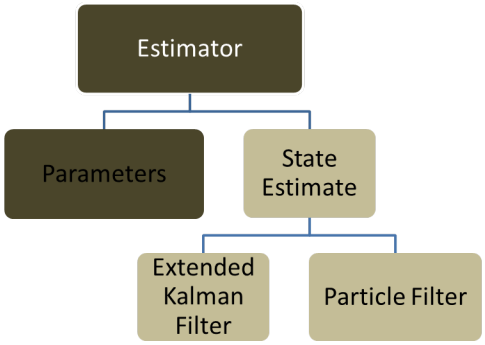


Figure 3.7: Estimator class diagram with subclasses.

Estimator – Parameters The *parameters* subclass contains user-defined initialization properties for the state estimation filters. These properties include the initial estimated state vector covariance matrix for all filters. For particle filter option, the number of particles used to represent the posterior distribution, the optimal jittering bandwidth, and the resampling threshold percentage of effective particles are also defined.

Estimator – State Estimate Along with filter-specific subclasses, the *stateestimate* subclass contains properties updated at every time step. This includes the estimated state vector, estimated state vector covariance matrix as well as the error vector and the normalized estimation error squared (NEES).

Estimator – State Estimate – Extended Kalman Filter The *EKF* subclass contains properties for the storage of the normalized innovation squared (NIS) and dimension of the measurement vector at each time step used in filter consistency analysis.

Estimator – State Estimate – Particle Filter The *particle filter* subclass contains properties to store the multivariate particles and their respective weights at each time step. The weights are only updated when a non-Kalman Filter particle filter is used, including the Sampling Importance Resampling Filter or the Regularized Particle Filter [112].

3.3 Urban Environment Development

To simulate a true urban canyon environment, urban environment generation software was developed to generate buildings based on user input. The design inputs can be based on real urban databases such as the Primary Land Use Tax-lot Output for New York City [113] or a user-defined fictitious city. Table 3.1 shows representative inputs used to generate urban environments for the simulations presented in this dissertation.

Table 3.1: Urban environment design specifications.

Urban Landscape Parameter	Value	Notes
Number of Blocks	5	
Buildings per Block	4	Number of buildings on each side of street
Open Space	3 – *East*	Modeled as buildings with zero height
Street Width	20 meters	Includes six lanes for traffic and sidewalks
Building Shape	Cuboid	
Building Height	40 meters-125 meters	Can be random or set for each building
Building Length	25 meters-40 meters	
Building Width	25 meters-40 meters	
Building Spacing	7.5 meters	
Antennas per block	1	Rising from roof of tallest building per block

When these specifications are provided as inputs to the urban environment generation software, the outputs produced are shown in Figure 3.8 and Figure 3.9 along with canyon maps and building maps.

The main data stored in canyon maps are the four canyon corners as defined by the extreme corners of the building pairs on both the southern edge of the block and the northern edge of the block. The building maps store building corner coordinates, building heights, and building numbers for each building. The buildings are numbered from West to East and

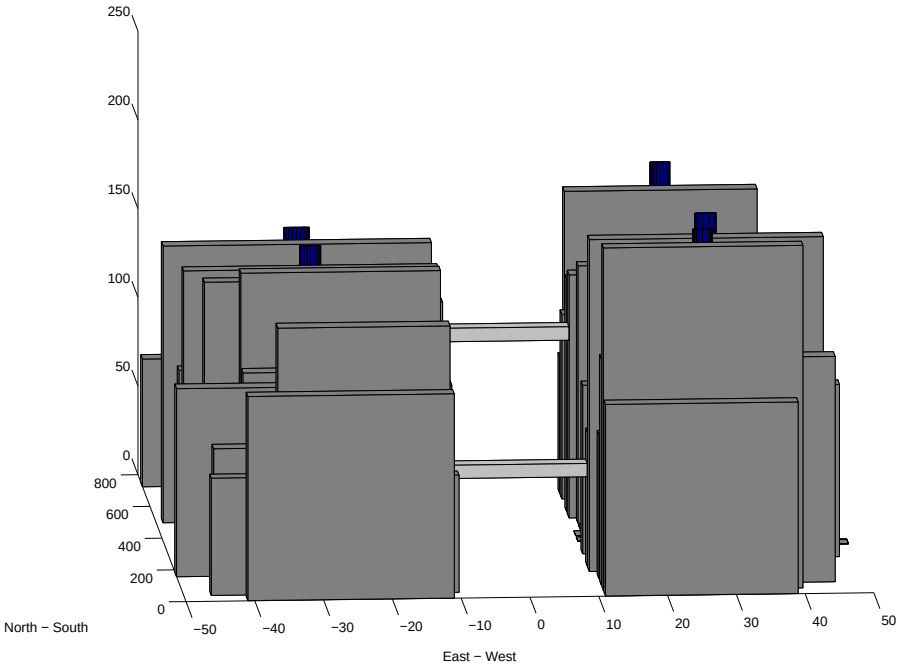


Figure 3.8: Example urban environment with skybridges.

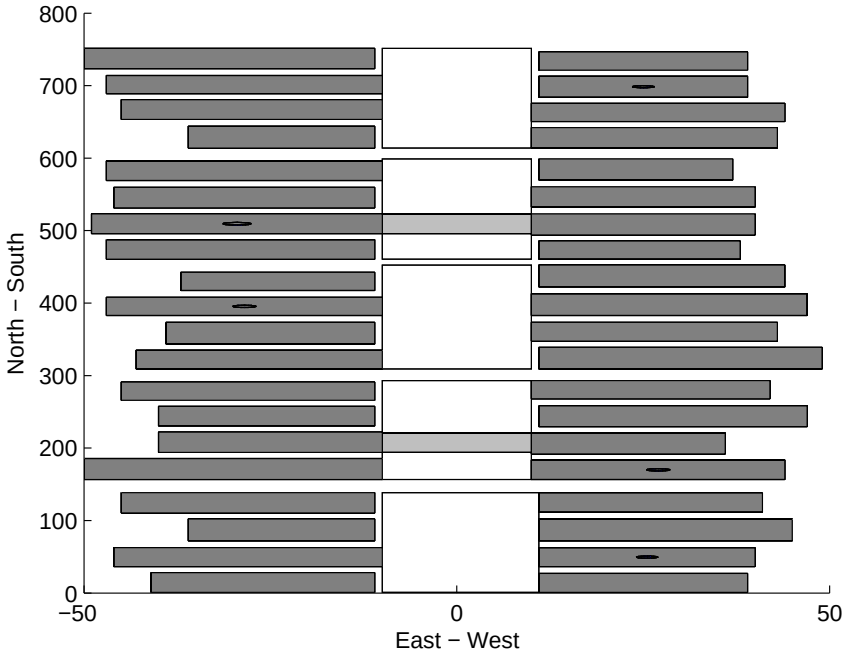


Figure 3.9: Example urban landscape with canyon boxes.

then South to North so that the southwesternmost building in the environment is assigned the number '1,' the building adjacent to the East is assigned number '2,' and so on. This building environment plays a critical role in determining the availability and accuracy of navigation sensor measurements.

3.4 UAS Model

The fixed-wing UAS aerodynamic model used for this research was taken directly from [42] and follows the form described previously in Chapter 2. Table 3.2 lists the physical parameters of the UAS pictured in Figure 3.10 and used for this research. The physical inertia matrix and aerodynamic coefficients for this UAS are listed in Appendix C. For the unmatched model, uncertainty percentages between 10% and 30% percent are applied to certain aerodynamic coefficients and the physical inertia matrix entries as will be described in Chapter 6. These parameters are indicated with an asterisk in Appendix C.

Table 3.2: UAS model physical parameters taken from [42].

Parameter	Value
Aircraft Mass, m	28 kilograms
Wing Surface Area, S	1.8 meters
Mean Aerodynamic Chord, $\bar{c}$	0.58 meters
Wing Span, b	3.1 meters
Propeller Diameter, D	0.79 meters
Engine Time Constant, τ_n	0.4 seconds

In this model, the aileron δ_a, elevator δ_e, and rudder δ_r, input deflection limits are normalized to the range $\{-1,\ 1\}$ and the engine revolutions/minute range is set to $\{1800,\ 6000\}$. The lower revolutions/minute limit is greater than zero because the UAS is gas-powered. Both ranges are necessary in the simulation to ensure the UAS autopilot is commanding only realistic control inputs.

The time-invariant UAS plant dynamics are propagated using the Runge-Kutta fourth-order numerical integration method [115] coded into simulation. This method generates

Figure 3.10: Photo of UAS used in model, courtesy of ETH [114].

an approximate solution at discrete points with fourth-order accuracy. For a given initial conditions vector $\mathbf{x}_0$, the state vector $\mathbf{x}$ is propagated forward in time to each subsequent discrete point using

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\bar{h}}{6}\left(\bar{\mathbf{k}}_1 + 2\bar{\mathbf{k}}_2 + 2\bar{\mathbf{k}}_3 + \bar{\mathbf{k}}_4\right) \tag{3.1}$$

where the step size $\bar{h}$ is calculated as

$$\bar{h} = t_{k+1} - t_k \tag{3.2}$$

and is set to 0.01 seconds for all simulations in this dissertation. The constant $\bar{\mathbf{k}}_1$ is calculated as

$$\bar{\mathbf{k}}_1 = f(\mathbf{x}_k, \mathbf{u}_k) \tag{3.3}$$

$\bar{\mathbf{k}}_2$ is calculated as

$$\bar{\mathbf{k}}_2 = f\left(\mathbf{x}_k + \frac{\bar{h}}{2}\bar{\mathbf{k}}_1, \mathbf{u}_k\right) \tag{3.4}$$

$\bar{\mathbf{k}}_3$ is calculated as

$$\bar{\mathbf{k}}_3 = f\left(\mathbf{x}_k + \frac{\bar{h}}{2}\bar{\mathbf{k}}_2, \mathbf{u}_k\right) \tag{3.5}$$

$\bar{\mathbf{k}}_4$ is calculated as

$$\bar{\mathbf{k}}_4 = f\left(\mathbf{x}_k + \bar{h}\bar{\mathbf{k}}_3, \mathbf{u}_k\right). \tag{3.6}$$

3.5 Linearization of the UAS Equations of Motion at Nominal Flight Conditions

To linearize the UAS equations of motion at given trim conditions, these conditions must first be calculated, then the state transition Jacobian matrix and control Jacobian matrix must be generated using these conditions. Once the model is linearized, the linear response must be checked to show that it accurately reflects the nonlinear response of the aircraft to control inputs.

3.5.1 Calculating Trim Conditions

When linearizing UAS equations of motion, a straight and level trim condition (flight path angle $\gamma = 0$ radians) is used as the baseline. The selected trim airspeed and altitude were chosen to be 30 meters per second and 50 meters respectively. The airspeed was selected since it was used for much of the analysis in [42]. Since the altitude has a minimal effect on the trim at a constant airspeed, it was selected to be sufficiently high, but still sufficiently lower than the tallest buildings in most major cities. Table 3.3 shows the trim values of all state variables and control inputs except the two inertial horizontal position coordinates, x_N and x_E. The table is divided into the longitudinal states and control inputs in the left column and the lateral states and control inputs in the right column where the longitudinal states are h, V_T, α, θ, q, δe, F_T and the lateral states are β, ϕ, ψ, p, r, δa, δr, respectively. The longitudinal inertial position, x_N, can be arbitrarily initialized for simulation, depending on the desired location within the environment. The altitude h is shown to four decimal places to demonstrate that small imprecision is introduced when using a numerical solver. The angle of attack, α, and pitch angle, θ, trimmed to equal values near 0.09 radians, and the

elevator, δ_e, trimmed to roughly 0.02 down, as indicated by the negative sign. The lateral inertial position, x_E, is initialized as a user input based on the desired position between the buildings in the canyon. All other lateral states and control inputs are approximately zero as expected in steady, level flight conditions. These states did not trim exactly to zero due to the imprecision in the UAS model and the use of a numerical solver.

Table 3.3: UAS Trim Conditions for V_T = 30 meters/second, h = 50 meters, γ = 0 degrees with a residual norm equal to 5.84×10^{-26}.

h	50.00 meters	β	0.0004 radians
V_T	30 meters/ second	ϕ	0.0006 radians
α	0.0887 radians	ψ	−0.0003 radians
θ	0.0887 radians	p	0 radians/second
q	0 radians/second	r	0 radians/second
δ_e	−0.0235 radians	δ_a	0.0001 radians
F_T	35.40 N	δ_r	−0.0006 radians

3.5.2 Linearizing about the Trim Conditions

With known trim conditions, the linearized state transition matrix, A, and control matrix, B, are generated by substituting the aerodynamic model values and trim conditions for the state and control inputs into the symbolic state transition Jacobian and control Jacobian derived in Section 2.3.1. This process yields a state transition matrix with values shown in (3.7) and a control matrix shown in (3.8). For these matrices the state vector was reordered to $[x_N\, h\, V_T\, \alpha\, \theta\, q\, x_E\, \beta\, \phi\, \psi\, p\, r]^T$, and the control vector reordered to $[\delta_e\, F_T\, \delta_a\, \delta_r]^T$ to separate the longitudinal and lateral states and control inputs. This allows the small but non-zero coupling between the longitudinal and lateral states to be seen in the upper right quadrant (rows 1 – 6 / columns 7 – 12) and lower left quadrant (rows 7 – 12 / columns 1 – 6) of

$$A_{lin} = \left(\begin{array}{cccccc|cccccc}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -30 & 30 & 0 & 0 & -0.0173 & -0.0101 & 0 & 0 & 0 \\
0 & 0 & -0.0840 & -0.9226 & -9.81 & 0 & 0 & -0.0058 & 0.0033 & 0 & 0 & 0 \\
0 & 0 & -0.0216 & -3.8590 & 0 & 1 & 0 & 0 & -0.0002 & 0 & -0.0004 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -0.0006 \\
0 & 0 & 0 & -4.7412 & 0 & -4.989 & 0 & 0 & 0 & 0 & 0 & 0 \\
- & - & - & - & - & - & - & - & - & - & - & - \\
0 & 0 & 0 & -0.0173 & 0 & 0 & 0 & 30 & -2.6561 & 30 & 0 & 0 \\
0 & 0 & 0 & -0.0001 & 0.0001 & 0 & 0 & -0.4871 & 0.3257 & 0 & 0.0885 & 0.9961 \\
0 & 0 & 0 & 0 & 0 & 0.0001 & 0 & 0 & 0 & 0 & 1 & 0.0889 \\
0 & 0 & 0 & 0 & 0 & 0.0006 & 0 & 0 & 0 & 0 & 0 & 1.0039 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -20.295 & 0 & 0 & -11.955 & 2.8374 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 24.3630 & 0 & 0 & 0.5289 & -3.117
\end{array}\right). \quad (3.7)$$

These values are approximately one order of magnitude smaller than any of the uncoupled values. For small deviations from trim, the linearized system can be approximated as decoupled as long as the coupling terms are small by comparison when propagating the states. There is no coupling between the lateral inputs and the longitudinal equations of motion or the longitudinal inputs and lateral equations of motion in

$$B_{lin} = \left(\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0.0356 & 0 & 0 \\
0 & -0.0001 & 0 & 0 \\
0 & 0 & 0 & 0 \\
28.6110 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 81.8280 & -2.8476 \\
0 & 0 & -3.6207 & 14.58
\end{array}\right) \quad (3.8)$$

as shown by the zeros in rows $1-6$ / columns $3-4$ and rows $7-12$ / columns $1-2$.

3.5.3 Verifying the Linear Model

Figures 3.11 and 3.12 show the linear and nonlinear response to small elevator and small thrust inputs, respectively, for the longitudinal UAS states. All linearized states track quite closely to the nonlinear states over the ten second interval shown, verifying the accuracy of the linear model for small elevator perturbations.

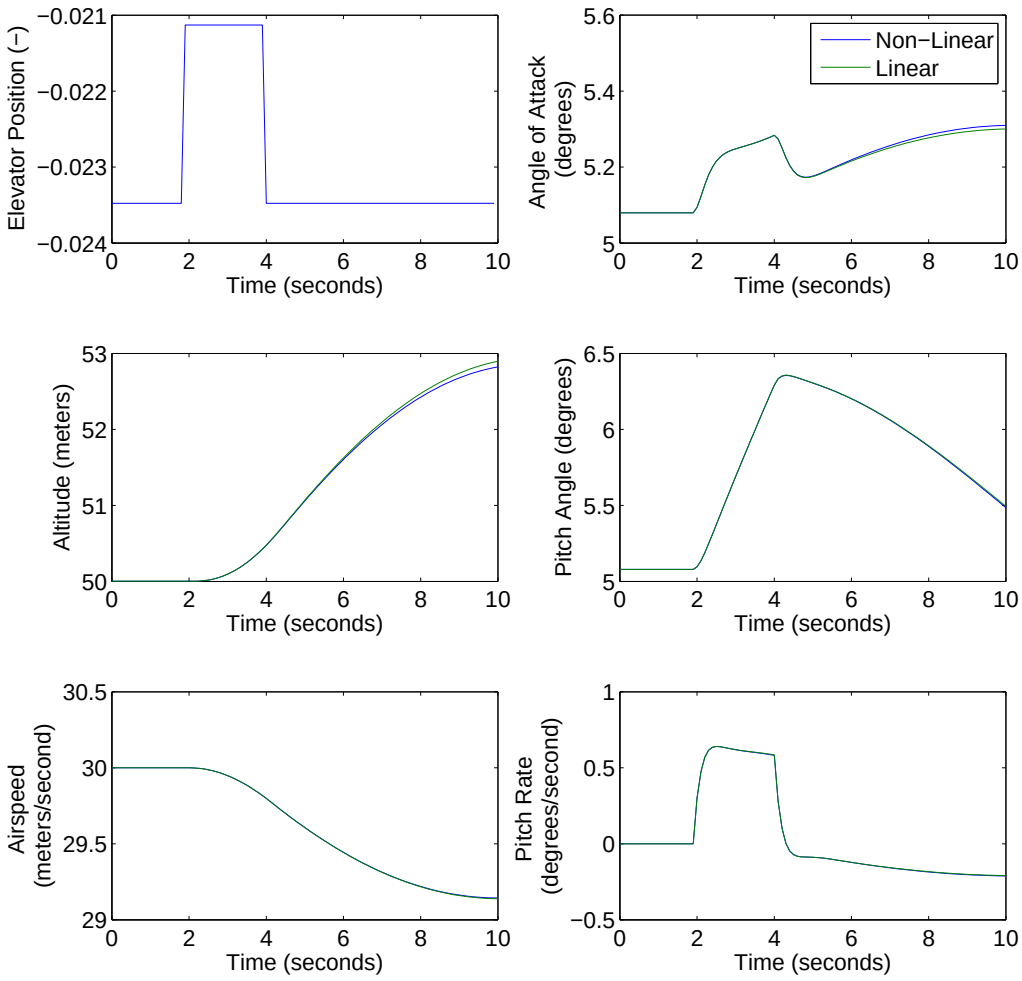


Figure 3.11: Comparison of nonlinear and linear responses of longitudinal states for a small elevator input.

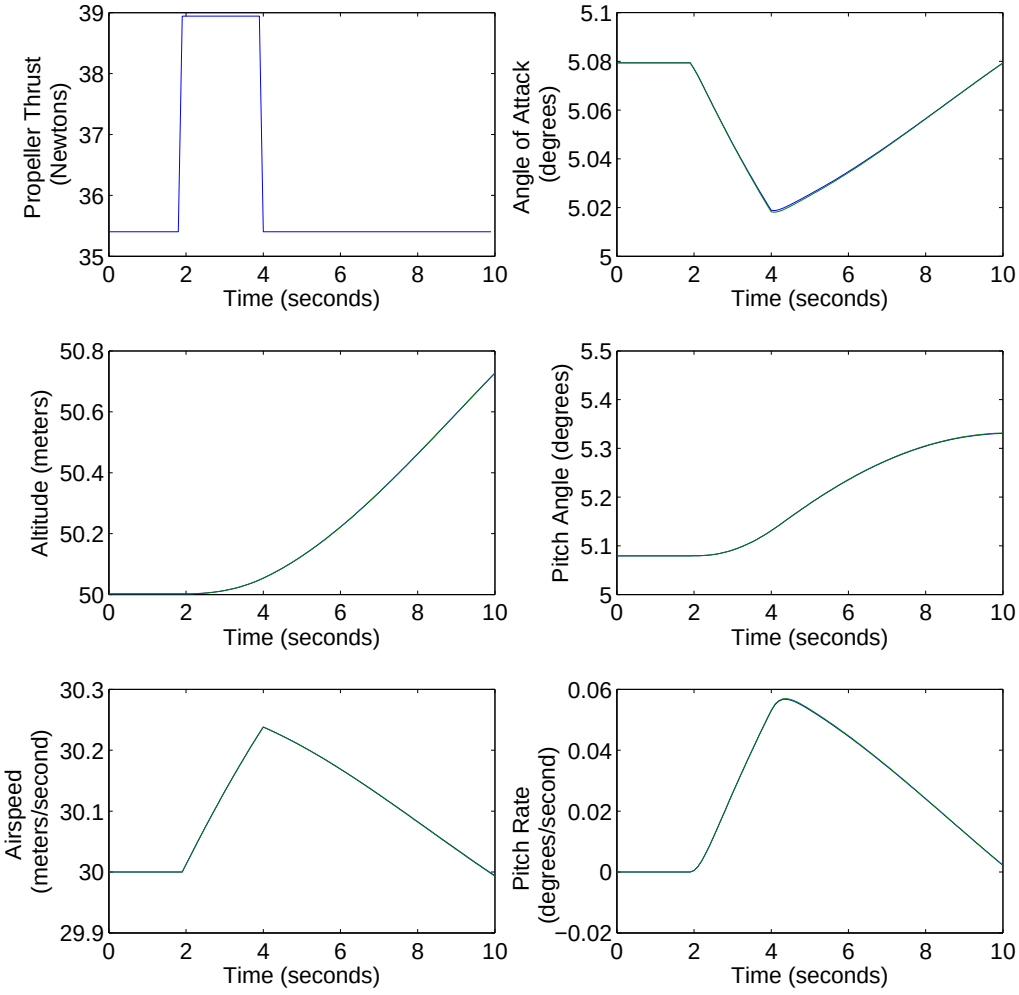


Figure 3.12: Comparison of nonlinear and linear responses of longitudinal states for a small thrust input.

Figure 3.13 shows the nonlinear and linear responses to a small aileron input for lateral position, angle of sideslip, roll angle, and roll rate. The roll rate response for the linear model tracks nearly identically to the non-linear model. Lateral position, angle of sideslip, and roll angle linear responses track the non-linear responses until roughly 5 *seconds*, at which point the linear response begins to diverge. The reason for these deviations is that the non-linear response accounts for how the increasing and decreasing roll angle and sideslip

angle affect the states of the system, while the linear model linearizes about trim values of both angles very close to zero radians. Once the roll angle and sideslip angle values increase or decrease from their trim values, the coefficients based on these trim values do not accurately emulate the non-linear model. These coefficients include $\frac{\partial \dot{x_E}}{\partial \beta}$, $\frac{\partial \dot{x_E}}{\partial \phi}$, $\frac{\partial \dot{\beta}}{\partial \theta}$, $\frac{\partial \dot{\beta}}{\partial \beta}$, $\frac{\partial \dot{\beta}}{\partial \phi}$, $\frac{\partial \dot{\phi}}{\partial q}$, $\frac{\partial \dot{\phi}}{\partial r}$. However, this behavior, consistent with [42], shows that for small perturbations from trim over small time periods, this linearized model response is acceptable.

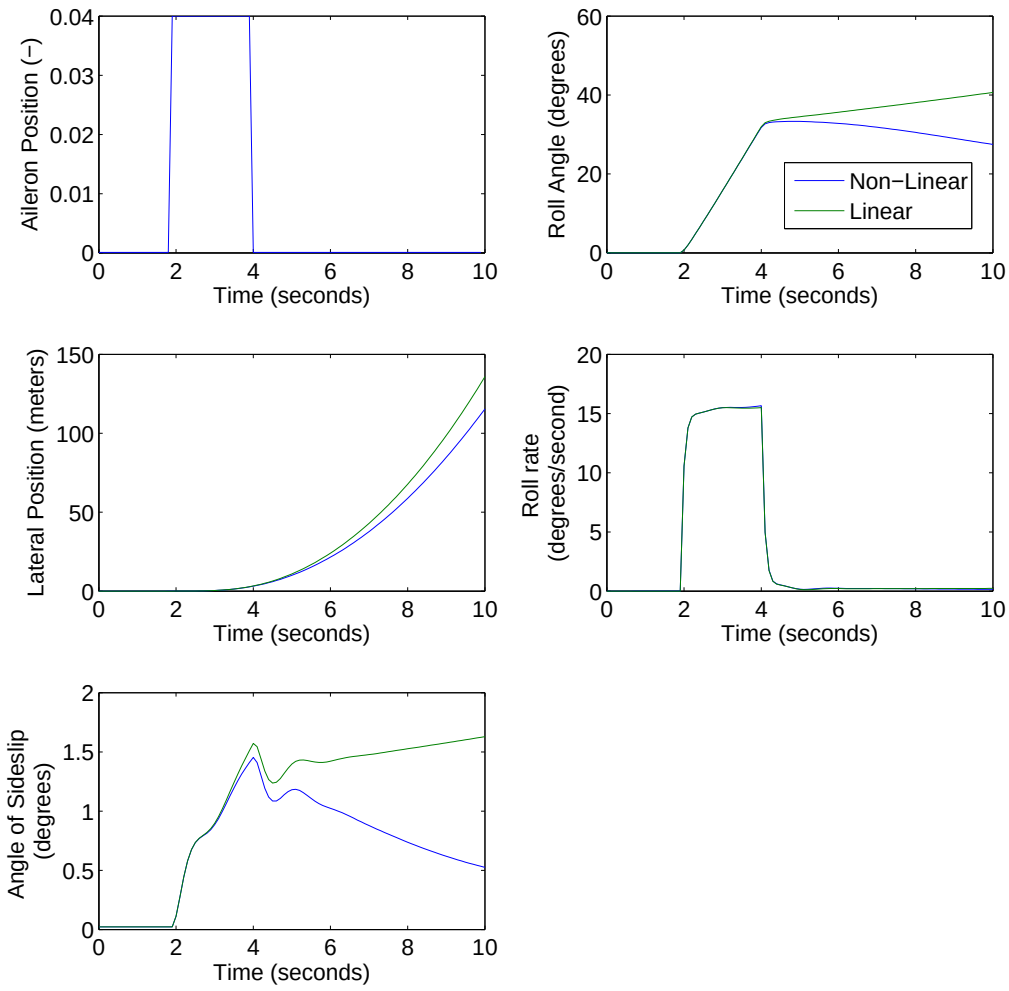


Figure 3.13: Comparison of nonlinear and linear responses of lateral states for a small aileron input.

Since the linear model sufficiently mimics the actual nonlinear dynamics of the UAS over small time periods given small perturbations, it can be used to design a linear controller to be implemented on the nonlinear UAS dynamics.

3.6 Linear Quadratic Regulator Design

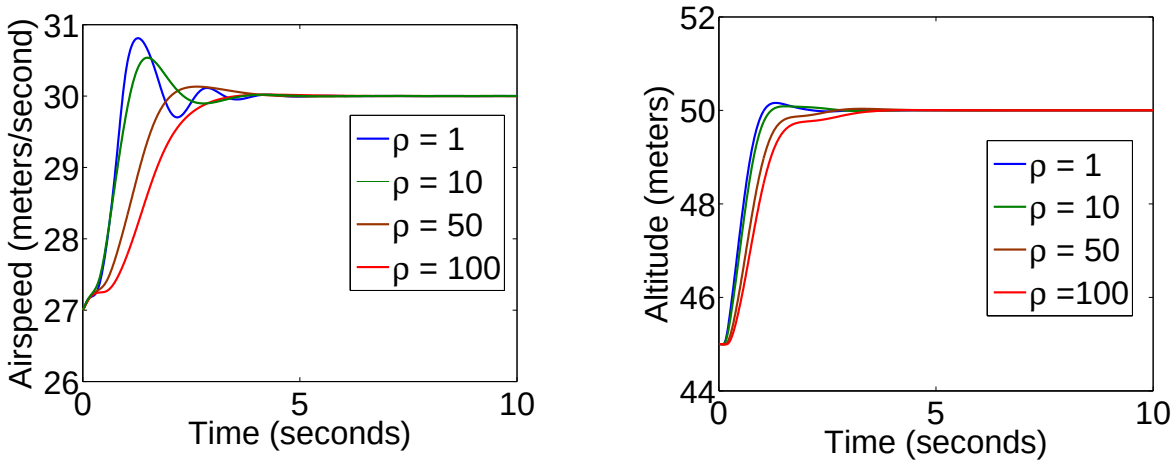
From Figure 3.1, the controller converts the error between the estimated state vector and the guidance reference commanded state vector into the control vector at each time step. The guidance reference command for the simulations in this research is the trim state vector at a given altitude unless another trajectory is specified. This relationship is

$$\mathbf{u}_{k-1} = K(\mathbf{x}_{cmd_{k-1}} - \hat{\mathbf{x}}_{k-1}) + \mathbf{u}_{trim} \tag{3.9}$$

An LQR controller was selected due to its straightforward gain matrix, which must be calculated using the linearized dynamics of the UAS. Using steady, level trim conditions, V_t = 30 meters/second, h=50 meters, $\gamma = 0$ degrees, allows the previously calculated A and B matrices, (3.7) and (3.8) to be used in the LQR process. The initial values of the $\bar{Q}$ matrix were set to an 11×11 identity matrix while the initial values of the $\bar{R}$ matrix were set to a 4×4 diagonal matrix containing the reciprocals of the square of each of the maximum control input values along the diagonal, as suggested by Bryson's Rule [41].

To find a set of gains for the controller that provide the best balance between rise time, overshoot, and settling time, four different scaling values were used. These included $\bar{\rho} = \{1, 10, 50, 100\}$. Each value was used as an input along with $\bar{R}$ and $\bar{Q}$ matrices to the MATLAB ® 'lqr' command to calculate a gain matrix. These matrices were used to determine how effectively the controller could regain trim conditions after a 10% deviation. Figure 3.14(a) shows the airspeed response for each of the scaling values. The value $\bar{\rho} = 10$ had the best tradeoff between the performance parameters. Figure 3.14(b) shows the altitude response for each of the scaling values. Again, the value $\bar{\rho} = 10$ had the best tradeoff,

although its response was very similar to the response with $\bar{\rho} = 1$.



(a) Airspeed Response

(b) Altitude Response

Figure 3.14: UAS response to a 10% deviation from trim for various LQR scaling values.

The 4×11 gain matrices for $\bar{\rho} = 10$ at a steady level flight trim condition and also at wings-level descending flight trim condition are available in Appendix D.

3.7 Chapter Summary

This chapter described UAS GNC simulation software in terms of the object oriented data classes. It also described the urban environment generation software, based on a suite of customizable input parameters for all aspects of the environment. It then detailed the UAS physical model, linearized the equations of motion about a steady, level trim condition, and developed the LQR controller for the simulation. Each result is used in subsequent chapters to examine urban UAS navigation performance under varying aircraft, sensor, and environmental conditions.

CHAPTER 4

Sensor Models and Filter Parameters for UAS Navigation in Urban Environments

This chapter focuses on the sensor and state estimation filter system blocks in Figure 3.1. It first explores the idea of a relative location categorization system to predict GPS availability and accuracy along with LTE accuracy for the filter. Next, the sensor measurement generation process is discussed. This includes the measurement sensitivity matrix, measurement error covariance matrices, and the measurement models for each sensor. The chapter concludes with a discussion of implementing delayed GPS, LTE, and VISION-OF measurements as well as selecting the right number of particles to guarantee a certain level of initial accuracy.

4.1 Urban Environment Relative Location Categorization

When a UAS operates in an urban environment, it may have a set of exteroceptive sensors only available in certain locations with measurement accuracy that varies due to the changing geometry of the surrounding buildings and obstacles. While these sensors use actual GPS or LTE signals to calculate their measurements, they may or may not have the ability to determine and report the accuracy of these measurements to the state estimation filter. One method to generate accuracy values for these sensors is to use empirical accuracy data based on the location of the sensor within the urban environment. This assumes that the

UAS has a priori knowledge of an accurate map of building locations and heights as expected measurement accuracy for sensors such as GPS and LTE will vary as a function of building heights and distances apart. This assumption is reasonable given the availability of building and map database information online today.

Relative locations are discretized into categories based on street-level position within a city block structure (horizontal plane) and altitude with respect to urban canyon buildings (vertical plane). Each horizontal-vertical position category is assigned specific GPS and LTE accuracies based on the literature. Using a feature known as Sensor Accuracy Mode (SAM), two different scenarios are explored. The first scenario presumes each sensor has the ability to determine its own accuracy. In this scenario the propagated horizontal-vertical position is used to look up sensor accuracy values used in both the generation of the sensor measurements and the state estimation filter. In the second scenario the sensors do not have the capability to generate their own accuracy values. In this case, the estimated horizontal-vertical position is used to determine the sensor accuracy filters for the state estimation filter with some uncertainty. However, in this second scenario, measurement generation still uses the same filter parameters and process as in the first scenario since these values are based on simulated ground truth data.

4.1.1 Categorizing Street-Level Position

The street-level or horizontal position categorization, or simply SL, is assigned three possible values listed below and shown in Figure 6.3(b).

1. Urban Canyon: $SL-1$
2. Intersection: $SL-2$
3. Adjacent Open Space: $SL-3$

When the UAS is within an urban canyon, defined by a block with buildings on either side of street, it is in the $SL-1$ category, as shown in Figure 4.1(a). Here UAS navigation

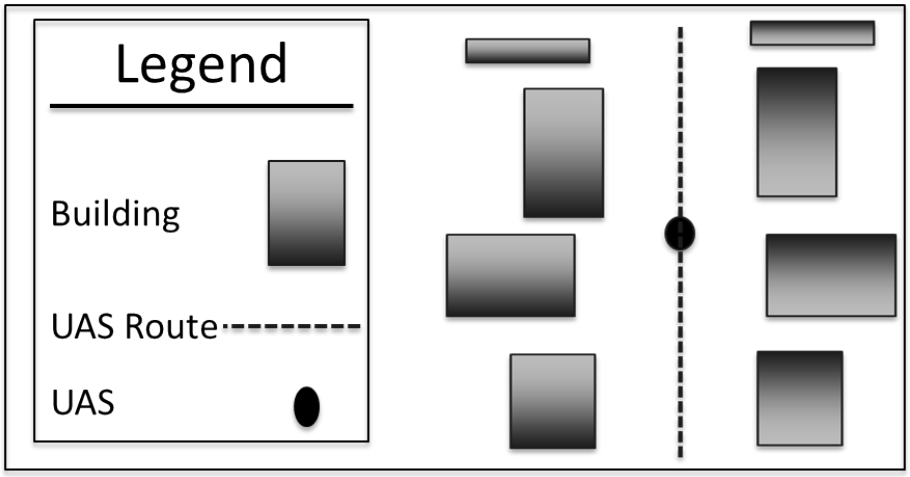
relies more on VISION, LTE, and GPS when available for measurements. Upon exiting the canyon at the end of a block, the UAS enters an intersection, $SL-2$ category, shown in Figure 4.1(b). This environment is more open than the canyon, allowing increased GPS availability and accuracy, increased LTE accuracy, but no VISION availability. UAS are in the third category, $SL-3$, when adjacent to an open space on one side of the street and buildings on the other side as shown in Figure 4.1(c). This environment allows for better GPS availability and accuracy than $SL-1$, but lower GPS availability and accuracy than $SL-2$. In $SL-3$, VISION is still available, but only gives measurements based on the one side with adjacent buildings. Open space on both sides within the urban canyon is not specifically given its own category as this is very similar to an intersection. When conducting urban missions, the UAS will encounter all three categories and must be able to gain as much measurement information as possible for use in state estimation.

The inputs for the SL categorization algorithm include UAS longitudinal position, canyon number, and building information for the current canyon. With these inputs, the algorithm uses a series of tests to determine SL category at the current time step as shown in Figure 4.2.

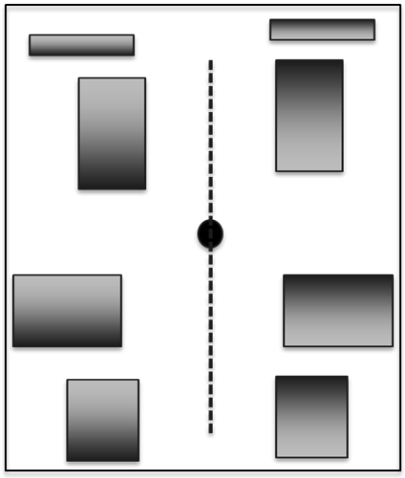
4.1.2 Categorizing Altitude with respect to Buildings

The altitude with respect to buildings categorization, or simply ALT, has three possible values listed below and shown in Figure 4.3.

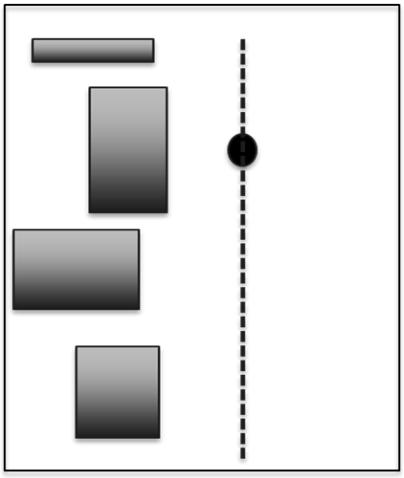
1. Above the tops of all buildings: $ALT-1$
2. Above the tops of some buildings: $ALT-2$
3. Below the tops of all buildings: $ALT-3$



(a) Canyon ($SL-1$)



(b) Intersection ($SL-2$)



(c) Adjacent Open Space ($SL-3$)

Figure 4.1: Street-level position categories (top view).

When the UAS is above the tops of all buildings ($ALT-1$), as shown in Figure 4.3(a), it relies heavily on accurate GPS position and airspeed estimates for navigation since a large view of the sky is available. As the UAS descends into a canyon, it may fly above the tops of some shorter buildings but still be below the tops of other skyscrapers. This is known as above the tops of some buildings ($ALT-2$), as shown in Figure 4.3(b). In this situation, GPS availability and accuracy begins to degrade as the signals no longer have line of sight to the UAS, but VISION availability increases. Once the UAS has descended fully into a canyon, it is below the tops of all buildings ($ALT-3$), as shown in Figure 4.3(c). Here,

1. Is the UAS longitudinal position in a canyon from the last time step AND North of the South side of the southernmost building in the current canyon AND South of the North side of the northernmost building in the current canyon?
   (a) Yes: Is there open space on one side?
      i. Yes: $SL-3$ - Adjacent Open Space
      ii. No: $SL-1$ - Urban Canyon
   (b) No: $SL-2$ - Intersection
2. Is the UAS in an intersection from the last time step but now adjacent to buildings?
   (a) Yes: Increment canyon number
      i. Is there an open space on one side?
         A. Yes: $SL-3$ - Adjacent Open Space
         B. No: $SL-1$ - Urban Canyon
   (b) No: $SL-2$ - Intersection

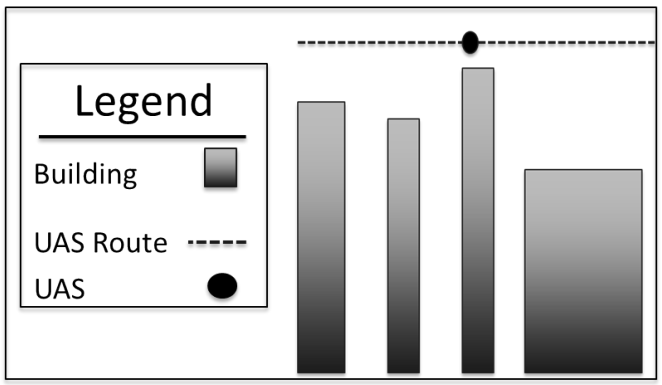
Figure 4.2: Street Level Position Categorization Algorithm.

the UAS may completely lose GPS availability, but may have more consistent VISION availability with buildings on either side of the street now visible. Being able to navigate safely in this category is important for the UAS to execute its low-altitude urban missions.

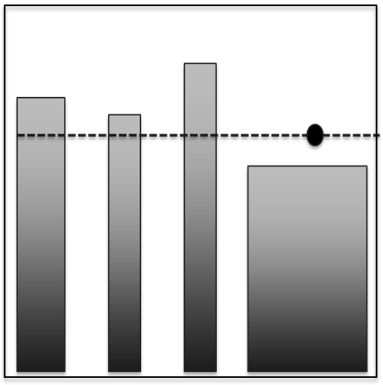
The inputs for the altitude categorization algorithm, shown in Figure 4.4, include the current SL category, UAS altitude, current canyon number, and building information for both the current canyon and the upcoming canyon. Figure 4.5 shows the UAS traveling through the intersection and its four bordering buildings (numbered $1-4$) referenced in the algorithm.

4.2 UAS Urban Environment Sensors

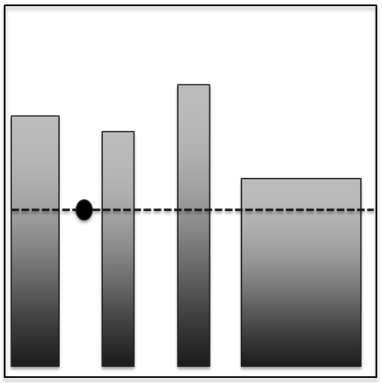
Figure 4.6 shows a system diagram of possible urban canyon navigation sensor solutions with available filtered sensor state noise covariance values. The individual sensor output



(a) Above Tops of All Buildings ($ALT-1$)



(b) Above Tops of Some Buildings ($ALT-2$)



(c) Below Tops of All Buildings ($ALT-3$)

Figure 4.3: Altitude with Respect to Buildings categories (side view).

noise characteristics are shown with solid lines and the integrated system output noise characteristics are shown as dotted lines. Each sensor type is currently available in a commercial off the shelf package and the integrated systems use combinations of these sensors in their published filtering techniques. The VISION-OF signal is considered to be measurements directly from a computer vision system using optical flow and the VISION/IMU signal uses odometry and localization to correct IMU drift.

For this research each sensor and integrated system is assigned a signal ID number to allow for quick reference, facilitate sensor addition and removal, and enable sensor modification to account for changes in performance. This signal ID and available filtered sensor states are shown in Table 4.1.

1. Is the UAS in an intersection ($SL-2$)?
   (a) Yes: Is the altitude of the UAS higher than the four buildings bordering the intersection?
      i. Yes: $ALT-1$ - Above Tops of all Buildings
      ii. No: Is the altitude of the UAS higher than at least one building bordering the intersection?
         A. Yes: $ALT-2$ - Above the Tops of Some Buildings
         B. No: $ALT-3$ - Below Tops of All Buildings
   (b) No: Is the altitude of the UAS higher than the tallest building in the current canyon?
      i. Yes: $ALT-1$ - Above Tops of All Buildings
      ii. No: Is the altitude of the UAS higher than at least one building in the current canyon?
         A. Yes: $ALT-2$ - Above Tops of Some Buildings
         B. No: $ALT-3$ - Below Tops of All Buildings

Figure 4.4: Street Level Position Categorization Algorithm.

Table 4.1: UAS Sensor Information.

	Signal ID Number	Measured States
GPS	S1	x_N, x_E, h, V_T
IMU	S2	$\phi, \theta, \psi, p, q, r$
VISION-OF	S3	V_T
ADS	S4	h, V_T, α, β
LTE	S5	x_N, x_E
GPS/IMU	S6	$x_N, x_E, h, V_T, \phi, \theta, \psi$
VISION/IMU	S7	$x_N, x_E, h, V_T, \phi, \theta, \psi$
ADS/IMU	S8	$\alpha, \beta, \phi, \theta, \psi$

4.3 Sensor Measurement and Error Covariance Generation

To use filtered sensor state accuracy information in a state estimation filter a sensor measurement must be generated and measurement noise covariance information must be de-

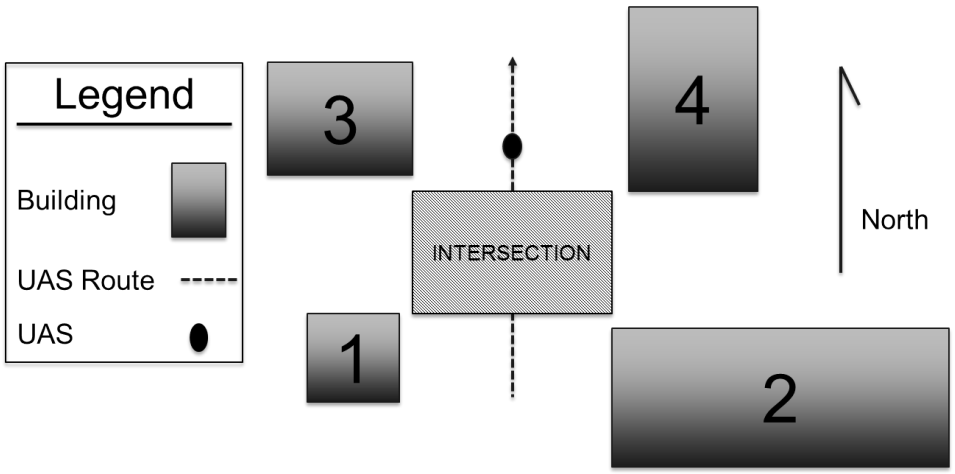


Figure 4.5: Cross-section of urban landscape intersection (top view).

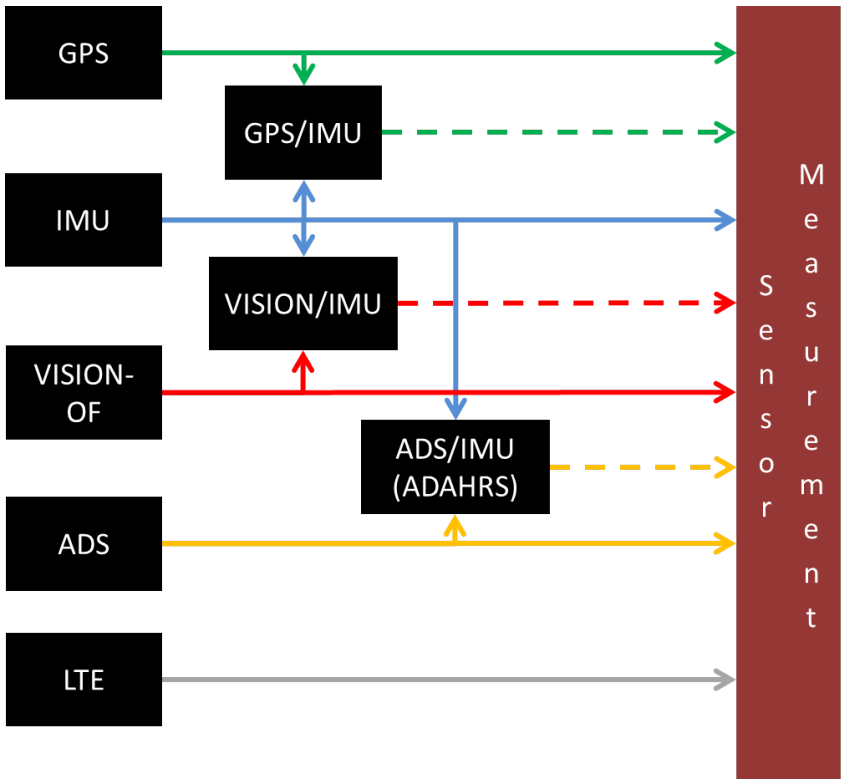


Figure 4.6: Sensor system diagram.

termined. In general, the sensor measurement generation part of the process is divided into an availability determination step, an accuracy determination step, and a measurement generation or acquisition step performed only if the measurement is available.

The input to the overall process is the propagated UAS plant dynamics state vector. Figure 4.7 shows the process that starts from this vector and outputs available sensor measurements and filter measurement noise covariance values for the representative sensors shown. To arrive at the output measurements, the propagated UAS plant dynamics state vector is first used to calculate the relative location categorization (SL, ALT), which dictates the availability of GPS, VISION-OF and VISION/IMU sensors as well as the true 1σ measurement noise values for all sensors except the IMU and ADS that are not influenced by buildings. Should any of the set of GPS and LTE sensors have the ability to determine their own measurement noise covariances, the filter measurement noise covariance matrix will match the measurement noise covariance matrix values as shown in the figure. Using a linear sensor measurement model is a simplifying assumption in the measurement generation process. This assumption should not be made when modeling the internal dynamics of any given sensor.

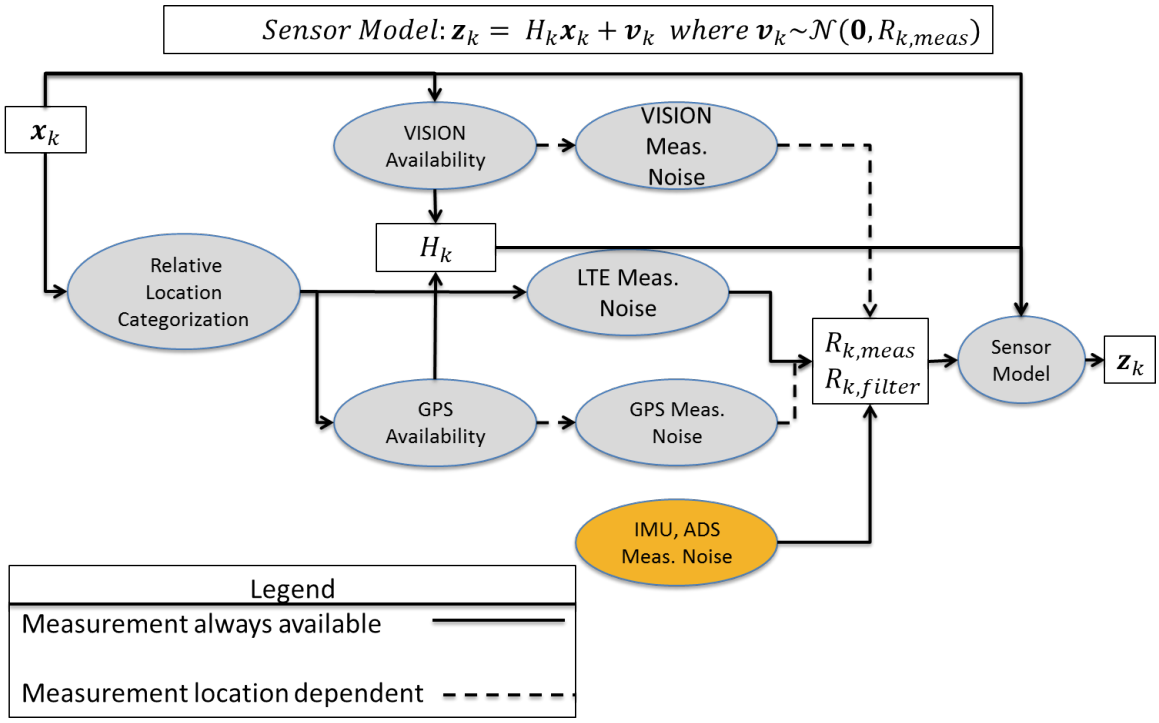


Figure 4.7: Sensor measurement and noise covariance generation process.

4.3.1 Sensor Availability

To best represent real-world conditions and performance, IMU, ADS, and LTE sensors are assumed to be always be available, with measurements taken at the sensor's sampling rate. However, GPS and both types of VISION sensor availabilities are subject to real-world conditions that may cause interruptions in measurement generation as well as changes in accuracy. Availability models are described for GPS and VISION sensors below.

4.3.1.1 GPS Measurement Availability for Simulation

GPS availability is determined through the probability-based algorithm shown in Figure 4.8. In this algorithm, availability probability decreases as the UAS altitude decreases until it is surrounded by buildings. The inputs to this algorithm are the current true ALT and SL values, a GPS Availability Lookup Table, shown in Table 4.2, and a number generated from the uniform distribution $U(0,1)$. The relative location values are used to look up the GPS availability probability from the GPS Availability Lookup Table, which contains values extracted from the literature as described below. This probability is then compared to the generated random number. If the GPS availability probability is greater than or equal to the random number, GPS is available at the current time step for measurement. If the GPS availability probability is less than the random number, GPS is not available for a measurement.

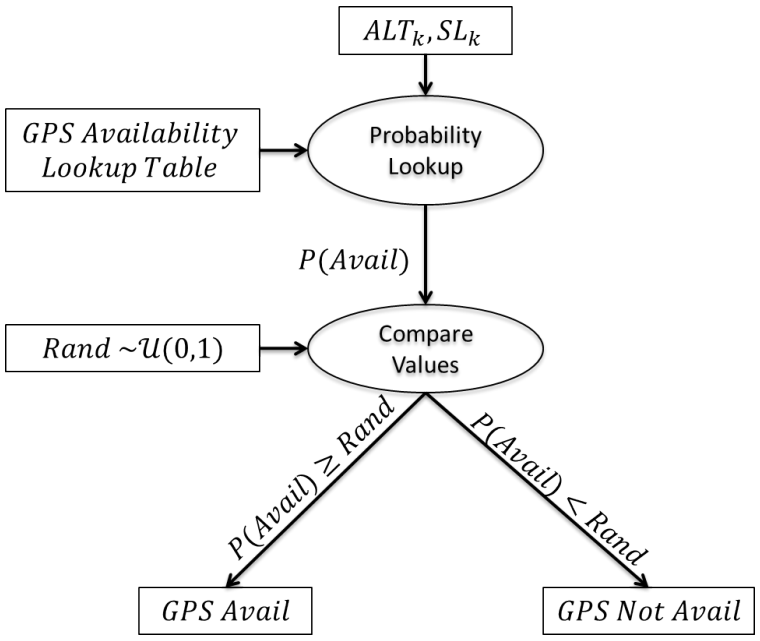


Figure 4.8: GPS measurement availability algorithm.

The basis of this algorithm is the assumption that as the UAS descends deeper into the urban canyon, less GPS satellites are in view. The same is true for canyons versus intersections as the intersections offer a wider view of the sky and therefore more satellites in view. Groves [116] proposed satellite availability as a function of building height and street-width using a global navigation satellite system simulation with three-dimensional models of cities around the world. Real-world experimental data such as in [17] and [20] would need to be collected in urban cores around the globe to determine the exact effects of building heights on GPS availability. As a starting point, GPS availability values are proposed in Table 4.2. The $ALT-2/SL-2$ value was set using the availability value in [10] and the $ALT-3/SL-2$ availability value was set using [29]. All other values for $ALT-2$ and $ALT-3$ were extrapolated. When the UAS is above all buildings, the SL value is not taken into account since it does not affect line of sight to satellites.

Table 4.2: Relative location-based GPS availability probabilities.

	$P(Avail)$
$ALT-1/SL-$	1
$ALT-2/SL-1$	0.45
$ALT-2/SL-2$	0.50
$ALT-2/SL-3$	0.55
$ALT-3/SL-1$	0.20
$ALT-3/SL-2$	0.25
$ALT-3/SL-3$	0.30

4.3.1.2 VISION Measurement Availability for Simulation

VISION-OF as well as VISION/IMU generate measurements only when the UAS is adjacent to buildings on one or both sides. As a result, no measurements are taken while the UAS is passing through an intersection ($SL-2$) or above all buildings ($ALT-1$). This is formalized in Figure 4.9. When the UAS is alongside an adjacent open space, only one of the two VISION sensors is able to generate a measurement. Figure 4.10 shows the port and starboard facing VISION sensors considered for this research. W is the width of the real-world image captured by the camera in meters, L is the perpendicular distance from the camera to the real-world object in meters, and $HFOV$ is the horizontal field of view in degrees.

---

1. Is the UAS in an intersection ($SL-2$) or above all buildings ($ALT-1$)?
   (a) Yes: VISION Unavailable
   (b) No: Is the UAS in an adjacent open space ($SL-3$)
      i. Yes: One VISION sensor available for measurement
      ii. No: Both VISION sensors available for measurement

Figure 4.9: VISION Availability Algorithm.

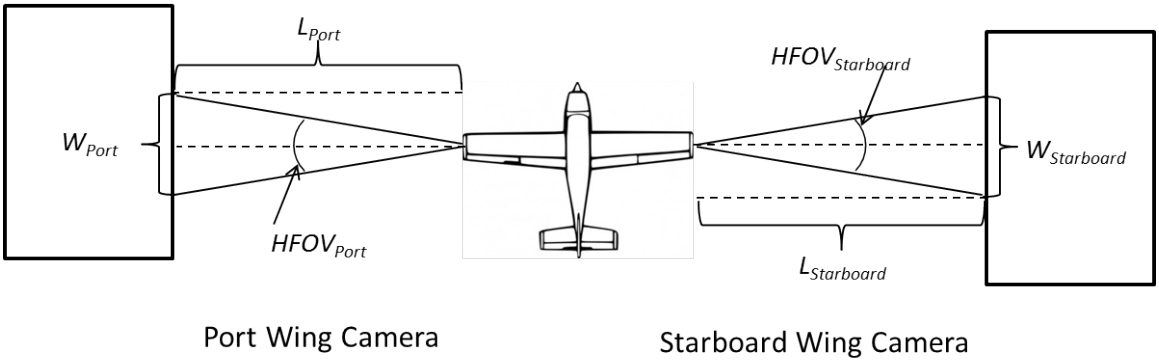


Figure 4.10: UAS VISION geometry.

4.3.1.3 Forming the Measurement Sensitivity Matrix, H_k

The output from the sensor availability calculations is the measurement sensitivity matrix, H_k. This $n_z \times n$ matrix is formed at the current time step by adding a row for each measured state for each available sensor where n_z is the number of available measurements at the time step. Each of these rows are filled with $n-1$ zeros, with a '1' in the column corresponding to the state to be measured using column ordering $[x_N\, h\, V_T\, \alpha\, \theta\, q\, x_E\, \beta\, \phi\, \psi\, p\, r]$ from left to right. This assumes that filtered sensor state values in the inertial frame are output to the sensor measurement model. Since real-world sensors generally produce raw measurements in the sensor frame, these raw measurements would need to be filtered as well as rotated if the sensor was not already aligned with the vehicle frame. An example is shown in (4.1) for the IMU, which measures $\theta, q, \phi, \psi, p, r$ where $n_z = 6$. If another sensor is also available at the current time step, rows are appended as necessary to account for all of the next sensor's measured states. Note that expected measurement availability at each time step is a function of both sensor availability and sensor sampling period.

$$H_{k_{IMU}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

4.3.2 Sensor Measurement Generation

When measurements are available, they are generated using

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k \tag{4.2}$$

where $\mathbf{v}_k$ is a noise term computed as $\mathcal{N}(0, R_k)$, with R_k as the measurement error covariance matrix. The simplified model with zero bias is used to allow focus on the state estimation process using several different sensors rather than the detailed modeling of any individual sensor. Covariance values R_k for each sensor are discussed below.

4.3.3 Sensor Error Covariance Determination for Available Sensors

To generate sensor measurements, the measurement model requires measurement noise covariance values R_k for all available sensors. For some sensors, this is a fixed value for each measured state, while it varies for other sensors based on the environment. As measurements become available versus dropping out, rows are appended to R_k and removed from R_k respectively so that each noise covariance value is in its own row in the column corresponding to the applicable measured state.

IMU and ADS sensors have measurement noise covariance values that are determined through experimental methods. These values are not generally influenced by the environment, so they can be set as constant values in the measurement generation block and state

estimation filter block of Figure 3.1. Even though environment dependent magnetometer data can be used in an IMU, it is not explicitly used in this research. However, sensors such as GPS, and LTE have measurement noise covariances that are dependent on the environment. This research allows for these noise covariance values to be determined either by the actual sensor in the sensor accuracy mode (SAM) *On* configuration or by using the estimated state vector in the sensor accuracy measurement Off configuration. This mode setting and its impact are discussed next.

4.3.3.1 Sensor Accuracy Mode

For sensors such as GPS and LTE it is assumed that measurement noise covariance values can be determined from the actual sensor diagnostic data. This is because both sensors generate measurements based on at least one overarching environmental factor. For GPS, this is the number of satellites that can be viewed by the receiver. In the case of LTE, the main factor is the number of hearable cellular network towers. Although both of these sensors generate measurements as shown in Figure 4.7 following the black arrows, when in the SAM Off mode, they do not have diagnostic data available to generate a measurement noise covariance value. In this case the filter estimates measurement noise covariance as shown in Figure 4.11 following the red arrows. This feature provides another capability to test current and future sensors regardless of their noise determination abilities.

4.3.3.2 Determining Sensor Measurement Noise Covariance Values

Figure 4.7 shows the measurement noise covariance values as outputs of a process dependent on the relative location values of the UAS. This is because both GPS and LTE measurement accuracy are largely correlated to the electromagnetic signal environment, which in turn, is correlated to the density of buildings and other structures. Table 4.3 assigns measurement error covariance values based on the relative location of the UAS within its immediate environment.

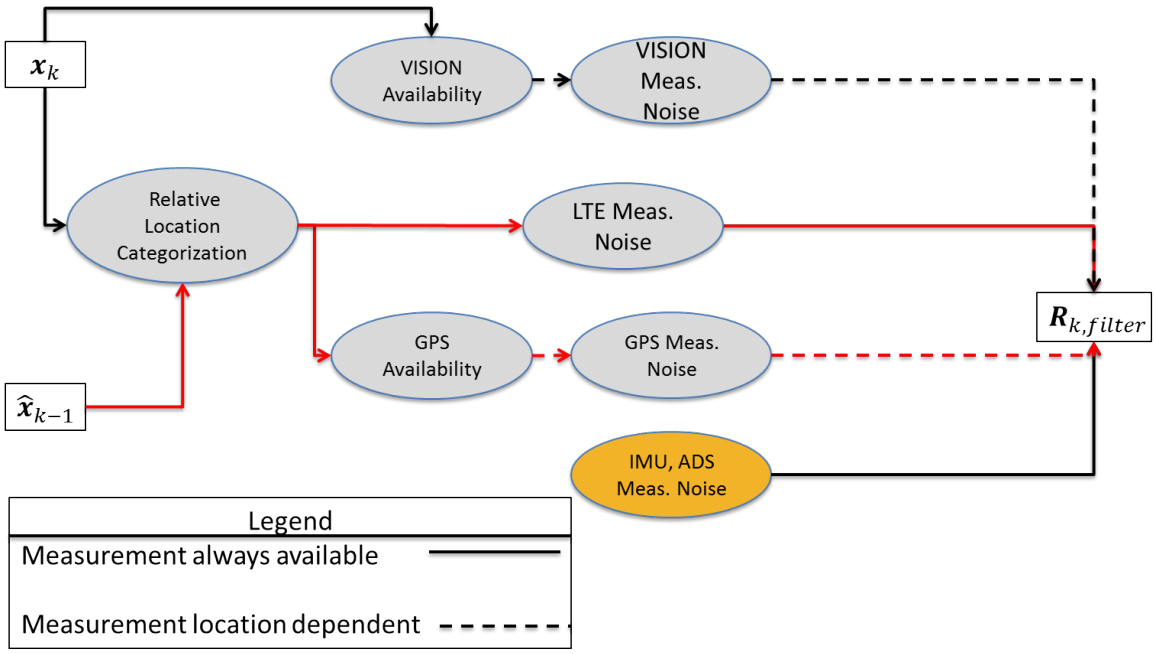


Figure 4.11: Filter measurement noise covariance matrix generation process for SAM Off.

Table 4.3: Measurement error covariance lookup table for GPS and LTE.

	$SL-1$	$SL-2$	$SL-3$
$ALT-1$	σ^2_{11}	σ^2_{12}	σ^2_{13}
$ALT-2$	σ^2_{21}	σ^2_{22}	σ^2_{23}
$ALT-3$	σ^2_{31}	σ^2_{32}	σ^2_{33}

When populated by simulation or real-world sensor data for GPS or LTE, the lookup table gives measurement error covariance values for each measured state as a function of the current ALT and SL values. If SAM is On, the current time-step ALT/SL pair is used, but if SAM is Off, the estimated ALT/SL pair from the previous time step is used. The initial ALT/SL pair is generated using the initial propagated UAS plant dynamics state vector while the initial estimated ALT/SL pair is generated using the initial estimated state vector. Both the true and estimated categorizations are calculated at each time step.

GPS The GPS and LTE inertial position noise covariance tables are shown in Table 4.4 and Table 4.5, respectively. The GPS $ALT-1$ position measurement noise covariance values were taken from [43] and [79] while the $ALT-3/SL-1$ values were taken from [17];

remaining position values were interpolated. While GPS gives the inertial velocity components, the no wind assumption allows this measurement to be used as the forward airspeed, V_T. Due to the lack of consistency in published GPS inertial velocity data regarding inertial airspeed measurement error covariance, which is the same as inertial velocity given the no-wind assumption, it is set to (0.2 meters/second)^2 for all relative location value combinations.

Table 4.4: Location-Based GPS Receiver Noise Covariance Data.

	$\sigma^2_{x_{N,E/h}}$ (meters^2)		
	$SL-1$	$SL-2$	$SL-3$
$ALT-1$	$3.67^2/7.2^2$	$3.67^2/7.2^2$	$3.67^2/7.2^2$
$ALT-2$	$6.96^2/19.59^2$	$6.45^2/17.99^2$	$5.68^2/15.60^2$
$ALT-3$	$10.25^2/31.98^2$	$9.23^2/28.78^2$	$7.69^2/23.99^2$

LTE The LTE OTDOA noise covariance values for inertial horizontal position as shown in Table 4.5 were generated from [53] as this was the only identified source of LTE positioning. These values were converted from horizontal range to the two separate components by assuming the noise was equal in both the North and East directions and taking the root-sum-square of the range. The accuracy is capped at 20 hearable towers consistent with a review of the Cell Reception website for Detroit, Michigan which showed approximately 22 cellular towers [1] in and around the skyscraper core. Higher density cores could use more hearable towers as appropriate. These nominal values assumed a general trend of more hearable eNodeBs when above the tops of some buildings of the table since there were an even mix of ground-based eNodeBs and roof-mounted eNodeBs in this area. LTE network simulations and data collection would be needed to determine exact eNodeB hearability maps and OTDOA accuracy for each specific urban environment.

VISION Recall that the simulation uses two types of VISION measurement inputs: VISION-OF refers to optical flow and VISION/IMU represents VISION odome-

[1] All will be assumed to be eNodeBs on the LTE network

Table 4.5: Location-Based LTE Noise Covariance Data.

	$\sigma^2_{x_{N,E}}$ (meters^2) (# of Hearable eNBs)		
	$SL-1$	$SL-2$	$SL-3$
$ALT-1$	4.38 (10-15)	4.38 (10-15)	4.38 (10-15)
$ALT-2$	4.38 (10-15)	3.48 (15-20)	4.38 (10-15)
$ALT-3$	6.26 (7-10)	4.38 (10-15)	6.26 (7-10)

try/localization information used to correct IMU drift, such as in [50]. Optical flow measurement noise standard deviation $\sigma_{OF_{pixels}}$ is generally reported in units of pixels/frame making a conversion to meters/second necessary. This is done by first determining the width of the real-world image W captured by the camera as shown in Figure 4.10. If the camera's focal length f is known, the image width is calculated using

$$W = \frac{L * d}{f} \tag{4.3}$$

where L is the perpendicular distance from the camera to the real-world object, d is the horizontal dimension of the image, and f is the focal length of the lens. The optical flow inertial airspeed noise covariance value is then scaled from (pixels/frame)^2 to (meters/second)^2 using

$$\sigma^2_{OF} = \Big(\frac{\sigma_{OF_{pixels}} FR * W}{HR}\Big)^2 \tag{4.4}$$

where FR is the camera frame rate in frames/second and HR is the camera horizontal resolution in pixels.

When VISION/IMU odometry/localization measurement noise values are used to correct IMU drift, they are taken directly from the literature [34], [65], [50] since building texture, time of day, or weather-related visibility conditions are not considered.

4.4 Accounting for Delayed Measurements using State Augmentation

With the multitude of sensors available for urban navigation, some are likely to have larger (non-negligible) delay times in producing measurements than others. This research assumes that the IMU, ADS, and VISION/IMU odometry/localization correction do not have delayed measurements since these sensors have a high sampling rate on the order of 50-100 Hz. However, GPS, LTE produce delayed measurements with delay times ranging from 0.1 seconds for GPS [51] to $4-10$ seconds for LTE according to the Polaris Wireless website and [117]. VISION-OF delay is 0.15 second delay. Appendix F shows how the state augmentation process is applied to GPS and LTE measurements.

4.5 Particle Filter Ensemble Sizing

Particle ensemble sizing is generally viewed as a trade off between using enough particles for an accurate representation of the distribution and computation time. Many times the particle ensemble is sized without an accompanying analysis of this design choice [24], [32]. This section describes an analysis aimed at ensuring the set of discrete particles accurately reflects the assumed distribution when all state vector elements are coupled. This problem can be explored by setting the desired covariance tolerance to a constant value and varying the state dimension. Alternatively ensemble sizing can be explored by varying the desired covariance tolerance and setting the state vector dimension to a constant value. Both are important because they give the user insight into the computational cost of the number of particles used to initialize the particle filter as the number of states increase and the desired tolerance decreases. This technique assumes that all particles can be drawn from the distribution at once using a command such as "mvnrnd" in MATLAB® to quickly execute the algorithm rather than build the distribution one particle at a time.

1. Set $N_p = 1$
2. Draw N_p particles from a given multivariate Gaussian distribution: $\mathbf{x}_{sample} \sim \mathcal{N}(\mu_{true}, \sum_{true})$
3. Calculate the particle ensemble covariance matrix $\sum_{sample}$
4. Determine if each entry in $\sum_{sample}$ is within the desired tolerance bounds
   (a) If all entries in $\sum_{sample}$ are within desired tolerance bounds → Go to step 5
   (b) *else* $N_p = N_p + 1$ → Go to step 2
5. Save N_p as number of particles necessary to represent $\mathcal{N}(\mu_{true}, \sum_{true})$ within desired covariance tolerance bounds

Figure 4.12: Particle Ensemble Sizing Algorithm.

The algorithm to determine the required particle ensemble size is specified in Figure 4.12. The only input is desired tolerance and the only output is the number of particles N_p necessary to ensure each entry in the $n \times n$ covariance matrix meets the desired tolerance constraint.

This algorithm must be executed a large number of times in a Monte Carlo simulation format to find an accurate average particle ensemble size, N_p, since this value will change each time the algorithm is executed as particles are sampled from different regions of the distribution.

Two different tests were run to show how results varied based on the dimension n of the state vector and also to investigate how results varied based on the covariance tolerance. The true distribution was $\mathcal{N}(\mathbf{0}_{n\times 1}, A^T A)$ where A was an $n \times n$ matrix drawn using the "rand" command in MATLAB®. Note that any $n \times 1$ vector could be used for the mean vector and any positive-semidefinite symmetric matrix could be used as the covariance matrix.

The first test varied the state dimension n from 1 to 12 while setting the covariance tolerance to 5% or 0.05. The Particle Ensemble Sizing Algorithm was run for 100, 1,000, and

10,000 Monte Carlo trials with the average number of required particles for each dimension shown in Figure 4.13. The number of particles increased from approximately 14 to 648 as the dimension of the vector increased for each set of trials. As expected, the trajectory became more smooth for 1,000 Monte Carlo trials, and 10,000 trial runs revealing a linear relationship between the number of required particles and the dimension of the state of the Gaussian random variable for a desired initial covariance tolerance.

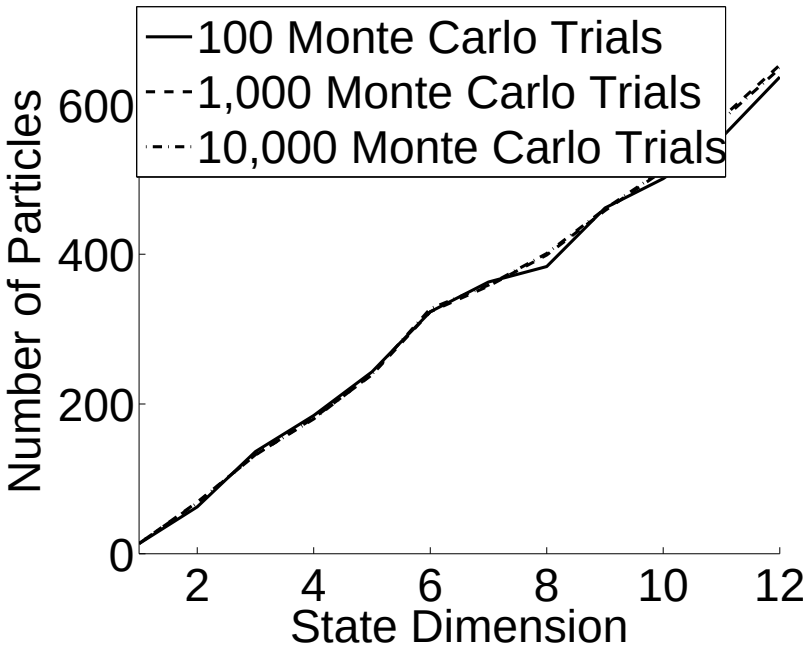


Figure 4.13: Minimum particle ensemble size for 5% covariance tolerance of a Gaussian random variable with increasing dimension.

The second test held the state dimension n constant at 12 while varying the covariance tolerance from 1% down to 10% to give particle filter users a good estimate of the number of particles necessary to initialize the filter. It was run for 100, 1,000, and 10,000 Monte Carlo trials with the average values shown below in Figure 4.14. The exponentially decreasing trajectory shows that to meet 10% tolerance requires roughly 215 particles, but this number increases 40 to 50 fold when 1% tolerance is desired. With such a large increase in required number of particles for smaller than 5% tolerance, users must make a design choice between speed of filter execution and desired accuracy based on the compu-

tational resources available and as well as the application.

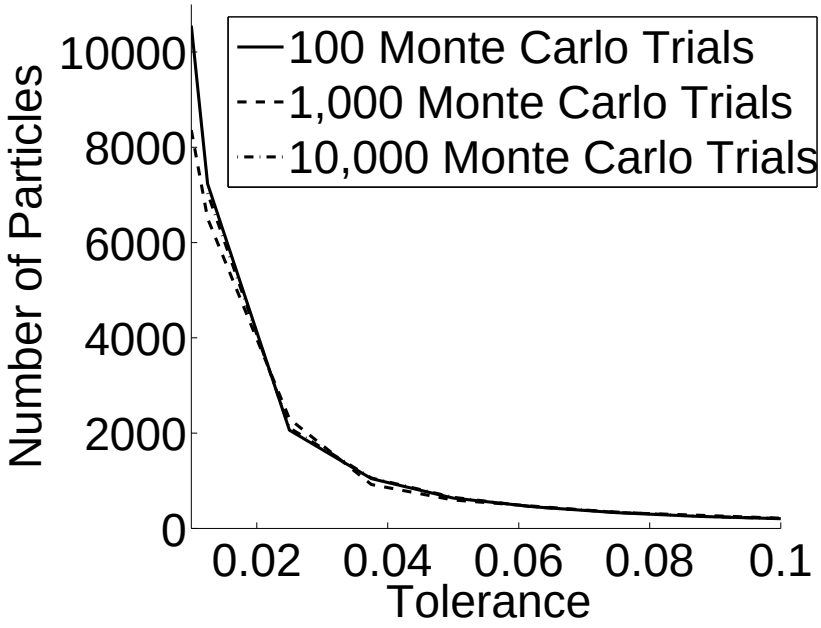


Figure 4.14: Minimum particle ensemble size for 12-state Gaussian random variable as a function of covariance tolerance.

4.6 Chapter Summary

This chapter first described a strategy for relative location categorization by street level (SL) and altitude (ALT) features that facilitate sensor availability and covariance specification. It then described the method used in this dissertation to generate sensor measurements and sensor measurement error covariances for filtered sensor state information from IMU, ADS, GPS, LTE, and VISION-OF. It focused primarily on the GPS, LTE, and VISION sensors as each of these have dynamic environment-based measurement error covariance matrices. There was also a discussion of sensor self-accuracy determination versus using the estimated position in the environment to determine noise covariance values to send to the filter. Additionally, a delayed-state filter implementation was described for GPS and LTE. Finally, a particle filter ensemble sizing study was conducted to determine a min-

imum number of particles required to ensures the ensemble covariance reflects the true distribution covariance to a user-defined tolerance.

CHAPTER 5

Accuracy of Navigation in a Homogeneous Urban Environment

This chapter describes UAS urban environment navigation results using different sensor sets at different altitudes in a simulated homogeneous environment. These sensor sets were selected to demonstrate the varying levels of navigation accuracy available using currently available algorithms and hardware based on literature and manufacturer specifications. The homogeneous environment provides a baseline where sensors are either available throughout the duration of the simulation or unavailable throughout the duration of the simulation. In this chapter the UAS plant dynamics propagation model and state estimation filter propagation model are assumed to be identical (or matched) to objectively evaluate test the sensor sets as a function of the environment.

5.1 Simulation Setup

5.1.1 Homogeneous Urban Environment

The homogeneous urban environment, shown in Figure 5.1, consists of two 100 meter high walls flanking a corridor with the UAS flight path set down the middle of the corridor. This environment is assumed to have constant VISION properties so that the measurement noise covariance is not affected by the longitudinal position within the canyon.

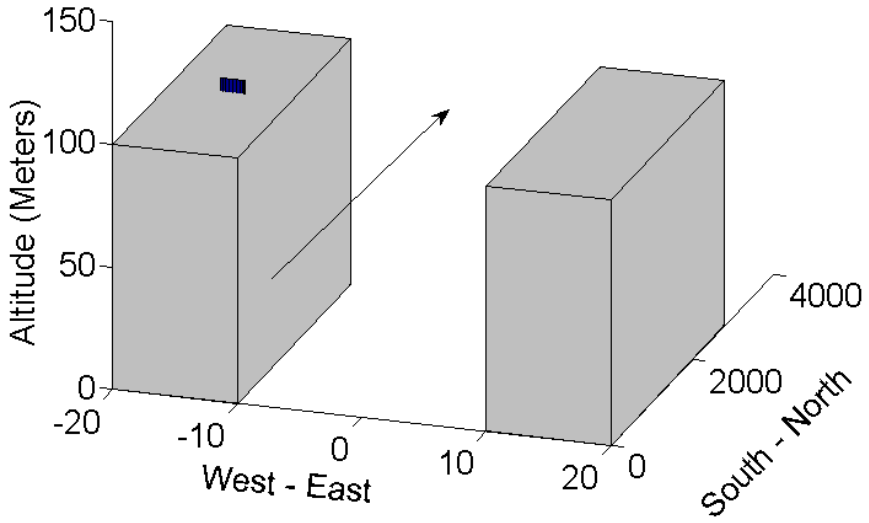


Figure 5.1: Urban Environment for simulations with direction of travel indicated.

5.1.2 Test Matrix

Six tests at varying altitudes, as shown in Table 5.1, were conducted to explore navigation accuracy as a function of GPS, VISION-OF, VISION/IMU, and LTE availability and covariance changes as a function of (SL, ALT) in the homogeneous environment.

Table 5.1: Simulation test matrix.

Test ID	Environment	Altitude, h	Available Signals
1	Open Space	125 meters	IMU, ADS, GPS
2	Open Space	125 meters	IMU, ADS, GPS, LTE
3	High Canyon	75 meters	$IMU, ADS, GPS, LTE, VISION/IMU \times 2$
4	Low Canyon	50 meters	$IMU, ADS\ LTE, VISION/IMU \times 2$
5	High Canyon	75 meters	$IMU, ADS, GPS, LTE, VISION-OF \times 2$
6	Low Canyon	50 meters	$IMU, ADS, LTE, VISION-OF \times 2$

Each test was conducted using the delays as shown in Table 5.2 for GPS, LTE, and VISION. Tests 1, 2, 5, 6 were conducted with zero GPS, LTE, and VISION delay to show the effect of sensor delay on the system. The first test served as a baseline using currently

available sensors with published accuracy in an open space where vision systems would not produce useful information. The second test was conducted in the same environment, but added the LTE sensor to determine what, if any, effect it had on navigation accuracy. The third test explored the effects of being high in an urban canyon, modeled by two long walls, while still having GPS available, although at a degraded level of accuracy. VISION/IMU odometry and localization information was also available for this test using two separate measurements streams (one vision camera pointing directly out laterally from each wing). The fourth test was conducted low in the canyon with GPS turned off to study the effects of using VISION/IMU as a GPS replacement. Tests 5 and 6 replaced VISION/IMU with just VISION-OF to provide vision-based inertial velocity, but no position or orientation measurements in the canyon. These two tests explored the navigation accuracy degradation when using optical flow only. Test 5 was also run for a 200 second duration using 250 Monte Carlo trials to examine the long term trends in navigation accuracy for the two different filter types and sensor delays.

5.1.3 Simulation Parameters

Table 5.2 shows the 1σ measurement noise values for each sensor. Each was taken from published results or sensor/integrated system specification sheets. A complete listing of values is shown in Appendix E. Table 5.3 shows the general simulation parameters. Although Figure 4.13 showed that almost 650 particles are necessary to discretely represent a Gaussian distribution to 5% covariance matrix tolerance initially, $1,000$ particles are used to gain slightly more initial accuracy.

The VISION-OF inertial velocity noise covariance was converted to (meters/second)^2 using the camera specifications from the PX4FLOW Smart Camera, a typical optical flow camera available on hobby websites. Assuming zero wind and travel in the North direction, the airspeed is assumed to be equal to the North inertial speed. The PX4FLOW camera has a resolution of 752×480 with a 16 millimeter focal length and $24 \times 24 \mu$ meters pixel size.

Table 5.2: Sensor simulation parameters

	Sampling Rate (Hz)	Sampling Delay (seconds)	Measured States (Noise Covariance Values)
GPS	1 Hz	0.1	x_N, x_E, h, V_T (See Table 5.4)
VISION/IMU Odometry/Localization	50	0	x_N, x_E, h (meters^2) [2.77, 3.39, 3.36]^2 $\dot{x}_N$, $\dot{x}_E$, $\dot{x}_h$ ((meters/second)^2) [0.70, 0.62, 0.26]^2 ϕ, θ, ψ (degrees^2) [0.760,0.807, 0.740]^2
VISION-OF Inertial Airspeed Only	6.67	0.15	$V_T = V_{North}$ (4.54 frames/second)^2
IMU	100	0	ϕ, θ, ψ (degrees^2) [2.71, 1.65, 8.27]^2 p, q, r ((degrees/second)^2) [0.6, 0.6, 0.6]^2
ADS	50	0	h, V_T [1.5 meters, 1 meter/second]^2 α, β [1 degree, 1 degree]^2
LTE	1/4	4	x_N, x_E (See Table 5.5)

Table 5.3: General simulation parameters.

Time step	0.01 seconds
Simulation Length	20 seconds (600 meters)
Number of Monte Carlo Trials	500
Filter Type	EKF/EnKF
Number of Particles	1,000

The optical flow algorithm and sampling data, shown in Table F.1, was taken from the Bartels algorithm [118] in the Middlebury Optical Flow Evaluation results for the synthetic urban image [75].

The GPS and LTE inertial position noise covariance tables are shown in Table 4.4 and Table 4.5, respectively. The GPS $ALT-1$ position measurement noise covariance values were taken from [43], [79] while the $ALT-3/SL-1$ values were taken from [17]. The remaining position values were interpolated and the velocity noise covariance value was taken from [58]. The LTE OTDOA noise covariance values were generated from [119] as this was the only identified source of LTE positioning accuracy as a function of number of hearable eNodeBs.

Table 5.4: Location-Based GPS Receiver Noise Covariance Data.

	$\sigma^2_{x_{N,E/h}}$ (meters^2)		
	$SL-1$	$SL-2$	$SL-3$
$ALT-1$	$3.67^2/7.2^2$	$3.67^2/7.2^2$	$3.67^2/7.2^2$
$ALT-2$	$6.96^2/19.59^2$	$6.45^2/17.99^2$	$5.68^2/15.60^2$
$ALT-3$	$10.25^2/31.98^2$	$9.23^2/28.78^2$	$7.69^2/23.99^2$

Table 5.5: Location-Based LTE Noise Covariance Data.

	$\sigma^2_{x_{N,E}}$ (meters^2) (# of Hearable eNBs)		
	$SL-1$	$SL-2$	$SL-3$
$ALT-1$	4.38 (10-15)	4.38 (10-15)	4.38 (10-15)
$ALT-2$	4.38 (10-15)	3.48 (15-20)	4.38 (10-15)
$ALT-3$	6.26 (7-10)	4.38 (10-15)	6.26 (7-10)

5.1.4 State Estimation Filters

As shown in Table 5.3, both the Extended Kalman Filter and the Ensemble Kalman Filter, a type of Particle Filter, are used in the simulations in this chapter. Using the two different filter types allows for comparison of the estimated state vector and covariance when they are empirically calculated once per time step as in the Extended Kalman Filter and calculated for several particles and then averaged as in the Ensemble Kalman Filter. Both filters are used when implementing delayed measurements, but only the Extended Kalman Filter is used when zero delay is assumed due to its rapid per time step execution rate.

The initial propagated UAS plant dynamics state vector and diagonal covariance matrix values are shown in Table 5.6 for a steady-level trim condition. For the Extended Kalman Filter and Ensemble Kalman Filter filters, the initial estimated state vector for each Monte Carlo simulation was drawn from $\mathcal{N}(\mathbf{x}_0, P_0)$. The constant process noise covariance matrix, Q, was set to $10^{-4} * I_{12}$. Since process noise structure was not the focus of this research and a specific process noise model was not published for the UAS dynamics model, a diagonal process noise covariance matrix is used. It will apply process noise to each state without correlation when propagating the UAS plant dynamics model.

Table 5.6: Initial UAS Plant Dynamics State and Initial Covariance Values.

State	Initial Value	Initial Covariance
x_N	5 meters	(1 meter)^2
x_E	0 meters	(1 meter)^2
h	50 meters	(1 meter)^2
V_T	30 meters/second	(5 meters/second)^2
α	0.0893 radians	$(1\pi/180$ radians$)^2$
β	0.0003 radians	$(1\pi/180$ radians$)^2$
ϕ	0.0003 radians	$(1\pi/180$ radians$)^2$
θ	0.0893 radians	$(1\pi/180$ radians$)^2$
ψ	0.0005 radians	$(1\pi/180$ radians$)^2$
p	0 radians/second	$(1\pi/180$ radians/second$)^2$
q	0 radians/second	$(1\pi/180$ radians/second$)^2$
r	0 radians/second	$(1\pi/180$ radians/second$)^2$

5.2 Results

5.2.1 Open Space Environment

Table 5.7 shows the longitudinal and lateral RMS position error value at the final time step for the two open space test environments. For both filters, the position error is similar with and without using the LTE measurement. However, in the no-delay case, the LTE measurement slightly aids in decreasing the longitudinal error, while it causes a slight increase in

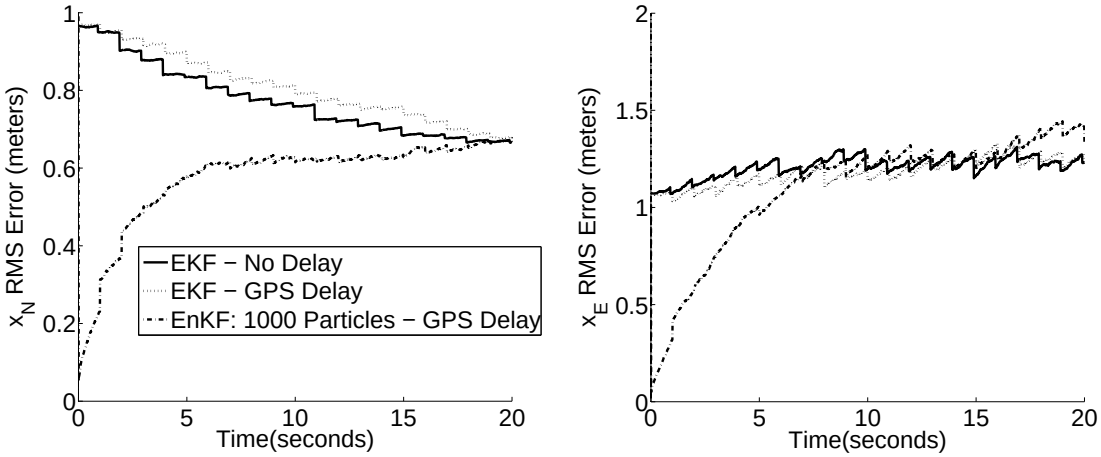
error for the delayed case using either filter. The difference in the magnitude of the lateral and longitudinal errors in the open space environments is due to the controller constantly attempting to correct the lateral position back to the center of the canyon. This causes small overshoots of the trim position throughout the duration of the simulation. The EnKF did not show any improvement in both longitudinal and lateral position errors as it was simply calculating the estimated state vector and covariance empirically instead of in closed-form to eliminate the need to generate the state transition Jacobian.

Table 5.7: RMS position error at final time step for open space test environments.

	Error in meters					
	$EKF(No\ Delay)$		$EKF(Delay)$		$EnKF(Delay)$	
Environment	x_N	x_E	x_N	x_E	x_N	x_E
Open Space (No LTE)	0.67	1.23	0.66	1.24	0.67	1.32
Open Space	0.64	1.25	0.71	1.19	0.68	1.35

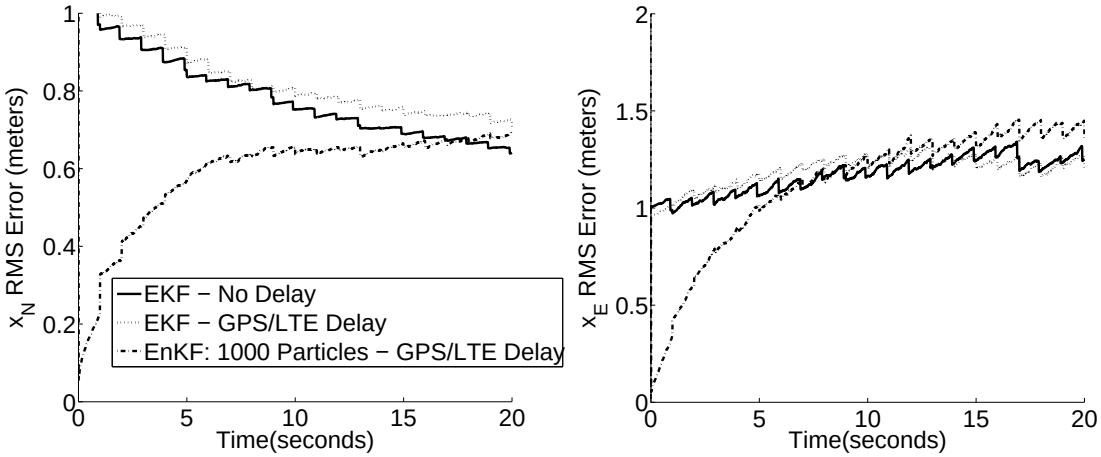
Figures 5.2 and 5.3 show the horizontal RMS position error trajectories of the two open space environments. The RMS position error trajectories are similar for both states when the EKF is used. The addition of LTE, with its 4-second delay, did not give any better performance since these measurements were not weighted as highly as the less-delayed GPS measurements in the correction step. The EnKF gives much lower initial RMS error for each state than the EKF because the initial estimated state vector is calculated as the mean value of $1,000$ twelve-dimensional particles. However, the error increases to EKF levels during the simulation as the forecasted mean is propagated with process noise. In both environments, the delayed EKF and EnKF longitudinal position errors approach the same value showing that in a steady level trajectory with a nearly constant state transition matrix, the EKF is tough to outperform.

Although the LTE measurements had little effect on increasing the accuracy of the position estimate, they did slightly increase the confidence in the estimate as shown in Figure 5.4. Here the EKF and EnKF position mean error and 3σ bounds are shown for the longitudinal position state (Figure 5.4(a)) and the lateral position state (Figure 5.4(b)). A



(a) Longitudinal Position (b) Lateral Position

Figure 5.2: RMS error trajectory for the open space environment with GPS.

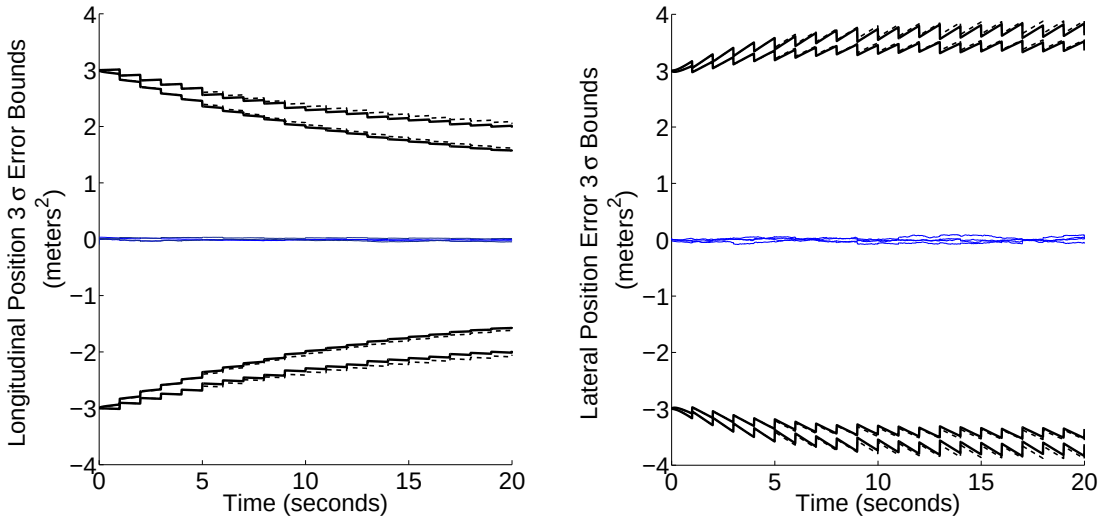


(a) Longitudinal Position (b) Lateral Position

Figure 5.3: RMS error trajectory for the open space environment with GPS and LTE.

legend is not included on this figure since the trajectories are very close together and use the same line types. The dotted lines with slightly higher 3σ values represent the open space environment with GPS only. The outer set of dotted lines is for the EKF and the inner set of dotted lines is for the EnKF. The solid lines represent the open space environment with the delayed LTE measurements added to the GPS. The outer set of solid lines is for the EKF and the inner set of solid lines is for the EnKF. The increase in the estimate confidence is shown by the slight divergence of the solid and dotted lines, with small but noticeable

shrinking of the bounds when the first delayed LTE measurement arrives at the 5 second point as more measurement information is received by the filter. Also, as expected, the EnKF does give tighter bounds since the covariance is empirically calculated, more closely approximating the error of the system. The error trajectories (shown in blue) hover near zero for both filters with and without LTE indicating that the error is Gaussian distributed over the set of Monte Carlo trials.



(a) Longitudinal Position

(b) Lateral Position

Figure 5.4: Effect of adding delayed LTE sensor on horizontal position error 3σ bounds.

5.2.2 Canyons

Table 5.8 shows the longitudinal and lateral RMS position error value at the final time step for the four canyon test environments. When using the VISION/IMU system to augment GPS and/or LTE, both filters give very small estimation errors in the range of $0.12 - 0.14$ meters in both horizontal directions. However, once the VISION augmentation is set to providing $V_T = V_N$ updates only, the error gets much worse when using the EKF with longitudinal RMS errors near 1 meter and lateral RMS errors approaching 2 meters in the realistic delayed measurement cases. Although the EnKF performed slightly better in the

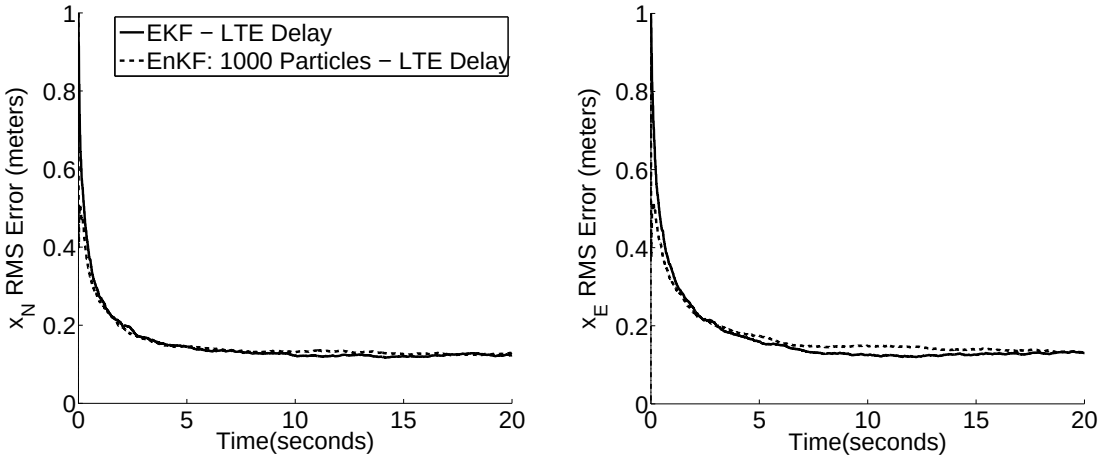
longitudinal direction with RMS errors just under 0.8 meters, the lateral position error got slightly worse just like in the open space tests for all environments as it attempted to keep the UAS down the centerline using less accurate, low-sampling rate, delayed measurements.

Table 5.8: RMS position error at final time step for canyon test environments.

	Error in meters					
	$EKF(No\ Delay)$		$EKF(Delay)$		$EnKF(Delay)$	
Environment	x_N	x_E	x_N	x_E	x_N	x_E
High Canyon (VISION/IMU)	-	-	0.13	0.12	0.14	0.14
Low Canyon (VISION/IMU/No GPS)	-	-	0.12	0.13	0.13	0.13
High Canyon (VISION OF)	0.89	1.73	0.92	1.79	0.74	1.93
High Canyon: 200 seconds (VISION OF)	1.06	2.31	1.04	2.38	1.61	3.39
Low Canyon (VISION OF/No GPS)	0.98	1.88	1.01	2.06	0.78	2.21

Figure 5.5 show the horizontal RMS position error trajectories of the low canyon environment at $h = 50$ meters. Both the lateral and longitudinal errors drop quickly and reach similar steady state error values due to the dominance of the two sets of high-sampling rate VISION/IMU measurements using both the EKF and EnKF. For the lateral error, this is in contrast to the open space environments where the error grows as the UAS attempts to control its lateral position using much slower position measurements. The results are similar for the high canyon environment at $h = 75$ meters and are not shown.

When the VISION/IMU is removed from the system and VISION-OF is added, the results look much different as shown in Figures 5.6 and 5.7 for the high and low canyon environments. These results show the same trends as the open space results since the only position sensors are GPS and LTE in the high canyon and only LTE in the low canyon. However, the RMS errors are higher within high canyons, especially when using the EKF, because the GPS measurements are less accurate, when available, and the LTE measure-

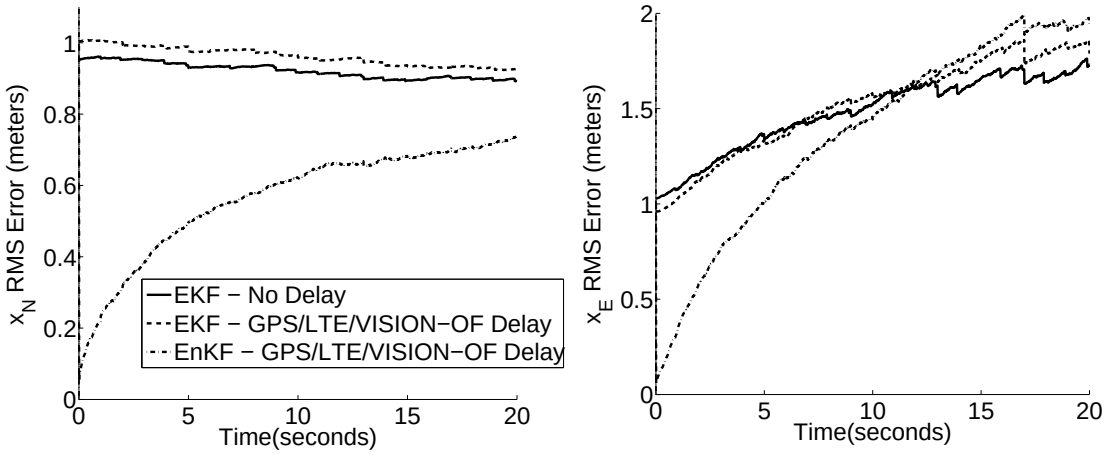


(a) Longitudinal Position (b) Lateral Position

Figure 5.5: RMS error trajectories for the low canyon environment with LTE and VISION/IMU.

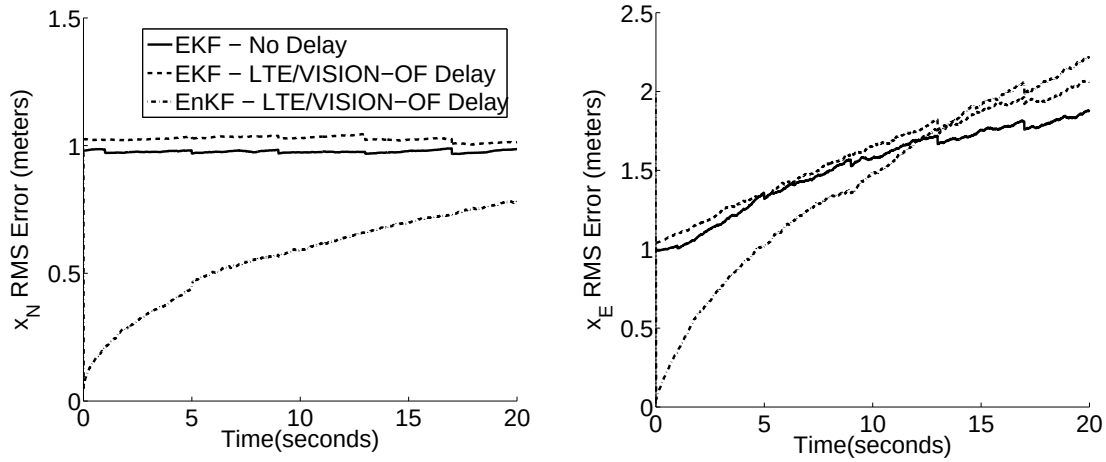
ments are only available every four seconds, and are delayed by the same amount. This effect is seen more in the lateral position errors as the controller attempts to center the UAS using this delayed measurement. The EnKF does provide a more accurate initial estimate of the longitudinal position with these low-sampling rate sensors, but it degrades over time and is still increasing toward the EKF trajectories at $t = 20$ seconds. Overall, GPS and/or LTE can only provide roughly 1 meter RMSE error in the longitudinal position estimate within an urban canyon and continuously worsening RMSE error in the lateral position estimate.

The 200-second high canyon test showed worse longitudinal and lateral RMS error results than the 20-second test indicated at the same conditions. From Table 5.8, the *nodelay* and *delay* EKF longitudinal error are roughly 1 meter and the lateral error hovers near 2.35 meters. However, as shown in Figure 5.8, these errors appear to have reached a steady-state value by this point, allowing accuracy conclusions to be drawn confidently. The vertical line in Figures 5.8(a) and 5.8(b) at the 20 second mark allows comparison between the error at 20 seconds and the error at 200 seconds. The major difference in the 20 second and 200 second runs is with the EnKF. With this filter, the longitudinal error is still growing



(a) Longitudinal Position (b) Lateral Position

Figure 5.6: RMS error trajectories for the high canyon environment with GPS, LTE and VISION-OF.

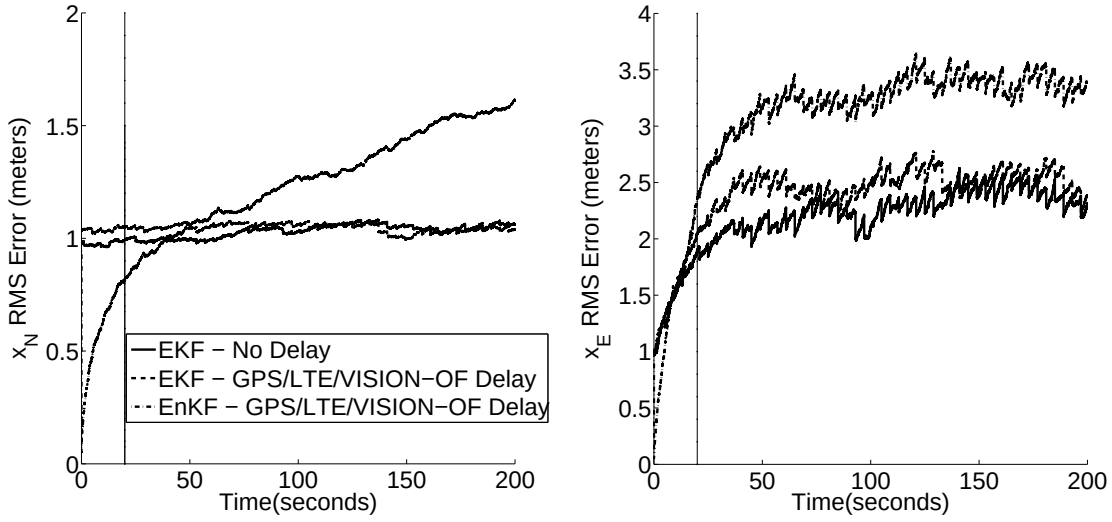


(a) Longitudinal Position (b) Lateral Position

Figure 5.7: RMS error trajectories for the low canyon environment with LTE and VISION-OF.

linearly at 200 seconds while the lateral error eventually reaches a steady-state value near 3.4 meters. The longitudinal position error will most likely keep climbing until it reaches the GPS $ALT-3/SL-1$ noise value of 10.25 meters since it considers all particles equally when calculating the estimate, regardless of the likelihood of those estimates being close to the actual value of the state. The lateral position error grew quickly to its steady-state

value since this state was being controlled. As the UAS plant dynamics lateral position value was propagated using a control input based on the estimate at the previous time step, it did not move back toward its commanded value of zero. This was due to the estimated value of the state at the previous time step being generally close to zero as it was the mean of $1,000$ particle values. With such a small difference between the estimated state value and its commanded value, the control input stayed small. As a result, the UAS plant dynamics state value had no way of being pushed back toward zero since it was already being estimated near zero. These trends show the difficulty in using an EnKF for state estimation and control with limited, inaccurate measurements.



(a) Longitudinal Position

(b) Lateral Position

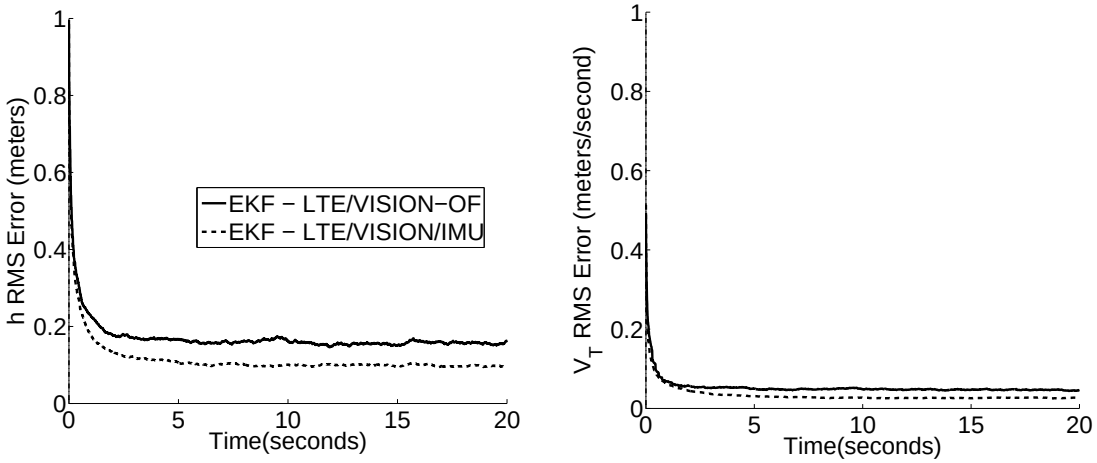
Figure 5.8: RMS error trajectories for a 200 second simulation in the high canyon environment with GPS, LTE and VISION-OF.

5.2.3 Altitude, Airspeed, and Attitude Performance

Since the altitude, airspeed, and attitude are measured by the reliable high sampling rate IMU and ADS, their performance does not change greatly when considering delayed systems versus ideal non-delayed systems. As such, the below results are presented using the delayed systems only comparing the error trajectories for the 50 meter LTE/VISION OF

test as the baseline and the 75 meter GPS/LTE/VISION/IMU to determine what, if any, effect the GPS and VISION/IMU sensors have on the accuracy of these states.

Figure 5.9 shows the altitude (Figure 5.9(a)) and airspeed (Figure 5.9(b)) RMSE trajectories for both low canyon environments. Since the ADS provides high-sampling rate measurements of both states in the LTE/VISION-OF configuration, the RMS error reaches a small steady state value quickly. When adding the high-sampling rate VISION/IMU, there is a slight decrease in steady-state error for both states. This drop in error is expected in the EKF since two independent altitude and airspeed measurements are available at every other time step to correct the predicted estimate.



(a) Altitude

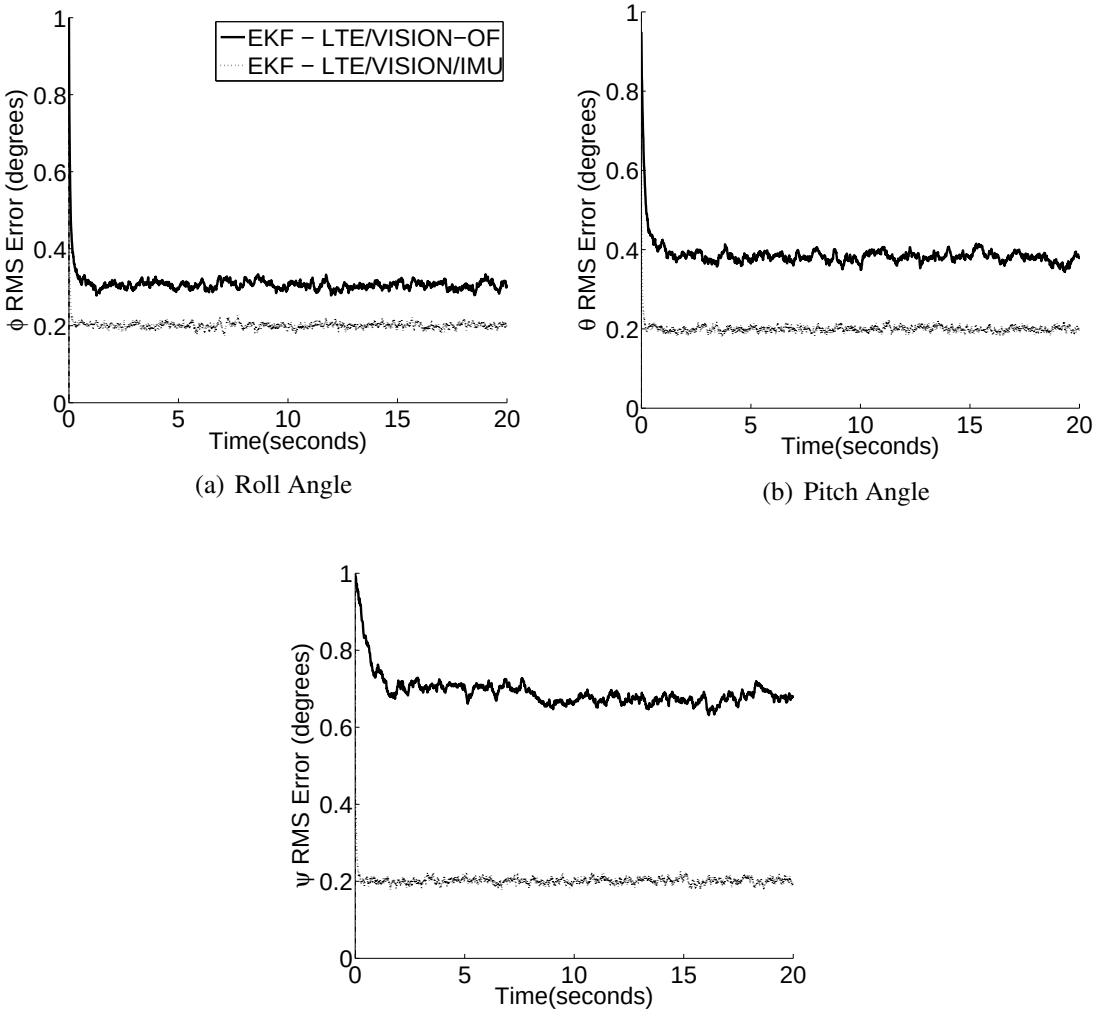
(b) Airspeed

Figure 5.9: RMS altitude and airspeed error trajectories for low canyon environments.

Figure 5.10 shows the attitude angle RMSE trajectories for roll (Figure 5.10(a)), pitch (Figure 5.10(b)), and yaw (Figure 5.10(c)). The IMU provides small steady-state RMS errors when in the LTE/VISION-OF configuration with roll and pitch error just under 0.4 degrees and yaw RMS error at 0.7 degrees. This trend of IMU roll and pitch being much more accurate than yaw is typical since yaw is primarily measured with a noisy magnetometer. However, once the VISION/IMU measurements are added, all three RMS errors drop to roughly 0.2 degrees as the VISION/IMU measurements are very accurate with low variances when using feature tracking-type localization algorithms.

5.3 Chapter Summary

Navigation accuracy in the open space and homogeneous urban canyon test environments was evaluated using several sensor suites that included combinations of GPS, LTE, VISION/IMU and VISION-OF along with the traditional IMU and ADS. In the open space environment with realistic sensor delay, the addition of LTE made a marginal difference in lateral position navigation accuracy, but slightly decreased the estimation error covariance. In the canyon with VISION/IMU navigation accuracy was on the order of 0.15 meters under the assumed simulation conditions, with little difference when eliminating GPS. When reverting to VISION-OF, the accuracy was on the order of $1-2$ meters for both lateral position states. When in the canyon using VISION-OF, the EnKF generally gave more accurate short-term results in the longitudinal direction while the EKF gave better results along the lateral direction. Longer simulations should be run in the future to determine if the EnKF can reach a steady-state longitudinal error value. Overall, VISION/IMU integrated navigation systems should be considered when possible as they are able to add accurate high sampling rate position and attitude measurements to the state estimation filter. LTE OTDOA is still immature as a UAS positioning sensor due to its low sampling rate, high delay period, and limited availability of accuracy data in literature but it may provide a means of detecting spoofed GPS data on a mass level, an analysis left to future work.



(a) Roll Angle

(b) Pitch Angle

(c) Yaw Angle

Figure 5.10: RMS attitude angle error trajectories for low canyon environments.

CHAPTER 6

Accuracy of Navigation in a Heterogeneous Urban Environment

This chapter studies navigation accuracy in a realistic urban environment using both the exact UAS dynamics propagation model, which includes the equations of motions along with the constant aerodynamic model parameters, and multiple randomly-generated UAS plant dynamics propagation models where the aerodynamic model parameters are varied from their exact values based on the literature. In the heterogeneous urban environment, buildings of varying heights, intersections, and open spaces allow the study of sensor performance in different relative location categories during straight and level flight as well as climbing and descending flight. Since GPS availability rates are not constant within an urban environment, these rates will be varied as a function of relative location category throughout many of the simulations presented within the chapter. This dissertation refers to models as *matched* when the same numerical values are used for constants and coefficients in the UAS plant dynamics and state estimation filter propagation models. *Unmatched* models have one or more different numerical values for the UAS plant dynamics model versus the filter dynamics model. Unmatched models require process noise tuning to ensure a consistent filter. The sensor models remain unchanged. Process noise tuning is accomplished through the use of the average normalized estimation error squared (ANEES), the average normalized innovation squared (ANIS), and the autocorrelation statistic, all introduced in Chapter 2.

Specific factors affecting the accuracy of UAS state estimation in the urban environments include both sensor availability and sensor self-accuracy detection, as first discussed in Chapter 4 as sensor accuracy mode (SAM). In reality, the UAS GPS sensor is sensitive to its altitude in the canyon with respect to the surrounding buildings. It may generate measurements when near the top of the urban canyon, but it can lose line of sight to enough satellites to lose measurement generation capability when flying lower in the canyon. This phenomenon can be modeled discretely by turning GPS off when the UAS is below $h = 75$ meters, as was the case in the previous chapter. However, a more realistic technique to account for varying availability is to use empirical availability rates as was shown in Table 4.2. Understanding the effects of sensor self-accuracy detection for both GPS and LTE is also important. If these sensors can determine their own measurement noise covariance values based on signal data, this information can be provided to the filter to give a more realistic estimate of the propagated UAS states. Otherwise, if these sensors are not able to generate this information, the filter must use estimated measurement noise covariance values based on environmental information from previous state estimates.

The remainder of the chapter is outlined as follows: the Simulation Description section discusses the simulation description in terms of building layout, sensors used and test matrices. The Matched Model Results section analyzes the large pool of matched model results where GPS availability, sensor accuracy mode, and the flight path are varied. The Unmatched Model Results section analyzes both filter tuning and navigation accuracy for a straight and level trajectory in the canyon. Finally, the Summary section provides overarching conclusions based the simulation results.

The main contribution of this chapter is to provide navigation accuracy results for multiple trajectories in the heterogeneous urban environment with varying GPS availability for both matched and unmatched models and filter tuning results for the unmatched model case. The main innovations of this chapter include the incorporation of position dependent sensor availability for GPS, sensor accuracy mode consideration when determining

measurement noise covariance, and using expected sensor availability in the filter tuning process.

6.1 Simulation Description

The general simulation parameters used in this chapter are shown below in Table 6.1.

Table 6.1: General simulation parameters.

Time-Step	0.01 seconds
Simulation Length	20 seconds (600 meters)
Number of Monte Carlo Trials	500
Filter Type	EKF
Base Process Noise Covariance	$Q = 10^{-4} * I_{12}$

6.1.1 Heterogeneous Urban Environment

The urban landscape shown in Figure 6.1 was used for all simulations in this chapter. Each block contains four buildings of varying heights except the east side of the third block, which is an adjacent open space. Since the mission altitudes within the environment were $h = 50$ meters and $h = 75$ meters, each of the urban environment relative location categories were encountered at least once. The $h = 125$ meters case was used as the baseline since both GPS and LTE were available at each of their sampling instances.

6.1.2 Sensors and Sensor Noise Covariance

Table 6.2 shows the 1σ measurement noise values for each sensor used in simulations within this chapter. The sensors include the $IMU, ADS, GPS, LTE, VISION - OF \times 2$, where $VISION - OF$ is an optical flow camera oriented perpendicular to the longitudinal axis of the UAS. The $\times 2$ denotes two cameras and two sets of measurements, with one along each wing. Each result in Table 6.2 was taken from published results or sensor/integrated system specification sheets.

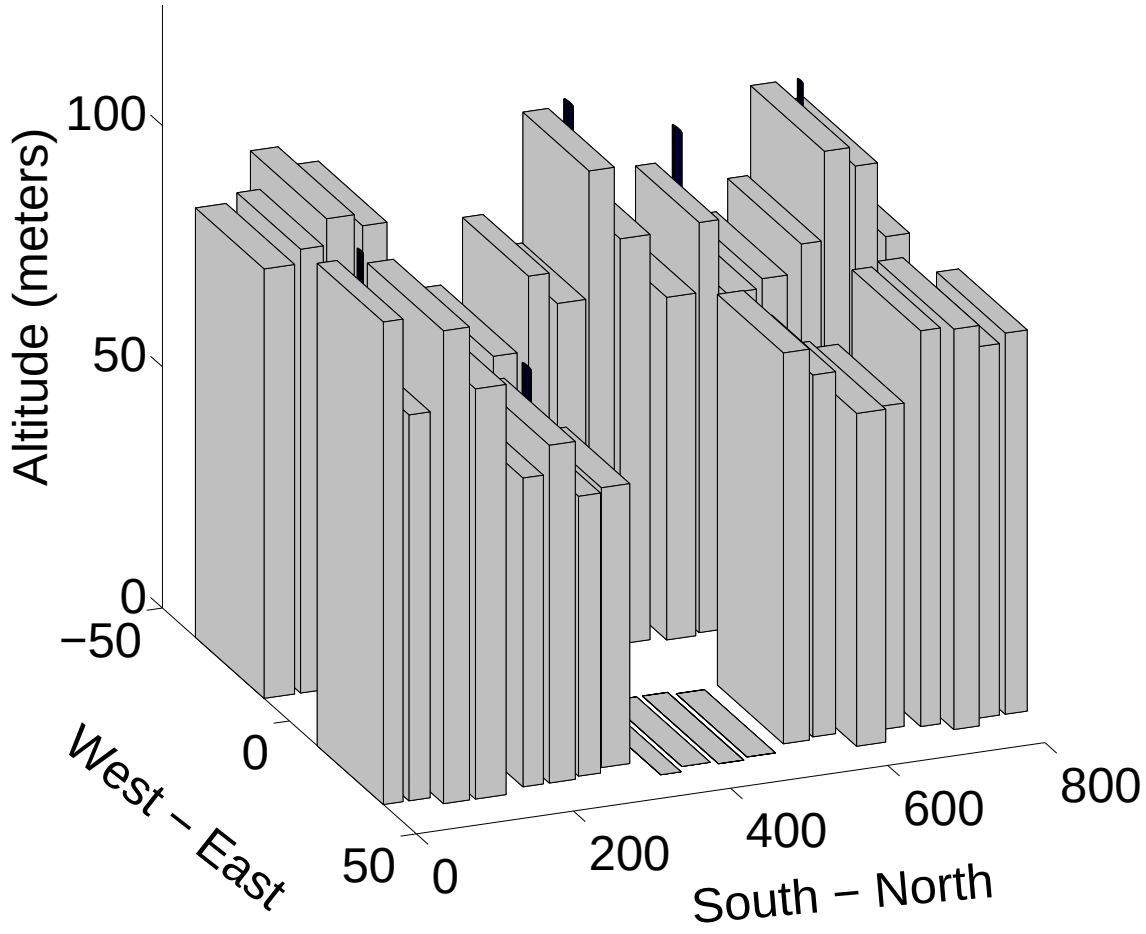


Figure 6.1: Example urban landscape used in simulations.

VISION-OF is used for all simulations in this chapter in order to focus on GPS availability as a main factor in receiving horizontal position measurements. Similar to the previous chapter, the VISION-OF inertial airspeed noise covariance was converted to (meters/second)^2 using the camera specifications from the PX4FLOW Smart Camera, a typical optical flow camera available on many hobby websites. These specifications included a resolution of 752×480 with a 16 millimeter focal length and 24×24 micrometer pixel size. The optical flow algorithm accuracy and computational speed data, shown in

Table 6.2: Sensor simulation parameters for heterogeneous urban environment testing.

	Sampling Rate (Hz)	Sampling Delay (seconds)	Measured States (Noise Covariance Values)
GPS	1 Hz	0.1	x_N, x_E, h, V_T (See Table 6.3)
VISION-OF Inertial Airspeed Only	6.67	0.15	$V_T = V_{North}$ (4.54 frames/second)^2
IMU	100	0	ϕ, θ, ψ (degrees^2) [2.71, 1.65, 8.27]^2 p, q, r ((degrees/second)^2) [0.6, 0.6, 0.6]^2
ADS	50	0	h, V_T [1.5 meters, 1 meter/second]^2 α, β [1 degree, 1 degree]^2
LTE	1/4	4	x_N, x_E (See Table 6.4)

Table F.1, was taken from the Bartels algorithm [118] in the Middlebury Optical Flow Evaluation results for the synthetic urban image [75].

The GPS and LTE inertial position noise covariance tables are both shown in Table 4.4 and Table 4.5 respectively. The GPS $ALT-1$ position measurement noise covariance values were taken from [43], [79] while the $ALT-3/SL-1$ values were taken from [17]. The remaining position values were interpolated and the airspeed noise covariance value was taken from [58]. The LTE OTDOA noise covariance values were generated from [119] as this was the only identified source of LTE positioning accuracy as a function of number of hearable eNodeBs.

Table 6.3: Location-Based GPS Receiver Noise Covariance Data.

	$\sigma^2_{x_{N,E/h}}$ (meters^2)		
	$SL-1$	$SL-2$	$SL-3$
$ALT-1$	$3.67^2/7.2^2$	$3.67^2/7.2^2$	$3.67^2/7.2^2$
$ALT-2$	$6.96^2/19.59^2$	$6.45^2/17.99^2$	$5.68^2/15.60^2$
$ALT-3$	$10.25^2/31.98^2$	$9.23^2/28.78^2$	$7.69^2/23.99^2$

Table 6.4: Location-Based LTE Noise Covariance Data.

	$\sigma^2_{x_{N,E}}$ (meters^2) (# of Hearable eNBs)		
	$SL-1$	$SL-2$	$SL-3$
$ALT-1$	4.38 (10-15)	4.38 (10-15)	4.38 (10-15)
$ALT-2$	4.38 (10-15)	3.48 (15-20)	4.38 (10-15)
$ALT-3$	6.26 (7-10)	4.38 (10-15)	6.26 (7-10)

6.1.3 Test Matrix

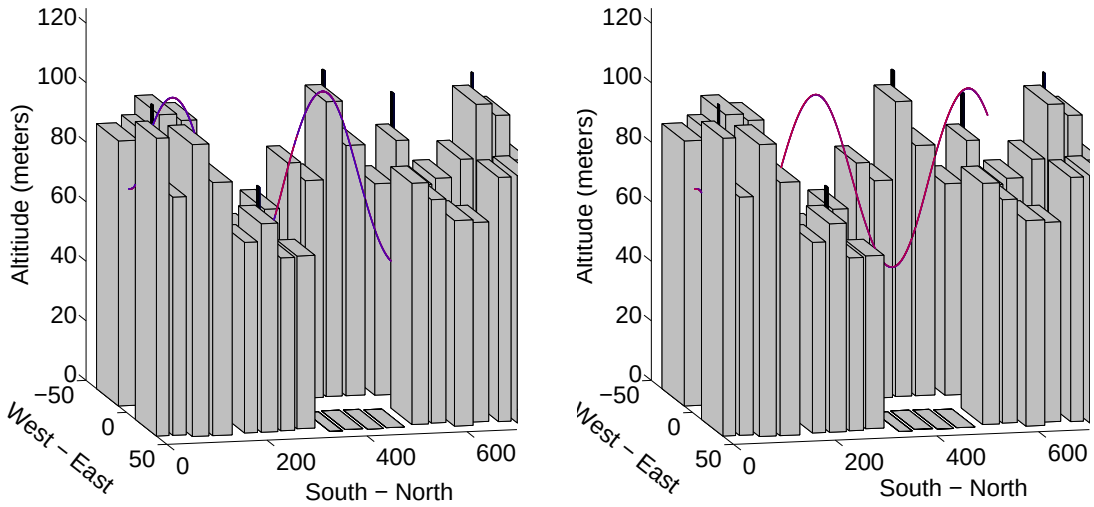
6.1.3.1 Matched Model

To test GPS availability and sensor self-accuracy detection, four sets of tests were conducted as shown in Table 6.5. Each test used a filter process noise covariance matrix set to $10^{-4} * I_{12}$. The first set of tests provided baseline accuracy values for the straight and level trajectory at the test altitudes using a GPS available/unavailable selection as in the previous chapter. The second set of tests allowed GPS availability to vary based on the UAS location within the environment to show any increase or decrease in navigation performance when a more realistic and dynamic model is used. Both of these first two tests assume that the sensor accuracy mode is *On* for GPS and LTE. The third test set switches the sensor accuracy mode to Off for GPS and LTE. This single change allows the second and third sets to be compared in order to determine what, if any, effect this change had on the system. The fourth and final set of tests commanded the UAS to fly a sinusoidal trajectory through the urban environment, starting at an altitude of 75 meters with a maximum altitude of 105 meters and a minimum altitude of 45 meters to ensure the UAS climbs above all buildings and below all buildings at some point along the trajectory. These trajectories are shown in Figure 6.2. The first run of this test had the UAS initiate a climb from the initial location

(Figure 6.2(a)) while the second run had the UAS descend from the initial point (Figure 6.2(b)). Since this is the only test that flies a non-straight and level mission trajectory, it cannot be compared quantitatively to the other tests, but the qualitative results can be used to determine accuracy differences.

Table 6.5: Simulation test matrix for UAS matched model.

Test ID	Altitude	Trajectory	GPS Availability	Sensor Accuracy Mode
1a	125 meters	Straight/Level	Available	On
1b	75 meters	Straight/Level	Available	On
1c	50 meters	Straight/Level	Unavailable	On
2a	125 meters	Straight/Level	Varies	On
2b	75 meters	Straight/Level	Varies	On
2c	50 meters	Straight/Level	Varies	On
3a	125 meters	Straight/Level	Varies	Off
3b	75 meters	Straight/Level	Varies	Off
3c	50 meters	Straight/Level	Varies	Off
4a1	75 meters	Sinusoidal: $h = 75 + sin(\frac{2\pi}{300}x_N)$	Varies	On
4a2	75 meters	Sinusoidal: $h = 75 - sin(\frac{2\pi}{300}x_N)$	Varies	On



(a) Initial climbing

(b) Initial descending

Figure 6.2: Sinusoidal trajectory through urban environment.

6.1.3.2 Unmatched Model

The accuracy of the unmatched filter was determined through a series of tests using 1% uncertainty in the UAS plant dynamics propagation model and the three different random models for UAS plant dynamics propagation with larger uncertainties. Each was conducted at an altitude $h = 75$ meters with varying GPS availability for a straight and level trajectory, similar to the conditions in Test 2b in Table 6.5. Two sets of tests are shown in Table 6.6. The first set simply uses the unmatched models with the same process noise for both the UAS plant dynamics propagation and filter dynamics propagation. This test was conducted for each unmatched model to determine if any one coefficient has a larger effect on the consistency of the filter than the others. With this baseline, the second set of tests raised the process noise entering the filter in order to weight the measurements higher than the filter predictions. This test set was conducted for all three unmatched models to generate comparison information. Additional tests were also run to fine tune the ANEES and ANIS values for unmatched Model 2 since it had the smoothest ANEES time history when untuned.

Table 6.6: Initial simulation test matrix for UAS unmatched model.

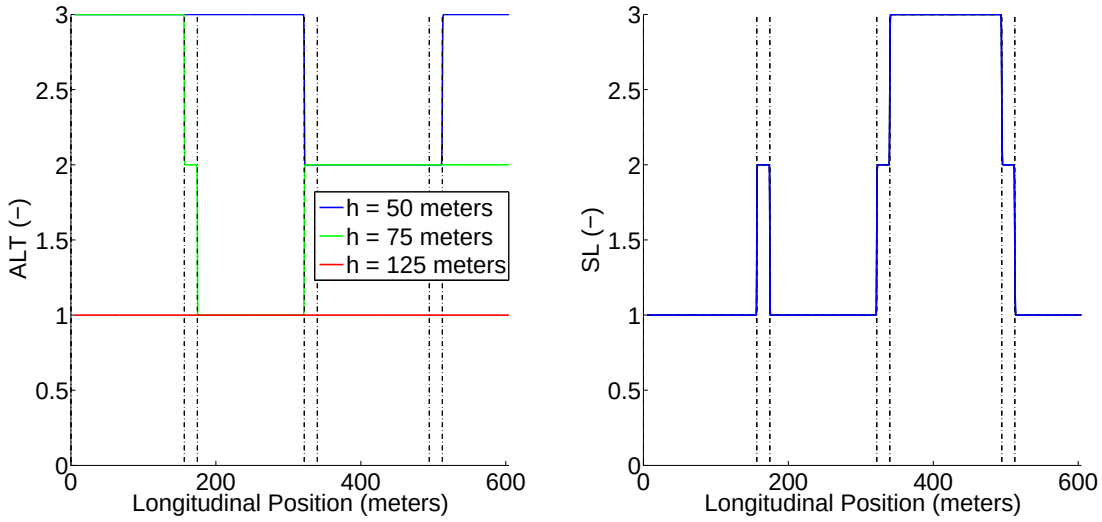
Test ID	Unmatched Model	UAS Plant Dynamics Process Noise Covariance
0	1% Uncertain	$10^{-4} * I_{12}$
1a	1	$10^{-4} * I_{12}$
1b	2	$10^{-4} * I_{12}$
1c	3	$10^{-4} * I_{12}$
2a1	1	$10 * 10^{-4} * I_{12}$
2a2	1	$100 * 10^{-4} * I_{12}$
2a3	1	$1000 * 10^{-4} * I_{12}$
2b1	2	$10 * 10^{-4} * I_{12}$
2b2	2	$100 * 10^{-4} * I_{12}$
2b3	2	$1000 * 10^{-4} * I_{12}$
2c1	3	$10 * 10^{-4} * I_{12}$
2c2	3	$100 * 10^{-4} * I_{12}$
2c2	3	$1000 * 10^{-4} * I_{12}$

6.2 Matched Model Results

This section discusses simulation RMSE results when using a matched model in the heterogeneous urban environment. It first analyzes the navigation accuracy with respect to GPS availability, then with respect to the sensor accuracy mode Off, and finally with the UAS flying sinusoidal trajectories through the urban environment.

6.2.1 GPS Availability

When analyzing the performance of the relative location-based GPS availability algorithm, the relative location trajectories must first be examined to determine where GPS is expected to be available. Figure 6.3 shows the ALT (Figure 6.3(a)) and SL (Figure 6.3(b)) trajectories for each altitude. The pairs of dotted vertical lines in each figure represent the entrance and exit of the intersections with the large white space between the pairs of line representing for the urban canyon. At h = 125 meters the ALT categorization stays above all buildings providing persistent GPS availability. At h = 75 meters the ALT categorization varies from below the tops of all buildings along the first block to above all buildings in the second block, to above the tops of some buildings along the third and fourth blocks. This variation results in sporadic GPS measurements initially, with more regular measurements available as the UAS moves through the environment. When the UAS is at h = 50 meters the ALT categorization begins below the tops of all buildings until just prior to the third block were it moves to below the tops of some buildings due to the adjacent open space along the third block. Once the UAS crosses into the fourth block, ALT is again below the tops of all buildings. At this altitude, GPS measurements will be much less frequent, but still not completely eliminated. This thorough sampling of ALT categories over runs spanning three separate altitudes ensures navigation accuracy through all possible ALT transitions will be reported. Since the same lateral profile is flown for all three altitudes, each of the three SL trajectories is the same, as shown in Figure 6.3(b).



(a) ALT (b) SL

Figure 6.3: Relative location categorization for straight and level trajectories through urban environment.

Figure 6.4 is a representative plot, using data from Monte Carlo Trial #500, to view trends in GPS measurement generation as a function of altitude and and surrounding urban canyon characteristics. It shows that the most GPS measurements occur at $h = 125$ meters since GPS is available to take a measurement every second, while this number decreases as the UAS altitude decreases due to loss of satellite line of sight. For this specific run only 8 of a possible 19 measurements are available at an altitude of 75 meters with 5 of the measurements coming in the second canyon where the UAS is above all buildings. At 50 meters, only 6 of 19 measurements available with multiple measurements only occurring in the third canyon. These values demonstrate the possible ineffectiveness of GPS as a reliable navigation sensor when the UAS is below the tops of buildings.

Figures 6.5 and 6.6 show the horizontal position and altitude/airspeed RMSE trajectories at $h = 75$ meters comparing the GPS constant availability method with the relative location-based algorithm. Each shows the trajectory at $h = 125$ meters for comparison. For the longitudinal position shown in Figure 6.5(a) the two techniques yield similar results, reaching a steady error of roughly 0.8 meters, because the measurements provided by the

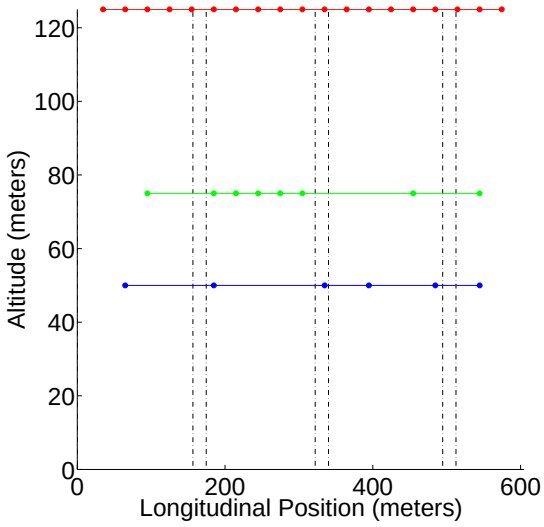
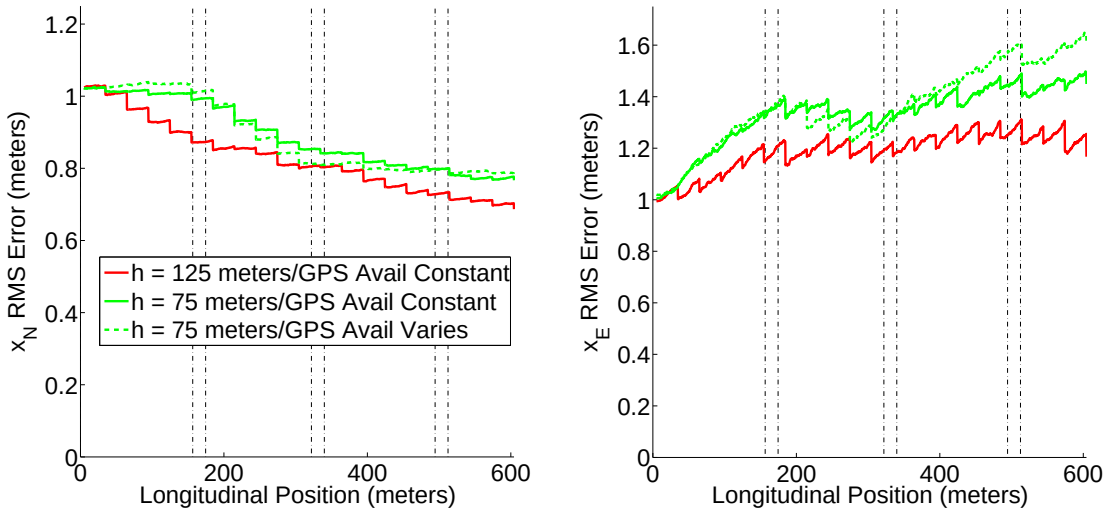


Figure 6.4: GPS measurement locations at each altitude using the varying GPS availability technique.

constant availability method were not sufficiently accurate upon exiting the second canyon to outperform the varying availability trajectory that generally had fewer measurements available. This was due to the UAS *ALT* category switching from above the tops of all buildings to above the tops of some buildings with its larger measurement noise values after that point. Neither method performed as well as the h = 125 meter trajectory as it had more accurate measurements available at each sampling point along the canyons, reaching a steady error of approximately 0.7 meters.

The lateral position trajectories shown in Figure 6.5(b) show the similar sawtooth trend to those in the previous chapter as the UAS attempted to center itself within the canyon. However, there was a linear increase in error at h = 75 meters using both GPS availability algorithms as the available measurements became less accurate in the third canyon. However, halfway though the third canyon, the constant availability trajectory error dropped while the varying availability trajectory error continued to climb, signifying the lack of available measurements for this algorithm. As both error trajectories continued to increase in the fourth canyon, the h = 125 meter error trajectory began to slowly decrease as the

UAS continued to receive more accurate measurements.



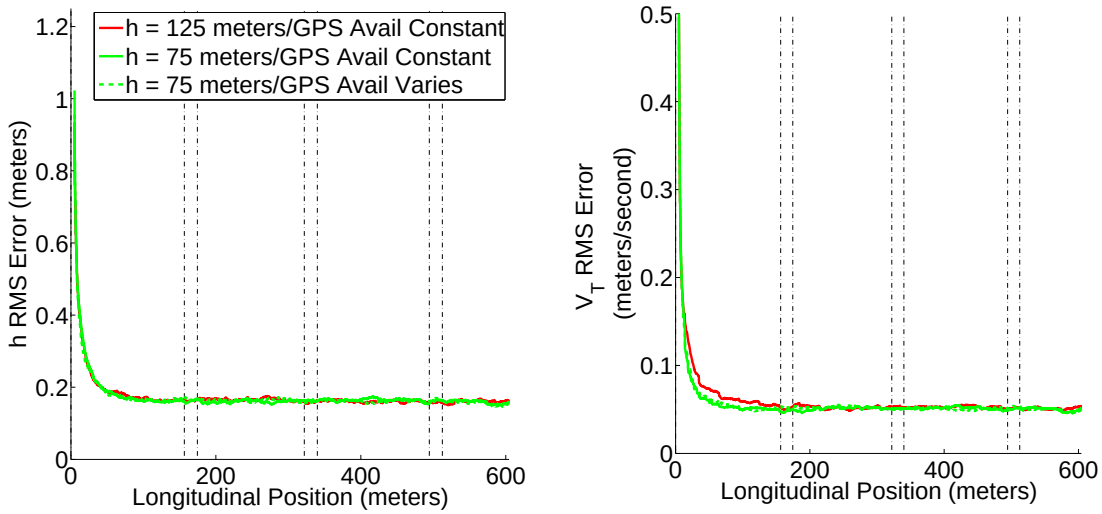
(a) Longitudinal

(b) Lateral

Figure 6.5: Horizontal position RMSE trajectories at $h = 75$ meters for both GPS availability methods with $h = 125$ meters baseline shown.

The altitude trajectories shown in Figure 6.6(a) show no real difference in accuracy performance between the two GPS availability algorithms at different altitudes since the ADS provided an accurate high sampling rate measurement regardless of GPS availability or accuracy. A steady-state error of 0.2 meters in altitude is sufficient to navigate in the urban environment, assuming all sensors are working properly. The airspeed trajectories in Figure 6.6(b) show a steep decrease in error regardless of the GPS availability method since the ADS also measures airspeed. However, the $h = 125$ meter trajectory does not decrease as fast as the $h = 75$ meter trajectories since it is not able to take advantage of optical flow airspeed when above all buildings. However, by the second canyon, a sufficient number of measurements have been received for all three trajectories to approach the same steady-state error value below 0.1 meters/second.

Figures 6.7 and 6.8 show the horizontal position and altitude/airspeed RMSE trajectories at $h = 50$ meters comparing the GPS constant availability method with the relative location-based algorithm. Each shows the trajectory at $h = 125$ meters for comparison.



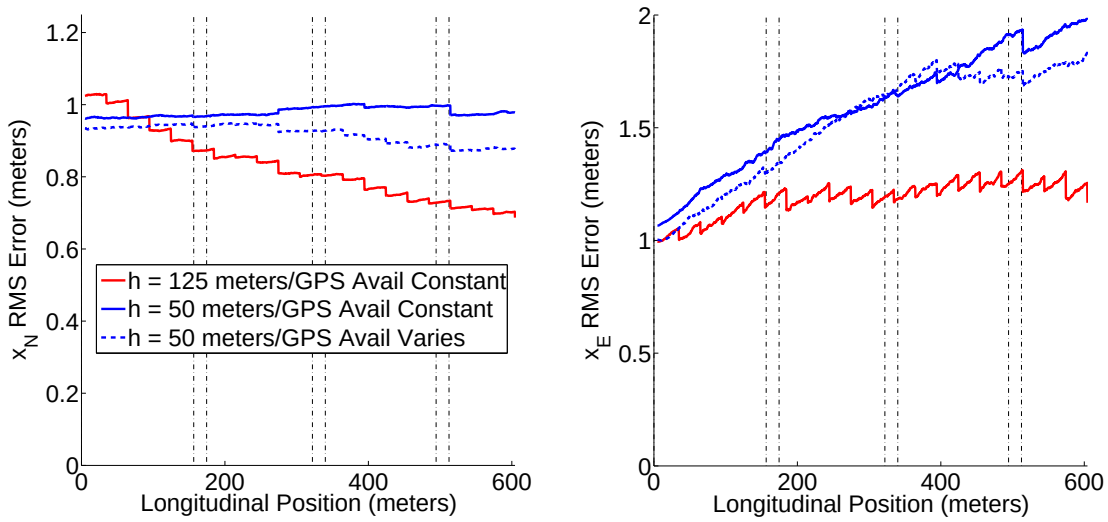
(a) Altitude

(b) Airspeed

Figure 6.6: Altitude and airspeed RMSE trajectories at $h = 75$ meters for both GPS availability methods with $h = 125$ meters baseline shown.

The longitudinal error trajectory in Figure 6.7(a) shows that the error worsens when GPS is unavailable and gets slightly better with variable GPS availability at $h = 50$ meters. The difference is based solely on the number of location measurements received. In the constant availability case, the only measurements received are those from the LTE sensors which are scarce and 4 seconds delayed when received. In the varying availability case, even if a small number of GPS measurements are received, they still augment LTE to provide a more accurate estimate. For the lateral position in Figure 6.7(b), both error trajectories at $h = 50$ meters increase with the constant availability increasing roughly linearly from 1 meter to 2 meters and the varying availability increasing from 1 meter to 1.75 meters until the third canyon and then leveling off as more position measurements are received. For the varying availability case, the error will grow in areas where measurements are scarce and level off in areas where measurements become available with more regularity. These error increases highlight the difficulty in controlling the lateral position of the UAS when so few measurements are available to increase the accuracy of the estimate.

The altitude and airspeed trajectories shown in Figures 6.8(a) and 6.8(b) respectively



(a) Longitudinal

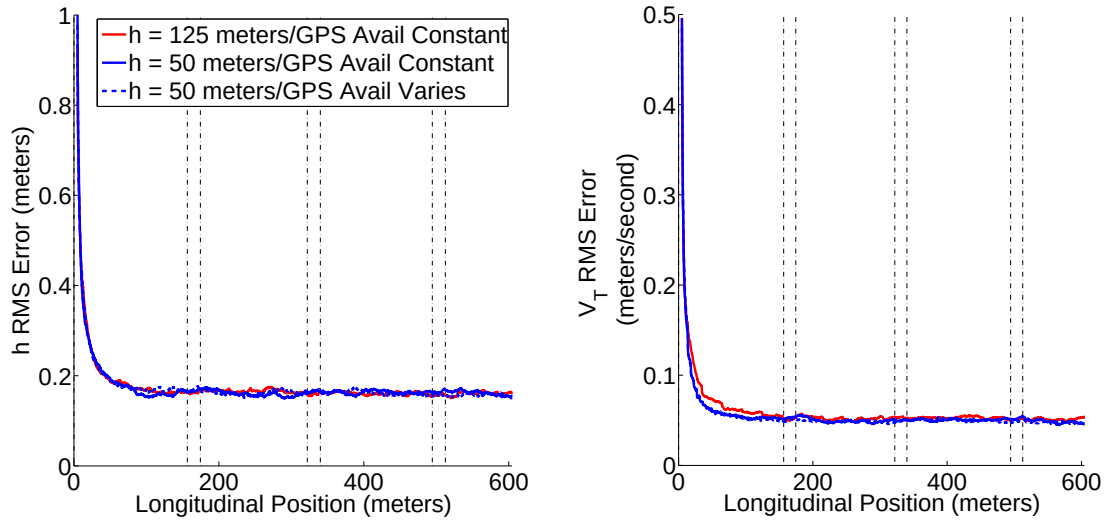
(b) Lateral

Figure 6.7: Horizontal position RMSE trajectories at $h = 50$ meters for both GPS availability methods with $h = 125$ meters baseline shown.

show similar results to those at $h = 75$ meters since the state accuracy is mostly dependent on the accuracy of the ADS altitude and airspeed measurements. As previously seen, the availability of $VISION - OF \times 2$ airspeed measurements allows the error to initially decrease more quickly in the first canyon at $h = 50$ meters than at $h = 125$ meters.

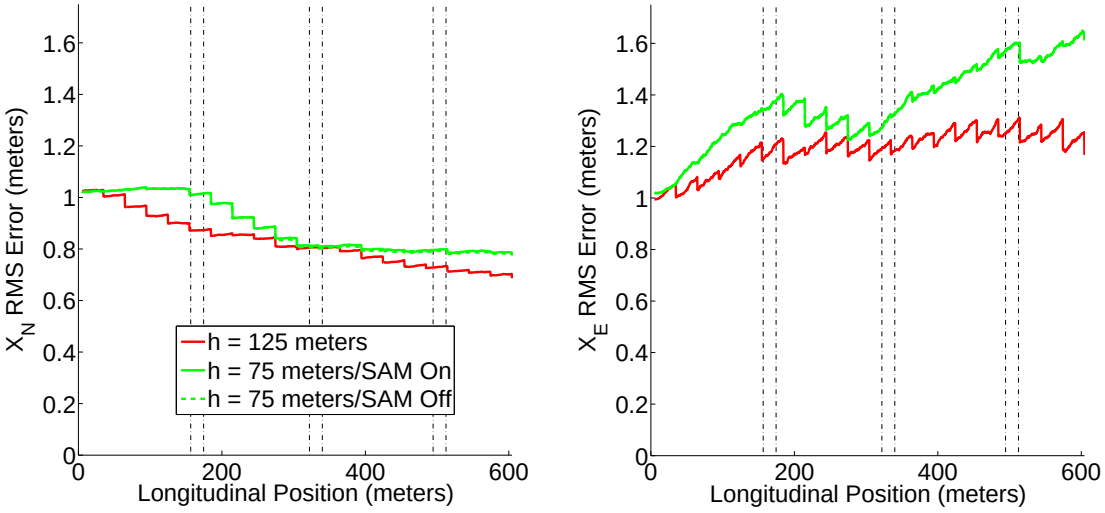
6.2.2 Sensor Accuracy Mode

Figures 6.9 and 6.10 show the horizontal position trajectories comparing the sensor accuracy mode (SAM) On and Off settings at $h = 75$ meters and at $h = 50$ meters. In the longitudinal case shown in Figures 6.9(a) and 6.10(a) and the lateral cases shown in Figures 6.9(b) and 6.10(b) both SAM settings yield approximately the same error trajectories. This behavior was expected in the canyons and intersections because the accurate altitude estimate was used to generate the ALT categorization. However, the fact that there was no variation in the error at the transition points between the two shows that the roughly 1 meter error in the longitudinal position estimate was sufficiently small when using a discrete relative location categorization algorithm.



(a) Altitude (b) Airspeed

Figure 6.8: Altitude and airspeed RMSE trajectories at at $h = 50$ meters for both GPS availability methods with $h = 125$ meters baseline shown.

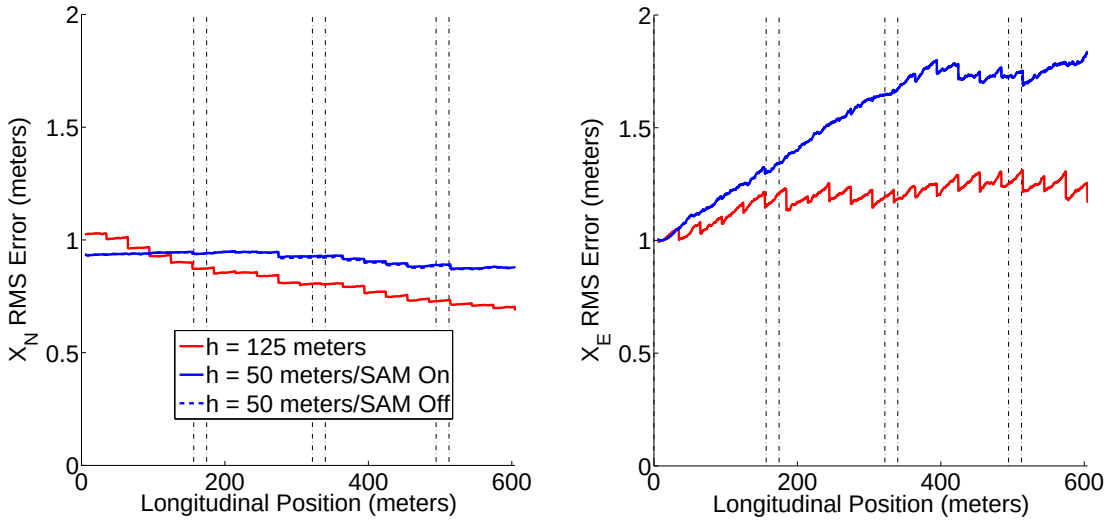


(a) Longitudinal (b) Lateral

Figure 6.9: Horizontal position RMSE trajectories at $h = 75$ meters for both GPS availability methods with $h = 125$ meters baseline shown.

6.2.3 Sinusoidal Flight Path through Environment

When flying a sinusoidal flight path though the urban environment with an initial altitude of $h = 75$ meters, the *ALT* profile will look different than the three trajectories shown in



(a) Longitudinal (b) Lateral

Figure 6.10: Horizontal position RMSE trajectories at h = 50 meters for both GPS availability methods with h = 125 meter baseline shown.

Figure 6.3(a) as it changes categorizations within both the urban canyons and intersections resulting in navigation performance changes. This is demonstrated in Figure 6.11 with both the initial climbing (shown as a solid line) and initial descending (shown as dotted line) flight paths. The *ALT* categorization for the initial climbing flight path starts below the tops of all buildings, then climbs to above the tops of all buildings, and then descends back to above the tops of some buildings by the time the UAS approaches the first intersection with the same transitions in the second intersection. Since the UAS is climbing when it enters the third canyon, it quickly transitions from above the tops of some buildings to above the tops of all buildings until it descends toward the next intersection. For the initial descending trajectory, the *ALT* categorization starts below the tops of all buildings and remains there until the second canyon where transitions to above the tops of all buildings as the UAS climbs. Since the UAS is at its highest altitude in the second canyon with shorter buildings, it stays above the tops of all buildings until it enters the third canyon where it is above the tops of some buildings. The UAS begins its second climb within the third canyon and ends up above the tops of all buildings by the time it enters the fourth

canyon. Since the commanded altitude trajectory was a sine function with a wavelength set to 300 meters, which is much larger than the block length, these frequent transitions between categorization values are to be expected.

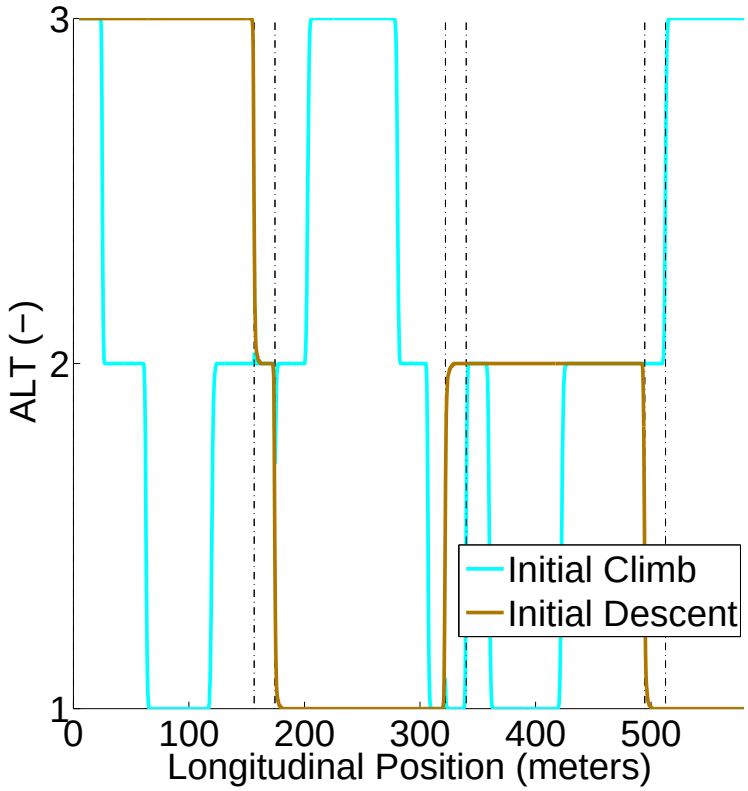
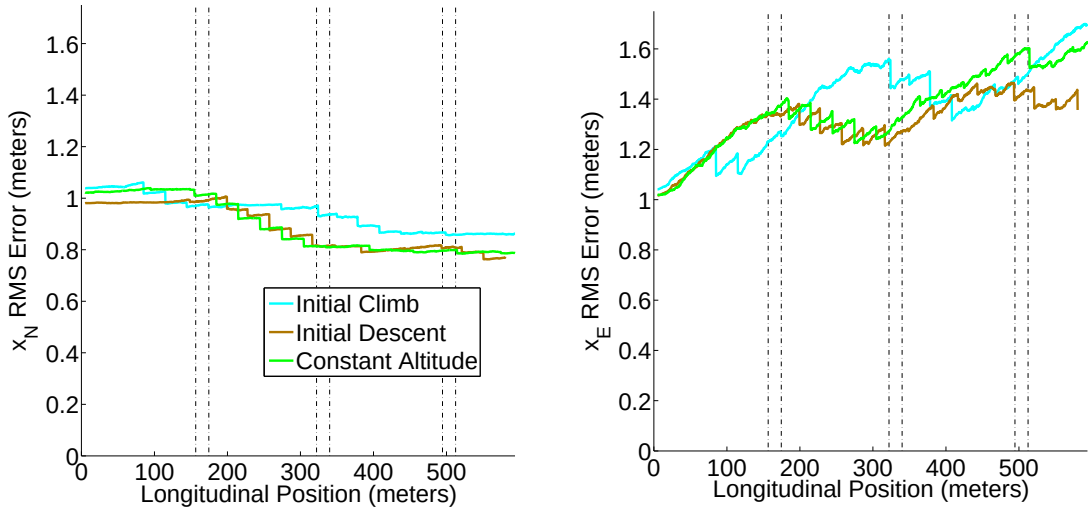


Figure 6.11: ALT categorization for sinusoidal trajectories through urban environment.

Figure 6.12 shows the horizontal position RMSE trajectories for both UAS sinusoidal trajectories and Figure 6.13 shows the altitude and airspeed RMSE trajectories for the same two trajectories with the initial altitude $h_0 = 75$ meters for both. For both horizontal position error trajectories in Figures 6.12(a) and 6.12(b), the initial descent flight path outperforms both the climbing flight path and the constant altitude flight path. This is due to its *ALT* categorization above all buildings for the entire length of the second canyon where the climbing flight is primarily below building tops in the second canyon. In the lateral case, the climbing flight path has a sharp drop in error during the first half of the third canyon as it climbs above the tops of all buildings, but this is brief as it descends back into the

canyon and its lateral error linearly increases. Even though a UAS may not fly these exact sinusoidal flight paths through the canyon, the noticeable trend is that rapid changes in altitude in turn induce rapid changes in RMS error.

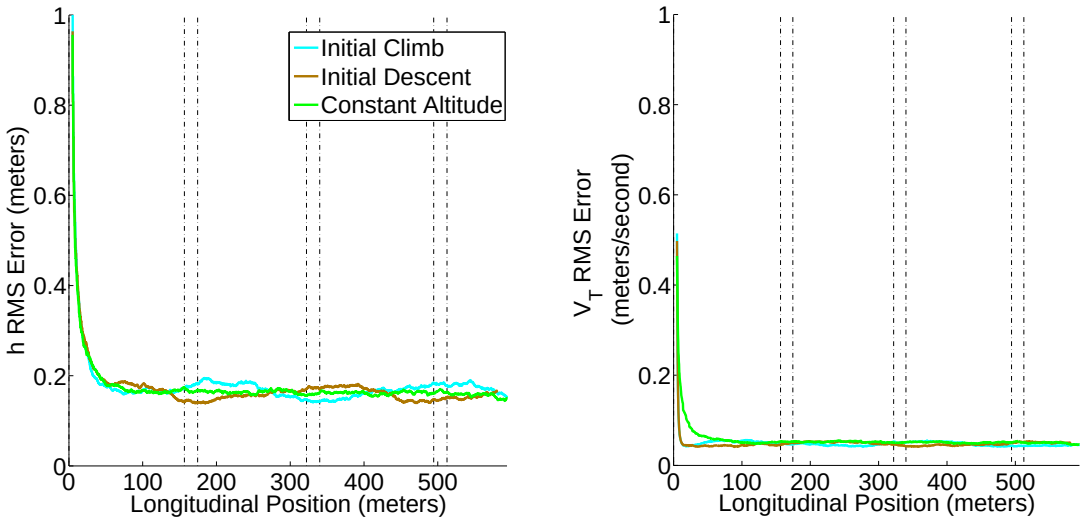


(a) Longitudinal

(b) Lateral

Figure 6.12: Horizontal position RMSE trajectories for sinusoidal trajectories through the urban environment where $h_0 = 75$ meters.

The altitude and airspeed trajectories for the sinusoidal flight paths are shown in Figure 6.13. Both follow the same general trend of a sharp decrease from the initial error to the steady-state due to the high frequency sampling ADS measurements. Figure 6.13(a) shows that the initial climb and initial descent flight paths have small but cyclic changes in altitude error along the length of the urban environment due to the changing ALT categorization. Generally when the UAS is higher in the canyon, the altitude estimate is more accurate and when the UAS is lower in the canyon it degrades slightly. Figure 6.13(b) shows the initial climbing and descending flight path error trajectories descending more steeply then the constant altitude error trajectory. Since all three cases start at $h = 75$ meters the climbing and descending flight paths would have received slightly more accurate initial optical flow airspeed measurements, but all three reached the steady-state error before the halfway point of the first canyon.



(a) Altitude (b) Airspeed

Figure 6.13: Altitude and airspeed RMSE trajectories for sinusoidal trajectories through the urban environment.

6.3 Unmatched Model Results

Since the real-world dynamic model parameters generally do not exactly match those calculated during the system identification process, uncertainty must be applied to these parameters. This true plant dynamics propagation can then be compared to the filtered estimates after filter tuning to determine state accuracy. A straightforward spring-mass-damper example is presented to show the effect of an unmatched model. The system is shown in Figure 6.14.

The linear equations of motion for the spring-mass-damper are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{6.1}$$

where x_1 is the position of the mass in meters, x_2 is the linear velocity of the mass in meters per second, k is the spring constant in Newtons per meter, c is the damping coefficient in kilograms per second, and m is the mass in kilograms.

The measurement model is

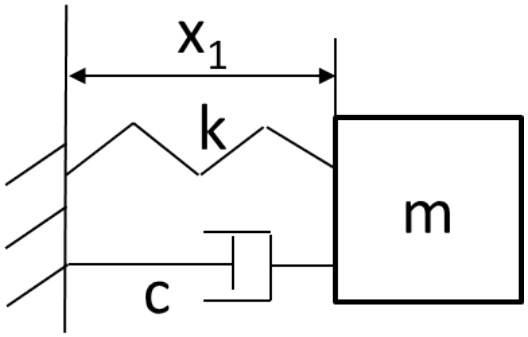


Figure 6.14: Spring-mass-damper system.

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{6.2}$$

where $\mathbf{v} \sim \mathcal{N}(0, R)$. To verify the simulation, it was run using a matched model for 10 seconds with a 0.01 second time step. The model constants were set to $m = 1$ kilogram, $k = 10$ Newtons per meter, and $c = 2$ kilograms per second for both the plant dynamics model and filter model. The plant states were initialized to $x_{1(0)} = 5$ meters and $x_{2(0)} = 0$ meters per second. The filter initial estimates were drawn from a normal distribution using the plant states as the means with the covariances equal to the initial filter covariance matrix

$$P = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.25 \end{bmatrix} \tag{6.3}$$

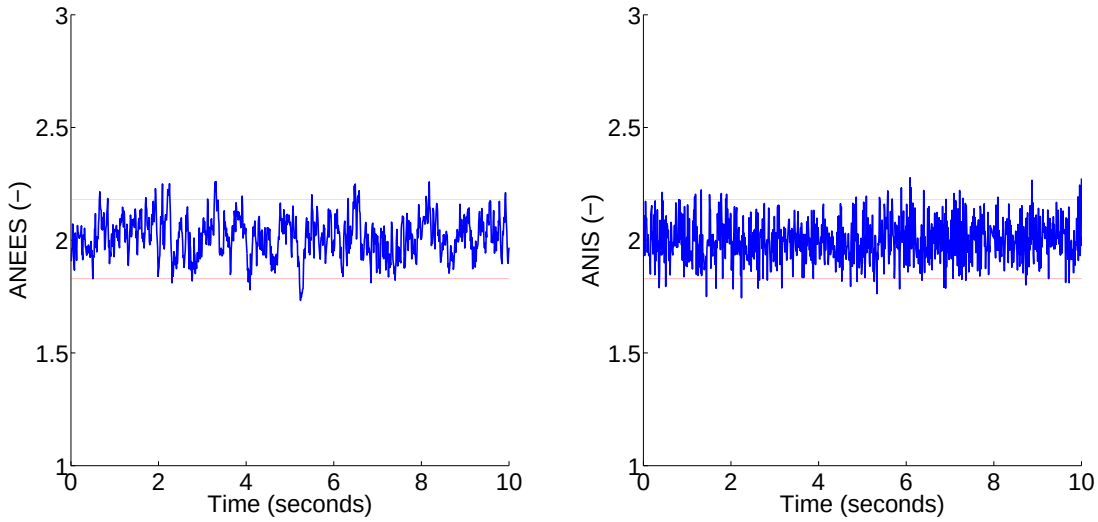
The process noise covariance matrix for both the plant dynamics propagation and filter propagation was set to

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.4}$$

and the measurement noise covariance was set to

$$R = \begin{bmatrix} 1.5 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{6.5}$$

Figure 6.15 shows the ANEES (Figure 6.15(a)) and ANIS (Figure 6.15(b)) trajectories for the spring-mass-damper system when using a matched model for the plant dynamics and filter dynamics. Here, the ANEES and ANIS values stay between the lower (red) and upper (green) chi-squared bounds for a 95% confidence interval, verifying that the filter covariance accurately reflects the error in the states and the innovation covariance accurately reflects the innovations being calculated by the filter.

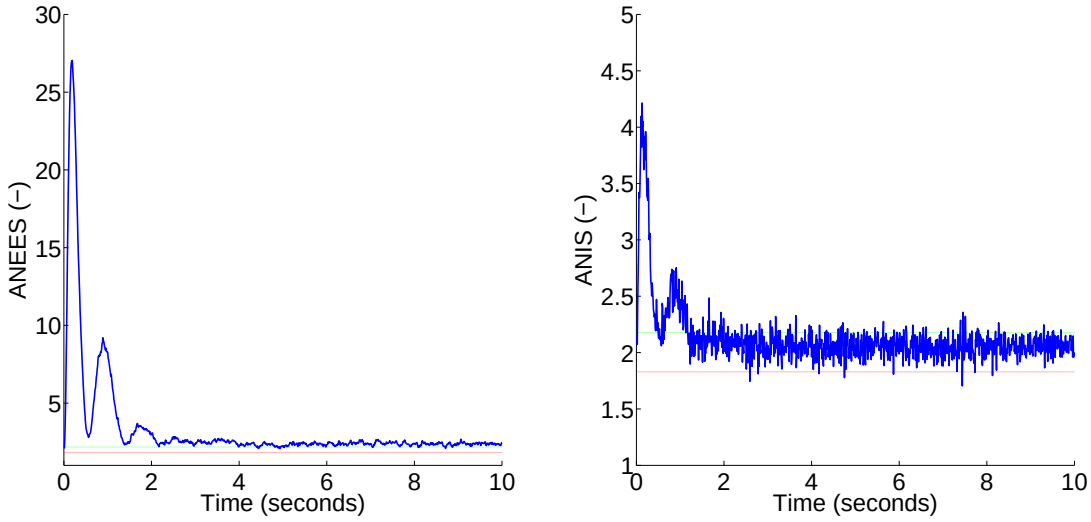


(a) ANEES (b) ANIS

Figure 6.15: Filter metrics for the matched spring-mass-damper system.

To determine the effects of using an unmatched model, the simulation was run keeping all of the above parameters the same except for the plant dynamics model constants. These were assumed to have a 20% uncertainty and were set to m = 0.8 kilogram, k = 12 Newtons per meter, and c = 1.6 kilograms per second. Figure 6.16 shows the ANEES (Figure 6.16(a)) and ANIS (Figure 6.16(b)) trajectories when using the unmatched model when

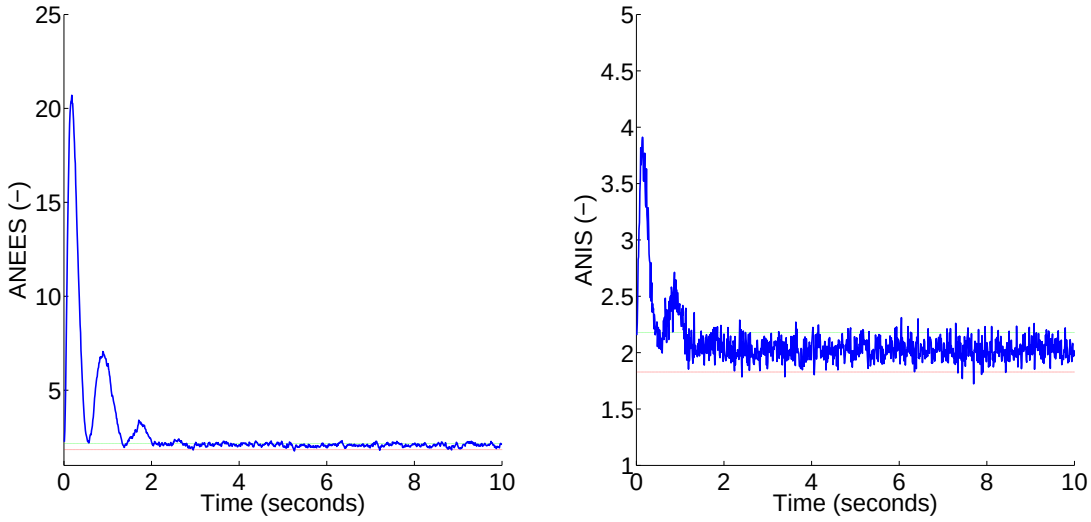
the filter is not tuned. Both the ANEES and ANIS values are above the upper chi-squared bounds, showing that the filter covariance is not predicting sufficient error in the states due to the unmatched model.



(a) ANEES (b) ANIS

Figure 6.16: Filter metrics for the unmatched and untuned spring-mass-damper system.

In order to tune the filter, the simulation was run again with increased filter process noise. Increased filter process noise increases the predicted covariance, which in turns causes an increase in Kalman gain. A larger Kalman gain will cause a larger correction based on the more accurate measurements. To raise the filter process noise, a scalar tuning value q_{tune} was used such that Q_{filter}=$q_{tune} * Q$. This method was selected as it allows the process noise on each state to be increased equally for a given scalar tuning value. It also facilitates straightforward ANEES/ANIS trajectory comparisons, as will be shown later in the chapter. Figure 6.17 shows the ANEES (Figure 6.17(a)) and ANIS (Figure 6.17(b)) trajectories when $q_{tune} = 10^{0.1}$ (1.25). With this filter process noise increase, both the ANEES and ANIS values drop back between the chi-squared bounds, showing that the filter is consistent. This same process will be used subsequently to tune the UAS navigation filter, but with quite different results.



(a) ANEES (b) ANIS

Figure 6.17: Filter metrics for the unmatched and tuned spring-mass-damper system with $q_{tune} = 10^{0.1}$.

6.3.1 Unmatched UAS Models

The same unmatching and filter tuning process can be applied to the UAS navigation simulation using the unmatched models shown below. Table 6.7 shows the nominal model parameters used by the state estimation filter along with slightly different uncertain parameters that are drawn from a uniform distribution centered on the nominal value with one percent bounds on either side. This one percent model is used to determine if filter consistency, as shown through ANEES and ANIS, is attainable without the need for tuning if system identification is able to yield highly accurate parameters for the aircraft.

Table 6.7: Uncertain UAS model aerodynamic coefficients.

Coefficient/Constant	Nominal Model	1% Uncertain
$CL_{\delta a}$	6.79×10^{-2}	6.74×10^{-2}
CL_{β}	-1.30×10^{-2}	-1.30×10^{-2}
CM_1	2.08×10^{-2}	2.07×10^{-2}
$CM_{\delta e}$	5.45×10^{-1}	5.46×10^{-1}
CM_{α}	-9.03×10^{-2}	-9.11×10^{-2}
$CM_{\tilde{q}}$	-9.83	-9.92
$CN_{\delta r}$	5.34×10^{-2}	5.30×10^{-2}
CN_{β}	8.67×10^{-2}	8.75×10^{-2}
$CN_{\tilde{r}}$	-2.14×10^{-1}	-2.16×10^{-1}
I_{xx} (kilogram-meter^2)	2.56	2.58
I_{yy} (kilogram-meter^2)	10.9	10.99
I_{zz} (kilogram-meter^2)	11.3	11.22
$I_{xz} = I_{zx}$ (kilogram-meter^2)	0.5	0.50

Table 6.8 shows the nominal model parameters used by the state estimation filter along with the uncertain parameters for three different randomly generated models. The bound on the percent uncertainty for each parameter, from Ducard [42], as shown in the table, is used to select the value from a uniform distribution centered about the nominal value. All coefficients are unitless while the moment of inertia constants are each in units of $kilogram - meter^2$. In keeping with [42], all other model parameters, listed in Appendix C are assumed to have no uncertainty although realistically they may have at least a small amount of uncertainty as they are typically generated through an experimental process.

Table 6.8: Unmatched UAS model aerodynamic coefficients.

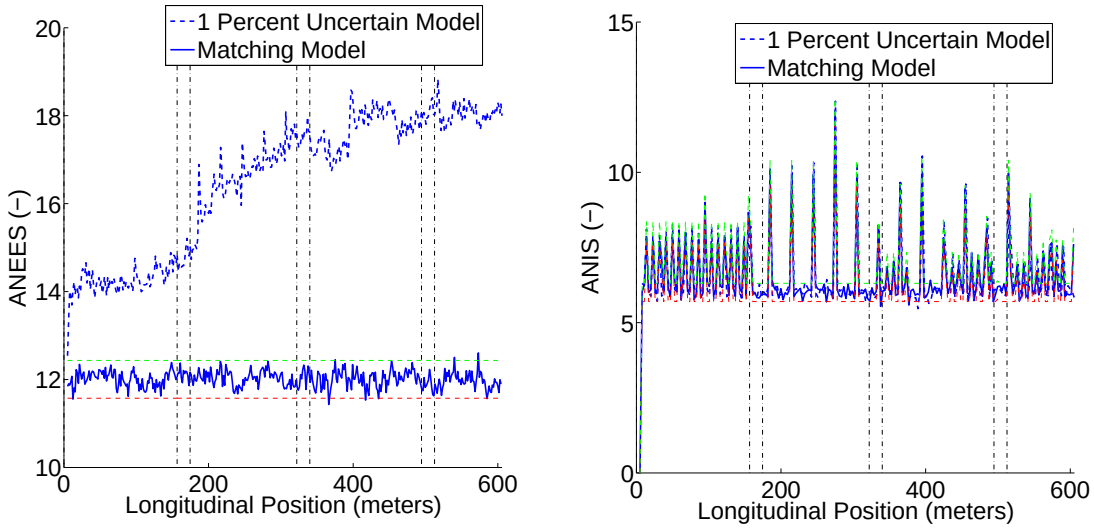
Coefficient/Constant	Nominal Model	Percent Uncertainty	Unmatched 1	Unmatched 2	Unmatched 3
$CL_{\delta a}$	6.79×10^{-2}	10%	7.05×10^{-2}	6.56×10^{-2}	7.43×10^{-2}
CL_{β}	-1.30×10^{-2}	30%	-1.06×10^{-2}	-1.69×10^{-2}	-1.44×10^{-2}
CM_1	2.08×10^{-2}	10%	2.11×10^{-2}	2.21×10^{-2}	2.24×10^{-2}
$CM_{\delta e}$	5.45×10^{-1}	20%	6.34×10^{-1}	5.16×10^{-1}	6.10×10^{-1}
CM_{α}	-9.03×10^{-2}	20%	-7.93×10^{-2}	-9.36×10^{-2}	-9.79×10^{-2}
$CM_{\tilde{q}}$	-9.83	20%	-7.98	-8.77	-9.38
$CN_{\delta r}$	5.34×10^{-2}	10%	4.98×10^{-2}	5.48×10^{-2}	5.14×10^{-2}
CN_{β}	8.67×10^{-2}	10%	8.66×10^{-2}	8.46×10^{-2}	8.10×10^{-2}
$CN_{\tilde{r}}$	-2.14×10^{-1}	10%	-1.98×10^{-1}	-2.07×10^{-1}	-2.14×10^{-1}
I_{xx}	2.56	5%	2.44	2.48	2.45
I_{yy}	10.9	5%	11.18	10.55	11.20
I_{zz}	11.3	5%	11.83	11.08	11.43
$I_{xz} = I_{zx}$	0.5	5%	0.49	0.52	0.49

6.3.2 Untuned Unmatched Model Results

When examining the performance of an unmatched model, ANEES time history can be used to determine if the filter is consistent. If the filter is consistent, the majority of the ANEES values will lie between the upper and lower 95% confidence bounds centered on n, the number of states in the system. With this type of behavior state errors can be considered to have a zero mean and be realistically estimated by the filter covariance. The ANIS is also a measure of filter consistency that examines the innovations using a lower and upper bound much in the same way as the ANEES. It measures how closely the covariance of the innovations at each time step match the filter-calculated innovation covariance.

The first comparison shows the matched model ANEES and ANIS time histories along with the same metrics for the 1% unmatched model in Figure 6.18. For ANEES time

history in Figure 6.18(a), the small differences in the 1% unmatched model results in increasing values in the first through third canyons from 12 to 18. The values level out in the fourth canyon where GPS measurements become scarce. This increase corresponds to an optimistic filter that will underestimate the actual state error, which is not acceptable in an urban environment. Even small differences in the model can drive the ANEES to quickly increase when the error covariance does not accurately predict the actual state error. The ANIS time history in Figure 6.18(b) shows almost identical results for the matched model and the 1% unmatched model. Since the ANEES does not show a consistent filter, but the ANIS does show a consistent filter, small tuning adjustments could be made to the filter to bring the ANEES into the consistency bounds if the true propagation model was known to 1%. However, it should not be at the expense of keeping the ANIS within the consistency bounds as this metric can be used in experimental filter tuning, while the ANEES cannot be used since the true state values are not known. The most important conclusion here is that an accurate system identification process with known measurement noise could yield a consistent real world filter with only small tuning corrections needed.

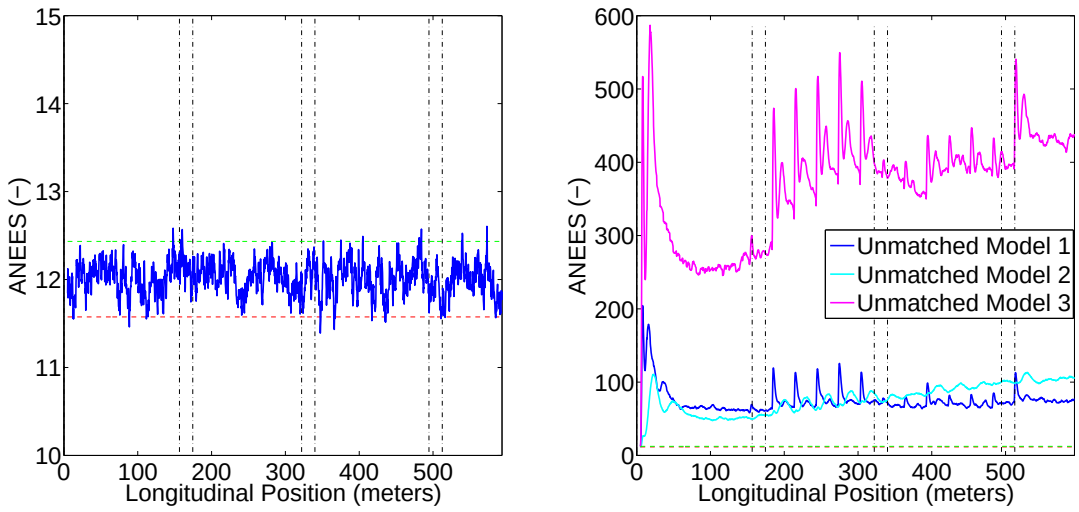


(a) ANEES

(b) ANIS

Figure 6.18: Filter metrics for straight and level trajectories through urban canyon at $h = 75$ meters using 1% unmatched model.

Figures 6.19 and 6.20 show the ANEES and ANIS plots respectively for the matched model along with the untuned and unmatched model tests with higher levels of uncertainty as described in Ducard [42]. The ANEES plots for all three unmatched models (Figure 6.19) are well above the green line that represents the upper bound on ANEES for filter consistency. This indicates an optimistic filter that underestimates the error covariance. Models 1 and 2 yield similar ANEES time histories that hover near 80 with Model 1 reaching a steady state and Model 2 slightly increasing. However, the ANEES value for Model 3 increases dramatically in the second canyon and spikes at almost 550. This is most likely due to Model 3 having the largest deviation from the exact filter dynamics model. This difference between the matched model in ANEES in Figure 6.19(a) and the unmatched models, as well as the differences in ANEES between the unmatched models show just how sensitive the filter is to changes in the model parameters.



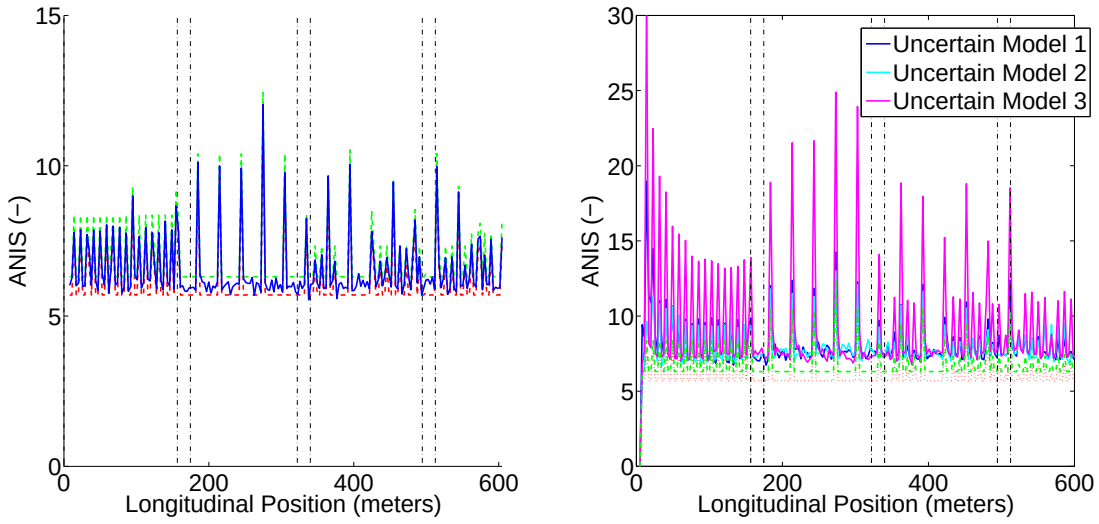
(a) Matched Model

(b) Untuned Unmatched Models

Figure 6.19: ANEES time history for straight and level trajectories through urban canyon at $h = 75$ meters using a matched model and three unmatched models.

The ANIS plots in Figure 6.20 show the same trends as the ANEES plots with the three unmatched model time histories all well above the upper bound for filter consistency. These discrepancies are due to the dependence of the innovation covariances on the predicted state

vector covariance from the current time-step. Since this covariance value is underestimated, as previously discussed, it will result in the underestimation of the innovation covariance, which in turn leads to large and inconsistent ANIS values. With proper tuning of the filter process noise covariance matrix, the ANIS time history should begin to look similar to Figure 6.20(a) where it stays within the confidence bounds indicating that the innovations are being represented accurately by the innovation covariance.



(a) Matched Model

(b) Untuned Unmatched Models

Figure 6.20: ANIS time history for straight and level trajectories through urban canyon at $h = 75$ meters using three unmatched models.

6.3.3 Tuned Unmatched Model Results

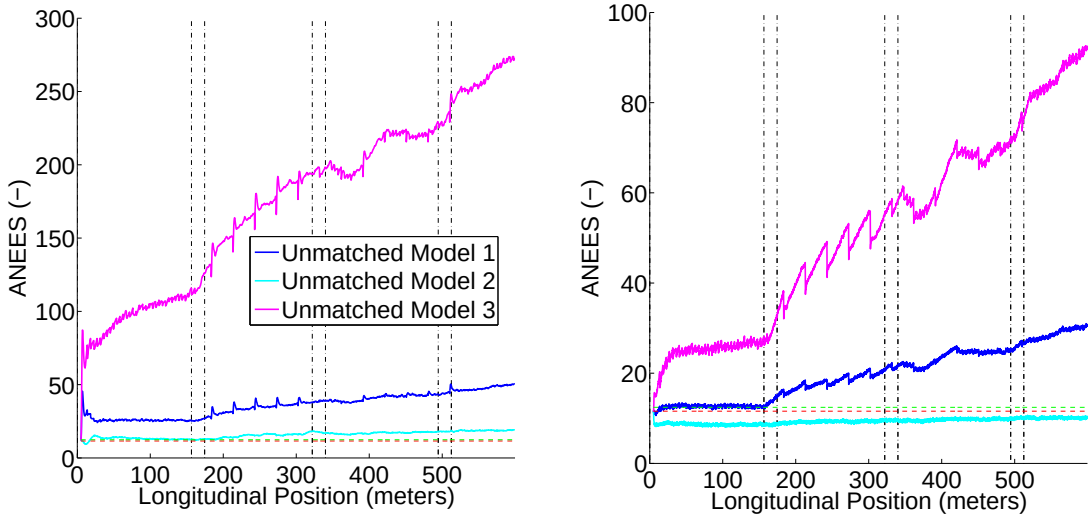
If the filter is not consistent when using the unmatched models, it must be tuned to bring these values within the consistency bounds as shown in Figures 6.19(a) and 6.20(a). This can be accomplished by adjusting the values of the filter process noise and the measurement noise covariance matrices. Since the measurement noise covariance values are known, the filter process noise covariance matrix can be adjusted. In general, the process noise covariance matrix values are artificially increased during tuning to intentionally raise the Kalman gain values. As a result, the filter weights the measurements more heavily. Mathemati-

cally, the increase in the process noise covariance matrix values raises both the predicted error covariance matrix values and the innovation covariance matrix values. With these two quantities increased, both the ANEES and ANIS are decreased for the same state error and innovation values.

6.3.3.1 Filter Metrics Verification

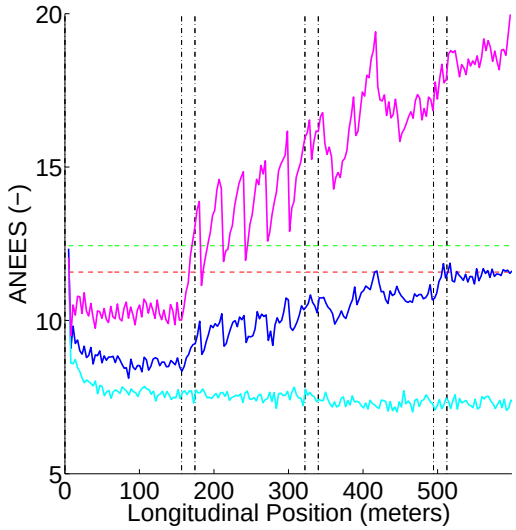
As shown in Table 6.6 tests $2a1 - 2c2$, the filter process noise covariance matrix values are increased by multiples of 10, 100, and 1000 respectively in an attempt to bring the ANEES and ANIS values into the consistent range so that the filter is accurately estimating the state error and innovations. Figures 6.21 and 6.22 show the ANEES and ANIS time histories respectively using these three filter process noise covariance matrix multipliers. As the process noise is increased from 10 (Figure 6.21(a)) to 100 (Figure 6.21(b)) to 1000 (Figure 6.21(c)), the ANEES value decreases toward the consistency range and eventually falls below the range at $q_{tune} = 1000$ for Models 1 and 2. This indicates that the filter has become pessimistic, meaning that its error covariance is larger than the actual state error. However, both unmatched models 1 and 3 showed the ANEES value starting to climb upon reaching the first canyon meaning that the error covariances were decreasing. Because the process noise increase lowers the ANEES values, it would be difficult to use a single parameter to tune the process noise covariance matrix such that the entire length of the ANEES time history would fall between the consistency bounds.

Any time the ANEES time histories decreased due to an increase in the filter process noise, we expect the ANIS time histories to also decrease. As the process noise is increased from 10 (Figure 6.22(a)) to 100 (Figure 6.22(b)) to 1000 (Figure 6.22(c)), the ANIS time history values also decreased through the consistency window until they were well below the lower bound. This indicated that the innovation covariance had increased to the point where the measurements were not being weighted as highly as necessary. Tuning filter process noise then becomes a balancing act to ensure it is not indiscriminately increased so



(a) $q_{tune} = 10$

(b) $q_{tune} = 100$

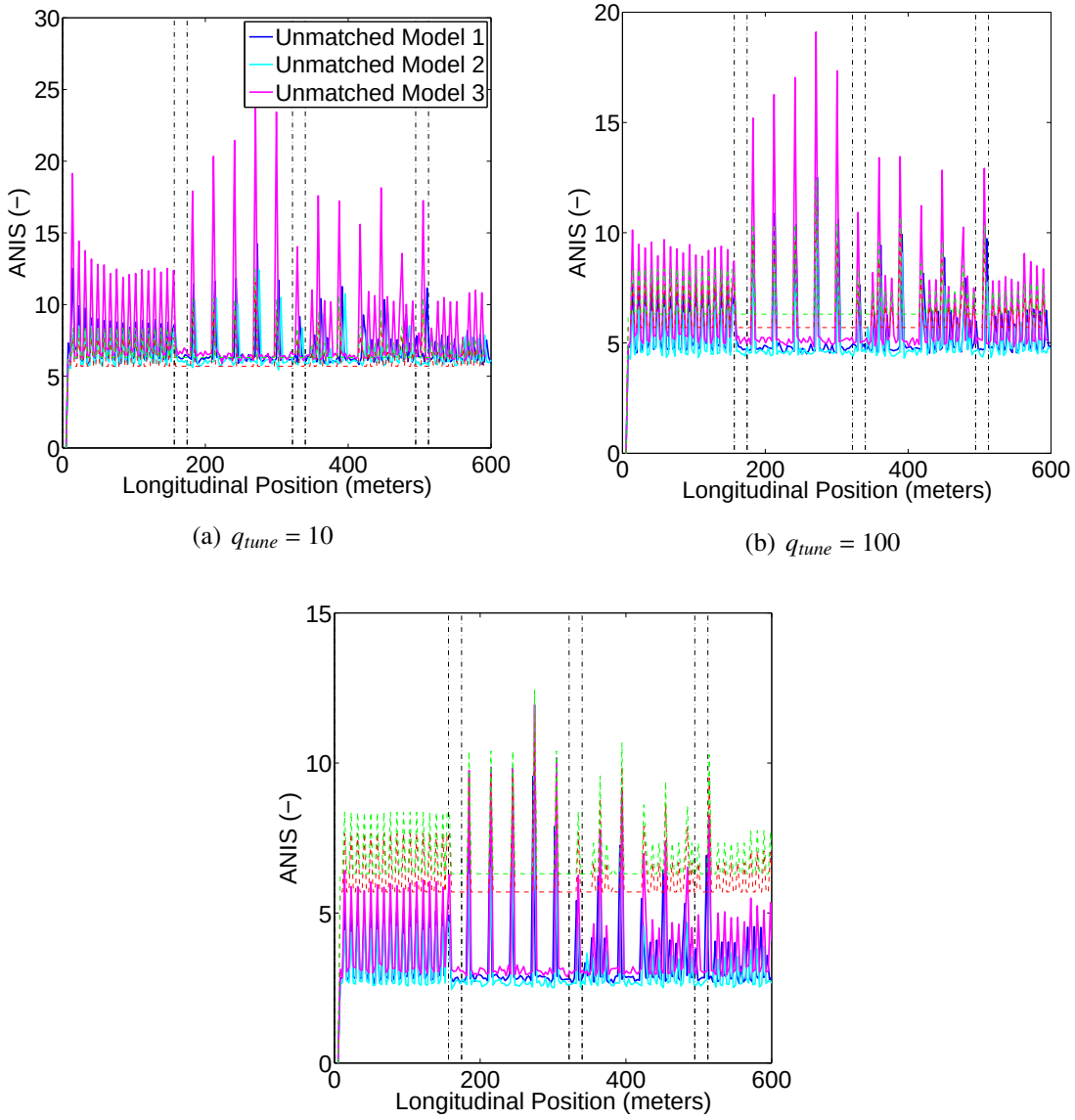


(c) $q_{tune} = 1000$

Figure 6.21: ANEES time history for straight and level trajectories through urban canyon at $h = 75$ meters using a matched model and three unmatched models.

that the ANIS values fall below the filter consistency bounds.

Further ANEES Tuning To further tune the filter, unmatched Model 2 with its fairly consistent ANEES values was selected to be run with q_{tune} values of 25 and 50 as shown in Figure 6.23. In both cases, the ANEES started below the lower consistency bound and



(a) $q_{tune} = 10$

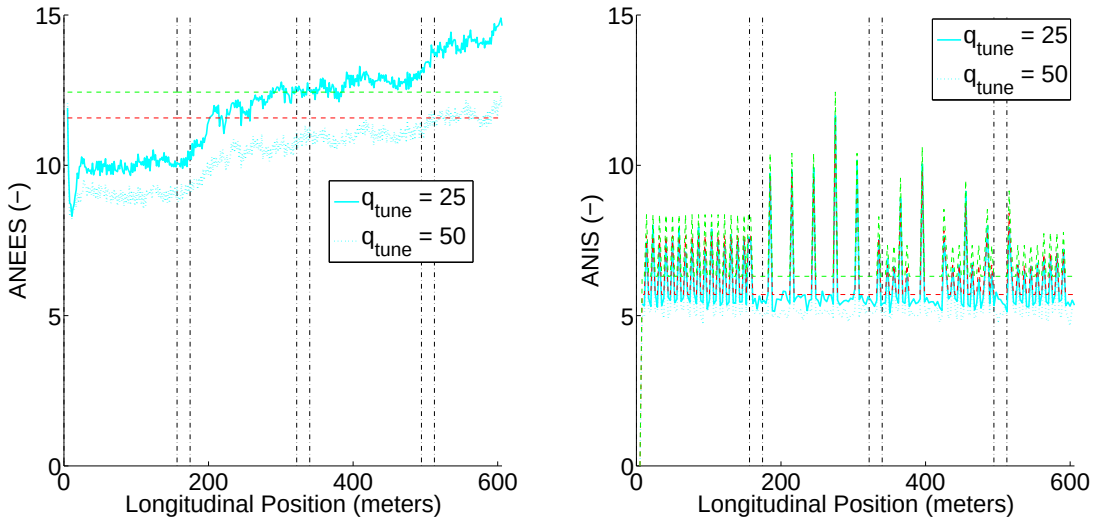
(b) $q_{tune} = 100$

(c) $q_{tune} = 1000$

Figure 6.22: ANIS time history for straight and level trajectories through urban canyon at $h = 75$ meters using matched model and three unmatched models.

increases as measurements are received and error covariance decreases. For $q_{tune} = 25$ the ANEES value stayed within the consistency bounds throughout most of the second canyon, but increased slightly in the third canyon and then climbed in the third intersection. The ANEES value for $q_{tune} = 50$ followed the same trend, but started at a slightly lower value as the filter was initially more pessimistic due to the higher process noise. The ANIS values

dipped just below the lower consistency bound in several locations as well indicating that the process noise was larger than necessary. However, a lower process noise would drive the ANEES above the upper consistency bound even closer to the beginning of the mission. This demonstrates the difficulty in tuning the filter process noise covariance by using a single tuning factor, especially in a dynamically changing sensor environment where more GPS measurements may be available when the UAS is in the second canyon and third canyon than are available in the other canyons.

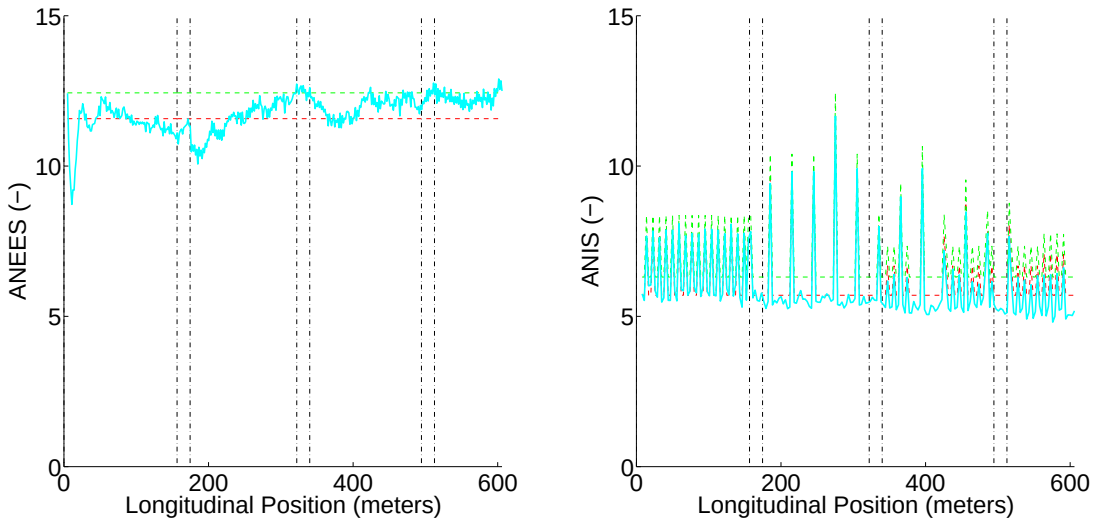


(a) ANEES (b) ANIS

Figure 6.23: Filter metrics for $q_{tune} = \{25, 50\}$ for straight and level trajectories through urban canyon at $h = 75$ meters using Unmatched Model 2.

With increasing ANEES values as the UAS moves through the canyon, one possible technique to improve ANEES-based performance would be to increase the filter process noise covariance matrix values in each canyon. To test this, q_{tune} was initialized to 15 in the first canyon then increased to $25, 35, 50$ respectively in each of the next three canyons. Using this technique, the ANEES value shown in Figure 6.24(a) was within the consistency bounds for parts of the first and second canyon, then for all of the third and fourth canyons. Since it did not creep very high above the upper bound, the ANEES only slightly over-predicted the accuracy of the state estimates in a handful of locations along the canyon.

However, the ANIS values decreased from canyon to canyon as the process noise increased from canyon to canyon and ended up below the lower consistency bound by the second canyon. Again, by increasing process noise, the ANIS showed an inconsistent filter while the ANEES showed a fairly consistent filter.

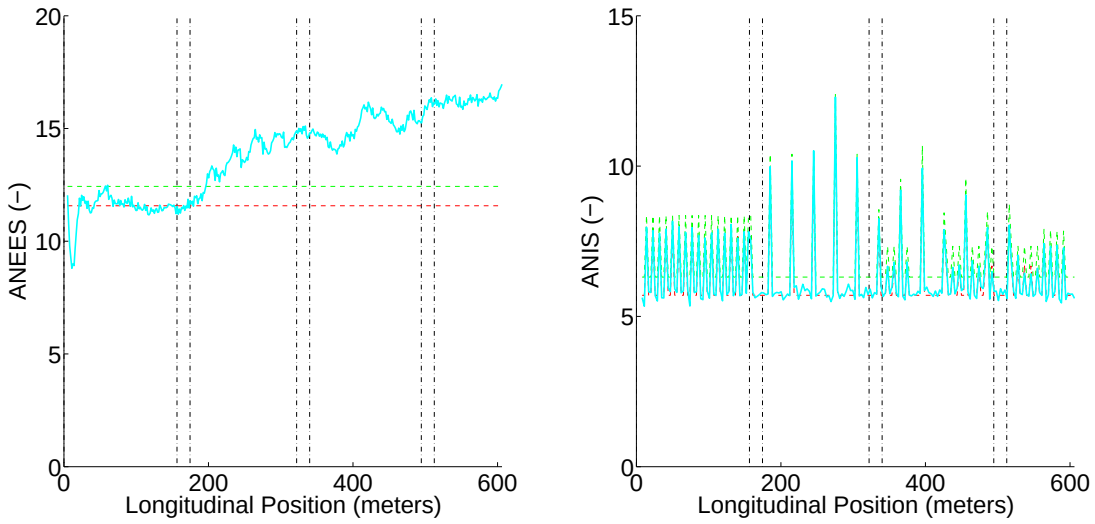


(a) ANEES (b) ANIS

Figure 6.24: Filter metrics for increasing q_{tune} for straight and level trajectories through urban canyon at $h = 75$ meters using Unmatched Model 2.

Further ANIS Tuning As stated in Bar-Shalom *et al.* [110] the ANEES can only be tested in simulation or a controlled laboratory environment since in the real world there is no way to measure the propagated UAS plant dynamics states. The ANIS test would be used to tune the filter in this situation since the measurements would be available. Following that idea while still considering ANEES, the simulation was run a with $q_{tune} = 15$ to determine if this value would yield a consistent filter in terms of ANIS values since Figure 6.23(b) shows the ANIS below the lower consistency bound for $q_{tune} = 25$ and Figure 6.21(a) shows the ANEES above the upper consistency bound for $q_{tune} = 10$. The results of this run, shown in Figure 6.25, reaffirm the expected trends. Figure 6.25(a) shows a slightly decreased ANEES in the first canyon, but then increased continuously once the

UAS entered the second canyon. As previously discussed, this increase can only be counteracted with canyon by canyon process noise increases. However, the ANIS, shown in Figure 6.25(b), stayed between the consistency bounds for much of the time history, with short intervals just below the lower bound. This indicates that a slightly smaller filter process noise would push the ANIS values into the consistent filter range, but would cause a further increase in ANEES.



(a) ANEES (b) ANIS

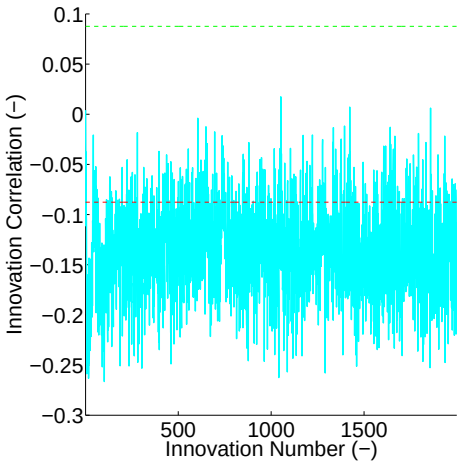
Figure 6.25: Filter metrics for q_{tune} = 15 for straight and level trajectories through urban canyon at h = 75 meters using Unmatched Model 2.

Innovation Autocorrelation The third requirement for a consistent filter is that the innovation sequences are considered to be white, meaning that they are statistically uncorrelated. For each of the sensors, the expectation is that the autocorrelation statistic has a zero mean and a variance of $1/N$. For q_{tune} = 15, Figure 6.26 shows the autocorrelation statistic along with the 95% bounds for each of the six active sensors. As seen in Figures 6.26(a) and 6.26(b), the high sampling rate IMU and ADS sample autocorrelation statistics are offset below the lower bound, which is expected as the innovation values decrease with an increase in filter process noise. The lower sampling rate GPS and LTE sample autocor-

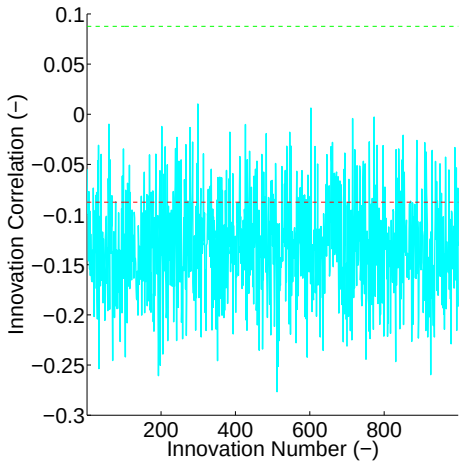
relation statistic time histories, shown in 6.26(c) and 6.26(d) have only a few data points (4 for GPS and 3 for LTE). The GPS autocorrelation statistic is slightly above the upper bound for the complete innovation sequence. However, it is difficult to draw any conclusions for GPS autocorrelation since only the four innovations that were present at the same time-step in each run are shown, due to varying GPS availability. The LTE autocorrelation statistic gives the best results as the values stay within the bounds for all four innovations. This is because the innovations are spread out four seconds in time, are sampled at the same time-steps for each run, and measure the states with the most uncertainty. Figures 6.26(e) and 6.26(f) show the sample autocorrelation statistical values for the VISION optical flow sensors. Both stay in the consistent range near zero for most of the trajectory showing that the appropriate amount of process noise is used for these particular sensors to ensure their innovations are uncorrelated.

6.3.4 Filter Accuracy Results

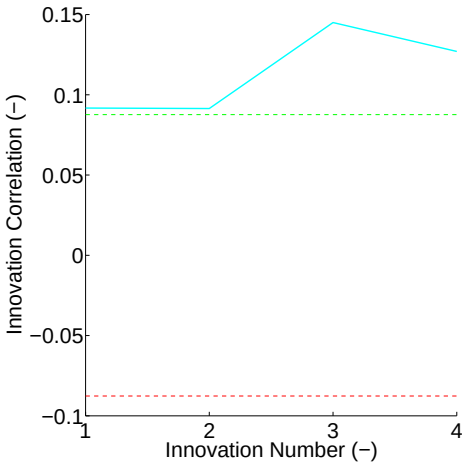
Once the filter is tuned, the RMS error is examined to determine the accuracy of the filter in estimating the propagated UAS plant dynamics states. RMS error plots are shown in Figure 6.27 - Figure 6.29, with time history plots for the unmatched and untuned case, the tuned ANEES case where q_{tune} increases in each canyon, and the tuned ANIS case where q_{tune} = 15. The three different cases were selected to show how different filter tuning methods affect the accuracy of the different states. In Figure 6.27 the three RMS position error plots are shown for the three different cases. For both longitudinal (Figure 6.27(a) and lateral (Figure 6.27(b)) positions, the untuned trajectory was generally between the two other trajectories in terms of accuracy. The longitudinal RMS position error continually increased along the urban environment with small decreases when measurements were received. This growth in error demonstrated the problem facing a UAS with an unmatched model when it does not have a reliable high sampling rate position measurement to measure a state that is not controlled. The results for the lateral RMS position error were slightly better due to



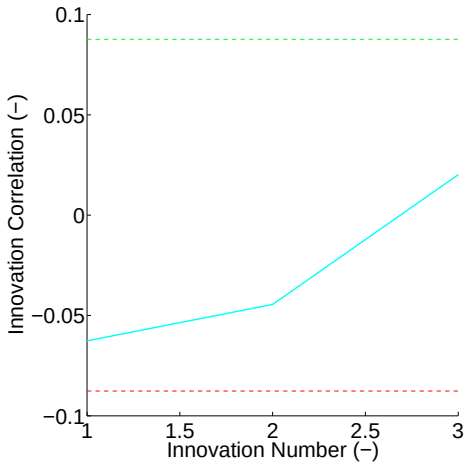
(a) IMU



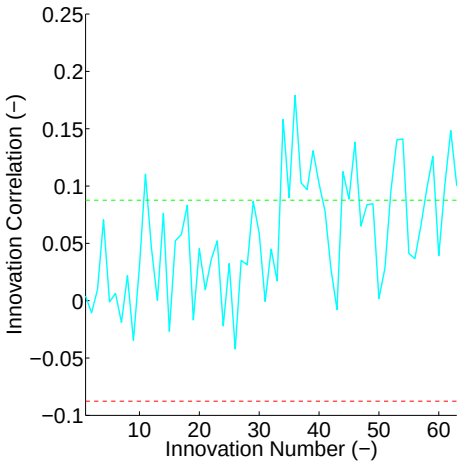
(b) ADS



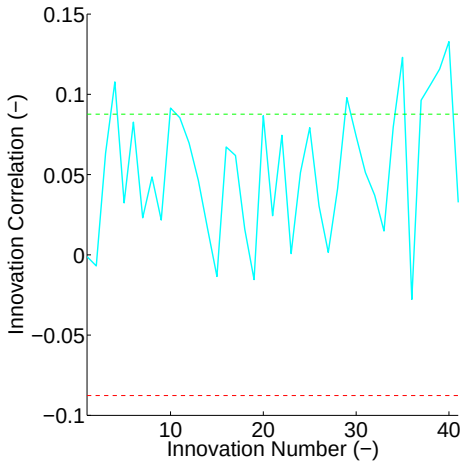
(c) GPS



(d) LTE



(e) VISION-OF LEFT



(f) VISION-OF RIGHT

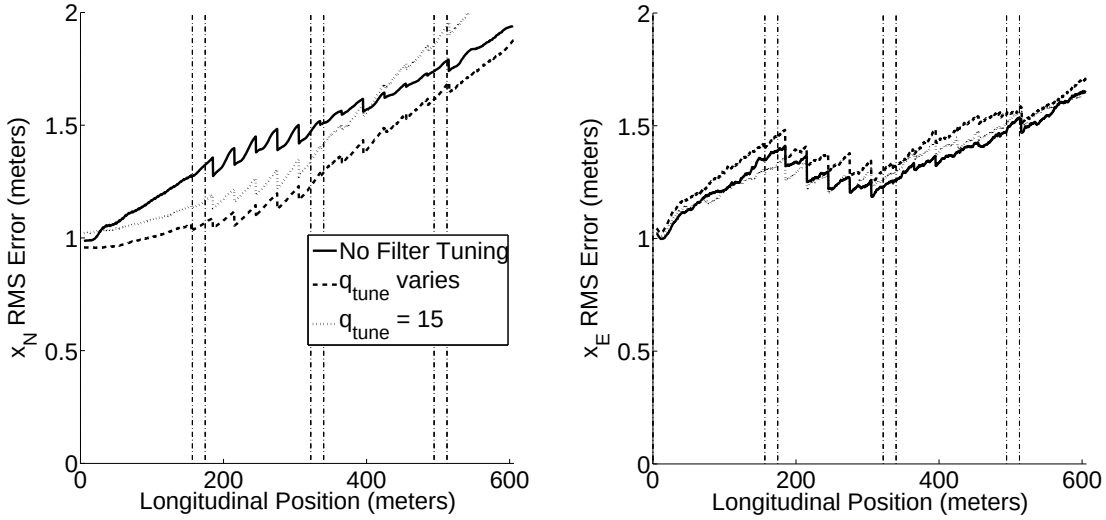
Figure 6.26: Innovation autocorrelation statistic time history for $q_{tune} = 15$ for straight and level trajectories through urban canyon at $h = 75$ meters using Unmatched Model 2.

the decrease in error in the second canyon. Here, the more frequent measurements produce accurate estimates as the controller drives the UAS to the center of the canyon.

Altitude RMS position error (Figure 6.27(c)) shows large differences between the untuned case and the two tuned cases. In the untuned case, after the initial sharp descent, the error climbed through the second half of the first canyon due to the difference in models. This demonstrates the effect of having a high sampling rate measurement combined with unmatched models. Even though the measurements were available, the process noise was not artificially inflated sufficiently to weight them higher in the Kalman gain calculation until the second canyon when more GPS measurements became available. For the two tuned cases, the trajectory followed the same path as the matched model tests since the increased process noise allowed for the frequently generated ADS altitude measurements to have a larger effect of the magnitude of the Kalman gain.

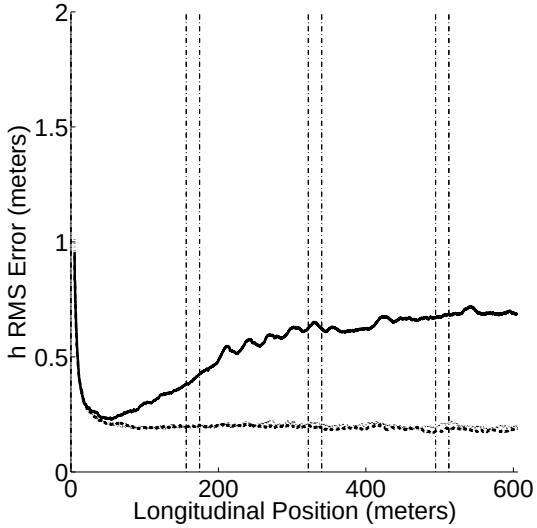
In Figure 6.28 the RMS error plots are shown for airspeed (Figure 6.28(a)), angle of attack (Figure 6.28(b)), and angle of sideslip (Figure 6.28(c)). The untuned airspeed trajectory shows a large spike within the first canyon corresponding to the jump in altitude error shown in Figure 6.27(c). As the altitude error began to level off in the second canyon, the airspeed error also leveled off with decreases each time a GPS measurement was received. Interestingly, the loss of the computer vision airspeed measurement had virtually no effect in any of the intersections, most likely due to its slow 0.15 second sampling rate and delay period. Once the filter was tuned, the two other cases showed small differences in error trajectories. As expected, the two trajectories essentially overlap in the first canyon since the q_{tune} value is the same for both. The case with variable q_{tune} was slightly more accurate for the remainder of the urban environment, but all three cases reach the same RMS error value by the end of the fourth canyon.

The angle of attack and angle of sideslip plots show small errors due to the availability of the ADS measurements throughout the environment. For these two states, the tuned cases were slightly less accurate than the untuned case again because artificial process



(a) Longitudinal position

(b) Lateral Position

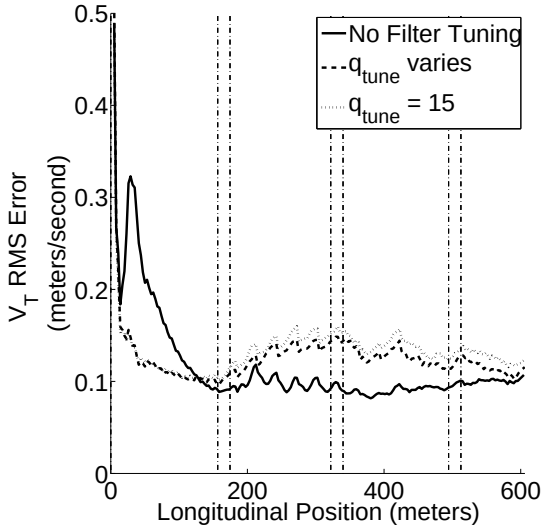


(c) Altitude

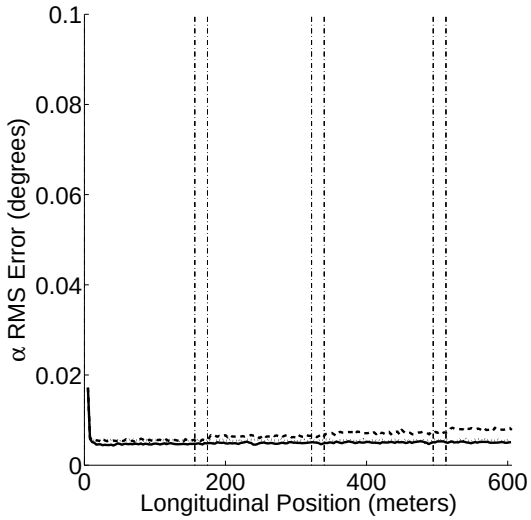
Figure 6.27: RMS position error time history for untuned and tuned cases for straight and level trajectories through urban canyon at h = 75 meters using Unmatched Model 2.

noise was injected into the filter. This was most clearly demonstrated in Figure 6.28(b) where the errors for the three cases were nearly identical in the first canyon, but as the process noise increased in the varying case, its error become larger in each canyon.

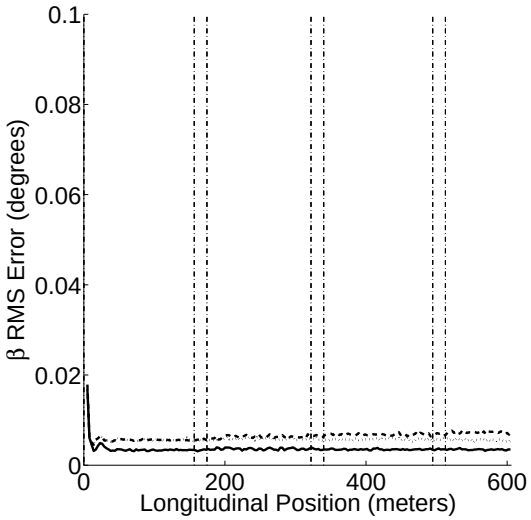
RMS error plots for the Euler angles in Figure 6.29 have similar tendencies to the angle of attack and angle of sideslip RMS plots. The untuned cases showed the most accurate



(a) Airspeed



(b) Angle of attack

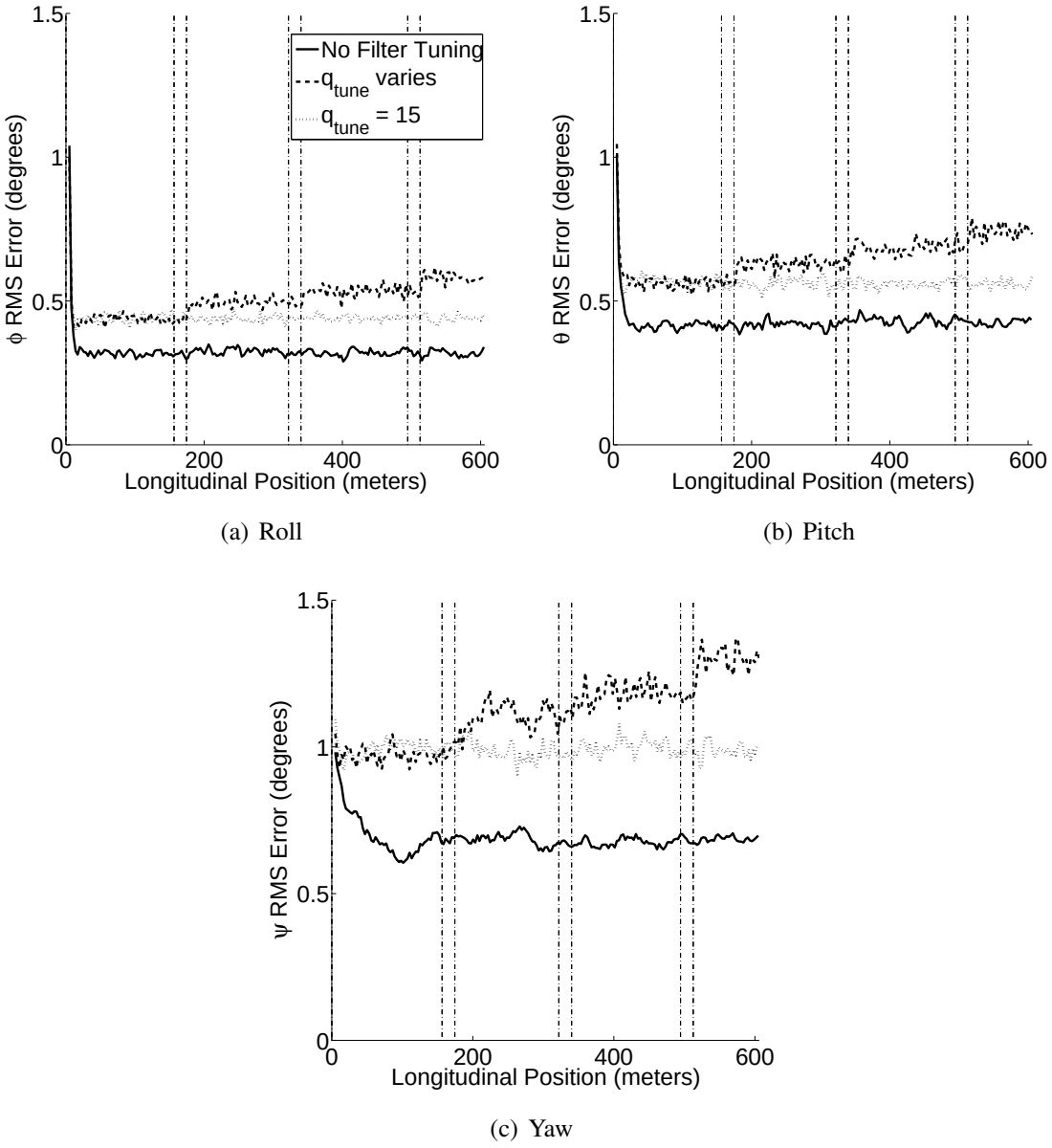


(c) Angle of sideslip

Figure 6.28: RMS airspeed and wind angle error time history for untuned and tuned cases for straight and level trajectories through urban canyon at h = 75 meters using Unmatched Model 2.

results while the varying process noise case showed increasing RMS error as the process noise was increased entering each canyon. In the case of the constant q_{tune} value, the error was virtually constant. However, as shown in the RMS position plots, these angle errors caused increasing position errors when horizontal position measurements were not

available.



(a) Roll

(b) Pitch

(c) Yaw

Figure 6.29: RMS Euler angle error time history for untuned and tuned cases for straight and level trajectories through urban canyon at h = 75 meters using Unmatched Model 2.

When using an Extended Kalman Filter with an unmatched model, the untuned case generally yields the most accurate estimates since it has the smallest filter process noise. The only state where this does not hold along the entire trajectory is altitude. Unfortunately, the untuned case also yields the most inconsistent filter, meaning that the UAS would not

actually know that these state estimates are the closest to the propagated UAS plant dynamics states as compared to the other tuned cases. These tradeoffs demonstrate why a good system identification process that allows the filter to be tuned using very little process noise is important to accurate and safe navigation within the urban canyon.

6.4 Chapter Summary

This chapter discussed UAS urban canyon simulations using both a matched model and process noise covariance values as well as unmatched models with both untuned and tuned process noise covariance values. In the matched model scenarios, simulations were run when varying GPS availability, switching the sensor accuracy mode on and off, and when following constant altitude sinusoidal trajectories through the urban environment. When comparing results with constant GPS availability rates to the varying GPS availability rates, lateral position navigation accuracy generally became more accurate as more GPS measurements were received at each of the altitudes. For altitude and airspeed accuracy, the higher sampling rate ADS provided sufficient data for the GPS measurements to have a minimal effect. The sensor accuracy mode switch did not have an effect on the navigation accuracy since a discrete relative location model was used, which would yield the same GPS and LTE accuracy for altitude bands that were always larger than the error in altitude. Again, the lateral position accuracy became worse in the canyon as fewer GPS measurements were received. When sinusoidal flight paths were flown through the canyon, position accuracy changed quickly as a function of *ALT* category, forcing a tradeoff between low altitude missions and higher navigation accuracy. This would motivate the use of VISION/IMU sensor localization when deep in the canyon with degraded GPS capabilities.

When unmatched models were used filter tuning was necessary to ensure the EKF-generated covariance properly represented the state error and that the innovation covariance was accurately reflecting the true difference between generated and predicted measure-

ments. This process was difficult as it was necessary to consider the ANEES, the ANIS, and the autocorrelation statistic metrics concurrently when tuning. The strategy of simply adding process noise without altering the calculated measurement noise eventually brought all three metrics near their consistent ranges near the beginning of the environment, but could not handle the resulting ANEES increase when more measurements becoming available further into the environment. Although ANIS and autocorrelation could be tuned this way, ANEES could not. Instead, it required the use of canyon-specific process noise covariance values, which caused the ANIS and autocorrelation to drop below the lower consistency bounds. This issue highlighted the difficulty in tuning process noise in an EKF when the UAS plant dynamics model has uncertainty in several of its parameters. For all but altitude, the untuned filter performed the same or better than the tuned filter due to the lack of artificial process noise added to the system. The only problem is that the user could not achieve this level of accuracy in real-world applications since the user would need to tune ANIS and autocorrelation to be confident that the filter error covariance properly captures the level of error in the estimates, especially in the narrow and safety-critical urban environment.

CHAPTER 7

Conclusions and Future Research Topics

To conduct UAS missions in a GPS degraded or denied urban environment, government, commercial, and academic organizations are actively researching multi-sensor solutions that take advantage of existing man-made electromagnetic signals. This dissertation investigated navigation accuracy for fixed-wing UAS using combinations of navigation sensors over a suite of urban environments. For the first time the attributes of the LTE OTDOA geolocation technique were used as part of a UAS navigation solution. A custom MATLAB® simulation incorporated and assessed a suite of statistical accuracy metrics using models of the urban environment, a UAS LQR controller and fixed-wing dynamics model, sensor availability models based on the UAS location within the environment, sensor measurement models utilizing filtered sensor state noise values, and an Extended Kalman Filter along with an Ensemble Kalman Filter using state augmentation to account for sensor measurement delay. Vertical and horizontal relative location partitions were created to categorically model the environment to infuse location-dependent sensor noise. Straight and level trajectories were flown in a homogeneous environment using matching true UAS dynamic models and filter models. The same trajectories, in addition to sinusoidal trajectories, were flown in a heterogeneous environment using matching models, while the straight and level trajectories were flown using unmatched models.

7.1 Conclusions

The first contribution of this research is the creation of an open-source end-to-end modular system-level UAS simulation that can account for many of the factors that are present in the real world. These include the integration of delayed measurements into the state estimation filtering module, considering several different sensor signals with different sampling rates and delays, and using dynamically changing sensor availability and sensor noise properties. Modeled sensor systems include the traditional IMU and ADS, along with man-made electromagnetic GPS and LTE signals. Multiple computer vision accuracy results were also used including feature matching to correct IMU output and optical flow for ground speed. By including these user-customizable capabilities in the simulation, it can be used in its current form with any filtered sensor state noise values of sensors with known sampling rates, time delays, and availability models. The urban environment simulation and modeling capability allows integrated navigation performance to be tested in cities as they currently exist or when the landscape is customized in a way that facilitates evaluation of GPS and LTE signal degradation.

The second contribution of this research is the development of environment-based availability (GPS) and noise covariance (GPS and LTE) models as a function of UAS location within the urban environment. A suite of discrete vertical and horizontal categories was developed using published signal availability and noise covariance data. This technique takes advantage of available building and street maps, considered as cloud data, for UAS navigation in combination with these onboard sensors. A benefit of using the map and sensors together is more rapid execution of navigation algorithms with less required computational requirements than using raw sensing capabilities only. This technique also allows more accurate sensor noise covariance values used in the filter based on location within the environment.

The third contribution of this research is a large compilation of sensor measurement noise statistical information for many classes of current and proposed UAS sensors based

on published literature throughout various engineering disciplines. By using this database, future researchers can focus their efforts on improving navigation algorithms in simulations rather than quantifying sensor noise on hardware. This data also allows comparison of measured levels of sensor/integrated navigation system noise in existing systems to data generated by simulations with new fused data sets or filters.

From on simulation-based testing, the first conclusion of this research is that fusion of IMU and ADS, GPS, LTE, and computer vision measurements using an EKF provide an increasingly accurate longitudinal position estimate for a UAS flying through an urban environment for missions up to 200 seconds, but a lateral estimate with decreasing accuracy as the UAS tried to center itself using slow, delayed, and inaccurate measurements. In a homogeneous environment the navigation accuracy was at its best when a vision-based feature matching IMU correction system was available. In a heterogeneous environment when the vision-based feature matching IMU correction system was fully removed, the accuracy was at its best when above all buildings as GPS measurements were more often available than when deeper in the urban canyon environment. Over the 200-second simulation duration with a consistent environment, the longitudinal accuracy does not decrease for this filter. EnKF results generally show increasing longitudinal and lateral error at 20 seconds, with the longitudinal error still increasing at 200 seconds as the lateral error reaches steady-state.

The second conclusion of this research is that the introduction of LTE OTDOA as a horizontal positioning technology does NOT provide any additional accuracy benefit with its long delay period of at least 4 seconds, low sampling rate, and relatively poor accuracy. However, it did stabilize the longitudinal error when no other horizontal position measurements were available. Given the current proprietary nature of the execution of the LTE positioning by the major carriers along with sparse and widely varying available literature, this was to be expected. However, adding this signal to the simulation did provide a persistent measurement within the urban canyon where GPS would likely be degraded or unavailable. Providing the framework for LTE will allow it to be used in future work as

its geolocation accuracy continues to improve and the network evolves towards complete independence from GPS. The latter would allow navigation to continue even with spoofed or jammed GPS signals.

The third conclusion of this research is that filter tuning for this specific UAS configuration in a heterogeneous environment is difficult to accomplish when attempting to use ANEES, ANIS, and autocorrelation as joint metrics. For the given process noise model and unmatched UAS, ANEES tuning required canyon-specific process noise covariance matrix values to counteract decreasing estimate uncertainty, while ANIS tuning required constant process noise covariance values regardless of canyon. Autocorrelation results showed that the tuning process becomes even more difficult when different sensors are tuned using different levels of artificial process noise.

7.2 Future Research Topics

This dissertation exposes many potential future areas of research to explore. These include further investigation of different sensor combinations with characterized noise values and state estimation filters, refining the UAS plant dynamics model to better understand the process noise, accounting for wind, understanding how the current urban environment affects sensor performance, and more concretely modeling GPS and LTE sensor accuracy to give finer resolution. Additional sensors not considered in this work could also be investigated. For example, downward-facing VISION sensors could take advantage of local landmarks such as road markings, traffic lights, etc, to provide position measurements. A LiDAR system should also be explored as this type of sensor can give extremely accurate measurements in a known environment while complementing both GPS and LTE. Markers and/or active sensors mounted in street and building infrastructure could also provide key data particularly in poorly-lit GPS-denied regions. Another state estimation filter type to consider is the particle filters that do not make any assumptions about Gaussian distributions of the

state estimate, process noise model, or measurement noise model. This gives flexibility to model process noise and sensor noise using the distributions that best fit experimental data. By weighting particles appropriately, it may reach a steady-state longitudinal error in less than 200 seconds. Several particle filter variants of this type exist. Each propagate particles differently and/or weights them differently based on incoming measurements. Direct urban environment navigation benefits include quick localization if the UAS loses knowledge of its location and trajectory and generating position estimates using inaccurate high sampling rate position measurements with non-Gaussian noise.

One assumption made in this research is that there is no wind affecting the flight of the UAS in the urban environment. Future work should include wind models in the UAS plant dynamics modeling realistic conditions in an urban environment. Wind can gust and can change direction quickly when deflected by buildings, billboards, overpasses, etc. These non-negligible wind effects may limit flight operations for certain size/weight UAS on any given day.

Another enhancement to the model is the characterization of the process noise covariance matrix since this matrix is generally not diagonal with equal values for all states. It is typically unique to each vehicle based on the how closely the vehicle's actual dynamics during flight, match the idealized fixed-wing rigid-body dynamics model. An estimate can be generated during real-world testing by tuning the filter for different forms of the process noise matrix since its form, in terms of individual values and correlated states, may vary in different flight configurations. During this tuning process the autocorrelation statistic should be monitored along with ANEES and ANIS to ensure all three filter consistency metric are as good as possible.

The use of real-world building and obstacle maps in the navigation system could provide the UAS with situational awareness that could be used in making decisions on which sensors to include in the navigation solution or how to weight different sensor measurements. The main challenges are how to store this information in an easy to retrieve format

either on board or off board but accessible, and how to account for temporary obstacles or permanent obstacles not updated on a map. This research suggested one basic method of storing building and obstacle information in an easy-to-retrieve format. However, as the cities get larger, the shapes of the buildings vary, and several different types of obstacles are present, storage of this information becomes increasingly difficult. The other challenge is to have the most current map of the area of operations at all times as well as an obstacle sensing capability. This is necessary because the many programs in existence that have modeled large cities around the world may or may not include obstacles such as overpasses, antennas, spires, billboards, construction cranes, power lines, trees, monuments, statues, etc. Even with the most up-to-date maps, construction equipment or other temporary obstacles can pop-up and must be avoided. Addressing this challenge will require the ability to quickly update maps, whether as part of the pre-mission checklist or en route using some type of cloud computing through Wi-Fi, the cell network, or satellites. Also, when the map fails to identify an obstacle, the UAS must have the capability to detect obstacles with enough standoff distance to safely avoid them using sensors such as LiDAR. Since obstacles are generally static, a sense and avoid type solution is not necessary. Rather, this problem can leverage existing collision detection and avoidance technology, such as that being developed in the emerging autonomous car field, to generate logical algorithms. By generating solutions to these challenges, UAS would have the necessary capabilities to safely fly in any urban environment. Real-world UAS mission issues such as loss of sensor data stream should also be considered to determine how these events affect map/sensor navigation.

Another future research area to consider is to more robustly model the interaction between the LTE transceiver and LTE network as well as the GPS receiver and GPS network in terms of providing position measurements for a UAS in the urban environment. While GPS models currently exist, the challenge would be to model the effects of the urban environment on the GPS signal and verify these values with real-world measurements. For the

LTE network, the open knowledge base from which to draw is currently small and requires system-level access and understanding of the cellular network. Also, the proprietary nature of the different cell carriers networks has slowed the ability of LTE OTDOA to be better understood and used. To solve this problem with real-world utility, the LTE network and associated position accuracy must be characterized using assumptions about which towers work together on the same network and how the building and obstacle infrastructure alters signal propagation. Since the OTDOA method is similar to the method used by GPS to generate measurements, basic knowledge of the LTE network layout could provide a more fine method of generating LTE-based measurements. Should the carriers ever allow the user to request OTDOA position data directly from the network, these models could be verified. In parallel it is expected that the carriers, their equipment providers, and the cellular network regulatory bodies will continually look for ways to provide OTDOA position fixes much more accurately and faster than the current 4 – *plus second* delay. Eventually, this GPS-independent man-made electromagnetic signal could completely replace GPS as an accurate and reliable positioning method for not just UAS navigation, but all urban environment applications. Until then, the LTE network can provide a level of cybersecurity in navigation by providing backup position measurements used to detect spoofed GPS signals.

APPENDIX A

Review of Optical Flow Techniques

A.1 Calculation Methods

Barron et al. [72] describes nine methods for calculating optical flow, divided into four broad categories: differential-based, region-based, energy-based, and phase-based. They argue that many of the methods share three unique stages during the optical flow field calculation. These stages include

1) Smoothing in order to extract signal structure of interest and signal to noise enhancement

2) Basic measurement extraction including both spatio-temporal derivatives and local correlation surfaces

3) Integration of the measurements, producing the 2-D optical flow vector field.

Differential-based methods, specifically Lucas/Kanade (LK), and the region-based methods are further described since these methods are primarily used in calculating the optical flow fields in published UAS urban canyon navigation work [69], [33].

A.1.1 Differential Based Methods

Lucas/Kanade (LK) [120] & Horn/Schunck (HS) [66] are two common differential based optical flow calculation methods used quite extensively in the field of optical flow. LK is a local method using a weighted least squares approach in a neighborhood of interest, while

HS is a global method that attempts to minimize an energy-like function consisting of a brightness term and a global smoothness term. The benefit of using HS is that it yields flow fields with 100% density, while LK, with its sparser flow fields, is more robust to noise [72]. These beneficial characteristics of each have been used to develop a combined local-global (CLG) method described in [121], discussed later in this section.

A.1.1.1 Lucas/Kanade and Horn/Schunck Algorithms

The primary assumption for both LK and HS is that the brightness constancy

$$I(x,y,t) = I(x+\delta x, y+\delta y, t+\delta t) \tag{A.1}$$

of the image does not change from frame to frame. Expanding the right hand side of (A.1) using a Taylor Series Expansion yields

$$I(x,y,t) = I(x,y,t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + H.O.T. \tag{A.2}$$

After neglecting the higher order terms and canceling out the $I(x,y,t)$ term in (A.2), the result is

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0. \tag{A.3}$$

Both sides of (A.3) are divided by δt and the limit is taken as $\delta t \to 0$, to give

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0. \tag{A.4}$$

Rewriting (A.4) using partial derivative shorthand notation and letting $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$ yields

$$I_x u + I_y v + I_t = 0. \tag{A.5}$$

Since (A.5) has two unknowns for each pixel, it cannot be used alone to solve for the

local optic flow. However, based on the form of (A.5), it can be rewritten as

$$(I_x, I_y) \bullet (u, v) + I_t = 0. \tag{A.6}$$

Moving I_t to the right hand side of (A.6) and dividing by the magnitude of the brightness gradient,$\|\nabla I\|$, yields

$$proj_{\nabla I}(u, v) = \frac{-I_t}{\|\nabla I\|}. \tag{A.7}$$

(A.7) gives the component of the optic flow vector that is parallel to the brightness gradient as shown in Figure A.1. In order to determine the normal component of the optical flow vector, more constraints are needed.

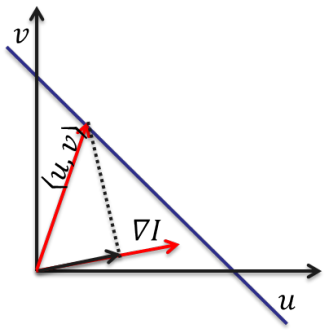


Figure A.1: Local optic flow vector component in the direction of the brightness gradient.

The Lucas and Kanade method [120] uses a neighborhood weighted least squares approach to estimate the optical flow for local regions of interest in an image rather than requiring the entire image be processed for each frame. The additional underlying assumption is that the neighborhood Ω contains n pixels having the same local flow velocity values, $\hat{v}$, giving rise to

$$E = \sum_{q \epsilon \Omega} W^2(q_i)[\nabla I(q_i) \bullet (u, v) + I_t(q_i)]. \tag{A.8}$$

Using the standard weighted least squares equation [72]

$$A^T W^2 A \hat{v} = A^T W^2 b \tag{A.9}$$

where

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \quad W = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & w_n \end{bmatrix} \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \tag{A.10}$$

and solving for $\hat{v}$

$$\hat{v} = (A^T W^2 A)^{-1} A^T W^2 b \tag{A.11}$$

yields

$$\hat{v} = \begin{bmatrix} \sum_i w_i^2 I_x(q_i)^2 & \sum_i w_i^2 I_x(q_i) I_y(q_i) \\ \sum_i w_i^2 I_x(q_i) I_y(q_i) & \sum_i w_i^2 I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_i w_i^2 I_x(q_i) I_t(q_i) \\ \sum_i w_i^2 I_y(q_i) I_t(q_i) \end{bmatrix} \tag{A.12}$$

where each entry is summed from $i = 1 : n$. This method allows for the local flow to be calculated at any pixel as long as $(A^T W^2 A)^{-1}$ exists for a given neighborhood.

Horn and Schunck sought to minimize the energy-like function

$$E = \int\int (I_x u + I_y v + I_t)^2 dxdy + \alpha^2 \int\int \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right) dxdy \tag{A.13}$$

over the image that includes a brightness term and global smoothness term. The global smoothness term consists of the summation of the squares of the partial derivatives of the two optical flow components in the x and y directions. A complete derivation with flow field solution can be found in [66].

To minimize (A.13) the iterative equations

$$u^{k+1} = \bar{u}^k - \frac{I_x [I_x \bar{u}^k + I_y \bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2} \tag{A.14}$$

and

$$v^{k+1} = \bar{v}^k - \frac{I_y[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2} \tag{A.15}$$

can be used where $\bar{u}^k$ and $\bar{v}^k$ are local flow averages at time step k.

Reference [66] suggests that the number of iterations for an image is dependent on the largest uniform region that must be filled in. In absence of that knowledge, the cross-section of the image may be used to estimate the number of iterations needed.

A.1.2 Region-Based Methods

Region-based methods calculate the optical flow field by matching a region or feature of an image, known as a patch P_v in consecutive frames. This patch, with $v \times v$ dimensions, is initially centered at (i, j) at time t, and moves to $(i+u,\ j+w)$ at time $t+1$.

At time $t+1$, a matching patch is found by minimizing

$$M = \sum_{(x,y)\epsilon P_v} |I_{t+1}(i+x+u,\ j+y+w) - I_t(i+x,\ j+y)| \tag{A.16}$$

comparing the pixel intensities of the patch at t and $t+1$ for each each pixel. The square target region is defined as $2n+1 \times 2n+1$ region with lower left corner and upper right corner pixel coordinates of $(i-n,\ j-n)$ and $(i+n,\ j+n)$ respectively, meaning that (u, w) resides in the $\{-n, ..., n\} \times \{-n, ..., n\}$ space. After the algorithm computes M for each pixel in the $2n+1 \times 2n+1$ region, it picks the minimum value for M and uses the (u, w) pair at that value as the motion of the center of P_v [33]. Since this method is very cumbersome due the number of calculations required for each new image frame, [33] suggests that a quantized version of the algorithm found in [122] be used in urban UAS applications.

A.1.2.1 Real-time Quantized Region Based Method

In order to limit the number of times (A.16) must be calculated, the real-time algorithm searches over time instead of over space by limiting the motion detection to no more than

one pixel per frame. It starts by using a small neighborhood where $n = 1$ to calculate motion between frames t and $t+1$, t and $t+2$, t and $t+3$, up to t and $t+D$, where D is the maximum time delay and defines the lower limit of detectable motion as $\frac{1}{D}$ pixels per frame. Now instead of evaluating pixel motions from 1 to n, the algorithm only needs to evaluate pixel motions from $\frac{1}{D}$ to 1. When $n = 1$, (u, w) would now reside in the $\{-1, 0, 1\} \times \{-1, 0, 1\}$ space, lowering the number of computations of (A.16) to 9 per frame or $9 \times D$ for D frames.

Pooling pixels together allows for a larger detectable motion and limits the number of pixels that must be calculated. For a $p \times p$ block of pixels where $p > 1$, the range of detectable motion is now $\frac{p}{D}$ to p and the number of pixels that must be calculated is reduced by $p \times p$.

APPENDIX B

Software Class Property Descriptions

This appendix lists the software classes and subclasses along with their property names, units if applicable, discrete values if applicable, allocated size if greater than a single scalar value, and the notation used within the software. Properties are listed by class in the order as defined in the software.

B.1 Simulation Class

B.1.1 Options

1. Environment, ($Wall/Clean/Skybridges$), *environment*
2. Feedback Type, (Full-State/State Estimate), $feedbackType$
3. Filter Type, (EKF/RPF), $filterType$
4. Initial Estimated State Distribution Type, (Gaussian/Uniform), $initialDistributionType$
5. Save Sim MC Average Data, (On/Off), *save*
6. Internal Sensor Accuracy Calculator, (On/Off), *sensorAccuracy*
7. Sensor Delay Compensation, (On/Off), *sensorDelayCompensation*
8. UAS Model, (Exact/Uncertain), $uasModelType$

B.1.2 Parameters

1. Time step (seconds), dt
2. Cycle time (seconds), $cycleTime$
3. Initial Position (1×3) (meters), $initialTruePosition$
4. Simulation Length (seconds), $length$
5. Number of Monte Carlo runs (-), $mcRuns$
6. Number of time steps (-), $numSteps$

B.1.3 Results

B.1.3.1 Monte Carlo Mean

1. Mean Altitude with respect to Buildings Value $(numSteps \times 1)$, alt
2. Mean Simulation Cycle Time (seconds), $cycleTime$
3. Mean Number of Measurements (-), nz
4. Mean Street-Level Value $(numSteps \times 1)$ (-), sl
5. Mean Control Input Vector $(4 \times 1 \times numSteps)$ (varies), u
6. Mean True State Vector $(12 \times 1 \times numSteps)$ (varies), x
7. Mean True State Vector Standard Deviation $(12 \times 1 \times numSteps)$ (varies), xSD
8. Mean Estimated Altitude with respect to Buildings Value $(numSteps \times 1)$ (-), $althat$
9. Mean Estimated State Covariance Matrix $(12 \times 12 \times numSteps)$(varies), $covariance$
10. Mean Estimated State Covariance Matrix Standard Deviation $(12 \times 12 \times numSteps)$(varies), $covarianceSD$

11. Mean State Error Vector ($12 \times 1 \times numSteps$) (varies), $error$

12. Mean State Error Vector Standard Deviation($12 \times 1 \times numSteps$)(varies), $errorSD$

13. Mean Estimated Street Level Value ($numSteps \times 1$), $slhat$

14. Mean Estimated State Vector ($12 \times 1 \times numSteps$) (varies), $xhat$

15. Mean Estimated State Vector Standard Deviation ($12 \times 1 \times numSteps$)(varies), $xhatSD$

16. Average Normalized Estimation Error Squared ($1 \times 1 \times numSteps$)(-), $ANEES$

17. Average Normalized Innovations Squared ($1 \times 1 \times numSteps$) (-), $ANIS$

18. Mean State Root Mean Squared Error ($12 \times 1 \times numSteps$) (varies), $RMSE$

19. Normalized Estimation Error Squared Standard Deviation ($1 \times 1 \times numSteps$) (-), $neesSD$

20. Normalized Innovations Squared Standard Deviation ($1 \times 1 \times numSteps$) (-), $neesSD$

B.1.3.2 Autocorrelation

1. Autocorrelation Value ($1 \times min(numofinnovations)$)(-), $autocorrelation$

B.2 Urban Environment Class

B.2.1 City Specs

1. Buildings per Block (-), $bldgsperBlock$

2. Number of Blocks (-), $numberofBlocks$

3. Open Space Block (-), $openSpace$

4. Building Spacing (meters), *bldgS pacing*
5. Street Width (meters), *streetWidth*
6. Building Length Range (1×2) (meters), *bldgLength*
7. Building Width Range (1×2) (meters), *bldgWidth*
8. Building Height Range (1×2) (meters), *bldgHeight*
9. Antenna Height Range (1×2) (meters), *antennaHeight*
10. Antenna Radius Range (1×2) (meters), *antennaRadius*
11. Antenna Minimum Building Height (meters), *antennaMinBldgHeight*
12. Number of Skybridges (-), *skybridgeQuantity*
13. Skybridge Height ($1 \times$ *skybridgeQuantity*), (meters), *skybridgeHeight*
14. Skybridge Connecting Buildings ($1 \times$ *skybridgeQuantity*), (meters), *skybridgeBldgs*
15. Skybridge Base Height ($1 \times$ *skybridgeQuantity*), (meters), *skybridgeBaseHeight*

B.2.2 Map

B.2.2.1 Canyon

1. Southwest Boundary Vector (1×2) (meters), SWB
2. Southeast Boundary Vector (1×2) (meters), SEB
3. Northeast Boundary Vector (1×2) (meters), NEB
4. Northwest Boundary Vector (1×2) (meters), NWB
5. Maximum Building Height within Canyon (meters), *maxHeight*
6. Minimum Building Height within Canyon (meters), *minHeight*

B.2.2.2 Canyon > Building

1. Building Number (-), *number*
2. Southwest Boundary Vector (1×2) (meters), SW
3. Southeast Boundary Vector (1×2) (meters), SE
4. Northeast Boundary Vector (1×2) (meters), NE
5. Northwest Boundary Vector (1×2) (meters), NW
6. Building Height (meters), *height*

B.2.2.3 Canyon > Obstacles > Antenna

1. Antenna Coordinates (1×2) (meters), *position*
2. Antenna Base Height (1×1) (meters), *baseHeight*
3. Antenna Top Height (1×1) (meters), *topHeight*
4. Antenna Radius (1×1) (meters), *radius*

B.2.2.4 Canyon > Obstacles > Skybridge

1. Southwest Corner Vector (1×2) (meters), SW
2. Southeast Corner Vector (1×2) (meters), SE
3. Northeast Corner Vector (1×2) (meters), NE
4. Northwest corner Vector (1×2) (meters), NW
5. Top Height (1×1) (meters), *topheight*
6. Base Height (1×1) (meters), *baseHeight*

B.2.3 Weather

1. Wind Vector (3×1) (meters/second), *wind*
2. Temperature (1×1) (degrees Kelvin), *temperature*
3. Visibility (1×1) (meters),*visibility*

B.2.4 Relative Location

1. Altitude with respect to Buildings $(mcRuns \times numSteps)$, (1: Above All Buildings/2: Above Some Buildings/3: Below all Buildings), *alt*
2. Street-Level Position $(mcRuns \times numSteps)$, (1: Urban Canyon, 2: Intersection, 3: Adjacent Open Space), *sl*
3. Current Canyon Number $(mcRuns \times numSteps)$, *canyonNumber*

B.3 UAS Dynamics

B.3.1 Model

1. Process Noise Matrix, (12×12) (varies), Q
2. Process Noise Matrix Scaling Factor (-), Qsf
3. Control Input Vector Dimension (-), *controlDim*
4. State Vector Dimension (-), *stateDim*
5. Pitch Rate Limits (1×2) (radians/second), *qLim*
6. Control Surface Deflection Limits (1×2) (-), *deltaLim*
7. Engine Revolutions per Minute Limits (1×2) (revolutions/second), *nLim*

8. Engine Thrust Limits (1×2) (Newtons), *thrustLim*

B.3.1.1 Parameters

1. Acceleration due to Gravity (meters per sec squared), g
2. Aircraft Mass (kg), m
3. *Aircraft Physical Inertia Matrix Entries (kilogram-meters^2), $\{Ixx, Iyy, Izz, Ixz\}$
4. Aircraft Wing Area (meters^2), S
5. Aircraft Wing Span (meters), b
6. Aircraft Mean Aerodynamic Chord (meters), c_bar
7. Propeller Diameter (meters), D
8. Engine Time Constant, (seconds) tau_n
9. Zero Angle of Attack Lift Coefficient (-), C_Z1
10. Angle of Attack Lift Coefficient (-), C_Z_alpha
11. Zero Angle of Attack/Angle of Sideslip Drag Coefficient (-), C_X1
12. Angle of Attack Drag Coefficient (-), C_X_alpha
13. Angle of Attack Squared Drag Coefficient (-), C_X_alpha2
14. Angle of Sideslip Squared Drag Coefficient (-), C_X_beta2
15. Angle of Sideslip Side Force Coefficient (-), C_Y1
16. Aileron Deflection Roll Coefficient (-), $C\_L\_delta\_a$
17. Angle of Sideslip Roll Coefficient (-), C_L_beta

18. Dimensionless Roll Rate Roll Coefficient (-), $C\_L\_p\_tilde$
19. Dimensionless Yaw Rate Roll Coefficient (-), $C\_L\_r\_tilde$
20. Zero Elevator Deflection/Angle of Attack/Dimensionless Pitch Rate Pitch Coefficient (-), C_M1
21. Elevator Deflection Pitch Coefficient (-), $C\_M\_delta\_e$
22. Angle of Attack Pitch Coefficient (-), C_M_alpha
23. Dimensionless Pitch Rate Pitch Coefficient (-), $C\_M\_q\_tilde$
24. Rudder Deflection Yaw Coefficient (-), $C\_N\_delta\_r$
25. Angle of Sideslip Yaw Coefficient (-), C_N_beta
26. Dimensionless Yaw Rate Yaw Coefficient (-), $C\_N\_r\_tilde$
27. Zero Advance Ratio Thrust Coefficient (-), C_F_T1
28. Advance Ratio Thrust Coefficient, (-) C_F_T2
29. Advance Ratio Squared Thrust Coefficient (-), C_F_T3

B.3.2 State Propagation

1. True State Vector ($11 \times numSteps \times mcRuns$) (varies), x
2. Control Input Vector ($4 \times numSteps \times mcRuns$) (varies), u

B.3.2.1 Guidance

1. Obstacle Avoidance Waypoint ($3 \times skybridgeQuantity$) (meters), $avoidWaypoint$
2. Obstacle Vertical Clearance (meters), $verticalClearance$
3. Commanded True State Vector ($11 \times numSteps \times mcRuns$) (varies), $xCmd$

B.3.3 Trim

B.3.3.1 Parameter

1. Flight Path Angle (radians), *gamma*
2. Altitude (meters), h
3. Heading (radians), *heading*
4. Airspeed (meters/second), Vt
5. Trim Conditions (11×1) (varies), dx

B.3.3.2 True States

1. True Trim State Vector (11×1) (varies), x
2. Non Linear Least Squared Residual Norm (-), $resnorm$

B.4 Sensor

B.4.1 Availability

1. Available Sensors, $availSensors$
2. Delayed Sensors, $delayedSensors$

B.4.2 Measurement

1. Actual Measurement Availability Timestep $(1 \times length/sampling\ period)$ (-), *avail*
2. Sensor Innovations $(nz \times length/sampling\ period)$ (-), *innovations*

B.4.3 Performance

1. Airspeed Error Covariance (1×3) (meters/second), $airspeedErrorCovariance$
2. Measurement Availability Index $(1 \times length/sampling\ period)$ (-), $availIndex$
3. Sensor Environment based Availability (3×3) (-), $environmentAvailability$
4. Measurement Noise Covariance (varies based on number of measurable states) (varies), $errorCovariance$
5. Time First Measurement Taken (seconds), $firstMeas$
6. Measurement Latency (seconds), $latency$
7. Sensor Orientation, (0/1/2) (-), $orientatation$
8. Number of Measured States, (-), $numStates$
9. Measurement Period (seconds), $period$
10. Measurement Sensitivity Matrix $(numStates \times stateDim)$ (varies), $sensitivity$
11. Measurement Taken Time Index $(numStates \times length/period)$ (-), $stateIndex$
12. Measured States $(1 \times numStates)$ (-), $states$
13. Sensor Status, (On/Off) (-), $status$

B.5 Estimator

B.5.1 Parameter

1. Filter Process Noise Covariance Matrix Tuning Parameter (-), $Qtune$
2. Particle Filter Optimal Jitter Bandwidth (12×1) (-), $hOpt$

3. Initial Estimate Covariance (1×12) (varies), *initialCovariance*

4. Particle Filter Number of Particles (-), *numberofParticles*

5. Particle Filter Resampling Threshold Number of Particles (-), *resampThreshold*

B.5.2 State Estimate

1. Normalized Estimation Error Squared ($1 \times numSteps \times mcRuns$) (-), *NEES*

2. Estimated State Covariance Matrix ($12 \times 12 \times numSteps \times mcRuns$) (varies), *covariance*

3. Estimated State Error Vector ($12 \times numSteps \times mcRuns$) (varies), *error*

4. Estimated State Vector ($12 \times numSteps \times mcRuns$) (varies), *xhat*

B.5.2.1 EKF

1. Normalized Innovations Squared ($mcRuns \times numSteps$) (-), *NIS*

2. Number of Measurements ($mcRuns \times numSteps$) (-), *nz*

B.5.2.2 Particle Filter

1. Particles ($12 \times numberofParticles \times numSteps \times mcRuns$), (varies), *particles*

2. Particle Weights ($12 \times numberofParticles \times numSteps \times mcRuns$) (-), *weights*

APPENDIX C

UAS Model Parameters

The UAS model presented in this research is for a fixed-wing 28 kilogram mass, 3.1 meter wingspan gas-powered radio-controlled aircraft described in Ducard [42]. All aerodynamic coefficients and the physical inertia matrix entries marked with an asterisk have uncertainty ranging between 10 and 30 percent as shown.

Table C.1: UAS model physical inertia matrix I^b in kilogram-meter^2 with 5% uncertainty.

$I_{xx} = 2.56$*	0	$I_{xz} = 0.5$*
0	$I_{yy} = 10.9$*	0
$I_{zx} = 0.5$*	0	$I_{zz} = 11.3$*

Table C.2: UAS model aerodynamic coefficients with varying levels of uncertainty.

Thrust		Roll	
CFT_1	8.42×10^{-2}	$*CL_{\delta a}$ (10%)	6.79×10^{-2}
CFT_2	-1.36×10^{-1}	$*CL_{\beta}$ (30%)	-1.30×10^{-2}
CFT_3	-9.28×10^{-1}	$CL_{\tilde{p}}$	-1.92×10^{-1}
Lift		$CL_{\tilde{r}}$	-3.61×10^{-2}
CZ_1	1.29×10^{-2}	Pitch	
CZ_{α}	-3.25	$*CM_1$ (10%)	2.08×10^{-2}
Drag		$*CM_{\delta e}$ (20%)	5.45×10^{-1}
CX_1	-2.12×10^{-2}	$*CM_{\alpha}$ (20%)	-9.03×10^{-2}
CX_{α}	-2.66×10^{-2}	$*CM_{\tilde{q}}$ (20%)	-9.83
$CX_{\alpha 2}$	-1.55	Yaw	
$CX_{\beta 2}$	-4.01×10^{-1}	$*CN_{\delta r}$ (10%)	5.34×10^{-2}
Side Force		$*CN_{\beta}$ (10%)	8.67×10^{-2}
CY_1	-3.79×10^{-1}	$*CN_{\tilde{r}}$ (10%)	-2.14×10^{-1}

Table C.3: UAS performance limits.

Parameter	Min	Max
Normalized Aileron Deflection, (-)	-1	1
Normalized Elevator Deflection, (-)	-1	1
Normalized Rudder Deflection, (-)	-1	1
Engine RPM @ 30 meters/second	51	100
Angle of Attack, degrees	-10	10
Pitch Rate, degrees/second	-50	50
Roll Rate, degrees/second	-45	45
Yaw Rate, degrees/second	-20	20

APPENDIX D

LQR Controller Gain Matrix for Steady Flight

This appendix contains the LQR gain matrices for the steady, level flight and wings-level descending flight trim conditions where $\bar{\rho} = 10$ for both matrices.

Table D.1: UAS LQR controller gains for steady level flight: V_T = 30 meters/second, h = 50 meters, γ = 0 degrees.

	h	V_t	α	θ	q	β	x_E	ϕ	ψ	p	r
δ_a	−0.0003	0	0.0007	−0.0052	0	5.9738	0.2191	0.5757	6.6334	0.2436	0.1324
δ_e	0.3117	−0.0360	−2.1909	5.1431	0.4067	0.0025	0.0003	−0.0016	0.0056	0	0
δ_r	−0.0001	0.0001	−0.0005	−0.0030	0	5.6482	0.2280	0.1291	6.2553	0.0271	0.2713
F_T	13.7652	79.1967	−106.4399	54.3512	−2.4391	−0.7129	−0.252	−0.0350	−0.5947	−0.0016	0.0067

Table D.2: UAS LQR controller gains for wings-level descending flight: V_T = 30 meters/second, h = 50 meters, γ = 0 degrees.

	h	V_t	α	θ	q	β	x_E	ϕ	ψ	p	r
δ_a	0	0	0	−0.0001	0	5.8037	0.1928	0.5802	6.5383	0.2429	0.1003
δ_e	0.3140	−0.0455	−2.2039	5.2469	0.3790	0	0	0	0.0001	0	0
δ_r	0	0	0	−0.0001	0	6.7218	0.2507	0.1612	7.4458	0.0216	0.2750
F_T	9.7590	79.7955	−86.9785	25.8216	−2.6102	−0.0124	−0.0004	−0.0005	−0.0105	0	0.0001

APPENDIX E

Sensor Measurement Noise Covariance Data

This appendix includes published sensor noise standard deviation data for several possible UAS sensors. Each of the sensor types are listed separately with the noise standard deviation values for each filtered sensor state. If no sensor measurement bias value is listed, it is assumed to be zero.

E.1 Inertial Measurement Unit & Inertial Navigation System/Attitude and Heading Reference System

Table E.1: IMU angular velocity measurement noise standard deviation data.

Data Type and Published Source	$\sigma_p = \sigma_q = \sigma_r$ (radians/second)
Simulation - Bryson & Sukkarieh [123]	0.002
HIL Simulation - Jung & Tsiotras [46]	0.002
Hardware - Bry *et al.* [124]	0.003
Simulation - Beard [47]	0.005
Simulation - Langelaan *et al.* [58]	0.01
Simulation - Ducard [42]	0.087
Simulation - Ramprasadh & Ayra [125]	0.1047

Table E.2: AHRS/INS Euler angle measurement noise bias and standard deviation data.

Data Type and Published Source	$\mu_\phi/\mu_\theta/\mu_\psi$ (radians)	$\sigma_\phi/\sigma_\theta/\sigma_\psi$ (radians)
Free Moving Platform Test - Perry *et al.* [126]	0/0/0	0.003/0.003/0.006
Langelaan *et al.* [58]	0/0/0	0.017/0.017/0.017
Hardware Spec - Microstrain Datasheet [60]	0/0/0	0.035/0.035/0.035
AHRS Simulation - Kingston & Beard [51]	0/0/0	0.052/0.052/0.052
GPS/INS Simulation - Lopes *et al.* [45]	$0/0/-0.016$	0.031/0.006/0.032
GPS/INS Flight Test - Lopes *et al.* [45]	$-0.002/0.01/-0.101$	0.036/0.039/0.144
Flight Test - Euston *et al.* [54]	$0/-0.023/-$	0.047/0.029/–

E.2 Air Data System

Table E.3: ADS indicated airspeed and aerodynamic angle measurement noise standard deviation data.

Data Type and Published Source	$\sigma_{V_T}/\sigma_\alpha/\sigma_\beta$ (meters/second radians radians)
Simulation - Beard [47]	$0.4/-/-$
Simulation - Langelaan *et al.* [127]	0.2/0.017/0.017
Simulation - Ramprasadh & Ayra [125]	0.808/0.0314/0.0164
Simulation - Ducard [42]	1.0/0.035/0.035
Free Moving Platform Test - Perry *et al.* [126]	$-/0.014/0.007$

Table E.4: ADS altitude measurement noise standard deviation data.

Data Type and Source	σ_h (meters)
Flight Test - Quigley *et al.* [63]	1.5
Simulation - Beard [47]	0.4

E.3 GPS

Table E.5: GPS lateral and vertical position measurement noise bias and standard deviation data.

Data Type and Source	$\sigma_{x_N} = \sigma_{x_E}$ (meters)	μ_h/σ_h (meters)
Old European Adjacent Open Space - Modsching *et al.* [20]	2.06	–
Old European Urban Core - Modsching *et al.* [20]	11.96	–
Simulation - Jung & Tsiotras [46], Langelaan et al. [58]	3	0/3
Nominal GPS Error Model - Beard [47]	3.67	0/7.2
Modern Urban Core (Best case) - MacGougan *et al.* [17]	7.65	−7.1/9.5

Table E.6: GPS inertial speed measurement noise standard deviation data.

Data Type and Source	$\sigma_{\dot{x}_N} = \sigma_{\dot{x}_E} = \sigma_{\dot{h}}$ (meters/second)
Simulation - Jung & Tsiotras [46], Langelaan *et al.* [58]	0.1

E.4 LTE

Table E.7: LTE network lateral position - OTDOA method measurement noise standard deviation data.

Number of eNBs [53]	$\sigma_{x_N} = \sigma_{x_E}$ (meters)
7	7.51
10	5.02
15	3.74
20	3.23
Simulation - 30 Neuland *et al.* [94]	25

E.5 Computer Vision

Table E.8: Computer vision position measurement noise bias and standard deviation data.

Data Source and Type	$\mu_{x_N}/\mu_{x_E}/\mu_h$ (meters)	$\sigma_{x_N}/\sigma_{x_E}/\sigma_h$ (meters)
Flight Test - Wang *et al*. [50]	6.807/7.055/7.718	2.774/3.385/3.635

Table E.9: Computer vision airspeed measurement noise bias and standard deviation data.

Data Source and Type	$\mu_{\dot{x}_N}/\mu_{\dot{x}_E}/\mu_{\dot{h}}$ (meters/second)	$\sigma_{\dot{x}_N}/\sigma_{\dot{x}_E}/\sigma_{\dot{h}}$ (meters/second)
Flight Test - Wang *et al*. [50]	0.590/0.467/0.373	0.698/0.626/0.257

Table E.10: Computer vision Euler angle measurement noise bias and standard deviation data,

Data Type and Source	$\mu_\phi/\mu_\theta/\mu_\psi$ (radians)	$\sigma_\phi/\sigma_\theta/\sigma_\psi$ (radians)
Flight Test - Hwangbo & Kanade [65]	$0.0106/-0.0086/-$	$0.0159/0.0196/-$
Flight Test - Wang *et al*. [50]	0.0116/0.0121/0.0141	0.0140/0.0132/0.129

APPENDIX F

State Augmentation Applied to GPS and LTE

This appendix shows how to account for a delayed GPS measurement, first taken $t = 0.9$ seconds, and available to the filter $t = 1$ second with a sampling frequency of 1 Hz and an LTE measurement first taken $t = 1$ second and first available to the filter at $t = 5$ seconds with a sampling frequency of 0.25 Hz. In this Appendix, the term 'available' is used in the context of a taken measurement sent to the filter after a delay rather than the UAS being in an (SL/ALT) location where the sensor is able to take a measurement. IMU and ADS are also considered as non-delayed sensors as shown in Table F.1.

Table F.1: Sensor simulation parameters.

	Sampling Period	Sampling Delay	Measured States
GPS	1 second	0.1 second	x_N, x_E, h, V_T
IMU	0.01 seconds	0 seconds	ϕ, θ, ψ, p, q, r
ADS	0.02 seconds	0 seconds	h, V_T, α, β
LTE	4 seconds	4 seconds	x_N, x_E

F.1 First GPS Measurement Taken

Switching to time step notation (for $dt = 0.01$ seconds) to eliminate decimal subscripts, the first GPS measurement is taken at time step $k = 91$. It will become available to the filter at $k = 101$. To account for this delay, the estimated state vector is augmented as shown in (F.1) where $\hat{\mathbf{x}}_{GPS_{91|91}}$ is a copy of the estimated state vector that will be used at $k = 101$

to calculate the delayed GPS measurement prediction. For the estimated state vectors the subscript $k1|k2$ is defined as the time step in which the estimate is added to the vector $k1$ and the time step upon which the estimate is conditioned upon $k2$. For $\hat{\mathbf{x}}_{GPS_{91|91}}$, subscript $k1 = 91$ does not change, but subscript $k2 = 91$ changes at each time step due to small corrections in this vector due to interim measurements. The augmented estimated state vector is shown below. Note that $\hat{\mathbf{x}}_{91|91}$ is the evolving estimated state vector generated by the filter.

$$\hat{\mathbf{x}}_{aug_{91|91}} = \begin{bmatrix} \hat{\mathbf{x}}_{91|91} \\ \\ \hat{\mathbf{x}}_{GPS_{91|91}} \end{bmatrix} \tag{F.1}$$

Anytime the estimated state vector is augmented or marginalized, the true state vector is also expanded or contracted as shown in (F.2) since these values are used in the linear measurement model: $\mathbf{z}_{aug_k} = H_{aug_k} * \mathbf{x}_{aug_k} + \mathbf{v}_k$ where $\mathbf{z}_{aug_k}$ is the augmented measurement vector, H_{aug_k} is the augmented measurement sensitivity matrix, and $\mathbf{x}_{aug_k}$ is the augmented true state vector.

$$\mathbf{x}_{aug_{91|91}} = \begin{bmatrix} \mathbf{x}_{91|91} \\ \\ \mathbf{x}_{GPS_{91|91}} \end{bmatrix} \tag{F.2}$$

When the estimated state vector is augmented, the estimated state vector covariance matrix must also be augmented as shown below in (F.3) where $P_{91|91}$ is the diagonal entry and $P_{91,91|91}$ is the cross-covariance with the first subscript representing the the current time step and the remaining two subscripts are defined in the same manner as for the estimated state vector.

$$P_{aug_{91|91}} = \begin{bmatrix} P_{91|91} & P_{91,91|91} \\ \\ P_{91,91|91} & P_{91|91} \end{bmatrix} \tag{F.3}$$

At $k = 92$, the state transition matrix is augmented as shown below in (F.4) to allow for propagation of the evolving estimated state vector covariance while holding the augmented estimated state vector covariance at its current value by adding a 12×12 identity matrix to

the lower right hand sub-block this matrix.

$$A_{aug_{92}} = \begin{bmatrix} A_{92} & 0_{12\times12} \\ 0_{12\times12} & I_{12\times12} \end{bmatrix} \tag{F.4}$$

The process noise covariance matrix is also augmented as shown below in (F.5) to account for the dimension of the estimated state vector.

$$Q_{aug_{92}} = \begin{bmatrix} Q_{92} & 0_{12\times12} \\ 0_{12\times12} & 0_{12\times12} \end{bmatrix} \tag{F.5}$$

Note that the augmented zero and identity matrices are square matrices with size equal to the number of states in the system.

For the correction step, the H matrix is augmented as shown in (F.6) to add 12 zeros to the end of each row. Those zeros to the right of the vertical line are necessary to account for the length of the augmented estimated state vector and n_z is the number of non-delayed measurements available to the filter at the time step from the IMU and ADS.

$$H_{aug_{92}} = \left[H \quad | \quad 0_{n_z\times12} \right] \tag{F.6}$$

From $k = 92$ through $k = 100$, the above augmented estimated state vector (F.1) and augmented covariance matrix (F.3) are propagated and then corrected using available non-delayed measurements at each time step.

F.2 First GPS Measurement becomes Available to Filter

At $k = 101$, the first GPS measurement becomes available to the filter changing the H matrix as shown below in (F.7) where $n_{z_{GPS}}$ is the number of delayed GPS measurements available

to the filter at $k = 101$.

$$H_{aug_{101}} = \begin{bmatrix} H_{101} & | & 0_{n_z \times 12} \\ 0_{n_{z_{GPS}} \times 12} & | & H_{GPS} \end{bmatrix} \tag{F.7}$$

After the correction step is complete the estimated state vector and covariance matrix are adjusted to marginalize the augmented states and covariances out of the system, shown in (F.8) and (F.9), respectively, since the estimated state vector at $k = 91$ is no longer needed.

$$\hat{\mathbf{x}}_{aug_{101|101}} = \begin{bmatrix} \hat{\mathbf{x}}_{101|101} \end{bmatrix} \tag{F.8}$$

$$P_{aug_{101|101}} = \begin{bmatrix} P_{101|101} \end{bmatrix} \tag{F.9}$$

F.3 First LTE Measurement is Taken

However, the first LTE measurement is taken at $k = 101$ and will become available to the filter at $k = 501$. The estimated state vector and covariance matrix from (F.8) and (F.9) are augmented as shown below in (F.10) and (F.11), respectively where $\hat{\mathbf{x}}_{LTE_{101|101}}$ represents the non-evolving estimated state vector that will be used in the measurement prediction at k=501.

$$\hat{\mathbf{x}}_{aug_{101|101}} = \begin{bmatrix} \hat{\mathbf{x}}_{101|101} \\ \hat{\mathbf{x}}_{LTE_{101|101}} \end{bmatrix} \tag{F.10}$$

$$P_{aug_{101|101}} = \begin{bmatrix} P_{101|101} & P_{101,101|101} \\ P_{101,101|101} & P_{101|101} \end{bmatrix} \tag{F.11}$$

At $k = 102$, the state transition matrix, process noise covariance matrix, and H matrix are augmented again as shown below in (F.12)-(F.14).

$$A_{aug_{102}} = \begin{bmatrix} A_{102} & 0_{12\times12} \\ 0_{12\times12} & I_{12\times12} \end{bmatrix} \tag{F.12}$$

$$Q_{aug_{102}} = \begin{bmatrix} Q_{102} & 0_{12\times12} \\ 0_{12\times12} & 0_{12\times12} \end{bmatrix} \tag{F.13}$$

$$H_{aug_{102}} = \begin{bmatrix} H_{102} & | & 0_{n_z\times12} \end{bmatrix} \tag{F.14}$$

From k = 102 through k = 190, the above augmented estimated state vector (F.10) and augmented covariance matrix (F.11) are propagated and then corrected using available non-delayed measurements at each time step.

F.4 Second GPS Measurement is Taken

At k = 191, the second GPS measurement is taken, which results in a longer augmented estimated state vector and larger augmented error covariance matrix as shown in (F.15) and (F.16), respectively.

$$\hat{\mathbf{x}}_{aug_{191|191}} = \begin{bmatrix} \hat{\mathbf{x}}_{191|191} \\ \hat{\mathbf{x}}_{LTE_{101|191}} \\ \hat{\mathbf{x}}_{GPS_{191|191}} \end{bmatrix} \tag{F.15}$$

$$P_{aug_{191|191}} = \begin{bmatrix} P_{191|191} & P_{191,101|191} & P_{191,191|191} \\ P_{191,101|191} & P_{101|191} & P_{191,101|191} \\ P_{191,191|191} & P_{191,101|191} & P_{191|191} \end{bmatrix} \tag{F.16}$$

For the estimated state vector and covariance matrix entries related to the augmented LTE states from k = 101, the second subscript is now 191 since the quantities are conditioned on the measurements received up through this time step.

At k = 192 the state transition matrix, process noise covariance matrix, and H matrix are augmented again as shown below in (F.17)-(F.19), now to account for delayed LTE and

delayed GPS measurements.

$$A_{aug_{192}} = \begin{bmatrix} A_{192} & 0_{12\times12} & 0_{12\times12} \\ 0_{12\times12} & I_{12\times12} & 0_{12\times12} \\ 0_{12\times12} & 0_{12\times12} & I_{12\times12} \end{bmatrix} \tag{F.17}$$

$$Q_{aug_{192}} = \begin{bmatrix} Q_{192} & 0_{12\times12} & 0_{12\times12} \\ 0_{12\times12} & 0_{12\times12} & 0_{12\times12} \\ 0_{12\times12} & 0_{12\times12} & 0_{12\times12} \end{bmatrix} \tag{F.18}$$

$$H_{aug_{192}} = \begin{bmatrix} H_{192} & | & 0_{n_z\times12} & 0_{n_z\times12} \end{bmatrix} \tag{F.19}$$

The process continues as the estimated state vector and error covariance matrix are augmented to account for GPS measurements taken at k = 291, 391, and 491 and marginalized at k = 301 and 401 as GPS measurements become available and are used in the innovation vector. Throughout this time, the non-evolving LTE portion of the augmented estimated state vector and error covariance matrix remain.

F.5 First LTE Measurement becomes Available to Filter

The first LTE measurement (and fifth GPS measurement) become available to the filter at k = 501. After the prediction step at this time step, the augmented estimated state vector and error covariance matrix become (F.20) and (F.21) as shown below.

$$\hat{\mathbf{x}}_{aug_{501|500}} = \begin{bmatrix} \hat{\mathbf{x}}_{501|500} \\ \hat{\mathbf{x}}_{LTE_{101|500}} \\ \hat{\mathbf{x}}_{GPS_{491|500}} \end{bmatrix} \tag{F.20}$$

$$P_{aug_{501|500}} = \begin{bmatrix} P_{501|500} & P_{501,101|500} & P_{501,491|500} \\ P_{501,101|500} & P_{101|500} & P_{501,101|500} \\ P_{501,491|500} & P_{501,101|500} & P_{491|500} \end{bmatrix} \tag{F.21}$$

To generate the measurements and measurement predictions at $k = 501$, the H matrix is augmented as shown in (F.22).

$$H_{aug_{501}} = \begin{bmatrix} H_{501} & | & 0_{n_z \times 12} & 0_{n_z \times 12} \\ 0_{n_{z_{LTE}} \times 12} & | & H_{LTE} & 0_{n_{z_{LTE}} \times 12} \\ 0_{n_{z_{GPS}} \times 12} & | & 0_{n_{z_{GPS}} \times 12} & H_{GPS} \end{bmatrix} \tag{F.22}$$

After the correction step the estimated state vector and error covariance matrix are adjusted to marginalize out the augmented states and covariances since the estimated state vector at $k = 101$ (LTE) and the estimated state vector at $k = 491$ (GPS) are no longer needed as shown below in (F.23) and (F.24), respectively. The state augmentation and marginalization process is repeated for GPS and LTE measurements as they are taken and become available to the filter throughout the simulation.

$$\hat{\mathbf{x}}_{aug_{501|501}} = \begin{bmatrix} \hat{\mathbf{x}}_{501|501} \end{bmatrix} \tag{F.23}$$

$$P_{aug_{501|501}} = \begin{bmatrix} P_{501|501} \end{bmatrix} \tag{F.24}$$

APPENDIX G

Guidance System Design in Urban Environments

When in the urban environment, a situationally-aware guidance system is necessary for safe and efficient flight. This problem has been examined by many, including [128], [37], [129]. This appendix proposes a simple urban canyon guidance system with the sole objective of generating commanded state values that allow the UAS to safely avoid obstacles by climbing above them or descending below them.

The default guidance mode in the simulation is to maintain the trim state set by the mission trajectory. The UAS switches to obstacle avoidance mode when it determines there is an obstacle in its flight path based on a priori knowledge of the urban environment. An obstacle is considered to be in the flight path if the UAS is on a trajectory to intersect the vertical plane extending from a user-defined obstacle avoidance waypoint above the obstacle to a user-defined waypoint below the UAS as shown in Figure G.1.

G.1 Maintaining the Trim State

To maintain the trim state, the UAS sends the trim state vector to the controller at each time step, as shown in (G.1), assuming the direction of travel is x_N. These trim values are then converted into control inputs using the static gains based on the current trim state of the UAS.

$$\mathbf{x}_{cmd} = \mathbf{x}_{trim} = [h_{trim}\, V_{T_{trim}}\, \alpha_{trim}\, \theta_{trim}\, q_{trim}\, x_{E_{trim}}\, \beta_{trim}\, \phi_{trim}\, \psi_{trim}\, p_{trim}\, r_{trim}]^T \tag{G.1}$$

G.2 Avoiding Obstacles

Once an obstacle is detected along the flight path using the known obstacle map, the UAS commands an avoidance maneuver toward an avoidance waypoint to clear of the obstacle. Upon clearing the obstacle, the UAS returns to its mission trajectory.

During the initialization of the simulation, obstacle avoidance waypoints are calculated for all obstacles in the urban environment. Both avoidance waypoints for each obstacle, (x_{avoid_N}, x_{avoid_E}, x_{avoid_h}), are calculated using (G.2) with a user-defined clearance above and below the obstacle. The North and East coordinates are equal for both waypoints, with the East coordinate assumed to be zero to ensure all obstacle avoidance climbing/descending is done as far away from buildings as possible.

$$\begin{gathered} x_{avoid_N} = \frac{obstacle_{NE} - obstacle_{SW}}{2} \\ x_{avoid_E} = 0 \\ x_{avoid\,above_h} = obstacle_{top\,height} + obstacle\ clearance \\ x_{avoid\,below_h} = obstacle_{base\,height} - obstacle\ clearance \end{gathered} \tag{G.2}$$

The UAS calculates a collision avoidance vector at each time step to determine if it is necessary to initiate an obstacle avoidance guidance command. This vector, $\mathbf{d}_{obs}$, contains the three elements listed in (G.3):

$$\mathbf{d}_{obs} = \begin{bmatrix} d_{long} \\ d_{vert} \\ \gamma_{avoid} \end{bmatrix} \tag{G.3}$$

These elements and all other obstacle avoidance geometry is shown in Figure G.1.

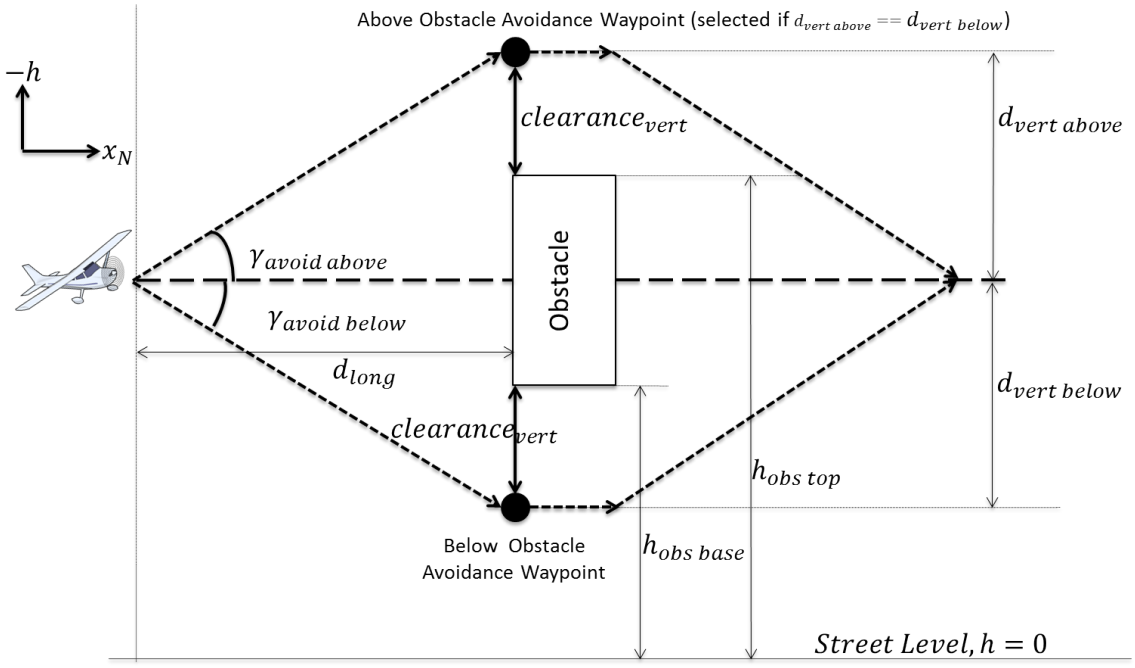


Figure G.1: UAS Obstacle avoidance in urban environment geometry.

The first element is the longitudinal distance from the UAS to the obstacle d_{long}. It is calculated using (G.4) to select the smaller distance where x_{obs} is the longitudinal position of the face of the obstacle closest to the UAS.

$$d_{long} = |\hat{x}_N - x_{obs}| \tag{G.4}$$

The second element is the vertical distance between the UAS and the closer avoidance waypoint, d_{vert}. The UAS selects from the two avoidance waypoints by calculating its vertical distance to each waypoint and then selecting the waypoint with the smaller distance,

as shown in (G.5). In the unlikely event that the UAS is exactly halfway between the two avoidance waypoints, it will default to the waypoint above the obstacle to keep it further from street-level

$$d_{vert} = min\{|\hat{x}_h - x_{avoid\,above_h}|, |x_{avoid\,below_h} - \hat{x}_h|\} \tag{G.5}$$

The third element is the necessary flight path angle to intersect the selected avoidance waypoint, γ_{avoid}. It is calculated using (G.6) with the selected avoidance waypoint. If the above obstacle avoidance waypoint is selected,$\gamma_{avoid\,above}$ is used and the below obstacle avoidance waypoint $\gamma_{avoid\,below}$ is used.

$$\begin{gathered}\gamma_{avoid\,above} = \tan^{-1}\left(\frac{x_{avoid\,above_h} - \hat{x}_h}{d_{long}}\right) \\ \gamma_{avoid\,below} = \tan^{-1}\left(\frac{\hat{x}_h - x_{avoid\,below}}{d_{long}}\right)\end{gathered} \tag{G.6}$$

The UAS initiates the avoidance maneuver once it is within a user defined distance of the obstacle on a collision trajectory. It completes this action by sending the selected obstacle avoidance waypoint altitude component to the controller, causing it to deviate from trim. Once the UAS has safely cleared the obstacle, the mission trajectory is continued as the guidance system resumes sending position and altitude trim commands to the controller. The user-defined avoidance waypoint vertical clearance and stand-off distance must balance mission needs with safety considerations and must keep the UAS in its flight envelope and sufficiently close to the reference trim state vector for the linearized controller to be effective.

BIBLIOGRAPHY

[1] "H.R. 658–112th Congress: FAA Modernization and Reform Act of 2012," February 2012.

[2] van Blyenburgh, P., "Unmanned Aircraft Systems: The Current Situation," *UAS ATM Integration Workshop*, UVS International, Paris, France, 2008.

[3] La Franchi, P., "Law enforcement UAV requirements under study by Thales UK," *Flight International*, July 2007.

[4] Dixon, P., "Sharing the skies," *Helicopters Magazine*, October-November-December 2011.

[5] "Canadian police push limits of civilian UAVs laws," *Homeland Security News Wire*, February 2011.

[6] Clarridge, C., "Police department demonstrates new drone, to help allay concerns," *The Seattle Times*, April 2012.

[7] "FAA gives Arlington, Texas police permission to use UAVs," *Homeland Security News Wire*, March 2013.

[8] Ma, C., Jee, G., MacGougan, G., Lachapelle, G., Bloebaum, S., Cox, G., Garin, L., and Shewfelt, J., "GPS signal degradation modeling," *Proceedings of the 14th International Technical Meeting of the Satellite Division*, ION, Salt Lake City, UT, September 2001, pp. 882–893.

[9] Prost, J., Godefroy, B., and Terrenoir, S., "City walking: improving GPS accuracy for urban pedestrians," *GPS World*, Vol. 19, No. 8, August 2008, pp. 32–37.

[10] Lu, M., Chen, W., and Chan, W., "Discussion of "building project model support for automated labor monitoring" by R. Sacks, R. Navon, and E. Goldschmidt," *Computing in Civil Engineering*, Vol. 18, No. 4, 2004, pp. 381–383.

[11] WiNG, "Wan Chai Overview 2008.jpg," Website, 2008.

[12] Grabowski, J., "Personal privacy jammers: locating Jersey PPDs jamming GBAS safety-of-life signals," *GPS World*, Vol. 23, No. 4, April 2012, pp. 28.

[13] Hanlon, M., "DARPA's surface navigation concept - without GPS," *Gizmag*, April 2007.

[14] Haigh, N., "BAE Systems locates opportunity to replace GPS," June 2012.

[15] "Wireless E911 location accuracy requirements," *PS Docket 07-114. Second Report and Order*, Federal Communications Commission, 2010.

[16] Wang, S., Min, J., and Yi, B., "Location based services for mobiles: technologies and standards," *Proceedings of the International Conference on Communication*, IEEE, Beijing, China, 2008, pp. 35–38.

[17] MacGougan, G., Lachapelle, G., Klukas, R., Siu, K., et al., "Degraded GPS signal measurements with a stand-alone high sensitivity receiver," *Proceedings of the National Technical Meeting*, Vol. 28, ION, San Diego, CA, 2002, p. 30.

[18] Mezentsev, O., Lu, Y., Lachapelle, G., and Klukas, R., "Vehicular navigation in urban canyons using a high sensitivity GPS receiver augmented with a low cost rate gyro," *Proceedings of the GPS Conference*, Vol. 15, ION, Portland, OR, 2002, pp. 263–369.

[19] Hide, C., Moore, T., Hill, C., and Park, D., "Low cost, high accuracy positioning in urban environments," *Journal of Navigation*, Vol. 59, No. 03, 2006, pp. 365–379.

[20] Modsching, M., Kramer, R., and ten Hagen, K., "Field trial on GPS Accuracy in a medium size city: The influence of built-up," *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication*, Hannover, Germany, 2006, pp. 209–218.

[21] Davidson, P., Hautamäki, J., Collin, J., and Takala, J., "Improved vehicle positioning in urban environment through integration of GPS and low-cost inertial sensors," *Proceedings of the the European Navigation Conference*, EUGIN, Naples, Italy, 2009.

[22] Mar, J. and Leu, J., "Simulations of the positioning accuracy of integrated vehicular navigation systems," *IEE Proceedings-Radar, Sonar and Navigation*, Vol. 143, No. 2, 1996, pp. 121–128.

[23] Vicek, C.and McLain, P. and Murphy, M., "GPS/dead reckoning for vehicle tracking in the 'urban canyon' environment," *Proceedings of the Vehicle Navigation and Information Systems Conference*, IEEE/IEE, Ottawa, Ontario, 1993, p. 461.

[24] Georgy, J., Iqbal, U., and Noureldin, A., "Quantitative comparison between Kalman filter and Particle filter for low cost INS/GPS integration," *Proceedings of the 6th International Symposium on Mechatronics and its Applications*, IEEE, Sharjah, UAE, 2009, pp. 1–7.

[25] Wei, W., Zongyu, L., and Rongrong, X., "INS/GPS/Pseudolite integrated navigation for land vehicle in urban canyon environments," *Proceedings of the Conference on Cybernetics and Intelligent Systems*, Vol. 2, IEEE, Singapore, 2004, pp. 1183–1186.

[26] Kloeden, H., Schwarz, D., Biebl, E., and Rasshofer, R., "Vehicle localization using cooperative RF-based landmarks," *Proceedings of the Intelligent Vehicles Symposium*, IEEE, Baden-Baden, Germany, 2011, pp. 387–392.

[27] Lu, M., Chen, W., Shen, X., Lam, H., and Liu, J., "Positioning and tracking construction vehicles in highly dense urban areas and building construction sites," *Automation in Construction*, Vol. 16, No. 5, 2007, pp. 647–656.

[28] Cui, Y. and Ge, S., "Autonomous vehicle positioning with GPS in urban canyon environments," *IEEE Transactions on Robotics and Automation*, Vol. 19, No. 1, 2003, pp. 15–25.

[29] Drevelle, V. and Bonnifait, P., "iGPS: global positioning in urban canyons with road surface maps," *IEEE Intelligent Transportation Systems Magazine*, Vol. 4, No. 3, 2012, pp. 6–18.

[30] Syed, S., "GPS based map matching in the pseudorange measurement domain," *Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation*, ION, Long Beach, CA, 2004, pp. 241–252.

[31] Boucher, C. and Noyer, J., "A hybrid particle approach for GNSS applications with partial GPS outages," *IEEE Transactions on Instrumentation and Measurement*, Vol. 59, No. 3, 2010, pp. 498–505.

[32] Georgy, J., Noureldin, A., and Goodall, C., "Vehicle navigator using a mixture particle filter for inertial sensors/odometer/map data/GPS integration," *IEEE Transactions on Consumer Electronics*, Vol. 58, No. 2, 2012, pp. 544–552.

[33] Muratet, L., Doncieux, S., Briere, Y., and Meyer, J., "A contribution to vision-based autonomous helicopter flight in urban environments," *Robotics and Autonomous Systems*, Vol. 50, No. 4, 2005, pp. 195–209.

[34] Wu, A., Johnson, E., and Proctor, A., "Vision-aided inertial navigation for flight control," *Journal of Aerospace Computing, Information, and Communication*, Vol. 2, No. 9, 2005, pp. 348–360.

[35] Graham, M., Steiner, T., and How, J., "Robust vision-aided navigation in urban environments," *Proceedings of the Guidance, Navigation, and Control and Co-located Conferences*, AIAA, Boston, MA, 2013.

[36] Hrabar, S. and Sukhatme, G., "Vision-based navigation through urban canyons," *Journal of Field Robotics*, Vol. 26, No. 5, May 2009, pp. 431–452.

[37] Shim, D., Chung, H., and Sastry, S., "Conflict-free navigation in unknown urban environments," *IEEE Robotics & Automation Magazine*, Vol. 13, No. 3, 2006, pp. 27–33.

[38] Soloviev, A., "Tight Coupling of GPS and INS for Urban Navigation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 46, No. 4, 2010, pp. 1731–1746.

[39] Tomić, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I., Ruess, F., Suppa, M., and Burschka, D., "Toward a fully autonomous UAV: research platform for indoor and outdoor urban search and rescue," *IEEE Robotics Automation Magazine*, Vol. 19, No. 3, 2012, pp. 46–56.

[40] Burl, J., *Linear Optimal Control: H_2 and H_∞ Methods*, Addison-Wesley Inc., Boston, MA, USA, 1st ed., 1998.

[41] Bryson, A. and Ho, Y., *Applied Optimal Control: Optimization, Estimation, and Control*, Taylor & Francis, 1975.

[42] Ducard, G., *Fault-tolerant Flight Control and Guidance Systems: Practical Methods for Small Unmanned Aerial Vehicles*, Advances in Industrial Control, Springer, 2009.

[43] Beard, R. and McLain, T., *Small Unmanned Aircraft*, Princeton University Press, 2012.

[44] Mensing, C. and Plass, S., "Positioning algorithms for cellular networks using TDOA," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 4, IEEE, Toulouse, France, 2006, p. IV.

[45] Lopes, H., Kampen, E., and Chu, Q., "Attitude determination of highly dynamic fixed-wing UAVs with GPS/MEMS-AHRS integration," *Proceedings of the Guidance, Navigation, and Control and Co-located Conferences*, AIAA, Minneapolis, MN, 2012.

[46] Jung, D. and Tsiotras, P., "Inertial attitude and position reference system development for a small UAV," *Proceedings of Infotech*, AIAA, Rohnert Park, CA, 2007, pp. 7–10.

[47] Beard, R., "State estimation for micro air vehicles," *Innovations in Intelligent Machines-1*, Springer, 2007, pp. 173–199.

[48] Aeroprobe Corporation, "Air data systems data sheet," 2014.

[49] Kim, J.and Brambley, G., "Dual optic-flow integrated navigation for small-scale flying robots," *Proceedings of the Australasian Conference on Robotics and Automation*, Brisbane, Australia, 2007, pp. 299–305.

[50] Wang, T., Wang, C., Liang, J., Chen, Y., and Zhang, Y., "Vision-aided inertial navigation for small unmanned aerial vehicles in GPS-denied environments," *International Journal of Advanced Robotic Systems*, Vol. 10, No. 276:2013, 2013.

[51] Kingston, D.and Beard, R., "Real-time attitude and position estimation for small UAVs using low-cost sensors," *Proceedings of the 3rd Unmanned Unlimited Technical Conference, Workshop, and Exhibit*, AIAA, Chicago, IL, 2004.

[52] Nemra, A. and Aouf, N., "Robust INS/GPS sensor fusion for UAV localization using SDRE nonlinear filtering," *IEEE Sensors Journal*, Vol. 10, No. 4, 2010, pp. 789–798.

[53] Kürner, T., "Requirements for X-Map-estimation in wireless self-organizing networks," FP& ICT-SOCRATES, Braunschweig, Germany, 2009.

[54] Euston, M., Coote, P., Mahony, R., Kim, J., and Hamel, T., "A complementary filter for attitude estimation of a fixed-wing UAV," *Proceedings of the International Conference on Intelligent Robots and Systems*, IEEE/RSJ, Nice, France, 2008, pp. 340–345.

[55] Lai, Y., Jan, S., and Hsiao, F., "Development of a low-cost attitude and heading reference system using a three-axis rotating platform," *Sensors*, Vol. 10, No. 4, 2010, pp. 2472–2491.

[56] de Marina, H., Espinosa, F., and Santos, C., "Adaptive UAV attitude estimation employing Unscented Kalman Filter, FOAM and low-cost MEMS sensors," *Sensors*, Vol. 12, No. 7, 2012, pp. 9566–9585.

[57] VectorNav, *AN014: VN-100 Velocity Compensation*, June 2014.

[58] Langelaan, J., Alley, N., and Neidhoefer, J., "Wind field estimation for small unmanned aerial vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 4, 2011, pp. 1016–1030.

[59] Dorobantu, A., Murch, A., Mettler, B., and Balas, G., "System identification for small, low-cost, fixed-wing unmanned aircraft," *Journal of Aircraft*, Vol. 50, No. 4, 2013, pp. 1117–1130.

[60] Lord Microstrain, "3DM-GX3-25 Miniature Attitude Heading Reference System," 2013.

[61] Beard, R. W., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., McLain, T., and Goodrich, M., "Autonomous vehicle technologies for small fixed-wing UAVs," *Journal of Aerospace Computing, Information, and Communication*, Vol. 2, No. 1, 2005, pp. 92–108.

[62] Yeo, D., Henderson, J., and Atkins, E., "An aerodynamic data system for small hovering fixed-wing UAS," *Proceedings of the Guidance, Navigation, and Control Conference*, AIAA, Chicago, IL, 2009.

[63] Quigley, M., Barber, B., Griffiths, S., and Goodrich, M., "Towards real-world searching with fixed-wing mini-UAVs," *Proceedings of the International Conference on Intelligent Robots and Systems*, Vol. 6543, IEEE/RSJ, Edmonton, Alberta, 2005, pp. 3028–3033.

[64] Man, H., Holt, R.and Wang, J., and Martini, R.and Netravali, R. M. I., "A new optical flow estimation method in joint EO/IR video surveillance," *Proceedings of the Defense and Security Symposium*, SPIE, 2007.

[65] Hwangbo, M. and Kanade, T., "Visual-inertial UAV attitude estimation using urban scene regularities," *Proceedings of the International Conference on Robotics and Automation*, IEEE, Shanghai, China, 2011, pp. 2451–2458.

[66] Horn, B. and Schunck, B., "Determining optical flow," *Artificial Intelligence*, Vol. 17, No. 1, 1981, pp. 185–203.

[67] Srinivasan, M., Lehrer, M., Kirchner, W., and Zhang, S., "Range perception through apparent image speed in freely flying honeybees," *Visual Neuroscience*, Vol. 6, No. 05, 1991, pp. 519–535.

[68] Serres, J., Ruffier, F., Viollet, S., and Franceschini, N., "Toward optic flow regulation for wall-following and centring behaviours," *International Journal of Advanced Robotic Systems*, Vol. 3, No. 2, 2006, pp. 147–154.

[69] Hrabar, S. and Sukhatme, G., "A comparison of two camera configurations for optic-flow based navigation of a UAV through urban canyons," *Proceedings of the International Conference on Intelligent Robots and Systems*, Vol. 3, IEEE/RSJ, Sendai, China, 2004, pp. 2673 – 2680.

[70] Hrabar, S. and Sukhatme, G., "Optimum camera angle for optic flow-based centering response," *Proceedings of the International Conference on Intelligent Robots and Systems*, IEEE/RSJ, Beijing, China, 2006, pp. 3922 –3927.

[71] Fleet, D. and Jepson, A., "Computation of component image velocity from local phase information," *International Journal of Computer Vision*, Vol. 5, No. 1, 1990, pp. 77–104.

[72] Barron, J., Fleet, D., and Beauchemin, S., "Performance of optical flow techniques," *International Journal of Computer Vision*, Vol. 12, No. 1, 1994, pp. 43–77.

[73] Arredondo, M., Lebart, K., and Lane, D., "Optical flow using textures," *Pattern Recognition Letters*, Vol. 25, No. 4, 2004, pp. 449–457.

[74] Sun, D., Roth, S., Lewis, J., and Black, M., "Learning optical flow," *European Conference on Computer Vision*, Marseille, France, 2008, pp. 83–97.

[75] Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., and Szeliski, R., "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, Vol. 92, No. 1, 2011, pp. 1–31.

[76] Paetzold, F. and Franke, U., "Road recognition in urban environment," *Image and Vision Computing*, Vol. 18, No. 5, 2000, pp. 377 – 387.

[77] He, Y., Wang, H., and Zhang, B., "Color-based road detection in urban traffic scenes," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 5, No. 4, Dec 2004, pp. 309–318.

[78] Newman, P., Sibley, G., Smith, M., Cummins, M., Harrison, A., Mei, C., Posner, I., Shade, R., Schroeter, D., Murphy, L., et al., "Navigating, recognizing and describing urban spaces with vision and lasers," *The International Journal of Robotics Research*, Vol. 28, No. 11-12, 2009, pp. 1406–1433.

[79] u-blox.com, "GPS: Essentials of Satellite Navigation," Compendium GPS-X-02007-D, 2009.

[80] Langley, R., "Dilution of precision," *GPS World*, Vol. 10, No. 5, 1999, pp. 52–59.

[81] Rankin, J., "An error model for sensor simulation GPS and differential GPS," *Proceedings of the Position Location and Navigation Symposium*, IEEE, Las Vegas, NV, 1994, pp. 260–266.

[82] Spirent, "An Overview of LTE Positioning," 2012, Rev. A 02/12.

[83] 3GPP, "TS 36.xxx Evolved universal terrestrial radio access (E-UTRA); LTE positioning protocol (LPP) (Release 9)," *Technical Specification Group Radio Access Network;*, September 2010.

[84] Ericsson, "Positioning with LTE," White Paper, September 2011.

[85] Kottkamp, M., Rössler, A., Schlienz, J., and Schutz, J., "LTE Release 9 Technology Introduction," White Paper, December 2011.

[86] 3GPP, "TS 36.355 Evolved universal terrestrial radio access (E-UTRA); LTE positioning protocol (LPP) (Release 10)," *Technical Specification Group Radio Access Network*, December 2011.

[87] 3GPP, "TS 36.455 Evolved universal terrestrial radio access (E-UTRA); LTE positioning protocol A; (LPPa) (Release 10)," *Technical Specification Group Radio Access Network*, September 2011.

[88] 3GPP, "TS 36.305 Evolved universal terrestrial radio access network (E-UTRAN); stage 2 functional specification of user equipment (UE) positioning in E-UTRAN (Release 10)," *Technical Specification Group Radio Access Network*, September 2011.

[89] Tam, S., Lee, H., Khasnabish, B., and Suraci, F., "MSF Whitepaper on Location Services in LTE Networks," April 2010, White Paper.

[90] Peral-Rosado, J. A. D., Lopez-Salcedo, J. A., Seco-Granados, G., Zanier, F., and Crisci, M., "Preliminary analysis of the positioning capabilities of the positioning reference signals of 3GPP LTE," *Proceedings of the 5th European Workshop on Signals and Signal Processing*, Toulouse, France, 2011.

[91] Yu, H., Huang, G., Gao, J., and Liu, B., "An efficient constrained weighted least squares algorithm for moving source location using TDOA and FDOA measurements," *, IEEE Transactions on Wireless Communications*, Vol. 11, No. 1, 2012, pp. 44–47.

[92] Bucher, R. and Misra, D., "A synthesizable VHDL model of the exact solution for three-dimensional hyperbolic positioning system," *VLSI Design*, Vol. 15, No. 2, 2002, pp. 507–520.

[93] Bull, J. F., "Wireless geolocation," *IEEE Vehicular Technology Magazine*, Vol. 4, No. 4, 2009, pp. 45–53.

[94] Neuland, M., Kurner, T., and Amirijoo, M., "Influence of positioning error on X-Map estimation in LTE," *Proceedings of the 73rd Vehicular Technology Conference*, IEEE, Budapest, Hungary, 2011, pp. 1 –5.

[95] Tao, S., *Mobile phone-based vehicle positioning and tracking and its application in urban traffic state estimation*, Ph.D. thesis, KTH Royal Institute of Technology, 2012, Licentiate Thesis.

[96] Levinson, J., Montemerlo, M., and Thrun, S., "Map-based precision vehicle localization in urban environments," *Proceedings of Robotics: Science and Systems*, Atlanta, Georgia, 2007.

[97] Culhane, A. A., *Development of an obstacle detection system for human supervisory control of a UAV in urban environments*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2007.

[98] Hokuyo Automatic Co., L., "Scanning laser range finder UTM-30LX-EW specification sheet," December 2012.

[99] Kalman, R., "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, Vol. 82, No. Series D, 1960, pp. 35–45.

[100] Welch, G. and Bishop, G., "An Introduction to Kalman filtering," *Technical Report, Department of Computer Science and Engineering, Univ. of North Carolina at Chapel Hill*, 2006.

[101] De Ruiter, A.and Damaren, C. and Forbes, J., *Spacecraft Dynamics and Control: An Introduction*, John Wiley & Sons, 2012.

[102] Ribeiro, M., "Kalman and Extended Kalman filters: concept, derivation and properties," Institute for Systems and Robotics, Lisbon, Portugal, 2004.

[103] Julier, S.and Uhlmann, J., "New extension of the Kalman filter to nonlinear systems," *Proceedings of the AeroSense 97 Conference*, SPIE, Orlando, FL, 1997, pp. 182–193.

[104] Gordon, N., Salmond, D., and Smith, A., "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEEE Proceedings F, Radar and Signal Processing*, Vol. 140, No. 2, 1993, pp. 107–113.

[105] Evensen, G., "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics," *Journal of Geophysical Research: Oceans (1978–2012)*, Vol. 99, No. C5, 1994, pp. 10143–10162.

[106] Gillijns, S., Mendoza, O., Chandrasekar, J., De Moor, B., Bernstein, D., and Ridley, A., "What is the ensemble Kalman filter and how well does it work?" *Proceedings of the American Control Conference*, Minneapolis, MN, 2006, pp. 4448–4453.

[107] Gan, Q. and Harris, C., "Comparison of two measurement fusion methods for Kalman-filter-based multisensor data fusion," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 37, No. 1, 2001, pp. 273–279.

[108] Roumeliotis, S. and Burdick, J., "Stochastic cloning: a generalized framework for processing relative state measurements," *Proceedings of the International Conference on Robotics and Automation*, IEEE, Washington, DC, 2002, pp. 1788–1795.

[109] Eustice, R., Singh, H., and Leonard, J., "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Transactions on Robotics*, Vol. 22, No. 6, 2006, pp. 1100–1114.

[110] Bar-Shalom, Y., Li, X., and Kirubarajan, T., *Estimation with Aplications to Tracking and Navigation: Theory Algorithms and Software*, John Wiley & Sons, 2004.

[111] Ristic, B., Arulampalm, S., and Gordon, N., *Beyond the Kalman filter: Particle filters for Tracking Applications*, Artech House Publishers, 2004.

[112] Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T., "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," , *IEEE Transactions on Signal Processing*, Vol. 50, No. 2, 2002, pp. 174 –188.

[113] "BYTES of the BIG APPLE," *Department of City Planning City of New York*, 2013.

[114] ETH, "Swiss Federal Insititute of Technology Zurich: aerobatic - guidance and control for aerobatic maneuvers of an unmanned airplane," 2006.

[115] Matthews, J. H., "Module for Runge Kutta Method for O.D.E.'s," 2003, California State University Fullerton.

[116] Groves, P., "Shadow matching: a new GNSS positioning technique for urban canyons," *Journal of Navigation*, Vol. 64, 7 2011, pp. 417–430.

[117] Ficek, M., Pop, T., and Kencl, L., "Active tracking in mobile networks: An in-depth view," *Computer Networks*, Vol. 57, No. 9, 2013, pp. 1936 – 1954.

[118] Bartels, C. and De Haan, G., "Smoothness constraints in recursive search motion estimation for picture rate conversion," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 20, No. 10, 2010, pp. 1310–1319.

[119] Amirjoo, M., Jorgueseski, L., Kurner, T., Litjens, R., Neuland, M., Scmelz, L., and Turke, U., "Cell outage management in LTE networks (COST 2100 TD(09) 941)," FP7-ICT-SOCRATES, 2009, Slides.

[120] Lucas, B. and Kanade, T., "An iterative image registration technique with an application to stereo vision," *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, 1981, pp. 674–679.

[121] Bruhn, A., Weickert, J., and Schnörr, C., "Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods," *International Journal of Computer Vision*, Vol. 61, No. 3, 2005, pp. 211–231.

[122] Camus, T., "Real-time quantized optical flow," *Journal of Real-Time Imaging*, Vol. 3, 1995, pp. 71–86.

[123] Bryson, M. and Sukkarieh, S., "Vehicle model aided inertial navigation for a UAV using low-cost sensors," *Proceedings of the Australasian Conference on Robotics and Automation*, Canberra, Australia, 2004, pp. 37–45.

[124] Bry, A.and Bachrach, A. and Roy, N., "State estimation for aggressive flight in GPS-denied environments using onboard sensing," *Proceedings of the International Conference on Robotics and Automation*, IEEE, Saint Paul, MN, 2012, pp. 1–8.

[125] Ramprasadh, C. and Arya, H., "Estimation of Aerodynamic Angles in a Mini Aerial Vehicle under Turbulent Atmosphere," *Proceedings of the Atmospheric Flight Mechanics Conference*, AIAA, Portland, OR, 2011.

[126] Perry, J., Mohamed, A., Johnson, B., and Lind, R., "Estimating angle of attack and sideslip under high dynamics on small UAVs," *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation*, ION, Savannah, GA, 2008, pp. 16–19.

[127] Langelaan, J., Spletzer, J., Montella, C., and Grenestedt, J., "Wind field estimation for autonomous dynamic soaring," *International Conference on Robotics and Automation*, IEEE, Saint Paul, MN, 2012, pp. 16–22.

[128] Singh, L. and Fuller, J., "Trajectory generation for a UAV in urban terrain, using nonlinear MPC," *Proceedings of the American Control Conference*, Vol. 3, 2001, pp. 2301–2308.

[129] Watanabe, Y., Lesire, C., Piquereau, A., Fabiani, P., Sanfourche, M., and Le Besnerais, G., "The ONERA RESSAC unmanned autonomous helicopter: visual air-to-ground target tracking in an urban environment," *Proceedings of the American Helicopter Society 66th Annual Forum*, Phoenix, AZ, 2010.