Height Estimation of a Blimp Unmanned Aerial Vehicle Using Inertial Measurement Unit and Infrared Camera

by

Hubert Villeneuve

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Mechanical Engineering

Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa

© Hubert Villeneuve, Ottawa, Canada, 2017

Abstract

Increasing demands in areas such as security, surveillance, search and rescue, and communication, has promoted the research and development of unmanned aerial vehicles (UAVs) as such technologies can replace manned flights in dangerous or unfavorable conditions. Lighter-than-air UAVs such as blimps can carry higher payloads and can stay longer in the air compared to typical heavier-than-air UAVs such as aeroplanes or quadrotors. One purpose of this thesis is to develop a sensor suite basis for estimating the position and orientation of a blimp UAV in development with respect to a reference point for safer landing procedures using minimal on-board sensors. While the existing low-cost sensor package, including inertial measurement unit (IMU) and Global Navigation System (GPS) module, could be sufficient to estimate the pose of the blimp to a certain extent, the GPS module is not as precise in the short term, especially for altitude. The proposed system combines GPS and inertial data with information from a grounded infrared (IR) camera. Image frames are processed to identify three IR LEDs located on the UAV and each LED coordinate is estimated using a Perspective-n-Point (PnP) algorithm. Then the results from the PnP algorithm are fused with the GPS, accelerometer and gyroscope measurements using an Extended Kalman Filter (EKF) to get a more accurate estimate of the position and the orientation. Tests were conducted on a simulated blimp using the experimental avionics.

Acknowledgements

I wouldn't have been able to complete this thesis without the support and encouragements from my supervisor, Eric Lanteigne, my friends and family. I would like to give special thanks to the people who helped with performing the tests: Ahmad Alsayed, François L'Écuyer, André Audette, and the IT technicians.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

ANN	Artificial Neural Network
EKF	Extended Kalman Filter
FL	Fuzzy Logic
GUI	Graphical user interface
IMU	Inertial Measurement Unit
IR	Infrared
KF	Kalman Filter
PnP	Perspective-n-Point
RFID	Radio Frequency Identification
UAV	Unmanned Aerial Vehicle

Mathematical Symbols

α	Angle

Δ	Uncertainty
δ	Partial derivative
$\hat{X}$	State vector estimation
λ	Latitude
ω	Angular velocity
ϕ	Roll angle about the x axis
ψ	Yaw angle about the z axis
σ_i^2	Variance of random variable i
σ_{ij}	Covariance of random variables i and i
$\mathbf{0}$	Zero matrix
θ	Pitch angle about the y axis
φ	Longitude
a	Linear acceleration
A_r	Area
b	Image width
d	Distance
E	Expected value operator
e	Error
F	State transition matrix
fl	Focal length
G	Control input transition matrix
g	Gravity constant

H	Measurement transition matrix
h	Image height
I	Identity matrix
K	Kalman gain matrix
O	Reference frame origin
P	State covariance matrix
p	Position coordinates
P_e	Perimeter
Q	Process noise covariance matrix
q	Process noise
R	Measurement noise covariance matrix
R	Rotation matrix
r	Measurement noise
s	Scale factor
t	Time
U	Control input vector
v	Vertical position on image
v_h	Ground velocity
w	Horizontal position on image
X	State vector
x	x axis
y	y axis

Z	Measurement vector
z	z axis

Superscripts

$-$	Prior estimate
b	Body reference frame
c	Camera reference frame
i	Image reference frame
n	World reference frame

Subscripts

0	Initial time step
cam	Measurements from camera
GPS	Measurements from GPS
k	Time step
px	Pixel units

Chapter 1

Introduction

1.1 Background

There has been a rise in Unmanned Aerial Vehicles (UAVs) research over the recent years for the attractive aspects they present [33]. For instance, they have longer flight endurance and lower energy consumption for hovering, as opposed to typical heavier-than-air Unmanned Aerial Vehicles (UAVs) such as quadcopters. Potential areas of application include surveillance, night operations, search and rescue missions, and data gathering.

For a UAV to have a level of conditional automation [34], it must be able to depart and land by itself. Landing is a delicate maneuver: precise control inputs are required as the UAV approaches the ground to prevent potential collisions and damaging the components. Knowing the position and orientation of the UAV with respect to an origin is crucial to perform such a task. A

widespread method of determining the pose of a UAV is to combine measurements from Inertial Measurement Unit (IMU) and GPS, sensors which can be low-cost, with a Kalman Filter (KF), a sensor fusion technique relatively simple to implement.

1.2 Problem Statement and Solution

Although landing an autonomous unmanned blimp presents additional challenges such as vulnerabilities to environmental disturbances, this thesis concentrates on finding the pose of a blimp UAV seen in Figure 1.1. The low-cost GPS receiver available on this UAV has low accuracy and precision with respect to height, which is problematic if one is to use a sensor fusion technique to estimate position and orientation for safe landing. As such, the main purpose is to find an alternative to using GPS, for height measurement. The secondary purpose is to provide a basis for a sensor package to estimate pose with respect to a reference point. The blimp system can be summarized in the block diagram of Figure 1.2 where the grayed boxes represent the focus of this thesis.

A novel approach to solve the problem is presented in this thesis. A pattern of IR LEDs, representing the front of the blimp, are detected by an infrared (IR) camera on a ground station. After simple image processing techniques, position and orientation can be estimated from camera frames using the Perspective-n-Point (PnP) algorithm. Instantaneous linear acceler-

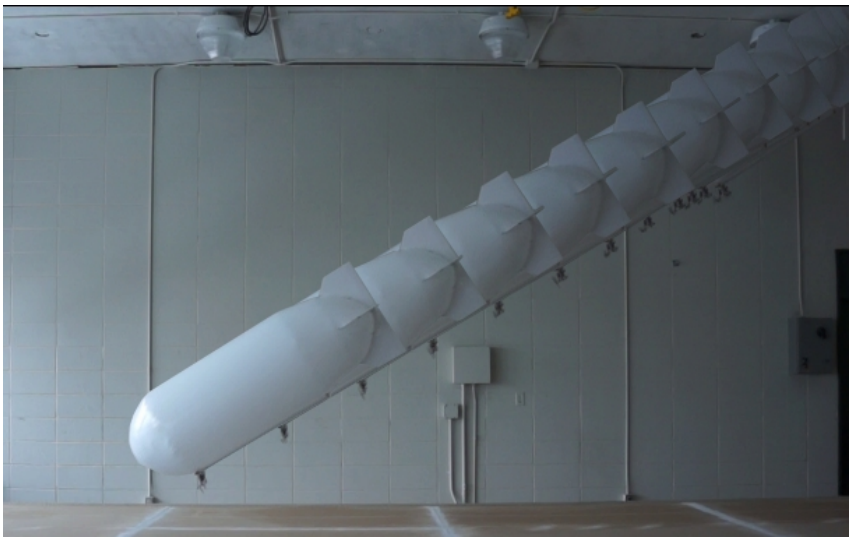


Figure 1.1: Composite image of blimp moving downwards [1].

ation and angular velocities measurements are obtained from the IMU on the blimp's gondola and transmitted to the ground station via Bluetooth. Pose estimation is then performed at the ground station by fusing the camera and IMU data using an Extended Kalman Filter (EKF). Having the camera on the ground station allows to reduce the weight on the blimp. By performing the pose estimation on the ground station, the computational load on the blimp's microcontroller is lessened.

1.3 Thesis layout

The core of this thesis will be divided into four major chapters. Sensors for pose estimation and sensor fusion algorithms used in existing UAV or mobile robot systems will be described in Section 1.4. Chapter 2 will elaborate on the sensor theory, the IR camera, and the sensor fusion technique, the EKF, used

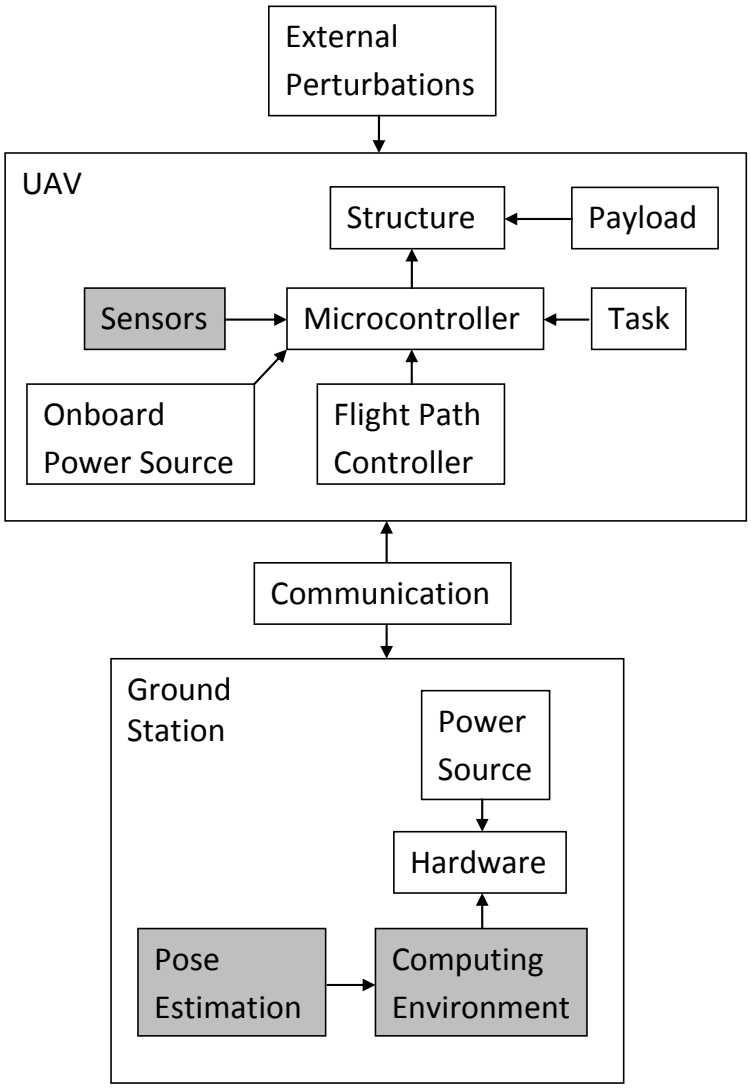


Figure 1.2: System Block Diagram.

to estimate the position and orientation of the UAV. A list of the equipment used in the experiments will be presented in Chapter 3. Finally, Chapter 4 will detail the various tests executed from the proposed method as well as discussing the results.

1.4 Literature review

This section will be divided in two main categories: sensor technologies and sensor data fusion algorithm. Each element will follow the same presentation formula: a brief description of the device and methodology, followed by advantages and disadvantages when applied to Unmanned Aerial Vehicles (UAVs) with examples.

1.4.1 Sensor Technologies

There are many types of sensors available for estimating the position of a UAV or a mobile robot. The following section examines six technologies.

IMU

An Inertial Measurement Unit (IMU) is a sensor package used to determine relative orientation and position. It often consists of a three-axis accelerometer and a three-axis gyroscope used to measure, respectively, the instantaneous linear accelerations and angular velocities. The accelerometer is noisy but reliable in the long term while the gyroscope drifts over time but is accurate

in the short term. The linear accelerations and angular velocities can be integrated over time to estimate the position and orientation of a UAV with respect to a fixed frame [35]. Figure 1.3 shows an MPU6050, an example of an IMU sensor. With a weight of 1.4 g, its output data rate can go up to 8000 Hz for gyroscope and 1000 Hz for accelerometer.

An IMU can work with an onboard computer on a UAV to give its relative orientation and location. There is no need for external communications and no dependence on environmental conditions such as lighting. One major disadvantage of IMU comes from integration: errors quickly accumulate over time, reducing the accuracy. Error sources can take many forms such as sensor misalignments, relative orientation, in the case where the takeoff location moved for instance, or vibrations. Moreover, measurements are discrete and can be rounded, leading to additional errors over time. Due to this, an IMU is almost never used by itself for UAV navigation.

Figure 1.3: 6 DOF IMU [2]

GNSS

The Global Navigation Satellite System (GNSS) consists of determining one's current location and velocity on the Earth from a network of satellites. The

GNSS system used in North America is the Global Positioning System (GPS). This thesis will refer to GNSS as GPS. As a signal is sent from a GPS receiver, the time taken to reach the satellites with direct line of sight is known. Then, the position is estimated by triangulation [36]. GPS can be fairly useful in knowing the position of a UAV on Earth. An example of a GPS sensor is the MTK3339, in Figure 1.4, with an update rate of 10 Hz and position accuracy of 1.8 m. [2]

The position given by GPS is generally reliable as it does not depend on previous data compared to IMU. But, it has lower sampling rate so quick changes in motion can not be recorded as well. Another disadvantage of GPS is its dependency on environmental conditions: it will not work properly if the signal is distorted or can not reach at least four satellites, such as when indoors or near large obstructions. Other weaknesses include less accurate measurement for altitude than for horizontal position and time required, in the order of several seconds, to get the first reading.

There exists some methods used to increase the performance of GPS and cover some of its weaknesses. Assisted GPS can give a quicker starting performance by acquiring initial information from an assistance server. Differential GPS or Real-Time Kinematic can improve positioning accuracy to the order of centimeters by reaching grounded stations.

Typical UAV applications combine GPS and IMU to measure accurate position and orientation as one sensor's strengths can cover the other's weak-

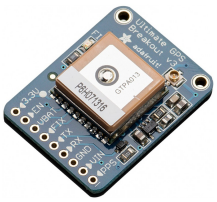


Figure 1.4: GPS Sensor [2]

nesses. For instance, Sharp et al use such a system as theoretical values for comparison with vision tracking on a helicopter UAV as it has an accuracy of 2 cm [37]. The system is implemented on a Yamaha R-50 helicopter seen in Figure 1.5 with its camera and on-board navigation computer. Nevertheless, it is possible to use IMU without GPS or the opposite up to a certain degree. Wendel et al develop GPS/IMU system for UAV localization [4]. One of their main focus is to maintain sufficient attitude accuracy during GPS outage using IMU measurements only. Some of their simulation results are shown in Figure 1.6. The angle measurement error, with roll angle in a) and pitch angle in b), is plotted for the proposed method, in red, versus a classical method, in blue. It can be seen that the proposed method keeps the error less than 2^o when GPS is unavailable between 120 s and 180 s, as opposed to the classical method with error going past 10^o. Cho et al succeed at using a single-antenna GPS receiver only for a fixed wing aircraft UAV navigation [5]. Attitude information relies on assumptions specific to the UAV design, however. The 3D plot in Figure 1.7 illustrates how the test trajectory of their UAV, in red, follows closely the predicted trajectory, in dashed blue. The horizontal trajectory is also shown in blue on the north-east plane. The GPS receiver used updates at a rate of

10 Hz.

Figure 1.5: Yamaha R-50 helicopter [3]

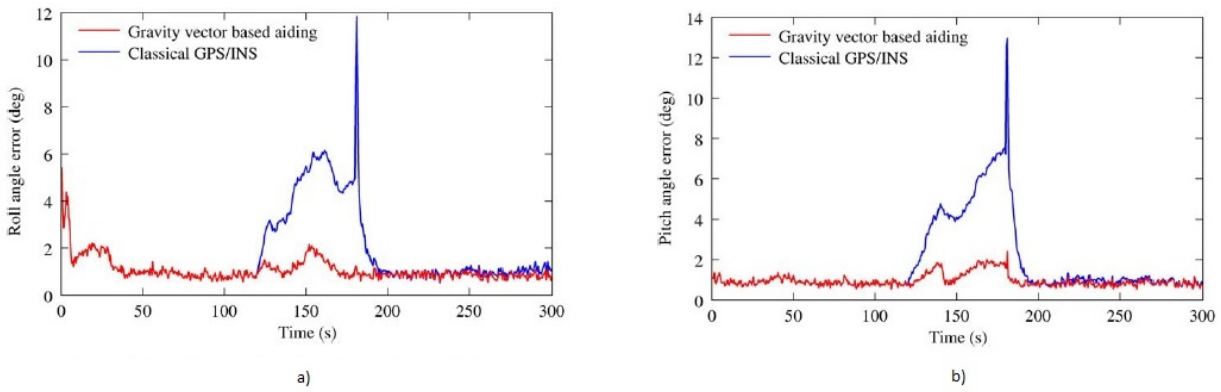


Figure 1.6: Gravity vector based aiding versus GPS/INS: a) roll angle, b) pitch angle [4]

Visible Light

Light is a part of the electromagnetic spectrum with wavelength between 400 and 700 nm. Computer vision in UAV applications often uses visible light

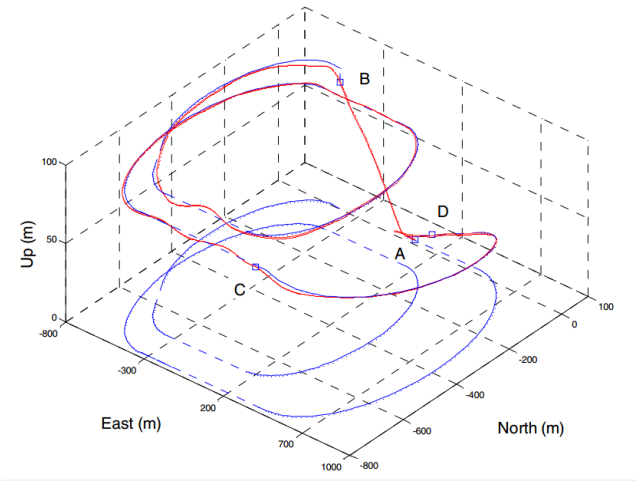
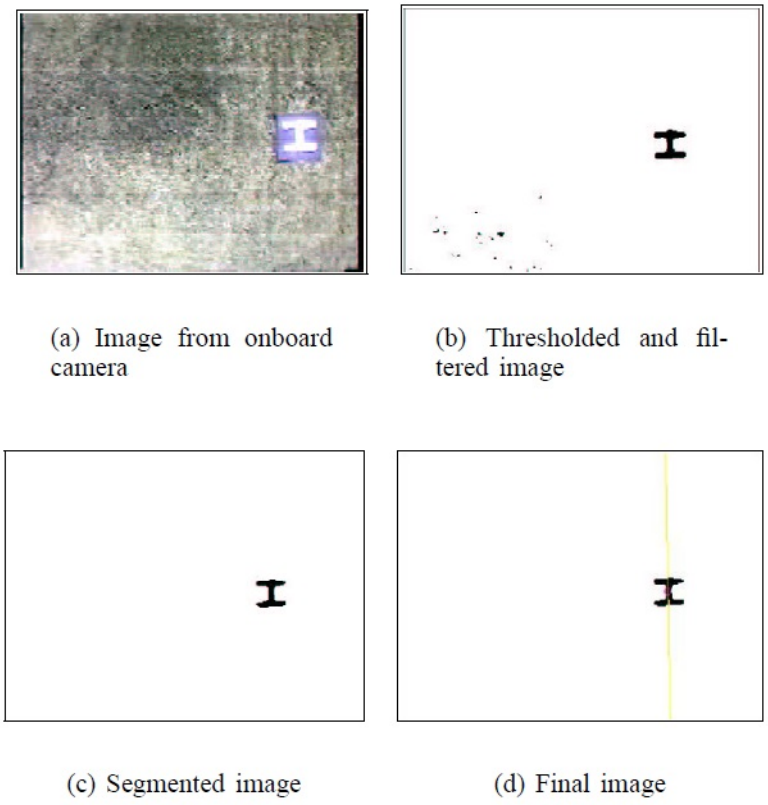


Figure 1.7: Real versus estimated trajectory [5]

cameras, which can weigh less than 60 g in the case of webcams. A typical practice is to have a visual camera onboard a drone to locate and track a target in real time. The images require processing to extract useful information for the UAV such as distance and orientation with respect to the landing pad. A common procedure for image processing, illustrated in Figure 1.8, is the following: convert the image to grayscale, convert it to binary using thresholding, split it in different regions with segmentation and attempt to recognize the target in one of the regions with pattern recognition [6]. It is also possible to correct the pictures from camera distortion using geometric relations such as affine transformation [7]. Figure 1.9 demonstrate an example of before and after moment invariant correction. Regarding pattern recognition, it is important to choose a target with a distinct pattern or large contrast with the background.

Cameras are relatively inexpensive, easily available and more intuitive for a

human being. Image processing can be fast and allows for object recognition. An advantage to using vision is that it can be used indoors, as opposed to GPS which must be used outdoors. A disadvantage is the necessity to having unobstructed visual contact with the target. As such, this method is less optimal in situations where the field of view is blocked due to obstacles or in no-line-of-sight applications.



(a) Image from onboard camera

(b) Thresholded and filtered image

(c) Segmented image

(d) Final image

Figure 1.8: Image Processing of H-shaped target [6]

As was mentioned previously, tracking using visible light cameras is commonplace in UAV research and their usage often shares many similarities. Saripalli et al explore visual tracking of a stationary and moving H-shaped target with a helicopter drone [6]. Images are acquired with an on-board cam-

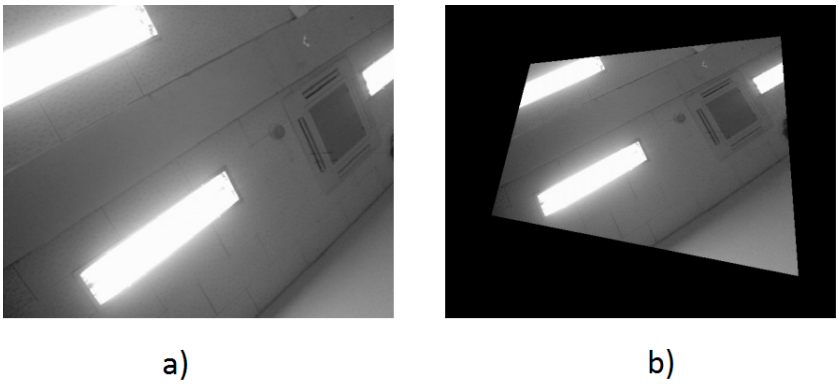


Figure 1.9: a) Raw Image, b) Undistorted image [7]

era and processed off-board at a rate of 10 frames per second. The results have an accuracy of 31 cm in and 6^o in orientation. Yang et al describe a method for vision-based landmark location estimation and application for helicopter autonomous landing on an H shape [8]. As seen in Figure 1.10, image processing is done in the following order: image acquisition, thresholding, edge detection, line detection, intersecting point location and pattern put on top of the original image. Simulation gave an average error of 1.144 pixels and processing a single image takes 0.66 s.

Bi et al land an AR.Drone using an on-board color camera on a moving platform with an H-shaped target [9]. As the target is green, they use a color RGB filter rather than going through grayscale for image processing. This is shown in Figure 1.11. The vision algorithm is deemed fast and effective enough. Lange et al manage to land a Hummingbird UAV on a distinct pattern consisting of four white concentric rings inside a black hexagon [10]. Images were both acquired with a Logitech QuickCam Pro 4000 camera and processed

on-board. The result of processing applied on the original picture is shown on the bottom of Figure 1.12. The algorithm runs at a frequency of 10 Hz to 100 Hz and position errors were less than 5 cm.

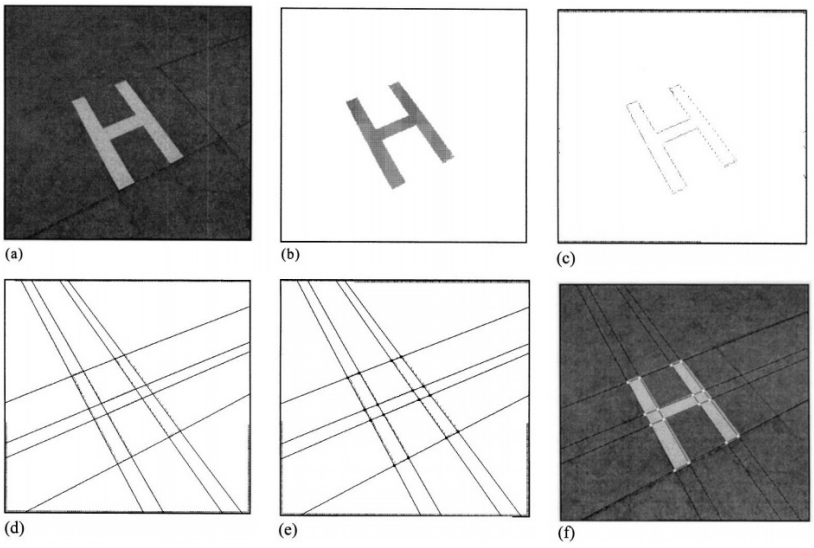


Figure 1.10: Image processing: a) Original image, b) thresholding, c) edge detection, d) line detection, e) collinear points detection, f) image processing with original image [8]

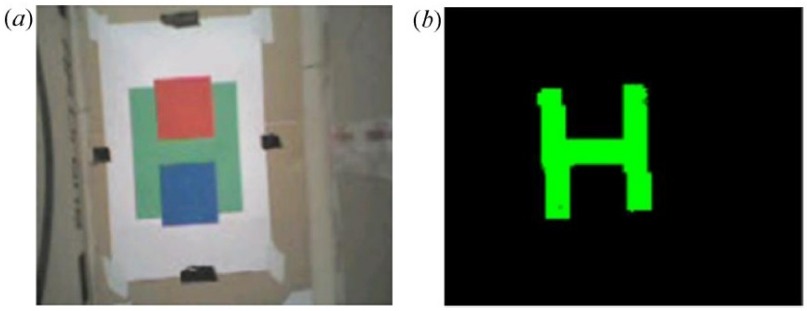


Figure 1.11: Green target: a) original image, b) processed image [9]

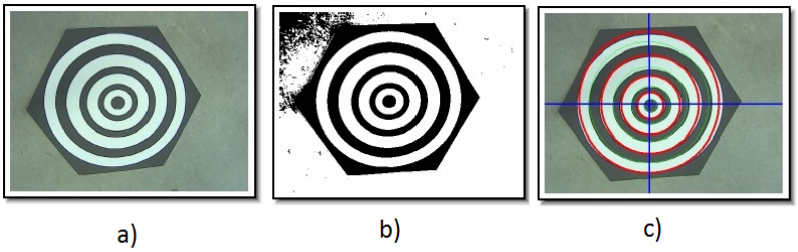


Figure 1.12: Concentric rings target: a) original image, b) binary image, c) detected rings [10]

Infrared Light

Infrared Radiation (IR) belongs to the electromagnetic spectrum with wavelength longer than 700 nm and of relatively low energy. As opposed to a visible light camera, an IR camera captures infrared waves in the shorter range. IR sources are then seen as additional light sources. Thermography, or thermal imaging, is a type of IR imaging where each pixels in one frame corresponds to a temperature. As such, a thermal imaging camera, working with longer IF waves in the 8000 to 15000 nm range, can illustrate the temperature gradient across a scene through a color scheme. For example, the bright hand in Figure 1.13 is relatively hotter than the two darker glasses. A small IR camera can weigh around 160 g. As far as image processing goes, it is done in a similar fashion as with visible light. [11]

IR cameras hold the same advantages as visible light cameras. For the purpose of robot localization, thermal imaging is preferable to visible light in the context where the view can be impeded by weather conditions. These can include night time, rain, snow or fog. While the range of an IR camera can

be affected by precipitations due to light scattering from water droplets for instance, it can still see better than a visible light camera in such a case. An IR camera also tends to be heavier than typical cameras.

Figure 1.13: Thermography example: hand holding a glass [11]

IR has been investigated many times for UAV localization purposes. Xu et al propose a method for UAV landing using an IR camera at close ranges [12]. The experimentation use an electrically powered T-shaped target coated with highly emissive material to increase contrast in temperature with the environment. The pattern is recognized at a speed of $17,2$ ms and rate of $97,2\%$ using an IR camera. Image processing is illustrated in Figure 1.14: the original image is at the first line, then the intensity histogram, thresholded image, binary and segmented image, and identified target. Lee et al present a technique for mobile robot localization using IR illumination and a regular camera [7]. An unpowered, artificial landmark made with IR reflecting material is photographed twice successively: with IR LED illumination, Figure 1.15 a), and without, Figure 1.15 b). By differentiating and correcting the distorted images, in Figure 1.15 c), the landmark can be identified with a 1% error average.

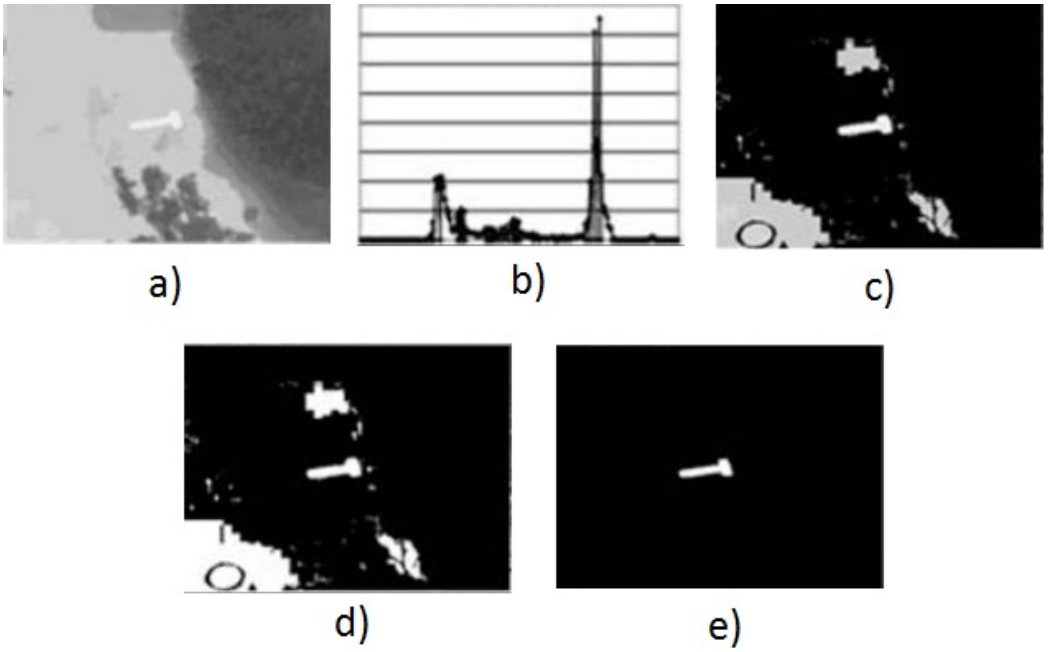


Figure 1.14: Thermal image processing: a) original images, b) intensity histogram, c) threshold, d) segmentation method 1, e) segmentation method 2 [12]

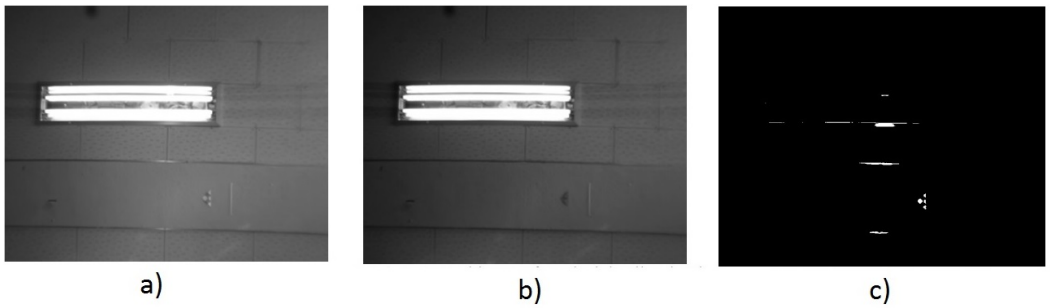


Figure 1.15: Image differentiation: a) pattern with IR illumination, b) pattern without IR illumination, c) image difference [7].

A tracking method using a cheap, consumer electronics Wii remote IR camera is presented by Wenzel et al [13]. The camera mounted on a UAV tracks a pattern of four IR lights, with geometry illustrated in Figure 1.16, fixed on a moving platform. The pose and orientation of the UAV is extracted from the pictures with the light pattern with an overall success rate of 90% and frequency of 50 Hz. Lastly, Yakimenko et al detail a method using IR to determine the relative position and orientation of a UAV with respect to a ship [14]. At large distances, it is viewed as a single hot spot. At closer distances, the ship is identified by three points with known distances as seen in Figure 1.17: the smokestack, and two intersections of the deck's edges. The correct pose and orientation at closer distance can be found using these three points at a rate of 20 Hz with their equipment. Flight testing on land, using an on-board IR camera, gave errors of ±5 m at distances fewer than 250 m.

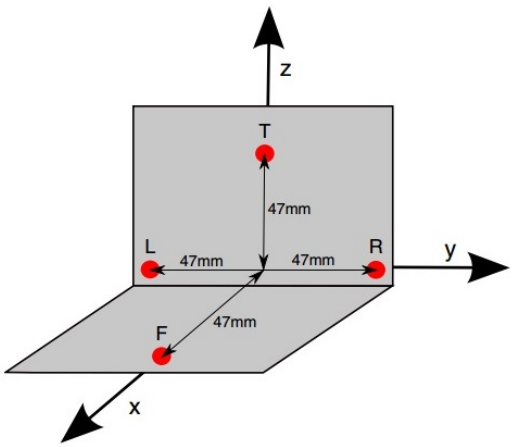


Figure 1.16: IR light pattern [13]

Figure 1.17: Ship point references: edges and smokestack [14]

RFID

Localization using radio waves, electromagnetic waves with wavelength longer than 1 mm, can be achieved with Radio-Frequency Identification (RFID) technology. An RFID tag, or transmitter, is composed of two main parts: an antenna and an Integrated Circuit (IC). The antenna serves to communicate the tag's information, which can be its unique identifier or additional information stored in its IC memory, to an RFID reader, such as in Figure 1.18. There exist two types of tags: active and passive. A passive transmitter is self-powered while an active tag requires external energy such as a battery. For this reason, active tags are more expensive, larger, and require regular battery changing. However, they are more resistant to outside conditions and have a greater communication range than passive transmitters. The range at which an RFID tags can be read can go up to 100 m. Lower end tags can have reading ranges as small as 2 cm. For mobile robot localization, a single reader, which can weigh as low as 10 g, would typically be on-board while multiple tags, which can weigh lower than 5 g, are used as targets. [15]

RFID technology has the benefits of not needing light nor a GPS network. The information on a tag can allow to identify particular objects. Tags can also be read when out of sight or behind walls, due to the nature of electromagnetic waves. For the same reason, however, signals can be affected by adverse weather or obstacles and not picked up properly by readers.

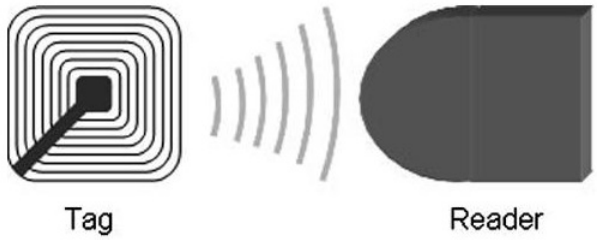


Figure 1.18: Main components of RFID [15]

Existing research on robot localization and RFID is limited to mobile robots, and use the signal itself or the tag's information. For example, Kim et al used an RFID tag as a goal to guide a mobile robot towards it in an obstacle free room [16]. The distance to the transponder is calculated from the signal strength and the orientation, through the strength pattern obtained by rotating the robot's directional antenna which acts as an RFID reader. The robot can reach the goal with an angle accuracy of $\pm 4^o$. Kim et al expanded on the idea to work in a cluttered environment by using two antennas instead [38]. Figure 1.19 shows a test with the mobile robot, transponder and obstacle in between. The plot below illustrates how the obstacle affects the signal strength.

In the second example, Park et al present a different method using passive

RFID embedded in the ground in a grid pattern to guide a mobile robot to a goal, concept illustrated in Figure 1.20. This enables a mobile robotic chair with RFID reader underneath to navigate in an obstacle free room. The robot can estimate its localization and pose by reading the tags, with a sensing range radius of 17 cm, as it moves towards the goal. The localization error is 13.3 cm in the x axis and 5.7 cm in the y axis. [17]

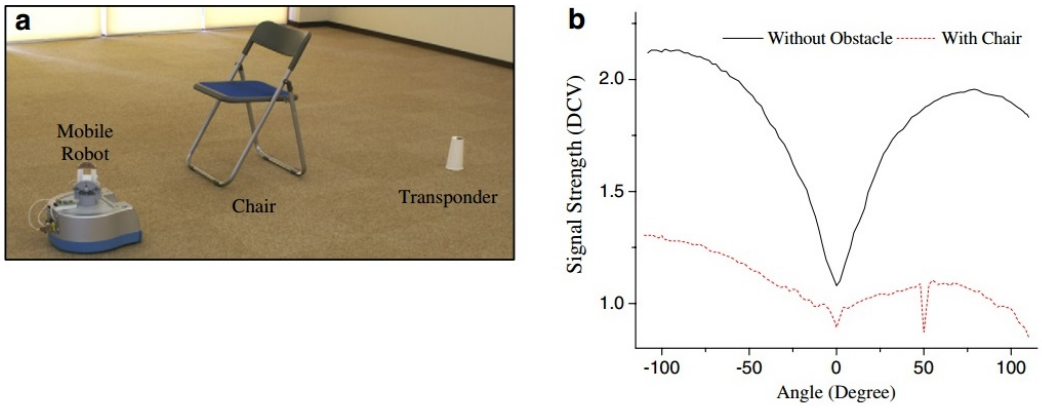


Figure 1.19: RFID robot: a) robot in cluttered environment, b) signal strength versus angle [16]

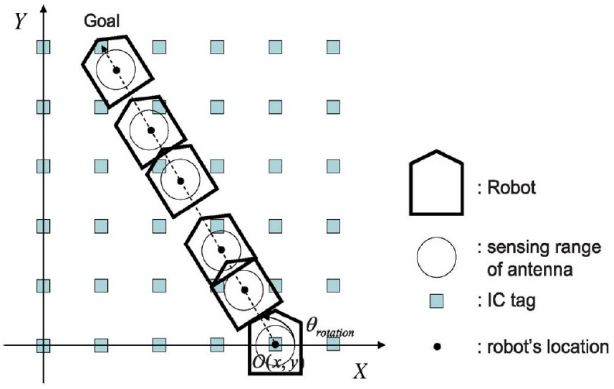


Figure 1.20: Concept of RFID grid pattern [17]

Ultra-wide band technology presents interesting properties that can benefit

RFID systems such as the ones previously described. For instance, the measurement of time-of-arrival allows for precise localization at submeter levels. Moreover, active tags using this technology have a lower energy consumption, thus becoming more attractive for lower-cost, larger distance applications [39].

Ultrasound

Ultrasound is defined as sound waves with frequencies higher than the upper human hearing limit of 20 kHz. Sonar sensors are used to estimate distances with objects by measuring the time of flight elapsed between the emission of an ultrasonic wave and its reception after being reflected. This is illustrated in Figure 1.21. Bats using echolocation to "see" and catch prey is a natural example of ultrasound localization. The maximum measuring range of a sonar transponder, combining emitter and receiver, can be up to 15 m. A transponder can weigh 50 g and gather data at a frequency of 10 Hz, such as for an XL-MaxSonar-WR1 Ultrasonic Range Finder. For UAVs, ultrasonic sensors are frequently used to measure altitude since emitted pulses can only be received by reflecting on solid objects, the ground in this case. [18]

Ultrasonic sensors have the advantages of being relatively inexpensive and not requiring light nor GPS to function. Sonars require surfaces to measure distance. As such, they are only useful for measuring altitude in open spaces. They also can't identify objects unlike RFID or vision-based methods. As such, they need to be combined with other sensors for landing purposes. A

major disadvantage of using ultrasound is that soundwaves can be skewed or reflected. For instance, if the object of interest is curved or angled with respect to the sensor, the soundwaves will not be reflected towards the receiver. More so, the sonar must be calibrated to accommodate changes in actual speed of sound due to temperature or pressure.

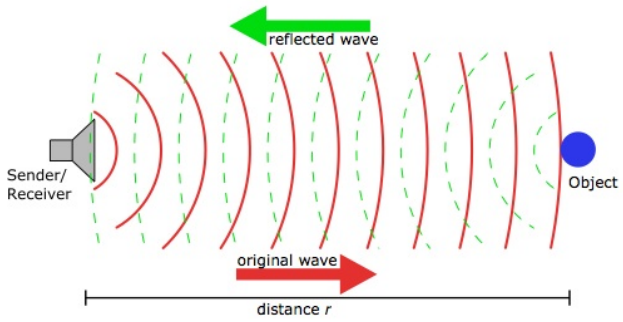


Figure 1.21: Principle of Ultrasound [18]

Mobile robot localization research relying uniquely on ultrasound are not common. Ghidary et al proposed a mobile robot positioning method combining ultrasonic and infrared signals [19]. The system consists of a transmitter mounted on the robot and six receivers installed on the ceiling. The robot position is known, at a rate of 10 ms, through ultrasonic range finding and triangulation when an infrared pulse is sent to the receivers. Figure 1.22 a) shows their results for static positioning with ten readings for each point. Figure 1.22 b) shows the measured heading angle when moving over different distances. It is concluded that the heading angle is accurate enough when measured above 50 cm. Mwakibinga et al inspire themselves from the bat echolocation navigation to determine the altitude and pitch of a small UAV and control its

tail-flap [20]. The idea is to mimic a bat's head, using a sonar speaker pointed towards the ground and two microphones angled at 45^o as seen in Figure 1.23. The altitude can be calculated from the time of flight and pitch angle, from the difference in amplitude between the two microphones. Accuracy for altitudes up to 8', over grass-like surface, is ± 1" and $\pm 5^o$ for pitch.

a)

Actual position (cm)	Mean measured Position	Max Error	Standard deviation
X=250 Y=280	X =250.8 Y =278.8	Δx=1.5 Δy=3	Std(x)=0.51 Std(y)=1.08
X=430 Y=210	X =429.4 Y =209.3	Δx=2 Δy=3	Std(x)=0.78 Std(y)=1.18
X=150 Y=150	X =148.5 Y =149.6	Δx=2.5 Δy=3.5	Std(x)=0.78 Std(y)=1.18

b)

	Error in measured angle in different distances (deg.)			
Actual heading(deg.)	At 20cm	At 30cm	At 40cm	At 50cm
0	9.8	6.6	4.9	3.2
90	10.1	5.9	5.2	3.1
45	13.4	8.8	6.9	4.1

Figure 1.22: Experimental results: a) static positioning with 10 readings per point (in cm), b) measuring heading angle [19]

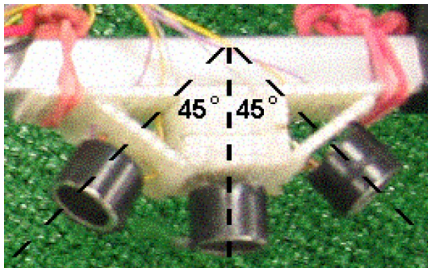


Figure 1.23: US sensors: emitter pointing downwards and receivers at 45^o [20]

1.4.2 Sensor Fusion

While a sensor can give data related to distances, it isn't sufficient by itself. For instance, errors over measurements can accumulate over time. A sensor fusion method can correct such difficulties as well as providing better estimates.

Particle Filter

A Particle Filter (PF) is an estimation method which can predict the next state of a system from the previous state and from sensor measurements. The idea is to represent the next state by a set of random samples drawn from the state. Then, a weight is calculated for each particle, representing the likeliness of being the right value given sensor measurements. The posterior is then the weighted average of the particles. If the algorithm stops here, it is known as Sequential Importance Sampling (SIS) and the particle weights can become too diluted with time. To solve this problem, a resampling step is added where the new particles are drawn and weighted from the finite number of particles of the first sampling. Here, the algorithm is referred to as Sampling Importance Resampling (SIR). Figure 1.24 shows an example of a PF. The state, in red, is propagated for each time step k. The measurement is shown in blue and particles, in grey. It can be seen that the particles in the SIR plot are less spread out compared to the SIS plot. [21]

Some advantages of PF are that they are not limited to normal distributions or linear systems. Also, the complexity of calculations is low. But, the method

has large computational requirements as thousands of particles are resampled every iteration.

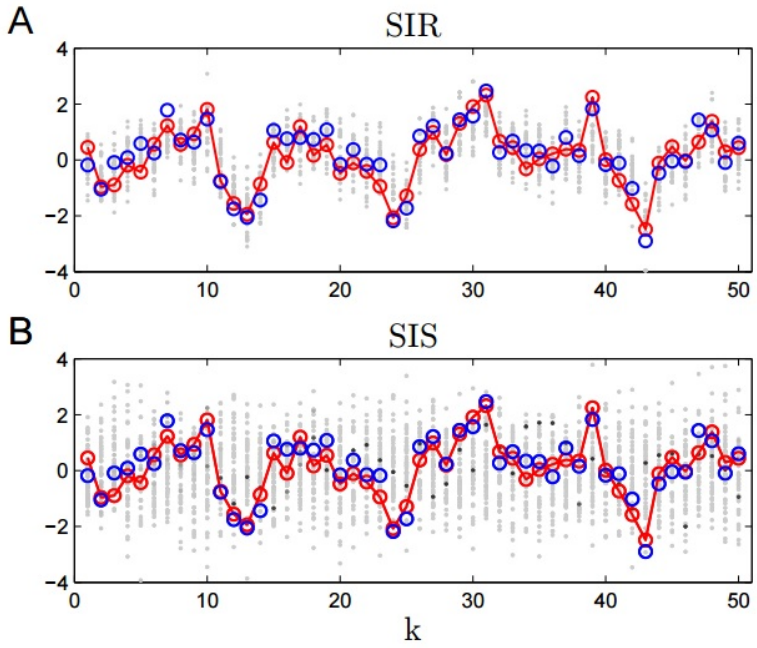


Figure 1.24: PF example: a) using SIR, b) using SIS [21]

Particle filtering has found its way in some UAV tracking applications. Caballero et al use a HERO-3 helicopter UAV to identify the position of grounded nodes [22]. The UAV regularly sends a signal with its position coordinates to the nodes which in turn sends their own coordinates to a computer, which estimates their position using PF with an error less than 5 m. Figure 1.25 illustrates the trajectory of the UAV in blue, the real node locations in green and the estimates in red. Ludington et al presents different vision-based operations using a GTMax helicopter UAV [23]. One of the operations consists of tracking a grounded moving van aided by a PF combining GPS, IMU and vision. Their system had relatively low error and was able to successfully follow

the target, as can be seen from the red shapes overlapping the van in Figure 1.26.

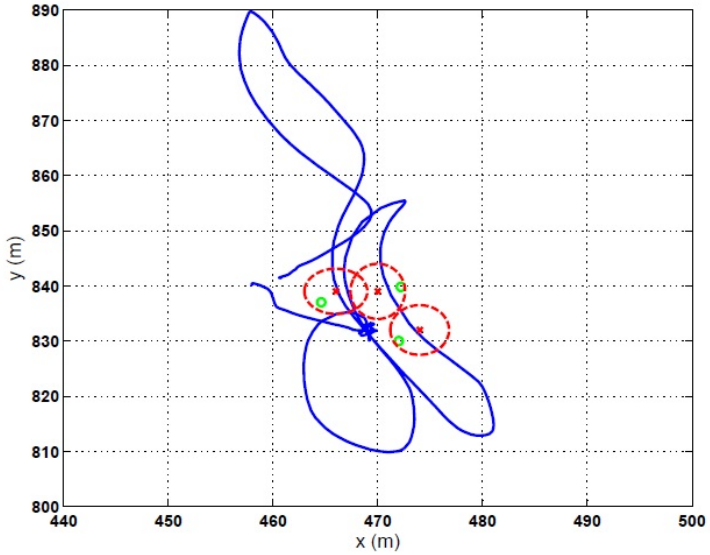


Figure 1.25: UAV trajectory and node, real and estimated positions [22]

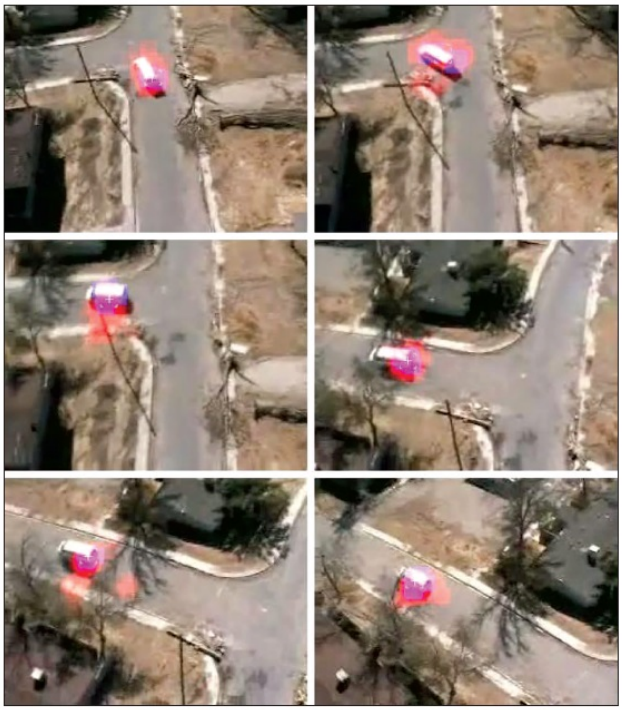


Figure 1.26: Van recognition [23]

Kalman Filter

A Kalman Filter (KF) is similar to a PF in that it can predict the next state using the previous state. The first step in a KF is to estimate the previous state as well as its corresponding covariance matrix estimation. The second step consists of updating from the sensor measurements. The Kalman gain is computed using the covariance matrix estimation. With the measurements, the covariance matrix is updated and an estimate of the next state can be calculated. For following states, the cycle simply repeats. This algorithm is summarized in Figure 1.27. [24]

The KF requires less computational load than the PF. KF is very precise, but is optimal only if the error can be described with a normal probability distribution. A weakness in the KF is that it requires the state to be estimated and relationship between the state and measurement to be based on linear equations. If the linearity of the KF can not be assured, variations of the KF can enable this assumption to remain valid, such as the Extended Kalman Filter (EKF) which approximates a nonlinear system as a linear one.

KF is very popular in UAV research. One other operation performed by Ludington et al with their GTMax helicopter is to approach a target using IMU measurements, vision-based navigation and an EKF [23]. The filter runs on an on-board computer with an updating rate of 100 Hz. As seen in Figure 1.28, the EKF method, in blue, is compared with raw GPS measurements, in green. The EKF is showed to follow commands, in red, with better accuracy

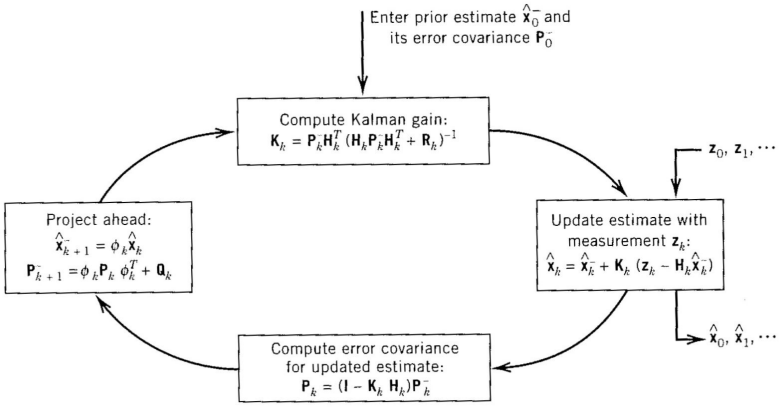


Figure 1.27: KF Algorithm [24]

than raw GPS. Wang et al use two different KF to estimate the position and velocity of an RMAX helicopter UAV during GPS outages [25]. The first KF gives position, velocity and acceleration from IMU measurements while the second KF estimates horizontal velocity and height using a laser range finder, camera and gyros. One set of their results is shown in Figure 1.29. The north, east and down (NED) positioning from the first KF, in dashed lines, is found to closely follow their respective GPS measurements, in solid lines. The data rates for both KF is set to 50 Hz, the maximum rate among the sensors used. Sensor information is sent to a ground computer via radio link for processing.

Fuzzy Logic

Fuzzy Logic (FL), or the logic of fuzzy sets, is a concept analogous to classical logic. Whereas in a classical set an element either belongs or not to it, an element will partially belong to a fuzzy set. Take the three shapes in Figure 1.30 a) for example. In a classical set, they would be classified in one of the

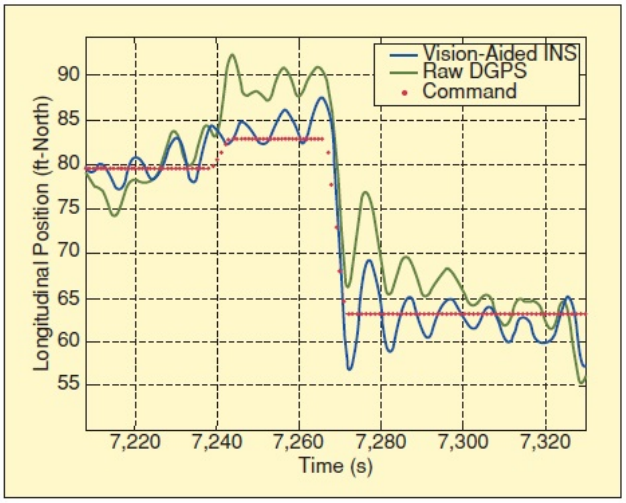


Figure 1.28: Vision based navigation results [23]

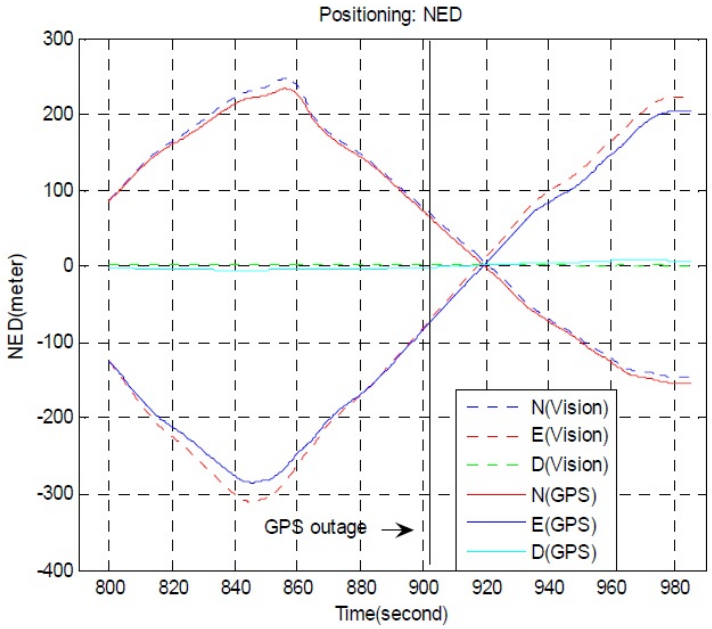


Figure 1.29: Horizontal positioning [25].

three categories, Figure 1.30 b), depending on their height and diameter: disk, cylinder or rod. In a fuzzy set, those shapes can be classified by more than one category. There are three main steps in FL. In the first, fuzzification, the system's inputs and outputs are decomposed in one or more fuzzy sets. The second step, fuzzy language inference, consists of assigning words to inputs, and outcomes to different input combinations. Lastly, the outputs are obtained as meaningful values via defuzzification. [26]

FL is then beneficial in many situations where the outcomes are not well defined. For example, actions to take for a UAV given various heights and temperatures. But, FL still requires to determine rules for the inference step. Also, the outputs can lack precision.

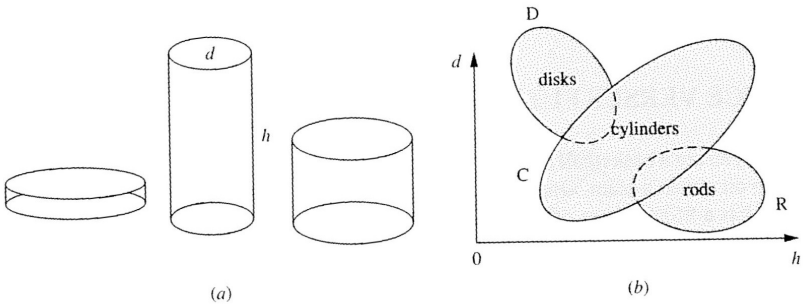


Figure 1.30: FL Example [26]

Fuzzy logic sensor fusion is more often seen in mobile robots than UAVs. An example is a mobile robot by Le which can search for fires, approach them and estimate their intensities [27]. Fuzzy logic is used to fuse temperature and light intensity values and determine the distance to the fire and its size. The linguistic values attributed to the temperature input are found in Figure 1.31.

Experimentation showed that the proposed system works. Lin et al attempt to determine the pose of an autonomous mobile robot using fuzzy adaptive extended information filtering scheme [28]. The system combines time of flight values from two US transmitters fixed to the ceiling and three receivers on the robot, and dead reckoning. Simulation showed that the proposed system can follow the true trajectory as opposed to an EKF-based method, as seen in Figure 1.32. An experiment where the robot is static gave position error less than 1.19" and heading error less than 3^o. Calculations are performed on an on-board computer.

dis(m)\temp	Large Fire°C	Medium Fire°C	Small Fire°C	No Fire°C
Close	high temperature	medium temperature	medium temperature	very low temperature
Medium	medium temperature	medium temperature	low temperature	very low temperature
Far	low temperature	low temperature	very low temperature	very low temperature

Figure 1.31: Language inference for temperature and distance [27]

Artificial Neural Network

An Artificial Neural Networks (ANN) are modeling techniques, inspired by neurons found in the human brain and used to build intelligent programs in that they can learn. An ANN is formed of layers containing threshold elements or artificial neurons, illustrated in Figure 1.33. Such an element can have different inputs x_{ik} with weights w_{ik} associated to them. Its output s_k

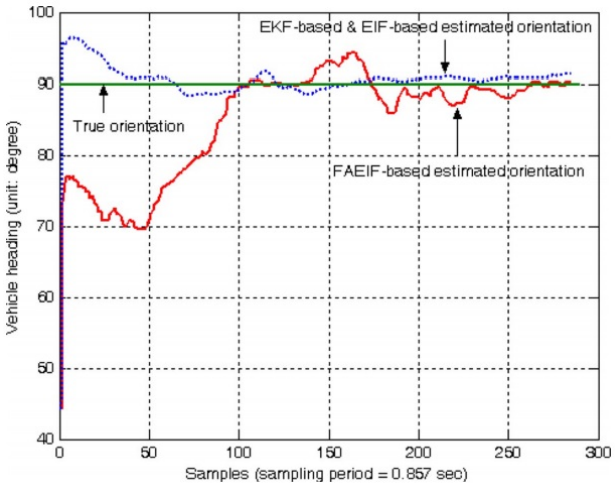


Figure 1.32: Dynamic test orientation [28].

will be triggered and propagated to the next layer of threshold elements only if the sum of input times weight is larger than a threshold value specific to the current element. During the learning process, an ANN will be closed loop and initially have random weights associated to each input. Then, those weights will update by using sets of known inputs and looping until the desired output d_k are obtained or, in other terms, the error ϵ_k is minimized. [29]

ANN is a very flexible modeling method. It can solve problems with complex interactions between variables and works for both nonlinear and linear systems. However, the learning process can be difficult to adjust properly. For instance, if the learning rate is too low, the process will take a while to converge. If it's too high, the weight values can diverge. Also, it can be near impossible to associate physical meaning to the relations between the threshold elements.

ANN are occasionally employed in robot localization. Racz et al estimate

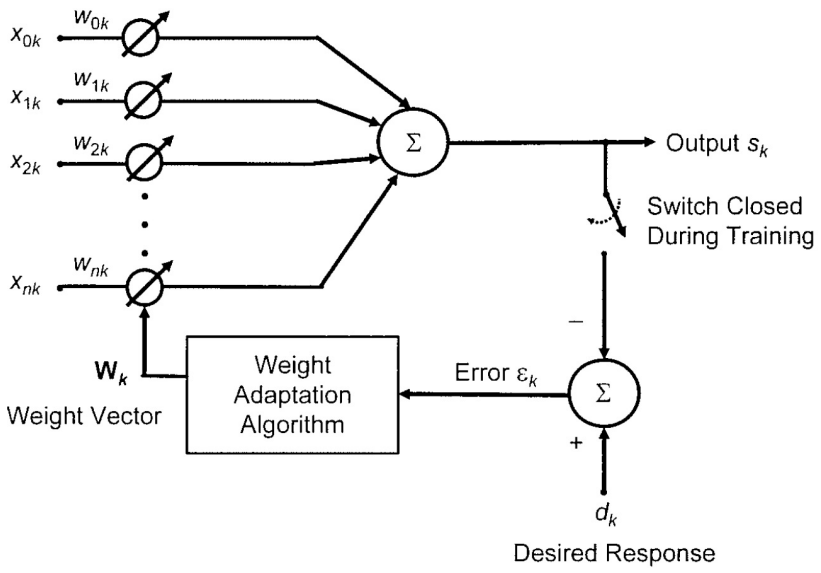


Figure 1.33: Artificial Neuron [29]

the location of a mobile robot on a grid with an ANN, containing fuzzy elements, combining inputs from ultrasonic range finders and dead reckoning [30]. The learning process for the ANN consisted of the robot wandering around the grid area while gathering data. Afterwards, for a thousand tests, about 3% had significant errors in predicting the correct location. Figure 1.34 shows those predictions, where each data point corresponds to a test at a certain distance and the location, north on Figure 1.34 a) and east on Figure 1.34 b), on the grid at which it was associated. From the straight curve representing the real location, It can be seen that the estimations are often in the correct category. Kaygisiz et al develop an ANN to replace GPS for localization with IMU measurements [31]. Testing was performed on a ground vehicle. It gathers data with GPS and IMU to train the ANN while moving on a test track for $6min$. Then, GPS is taken out for 4 min and is replaced by the ANN.

Results show that in ANN mode, the error in the north direction is of 200 m, in Figure 1.35 a), and 10 m in the east direction, in Figure 1.35 b). In both cases, the errors stay constant while the errors from IMU measurements keep accumulating.

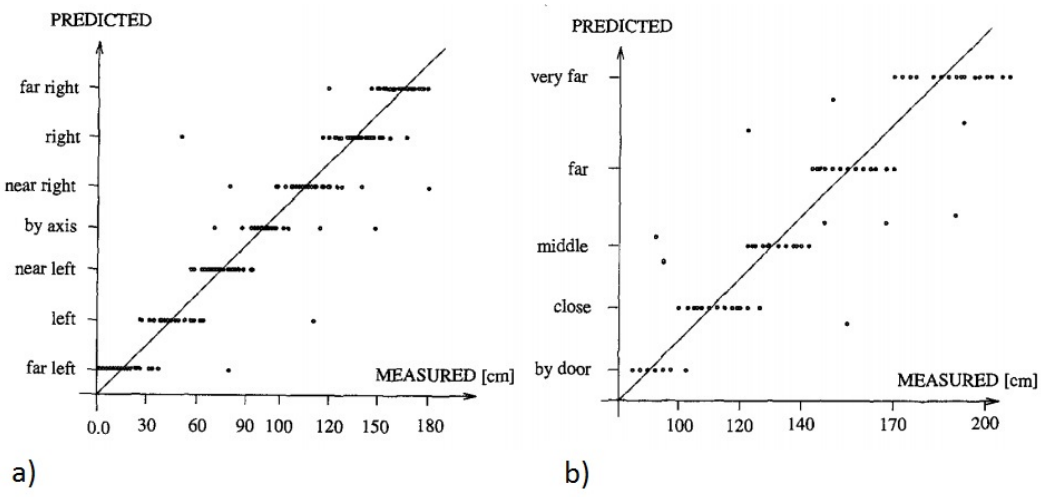


Figure 1.34: Predicted and real location tests [30]

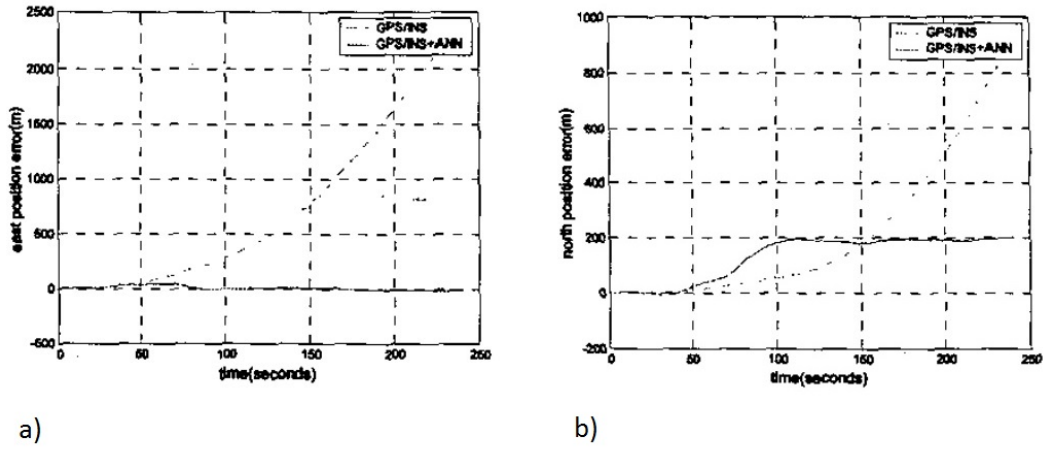


Figure 1.35: Results: a) north direction, b) east direction [31].

Chapter 2

Theory

The relative position and orientation of a blimp UAV will be estimated using the following method: the relative position is estimated using the Perspective-n-Point (PnP) algorithm applied to images obtained from a grounded IR camera seeing a triangular pattern of three IR LEDs fixed to the bow of the UAV. Values obtained from the PnP algorithm are fused with IMU data in an EKF to get a state estimation. This data is then compared with another EKF using GPS measurements instead of image data and serving as theoretical measurements.

This section describes the aforementioned method including the coordinate systems used for this thesis, vision related theory, on-board sensors and Kalman Filter theory.

2.1 Coordinate Systems

Four different coordinate systems are used for the presented method. The image coordinates ${}^{i}(fl, v, w)$ indicate the coordinates of the blimp on the image frames taken by the camera, to which the camera coordinates ${}^{c}(x, y, z)$ is attached to. The body coordinates ${}^{b}(x, y, z)$ are attached to the blimp, represented by the geometric center of the LED pattern for simplification. Every measurement is converted to the fixed world coordinates ${}^{n}(x, y, z)$ which share their origin with the camera coordinates.

2.1.1 World Coordinates

The world coordinates ${}^{n}(x, y, z)$ serve as the reference for all measurements. It is defined as a reference frame fixed to the origin of the camera coordinates, but does not rotate with the camera. The axes are expressed in the north-east-down convention. That is, the ${}^{n}x$ axis always points north, ${}^{n}y$ always points east and ${}^{n}z$ always points downwards.

2.1.2 Body Coordinates

The body coordinate system ${}^{b}(x, y, z)$ used to describe the blimp is shown in Figure 2.1. For each axis ${}^{b}x$, ${}^{b}y$ and ${}^{b}z$, the respective Euler angles ${}^{b}\phi$, ${}^{b}\theta$ and ${}^{b}\psi$ are measured with respect to the world coordinates. These angles are conventionally known, for aircrafts, as roll, pitch and yaw.

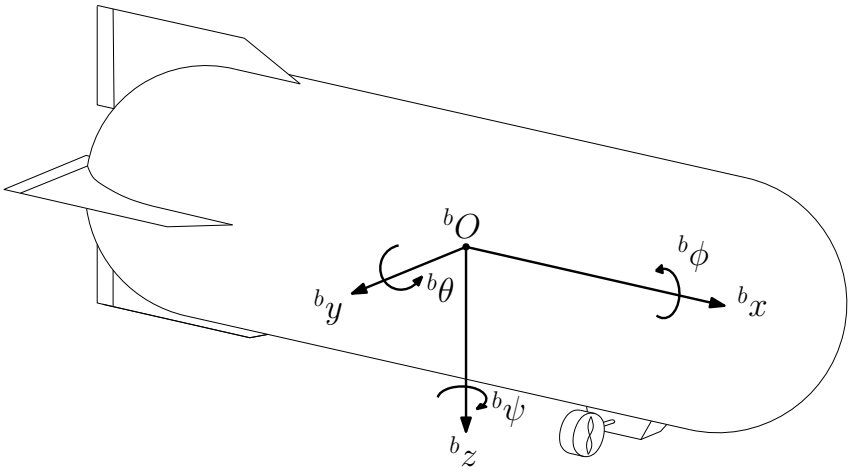


Figure 2.1: Blimp coordinate system [1]

2.1.3 Camera and Image Coordinates

Figure 2.2 shows a camera C viewing a point B in space. The point is projected as point B' on the image frame, represented as a blue rectangle, with height h and width b. The camera coordinates ${}^c\vec{p} = {}^c(x, y, z)$ are fixed to the camera and move with it. The camera angles ${}^c\phi$, ${}^c\theta$ and ${}^c\psi$ are measured with respect to the world axes. The image coordinates ${}^i\vec{p} = (fl, v, w)$, where v is the vertical position and w the horizontal position in pixels, also move with the camera. But, their origin ${}^iO$ is located at a distance equal to the focal length fl of the camera along the ${}^cx$ axis. Angles ${}^i\phi$, ${}^i\theta$ and ${}^i\psi$ obtained from the image coordinates, which will be detailed in Subsection 2.2.4, are measured with respect to the camera frame.

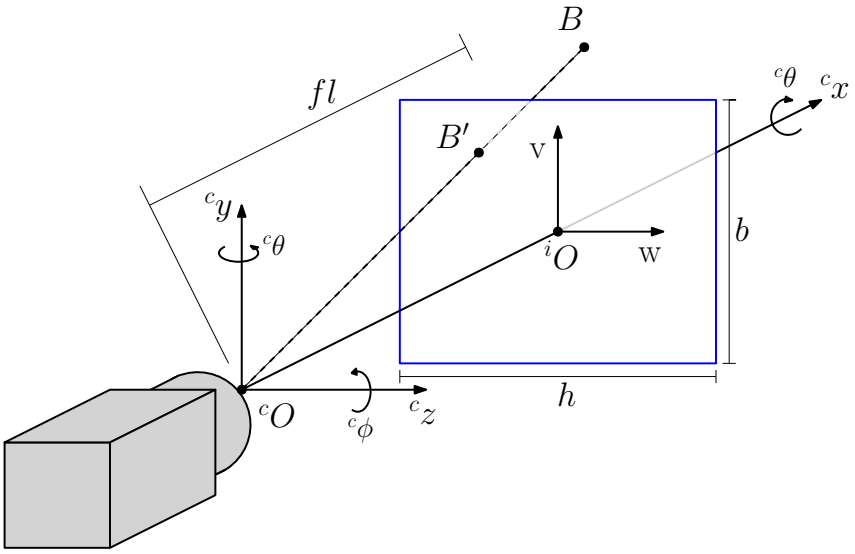


Figure 2.2: Camera and image coordinate system

2.1.4 Conversion Between Coordinate Systems

As measurements are obtained in different coordinate systems, it is necessary to convert them to the world coordinates. The image coordinates can be converted to camera coordinates with a multiplication, deduced by applying trigonometry on Figure 2.3:

$$^{c}\vec{p} = \left(\frac{^{c}x}{fl}\right) {}^{i}\vec{p} \tag{2.1}$$

The coordinates of the image reference frame can be expressed in pixels or meters. The conversion between the two units can be done from the following:

$$\frac{v}{v_{px}} = \frac{h}{h_{px}} \approx \frac{w}{w_{px}} = \frac{b}{b_{px}} = \frac{fl}{fl_{px}} \tag{2.2}$$

More specifically, h and b represent the size of the sensor in meters while the

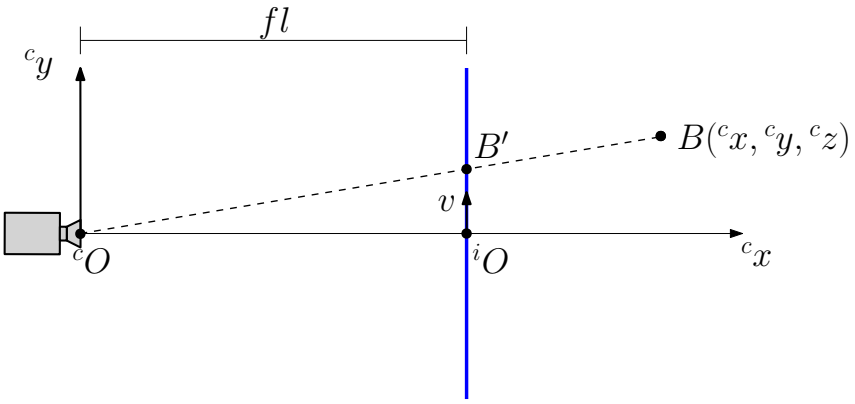


Figure 2.3: Conversion between coordinate systems

px subscript represents their value in pixels. In reality, the height and width ratios are not the same, but close enough to be considered approximately equal.

The camera and world coordinates share their origin, but there is a rotation required to go from one system to the other. It is possible to do so using

rotation matrices, recalling the following relations:

$$
\begin{aligned}
R_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\
R_y(\theta) &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \\
R_z(\psi) &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
R = R_z(\psi)R_y(\theta)R_x(\phi); R^T &= R^{-1}
\end{aligned}
\tag{2.3}
$$

Then, the conversion can be written as:

$$
{}^n\vec{p} = {}^nR_c\,{}^c\vec{p} = R_z({}^c\psi)R_y({}^c\theta)R_x({}^c\phi - 90^o)\,{}^c\vec{p} \tag{2.4}
$$

An additional -90^o in the R_x rotation is used to match the ${}^cz$ and ${}^nz$ axes. The blimp coordinates can be converted to the world coordinates by rotating the coordinates as follow:

$$
{}^n\vec{p} = {}^nR_b\,{}^b\vec{p} = R_z({}^b\psi)R_y({}^b\theta)R_x({}^b\phi)\,{}^b\vec{p} \tag{2.5}
$$

Lastly, a conversion can be established between the angles from the body, camera and image coordinate frames:

$$\begin{bmatrix} {}^{b}\phi \\ {}^{b}\theta \\ {}^{b}\psi \end{bmatrix} = \begin{bmatrix} {}^{c}\phi + {}^{i}\phi - 90^{o} \\ {}^{c}\theta + {}^{i}\theta \\ {}^{c}\psi + {}^{i}\psi \end{bmatrix} \tag{2.6}$$

2.2 Vision

It is possible to extract position and orientation information from the IR camera with the PnP algorithm. To do so, the images of the blimp, represented by a pattern of three IR LEDs, must be processed to be recognized. Propagation of uncertainty of data obtained from the camera will also be described in this section.

2.2.1 PnP problem

The position of the LEDs with respect to the camera can be obtained using a method called the PnP problem, as described in [14] and presented here for the sake of clarity.

From a set of n points projected on a camera frame, knowing their relative position, find the pose of the camera. It was shown in [40] that the PnP problem with $n = 2$ points results in infinite solutions. As such, three points, the minimum number required for a finite number of solutions will be used.

Consider Figure 2.4 with $n = 3$ points ${}^{c}p$ with their projection ${}^{i}p$:

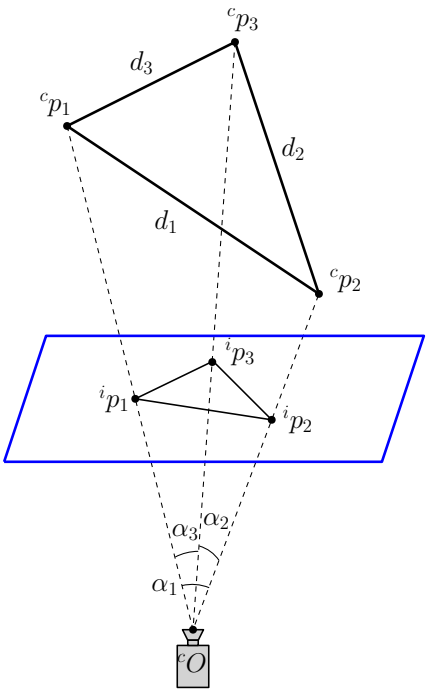


Figure 2.4: P3P problem with image plane

The position of the points ${}^{c}\vec{p}_j$ have a distance of d_j between them and an angle of α_j, for $j = 1, 2, 3$. From these variables, the law of cosines can be applied to find ${}^{c}\vec{p}_j$:

$$\begin{aligned} d_1^2 &= \|{}^{c}\vec{p}_1\|^2 + \|{}^{c}\vec{p}_2\|^2 - 2\|{}^{c}\vec{p}_1\|\|{}^{c}\vec{p}_2\| \cos\alpha_1 \\ d_2^2 &= \|{}^{c}\vec{p}_2\|^2 + \|{}^{c}\vec{p}_3\|^2 - 2\|{}^{c}\vec{p}_2\|\|{}^{c}\vec{p}_3\| \cos\alpha_2 \\ d_3^2 &= \|{}^{c}\vec{p}_3\|^2 + \|{}^{c}\vec{p}_1\|^2 - 2\|{}^{c}\vec{p}_3\|\|{}^{c}\vec{p}_1\| \cos\alpha_3 \end{aligned} \tag{2.7}$$

Recalling the dot product definition:

$$\vec{p_i} \cdot \vec{p_j} = \|\vec{p_i}\|\|\vec{p_j}\| \cos\alpha \tag{2.8}$$

where $i \neq j$, the angles can be removed from the equations in 2.7:

$$\begin{aligned} d_1^2 &= \|{}^c\vec{p}_1\|^2 + \|{}^c\vec{p}_2\|^2 - 2({}^c\vec{p}_1 \cdot {}^c\vec{p}_2) \\ d_2^2 &= \|{}^c\vec{p}_2\|^2 + \|{}^c\vec{p}_3\|^2 - 2({}^c\vec{p}_2 \cdot {}^c\vec{p}_3) \\ d_3^2 &= \|{}^c\vec{p}_3\|^2 + \|{}^c\vec{p}_1\|^2 - 2({}^c\vec{p}_3 \cdot {}^c\vec{p}_1) \end{aligned} \tag{2.9}$$

Since the coordinates of ${}^i\vec{p}_j$ are known, (2.9) can be converted from one coordinate system to the other using (2.1):

$$\begin{aligned} d_1^2 fl^2 &= {}^c x_1^2 \|{}^i\vec{p}_1\|^2 + {}^c x_2^2 \|{}^i\vec{p}_2\|^2 - 2{}^c x_1 {}^c x_2 ({}^i\vec{p}_1 \cdot {}^i\vec{p}_2) \\ d_2^2 fl^2 &= {}^c x_2^2 \|{}^i\vec{p}_2\|^2 + {}^c x_3^2 \|{}^i\vec{p}_3\|^2 - 2{}^c x_2 {}^c x_3 ({}^i\vec{p}_2 \cdot {}^i\vec{p}_3) \\ d_3^2 fl^2 &= {}^c x_3^2 \|{}^i\vec{p}_3\|^2 + {}^c x_1^2 \|{}^i\vec{p}_1\|^2 - 2{}^c x_3 {}^c x_1 ({}^i\vec{p}_3 \cdot {}^i\vec{p}_1) \end{aligned} \tag{2.10}$$

which in turn can be further simplified:

$$\begin{aligned} D_1 &= A_1 {}^c x_1^2 + A_2 {}^c x_2^2 - 2C_{12} {}^c x_1 {}^c x_2 \\ D_2 &= A_2 {}^c x_2^2 + A_3 {}^c x_3^2 - 2C_{23} {}^c x_2 {}^c x_3 \\ D_3 &= A_3 {}^c x_3^2 + A_1 {}^c x_1^2 - 2C_{31} {}^c x_3 {}^c x_1 \end{aligned} \tag{2.11}$$

by posing the following:

$$\begin{aligned} D_j &= d_j^2 fl^2 \\ A_j &= \|{}^i\vec{p_j}\|^2 \\ C_{jk} &= {}^i\vec{p_j} \cdot {}^i\vec{p_k} \\ for \quad j,k &= 1,2,3; j \neq k \end{aligned} \tag{2.12}$$

Finally, the system in (2.11) is used to solve for ${}^cx_j$ and from (2.1), it is possible to determine ${}^cy_j$ and ${}^cz_j$. In Matlab, "vpasolve" is used to find the solution of (2.11) as it is able to solve nonlinear system of equations as well as present every solution to that system.

2.2.2 Image Processing

For the PnP algorithm to be applicable, the blimp must be recognized as a series of points on the image. Since the chosen sensor is an IR camera, those points can be represented by IR LEDs. After a frame is taken with the camera, some operations must be done on it to identify the LEDs. The sequence of operations is similar to the ones found in various UAV projects where image processing is necessary, such as [12] or [37]. Figure 2.5 shows an example of the series of operations used for this thesis:

Figure 2.5 a) shows a gray scale picture with a single IR LED in front of a black cardboard, to help separate it from the daylight behind. The first step is to convert this image to black and white, or binary, as in Figure 2.5 b).

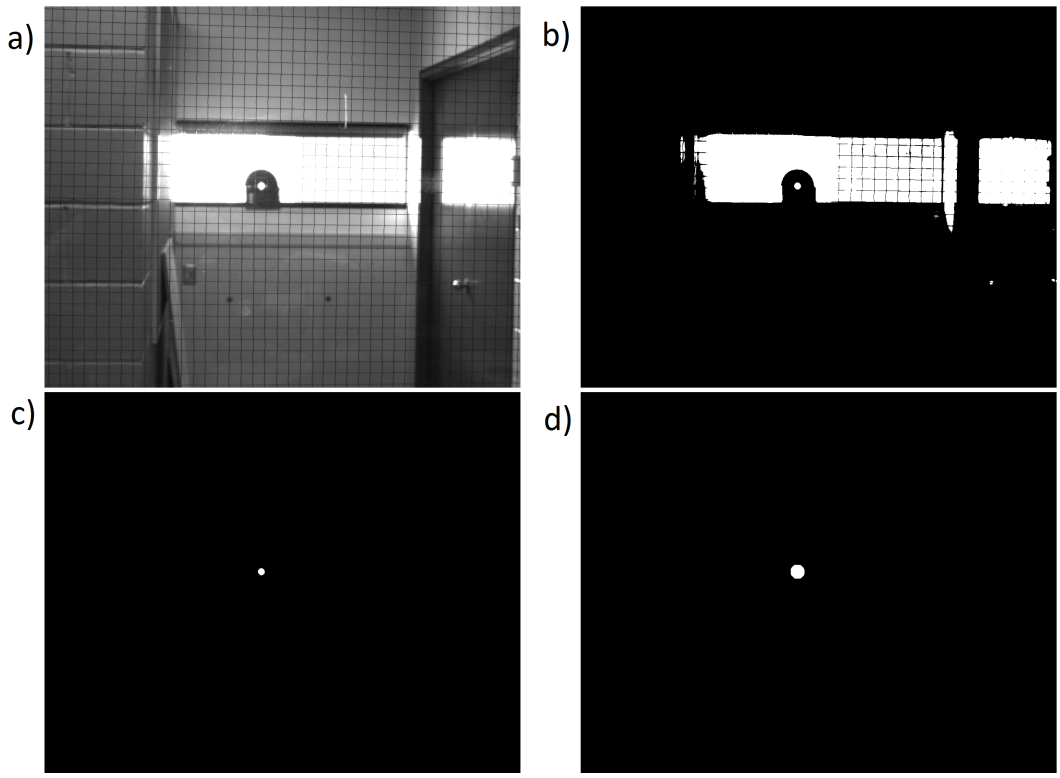


Figure 2.5: a) Original image, b) Binary image, c) Pattern recognition, d) Dilation

This serves as an initial information filter. To do so, the following operation is performed on every pixel in the image:

$$value = \begin{cases} 1, & \text{if } value \geq threshold \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

where *value* is the luminous intensity of a pixel, ranging on a scale of 0, black, to 255, white. In (2.13) , if the *value* is greater than or equal to the *threshold*, the pixel gets a value of 1, or white. In this case, the IR LEDs are very bright when facing the camera, so the *threshold* is taken as 254.

The next step consists of segmenting the image into regions delimited by black or white. Then, it is possible to evaluate each region to search for specific properties or patterns. Since an IR LED's illumination is circular, the regions will be evaluated for roundness with the following formula:

$$roundness = \frac{4\pi A_r}{P_e^2} \tag{2.14}$$

were P_e is the region's perimeter and A_r, its area. *roundness* can take a value between 0 and 1, where 1 is a perfect circle. By choosing a threshold value for *roundness* and applying a similar operation than (2.13), only the circular regions will remain on the image as can be seen in Figure 2.5 c).

Using the "imfindcircles" function in Matlab, the center coordinates and radii of the circles are obtained. Since the function has difficulty identifying

very small circles, a dilation operation is performed on Figure 2.5 c) to increase the size of the white region. The size increase is equal to half the size in pixels of a diamond-shaped structuring element used in the dilation. The result of this operation is seen in Figure 2.5 d). Provided that there is no unwanted circles on the processed image, it is then possible to represent the blimp with IR LEDs.

2.2.3 LED Triangle Pattern

The blimp is to be identified by a pattern of three IR LEDs to meet the minimum of three points required by the PnP algorithm. The pattern is located on the front of the blimp, as seen in Figure 2.6, since it is assumed for this thesis that the blimp will be approaching the camera for landing in the forward position. The pattern is an equilateral triangle shape that covers the widest area possible. Moreover, it is assumed that, from the nature of the blimp, the roll angle will never be greater than $\pm 60^o$. This way, LED 1 will always be on the bottom left of the image, LED 2 on the top and LED 3 on the bottom right.

2.2.4 Orientation Estimation

The PnP algorithm outputs the coordinates of the LEDs in camera coordinates. Those can be used, along with the assumptions and LED denotation from the previous subsection, to estimate the blimp's Euler angles. The dia-

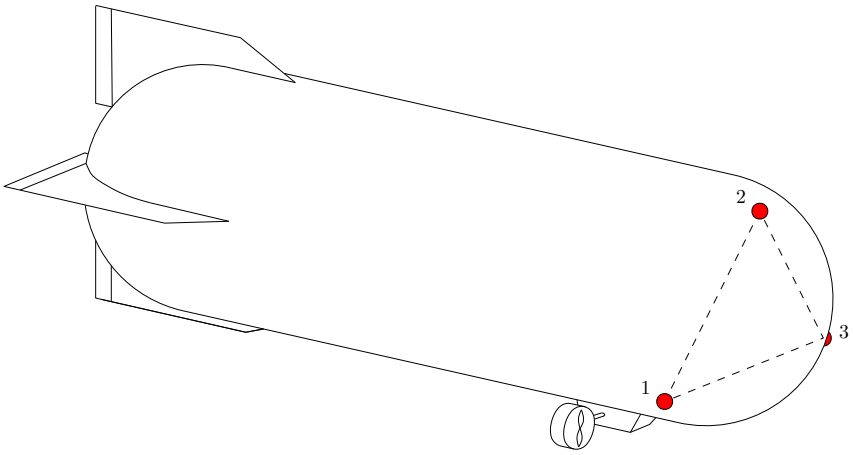


Figure 2.6: LED Triangle Pattern

grams in Figure 2.7 shows how they can be found using basic trigonometry. Since the angles are found from the image, they are denoted with the superscript i.

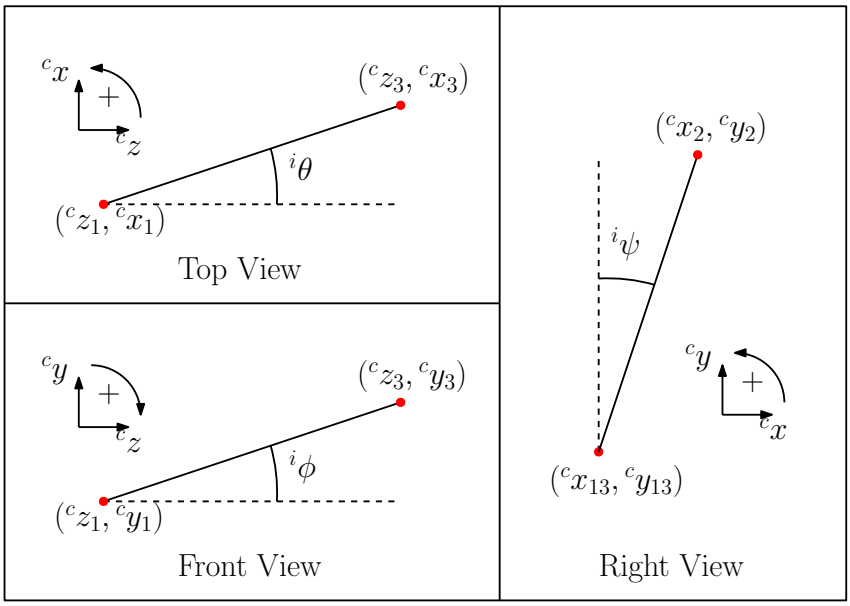


Figure 2.7: Euler angles estimation from LED coordinates: $^i\phi$ (top left), $^i\theta$ (bottom left) and $^i\psi$ (right)

The angles can be calculated with the following equations:

$$
\begin{aligned}
{}^{i}\phi &= -\arctan\left(\frac{{}^{c}(y_3 - y_1)}{{}^{c}(z_3 - z_1)}\right) \\
{}^{i}\theta &= \arctan\left(\frac{{}^{c}(x_3 - x_1)}{{}^{c}(z_3 - z_1}\right) \\
{}^{i}\psi &= -\arctan\left(\frac{{}^{c}(y_2 - y_{13})}{{}^{c}(x_2 - x_{13})}\right) \\
{}^{c}(x, y)_{13} &= \frac{{}^{c}((x, y)_1 + {}^{c}((x, y)_3}{2}
\end{aligned}
\tag{2.15}
$$

It should be noted that from (2.15), the calculated angles can only take a value within $\pm 90^o$. In the event that the blimp is moving away from the camera, ${}^{i}\psi$ would get a value of $\pm[90^o, 180^o]$. Then, since it is impossible to determine whether or not the blimp is approaching the camera from a single image, the heading must be determined using other sensors such as GPS.

2.2.5 Vision Uncertainties

The uncertainties in the position and orientation measurements obtained from vision can be summarized in the following equations:

$$
\begin{bmatrix} \Delta^n x \\ \Delta^n y \\ \Delta^n z \end{bmatrix} = {}^{n}R_c \left(\begin{bmatrix} \Delta^c x_i \\ \Delta^c y_i \\ \Delta^c z_i \end{bmatrix} + \begin{bmatrix} \Delta^c x_c \\ \Delta^c y_c \\ \Delta^c z_c \end{bmatrix} \right) \tag{2.16}
$$

$$\begin{bmatrix} \Delta^n\phi \\ \Delta^n\theta \\ \Delta^n\psi \end{bmatrix} = {}^nR_c \left(\begin{bmatrix} \Delta^i\phi \\ \Delta^i\theta \\ \Delta^i\psi \end{bmatrix} + \begin{bmatrix} \Delta^c\phi \\ \Delta^c\theta \\ \Delta^c\psi \end{bmatrix} \right) \tag{2.17}$$

where Δ indicates the uncertainty. In (2.16), the uncertainty terms come from the PnP algorithm and the camera, subscripts p and c respectively. Equation (2.15) and the camera angles contribute to the uncertainties in (2.17). These four sets of uncertainty terms will be explained in this subsection.

The PnP algorithm outputs ${}^cx$ calculated with the Matlab built-in function "vpasolve". The outputs were found to slightly differ given consecutive two image frames with no visible difference between them. Then, ${}^cx$ data was gathered for various distances and its standard deviation was taken as $\Delta^c x_p$. Details about the procedure and the value of $\Delta^c x_p$ can be found in Section 4.1.

Since ${}^cy$, ${}^cz$, ${}^i\phi$, ${}^i\theta$ and ${}^i\psi$ are deduced from ${}^cx$, v and w, their corresponding uncertainties can be calculated using the propagation of uncertainty equation found in [41]:

$$\sigma_F^2 = \left(\frac{\delta F}{\delta X}\right)^2 \frac{\sigma_X^2}{m} + \left(\frac{\delta F}{\delta Y}\right)^2 \frac{\sigma_Y^2}{m} + 2\left(\frac{\delta F}{\delta X}\right)\left(\frac{\delta F}{\delta Y}\right)\frac{\sigma_{XY}}{m} \tag{2.18}$$

where X nd Y are two random variables, F a function of X and Y, σ_X^2 and σ_Y^2 the variances, σ_{XY} the covariance and m the number of measurements for a single state or time step. For the purpose of this thesis, measurements of a

certain state are taken once. Then, (2.18) can be simplified as:

$$\Delta F = \sqrt{\left(\frac{\delta F}{\delta X}\Delta X\right)^2 + \left(\frac{\delta F}{\delta Y}\Delta Y\right)^2} \tag{2.19}$$

where Δ is the standard deviation of the variables, taken as the uncertainties. This equation can be applied to (2.1) to get $\Delta^c y_i$ and $\Delta^c z_i$:

$$\begin{aligned} \Delta^c y_i &= \frac{^c y}{fl}\sqrt{(v\Delta^c x_i)^2 + (^c x \Delta v)^2} \\ \Delta^c z_i &= \frac{^c z}{fl}\sqrt{(w\Delta^c x_i)^2 + (^c x \Delta w)^2} \end{aligned} \tag{2.20}$$

Then, applying (2.19) to (2.15) yields the following uncertainty values:

$$\begin{aligned} \Delta^i \phi &= \frac{\sqrt{(^c(z_3 - z_1)2\Delta^c y_i)^2 + (^c(y_3 - y_1)2\Delta^c z_i)^2}}{^c(z_3 - z_1)^2 + {}^c(y_3 - y_1)^2} \\ \Delta^i \theta &= \frac{\sqrt{(^c(z_3 - z_1)2\Delta^c x_i)^2 + (^c(x_3 - x_1)2\Delta^c z_i)^2}}{^c(z_3 - z_1)^2 + {}^c(x_3 - x_1)^2} \\ \Delta^i \psi &= \frac{\sqrt{(^c(z_2 - z_{12})\Delta^c y_i)^2 + (^c(y_2 - y_{12})\Delta^c z_i)^2}}{^c(z_2 - z_{12})^2 + {}^c(y_2 - y_{12})^2} \end{aligned} \tag{2.21}$$

The camera angles ${}^c\phi_c$, ${}^c\theta_c$ and ${}^c\psi_c$ can be estimated using an IMU attached to the camera or a set of instruments such as a compass and a protractor. For this thesis, the latter measuring instruments were used to simplify the experimental setup, so the uncertainties are equal to half their smallest measurement. $\Delta^c\phi_c$, $\Delta^c\theta_c$ and $\Delta^c\psi_c$ induce additional uncertainty to the position ${}^c x$, ${}^c y$ and ${}^c z$ as follow:

Figure 2.8 shows the top view of the world coordinates. For a certain position ${}^c x$ measured at an angle ${}^c\psi$, it can be seen that a small rotation of

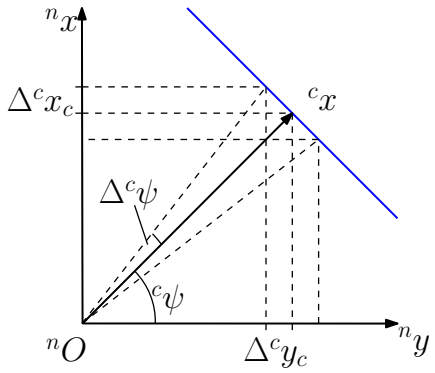


Figure 2.8: Position error induced by $\Delta^c \psi_c$

$\pm\Delta^c\psi$ will deviate the positions $^n x$ and $^n y$ by a small amount. For all three axes, this can be calculated using a rotation matrix and keeping the largest uncertainty:

$$\begin{bmatrix} \Delta^c x_c \\ \Delta^c y_c \\ \Delta^c z_c \end{bmatrix} = \max \begin{cases} |R_z(\Delta^c\psi)R_y(\Delta^c\theta)R_x(\Delta^c\phi)^c\vec{p}| \\ |R_z(-\Delta^c\psi)R_y(-\Delta^c\theta)R_x(-\Delta^c\phi)^c\vec{p}| \end{cases} \tag{2.22}$$

2.3 On-board Sensors

Available equipment on the blimp includes an IMU and a GPS receiver. IMU can help augment the accuracy of vision while GPS can be used as theoretical values for comparison. This section will discuss how to use the readings obtained with theses sensors.

2.3.1 IMU

Raw readings of IMU linear accelerations and angular velocity must be processed to be usable. A bias must first be determined and removed from the readings. Then, a scale factor must be applied to the unbiased readings to convert them to meaningful units. The equations to do so are the following:

$$
\begin{aligned}
{}^{b}\vec{a} &= \frac{{}^{b}\vec{a}_{raw} - \vec{b_a}}{s_a} \\
{}^{b}\vec{\omega} &= \frac{{}^{b}\vec{\omega}_{raw} - \vec{b_\omega}}{s_\omega}
\end{aligned}
\tag{2.23}
$$

where ${}^{b}\vec{a}$ and ${}^{b}\vec{\omega}$ are the output linear acceleration and angular velocity, respectively, in body coordinates, with biases $\vec{b}$ and scale factors s. The scale factors are usually found in the IMU specification sheet. The bias changes on every start-up [42]. A quick method to determine the biases for each axis, before perfoming tests, is to place the axis downwards, to match the gravity vector, and take the mean of a large number of IMU readings.

Temperature can affect the scale factors and biases obtained for low-cost IMU, as stated in [43]. As such, values obtained from the IMU may drift during testing and require more frequent calibration. For this thesis, the effect of temperature was neglected to simplify the manipulations required for the tests.

From the IMU readings, the following equations can be used to determine the blimp's angular velocities and linear accelerations with respect to the world

coordinates [42]:

$$\begin{bmatrix} {}^{n}\dot{\phi} \\ {}^{n}\dot{\theta} \\ {}^{n}\dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin{}^{b}\phi \tan{}^{b}\theta & \cos({}^{b}\phi) \tan{}^{b}\theta \\ 0 & \cos{}^{b}\phi & -\sin{}^{b}\phi \\ 0 & \sin{}^{b}\phi \sec{}^{b}\theta & \cos{}^{b}\phi \sec{}^{b}\theta \end{bmatrix} \begin{bmatrix} {}^{b}\omega_x \\ {}^{b}\omega_y \\ {}^{b}\omega_z \end{bmatrix} \tag{2.24}$$

$$\begin{bmatrix} {}^{n}\ddot{x} \\ {}^{n}\ddot{y} \\ {}^{n}\ddot{z} \end{bmatrix} = {}^{n}R_b \left(\begin{bmatrix} 0 & -{}^{b}\dot{z} & {}^{b}\dot{y} \\ {}^{b}\dot{z} & 0 & -{}^{b}\dot{x} \\ -{}^{b}\dot{y} & {}^{b}\dot{x} & 0 \end{bmatrix} \begin{bmatrix} {}^{b}\omega_x \\ {}^{b}\omega_y \\ {}^{b}\omega_z \end{bmatrix} - \begin{bmatrix} -\sin{}^{b}\theta \\ \cos{}^{b}\theta \sin{}^{b}\phi \\ \cos{}^{b}\theta \cos{}^{b}\phi \end{bmatrix} g + \begin{bmatrix} {}^{b}a_x \\ {}^{b}a_y \\ {}^{b}a_z \end{bmatrix} \right) \tag{2.25}$$

In (2.25), the first term on the right hand side is the acceleration induced by rotation. The second term is the gravity vector removal and the third term, the acceleration readings of the IMU. If the blimp is assumed to be in constant velocity, it is possible to approximate ${}^{b}\phi$ and ${}^{b}\theta$ by comparing the acceleration measurement to the gravity vector:

$${}^{b}\vec{a} = {}^{b}R_n\vec{g} = (R_y({}^{b}\theta)R_x({}^{b}\phi))^T\vec{g} =$$

$$\begin{bmatrix} \cos{}^{b}\theta & 0 & -\sin{}^{b}\theta \\ \sin{}^{b}\phi \sin{}^{b}\theta & \cos{}^{b}\phi & \sin{}^{b}\phi \cos{}^{b}\theta \\ \cos{}^{b}\phi \sin{}^{b}\theta & -\sin{}^{b}\phi & \cos{}^{b}\phi \cos{}^{b}\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = g \begin{bmatrix} -\sin{}^{b}\theta \\ \sin{}^{b}\phi \cos{}^{b}\theta \\ \cos{}^{b}\phi \cos{}^{b}\theta \end{bmatrix} \tag{2.26}$$

where g is the gravity constant. From (2.26), ${}^b\phi$ and ${}^b\theta$ can be found as:

$$\begin{aligned} {}^b\phi &= \arctan\left(\frac{-{}^ba_x}{\sqrt{{}^ba_y^2 + {}^ba_z^2}}\right) \\ {}^b\theta &= \arctan\left(\frac{{}^ba_y}{{}^ba_z}\right) \end{aligned} \tag{2.27}$$

In (2.26), there is no rotation along the z axis as the gravity vector is independant from ${}^b\psi_z$. It is possible to find this angle using an additional directional vector. A magnetometer, which gives the direction of magnetic north, could give such a value.

2.3.2 GPS

Typical GPS coordinates can be represented in cartesian coordinates x, y and z or geodetic coordinates longitude φ, latitude λ and altitude h, as seen in Figure 2.9. Longitude is an angle corresponding to the east and west directions, with the Greenwich meridian at at 0^o. Longitude can take values of $\pm 180^o$, positive being east. Latitude is an angle corresponding to the north and south directions, where the equator is at 0^o. Latitude can take values of $\pm 90^o$, positive being north.

If the reference frame is located at the Earth's center of mass and rotates with the Earth, it is known as Earth Centered Earth-Fixed (ECEF). A widely used standard to express GPS coordinates is the World Geodetic System 1984 (WGS-84). The reference frame used is ECEF and expressed in geodetic pa-

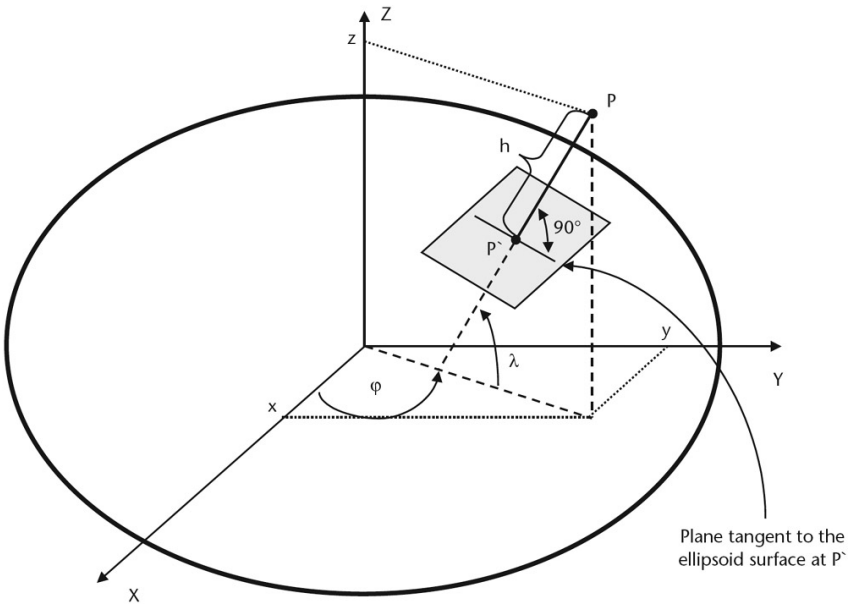


Figure 2.9: Cartesian and geodetic coordinates [32]

rameters. Moreover, since the Earth is not a perfect sphere, it is modeled as an ellipsoid.

To make use of latitude and longitude, they must be converted to units of length. This can be done by knowing the coordinates of a reference point and linearizing the length of one degree on the Earth's surface. With the WGS84 standard, the conversion ratio of a degree to meters varies with specific longitude and latitude. The equations below are used to linearize the conversion for the WGS84 standard [44] :

$$s_\lambda(m) = (111132.92 - 559.82cos(2\lambda) + 1.175cos(4\lambda) - 0.0023cos(6\lambda))m$$

$$s_\varphi(m) = (111412.84cos(\varphi) - 93.5cos(3\varphi) + 0.118cos(5\varphi)) \tag{2.28}$$

where d is the conversion factor. The blimp used for this thesis does not travel very far during tests. Hence, the conversion factors can be assumed constant.

In [45], it is stated that the heading angle ${}^{n}\psi$ can be approximated from GPS coordinates in the absence of a magnetometer using velocity. If the velocity measurement is not detailed enough, the heading angle can be calculated from two consecutive position coordinates:

$$ {}^{b}\psi \approx \arctan\left(\frac{s_{\varphi}(m)(\varphi_2 - \varphi_1)}{s_{\lambda}(m)(\lambda_2 - \lambda_1)}\right) = \arctan\left(\frac{{}^{n}y_2 - {}^{n}y_1}{{}^{n}x_2 - {}^{n}x_1}\right) \tag{2.29} $$

The uncertainty associated to (2.29) is obtained similarly to (2.21):

$$ \Delta^{b}\psi = \frac{\sqrt{(({}^{n}y_2 - {}^{n}y_1)2\Delta^{n}x)^2 + (({}^{n}x_2 - {}^{n}x_1)2\Delta^{n}y)^2}}{({}^{n}y_2 - {}^{n}y_1)^2 + ({}^{n}x_2 - {}^{n}x_1)^2} \tag{2.30} $$

Since (2.29) and (2.21) rely on the differences between the coordinates of two measurements, they can not be used if the blimp is not moving. In such a case, two GPS units located at the front and back of the blimp could be employed, replacing the two consecutive position coordinates in (2.29) and (2.21).

It was mentioned in Subsection 2.2.4 that GPS could be used to correct

${}^{i}\psi$. To do so, the following conditions are used:

$$approach = \begin{cases} true, {}^{b}\psi \leq [-{}^{c}\psi - 90^{o}, -{}^{c}\psi + 90^{o}] \\ \\ false, otherwise \end{cases}$$
$$(2.31)$$
$${}^{i}\psi = \begin{cases} {}^{c}\psi - 180^{o}, (approach = true) \wedge ({}^{b}\psi < 0) \\ \\ {}^{c}\psi + 180^{0}, (approach = true) \wedge ({}^{b}\psi \geq 0) \end{cases}$$

Lastly, there is the possibility of the GPS measuring ground velocity ${}^{n}v_{h}$. In this case, the velocities along the x and y axes can be approximated as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \approx {}^{n}v_{h} \begin{bmatrix} \cos {}^{b}\psi \\ \sin {}^{b}\psi \end{bmatrix} \tag{2.32}$$

2.4 Data Fusion

The purpose of this section is to explain how the measurements and dynamic equations can be fused to obtain a better state estimate. As mentioned at the beginning of Chapter 2, the Extended Kalman Filter (EKF) was selected to estimate the blimp states from the various sensors. The Kalman Filter (KF) will first be presented to cover the basics, then the EKF will be presented, as in [24] and [46]. Afterwards, a method to discretize a continuous-time model will be described followed by the matrix initialization for the UAV application.

2.4.1 Kalman Filter

The KF is used to estimate the state vector X_k, containing the system's variables of interest, at a time step k using a series of measurements Z_k. From the KF, X_k is described with the following linear model:

$$X_k = F_k X_{k-1} + G_k U_k + q_{k-1} \tag{2.33}$$

where F_k is a state transition matrix relating X_{k-1} to X_k, U_k is a vector containing control inputs to the system, G_k is a matrix relating U_k to X_k, and q_{k-1} is the noise associated to X_{k-1}. The model has an associated measurement vector Z_k given by:

$$Z_k = H_k X_k + r_k \tag{2.34}$$

where H_k is a matrix relating X_k to Z_k, and r_k the zero mean measurement noise.

The choice of matrices F_k, G_k and H_k are relevant to the model. As an example, consider a particle whose state, one dimensional position and velocity, can be described by the kinematic equations:

$$p_k = p_{k-1} + \dot{p}_{k-1}\Delta t + \frac{\ddot{p}_k \Delta t^2}{2}$$

$$\dot{p}_k = \dot{p}_{k-1} + \ddot{p}_k \Delta t$$

where p is the particle position and Δt, the time interval. In matrix form, the

kinematic equations can be written as:

$$\begin{bmatrix} p \\ \dot{p} \end{bmatrix}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \ddot{p}_k$$

Assuming no noise in this model, each term can be associated to a variable in (2.33), where:

$$X_k = \begin{bmatrix} p \\ \dot{p} \end{bmatrix}_k ; F_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} ; G_k = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} ; U_k = \ddot{p}_k$$

Suppose, for this example, that *pos* and *vel* are the direct measurements of p and $\dot{p}$. In matrix form, the measurements can be written as:

$$\begin{bmatrix} pos \\ vel \end{bmatrix}_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \end{bmatrix}_k$$

If there are no noise in the measurements, the equivalence with (2.34) are:

$$Z_k = \begin{bmatrix} pos \\ vel \end{bmatrix}_k ; H_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

For this thesis, the states of the UAV to be estimated are the positions, velocities and angles in world coordinates:

$$X_k = \begin{bmatrix} {}^n x & {}^n y & {}^n z & {}^n \dot{x} & {}^n \dot{y} & {}^n \dot{z} & {}^b \phi & {}^b \theta & {}^b \psi \end{bmatrix}_k^T \tag{2.35}$$

As for the measurements, it was decided that they were taken from the camera or the GPS, since both can measure distance. The camera gives the position of the IR LED pattern center of gravity and the angles with respect to the camera, both described in Section 2.2. Hence:

$$Z_{k,cam} = \begin{bmatrix} {}^{c}x & {}^{c}y & {}^{c}z & {}^{i}\phi & {}^{i}\theta & {}^{i}\psi \end{bmatrix}_{cam}^{T} \tag{2.36}$$

The GPS used for this thesis gives position and ground velocity ${}^{n}v_h$ given in (2.32). The angle ${}^{b}\psi$ can also be estimated from (2.29). The corresponding Z_k then becomes:

$$Z_{k,GPS} = \begin{bmatrix} {}^{n}x & {}^{n}y & {}^{n}z & {}^{n}\dot{x} & {}^{n}\dot{y} & {}^{b}\psi \end{bmatrix}_{GPS}^{T} \tag{2.37}$$

The noises q_{k-1} and r_k in equations (2.33) and (2.34) can take the form of error in the state or noise in the measurements. The noises have zero mean, but have covariance matrices Q_{k-1} and R_k. The diagonal elements in these matrices are variances while the non-diagonal elements are covariances. The

covariance matrices are obtained with the following:

$$E[q_i q_j^T] = \begin{cases} Q_{k-1}, i = j \\ \\ \mathbf{0}, i \neq j \end{cases}$$
$$E[r_i r_j^T] = \begin{cases} R_k, i = j \\ \\ \mathbf{0}, i \neq j \end{cases} \tag{2.38}$$
$$E[q_i r_j^T] = \mathbf{0}, \forall i, j$$

where i and j represent different time steps, and E is the expected value operator. From this equation, it can be seen that the noises are independent in time as well as having no cross-correlation.

If X_k is the real state, $\hat{X}_k^-$ is defined as a prior, or initial, state estimate of time step k and $\hat{X}_k$, the posterior, or updated, state estimate at k. Then, the prior estimation error e_k^- and posterior estimation error e_k can be introduced:

$$\begin{aligned} e_k^- &= X_k - \hat{X}_k^- \\ e_k &= X_k - \hat{X}_k \end{aligned} \tag{2.39}$$

with their respective error covariance matrices P_k^- and P_k:

$$\begin{aligned} P_k^- &= E[e_k^- e_k^{-T}] \\ P_k &= E[e_k e_k^T] \end{aligned} \tag{2.40}$$

The idea is to write $\hat{X}_k$ as a linear combination of a known $\hat{X}_k^-$, and a weighted difference K_k between Z_k and an estimation of Z_k given by $H_k\hat{X}_k^-$:

$$\hat{X}_k = \hat{X}_k^- + K_k(Z_k - H_k\hat{X}_k^-) \tag{2.41}$$

Combining (2.39), (2.40) and (2.41), P_k can be rewritten as:

$$P_k = (I - K_k H_k)P_k^- \tag{2.42}$$

where I is the identity matrix. By minimizing (2.42), an equation for K_k, named the Kalman gain, can be obtained:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1} \tag{2.43}$$

Equations (2.41) to (2.43) served to update the prior state estimate $\hat{X}_k^-$ to $\hat{X}_k$ at a precise time step. Then, more equations must be established to update the state estimate from one time step to the next. This can be done from (2.33):

$$\hat{X}_k^- = F_k\hat{X}_{k-1} + G_k U_k \tag{2.44}$$

Between this equation and (2.33), the noise term q_{k-1} is omitted due to (2.38) and for having zero mean [24]. The covariance P_{k-1} can also be updated using

(2.39), (2.40) and (2.44):

$$P_k^- = F_k P_{k-1} F_k^T + Q_{k-1} \tag{2.45}$$

Figure 2.10 represents the KF algorithm. From $\hat{X}_0^-$ and P_0, time update is performed. That is, $\hat{X}_k^-$ and P_k^- are predicted. Then, K_k is computed, X_k^- and P_k^- are updated to X_k and P_k and the process is repeated for the next time step.

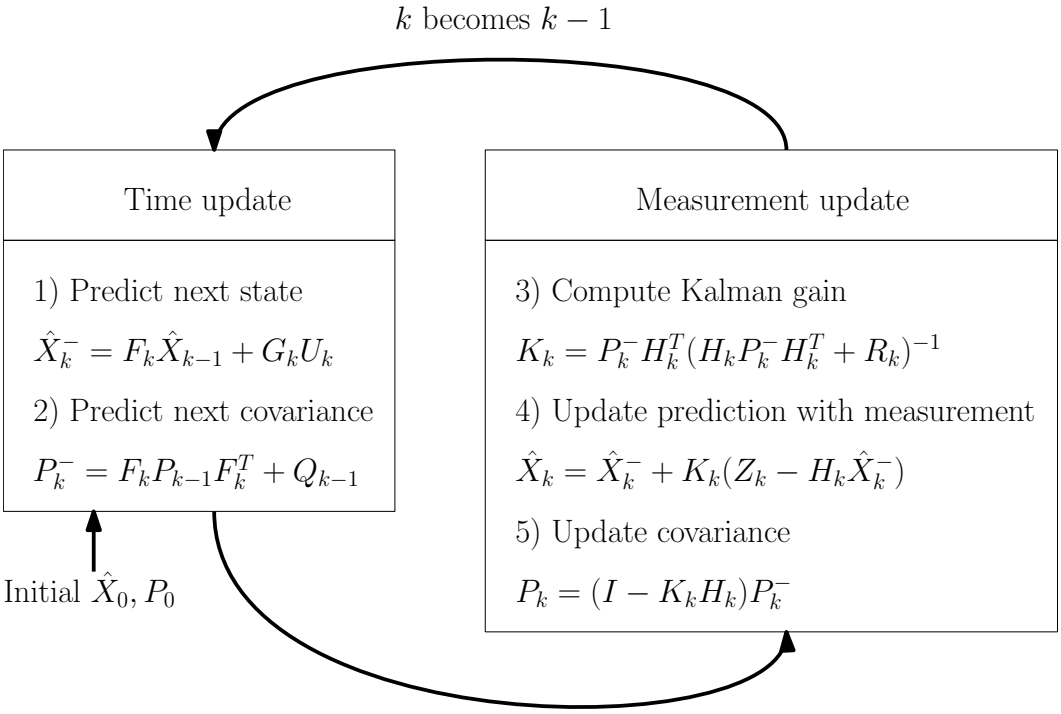


Figure 2.10: Kalman Filter Algorithm

2.4.2 Extended Kalman Filter

The EKF was developed for systems presenting state nonlinearities. Consider X_k and Z_k, described in the more general form of (2.33) and (2.34):

$$X_k = f(X_{k-1}, U_k) + q_{k-1} \tag{2.46}$$

$$Z_k = h(X_k) + r_k \tag{2.47}$$

where f and h are nonlinear functions. Then, similarly to the KF, the EKF equations can be written as:

$$\hat{X}_k = \hat{X}_k^- + K_k(Z_k - h(\hat{X}_k^-)) \tag{2.48}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{2.49}$$

$$P_k = (I - K_k H_k) P_k^- \tag{2.50}$$

$$\hat{X}_k^- = f(\hat{X}_{k-1}, U_k) \tag{2.51}$$

$$P_k^- = F_k P_{k-1} F_k^T + Q_{k-1} \tag{2.52}$$

Equations (2.48) to (2.52) work similarly to their KF counterpart (2.41) to (2.45) with the exception of the state transition matrix F_k and state measurement matrix H_k. In the EKF, Jacobian matrices are used to linearize the functions f and h with respect to X_k. These matrices are computed at each

time step using the following partial derivation:

$$F_k = \frac{\delta f(X_k, U_k, 0)}{\delta X} \tag{2.53}$$

$$H_k = \frac{\delta h(X_k, 0)}{\delta X} \tag{2.54}$$

with the noise terms set to zero. Figure 2.11 shows the EKF algorithm flowchart.

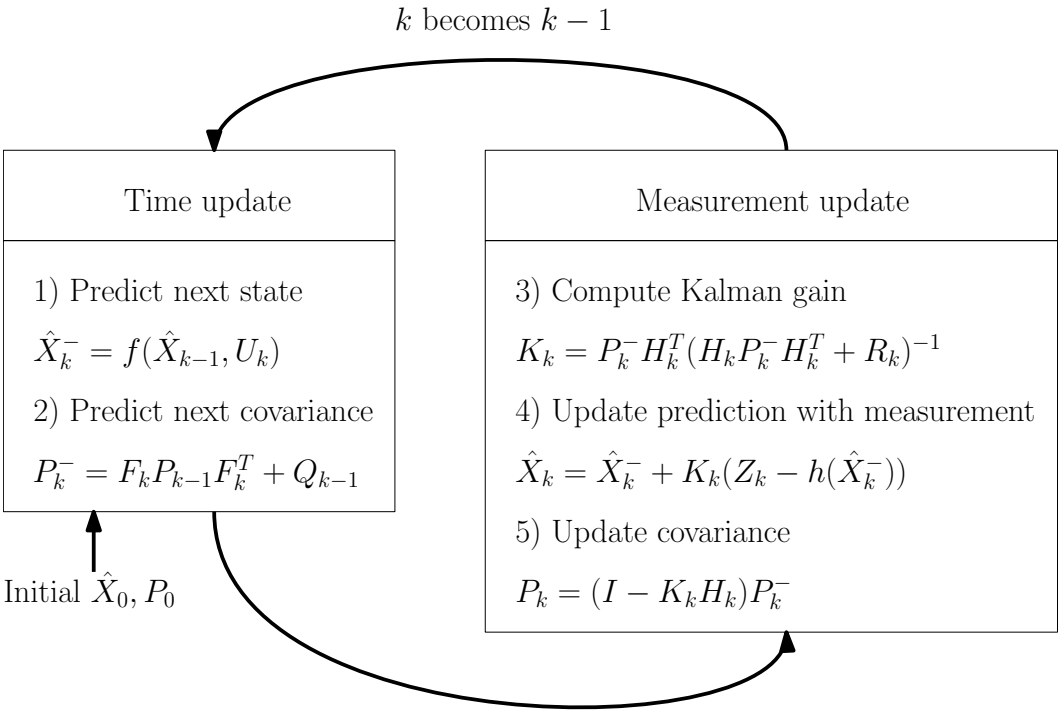


Figure 2.11: Extended Kalman Filter Algorithm

2.4.3 Continuous to Discrete

In both the KF and EKF algorithms, X_k is a discrete-time model of the state. If the system is obtained in continuous-time form, it must be discretized to be

used with the KF or EKF equations. Consider the following state equation in continuous-time form:

$$\dot{X}(t) = F(t)X(t) + q(t) \tag{2.55}$$

where $\dot{X}(t)$ is the time derivative of the state $X(t)$, $F(t)$ a Jacobian matrix obtained from (2.53) and $q(t)$ the continuous-time process noise with covariance $Q(t)$. The continuous-time measurement equation is not presented as it is equivalent to its discrete-time form.

In [24], a method to discretize such a model is introduced, posing matrices A and B:

$$A = \begin{bmatrix} -F(t) & Q(t) \\ 0 & F(t)^T \end{bmatrix} \Delta t \tag{2.56}$$

$$B = e^A = \begin{bmatrix} ... & F_k^{-1} Q_k \\ 0 & F_k^T \end{bmatrix} \tag{2.57}$$

The discrete matrices F_k and Q_k can then be extracted from (2.57). The upper-left elements of B are omitted for not being necessary to find F_k and Q_k.

2.4.4 Matrix Selection

This section describes the selection process for the matrices used in this thesis for data fusion. In Subsection 2.3.1, (2.24) and (2.25) are the time derivatives of the angles and velocities of X_k in (2.35). These relations can be used to write

the state model in continuous-time form of (2.55) and the resulting model is given by:

$$\begin{bmatrix} {}^{n}\dot{x} \\ {}^{n}\dot{y} \\ {}^{n}\dot{z} \\ {}^{n}\ddot{x} \\ {}^{n}\ddot{y} \\ {}^{n}\ddot{z} \\ {}^{b}\dot{\phi} \\ {}^{b}\dot{\theta} \\ {}^{b}\dot{\psi} \end{bmatrix} = \begin{bmatrix} {}^{n}\dot{x} \\ {}^{n}\dot{y} \\ {}^{n}\dot{z} \\ \begin{bmatrix} 0 & -{}^{n}\dot{z} & {}^{n}\dot{y} \\ {}^{n}\dot{z} & 0 & -{}^{n}\dot{x} \\ -{}^{n}\dot{y} & {}^{n}\dot{x} & 0 \end{bmatrix} \begin{bmatrix} {}^{b}\omega_x \\ {}^{b}\omega_y \\ {}^{b}\omega_z \end{bmatrix} - \begin{bmatrix} -\sin{}^{b}\theta \\ \cos{}^{b}\theta \sin{}^{b}\phi \\ \cos{}^{b}\theta \cos{}^{b}\phi \end{bmatrix} g + {}^{n}R_b \begin{bmatrix} {}^{b}a_x \\ {}^{b}a_y \\ {}^{b}a_z \end{bmatrix} \\ \begin{bmatrix} 1 & \sin{}^{b}\phi \tan{}^{b}\theta & \cos({}^{b}\phi) \tan{}^{b}\theta \\ 0 & \cos{}^{b}\phi & -\sin{}^{b}\phi \\ 0 & \sin{}^{b}\phi \sec{}^{b}\theta & \cos{}^{b}\phi \sec{}^{b}\theta \end{bmatrix} \begin{bmatrix} {}^{b}\omega_x \\ {}^{b}\omega_y \\ {}^{b}\omega_z \end{bmatrix} \end{bmatrix} + q(t) \tag{2.58}$$

$F(t)$ can be obtained from the Jacobian of (2.58), similar to (2.53).

As for $q(t)$ in (2.58), its definition is not important. However, its covariance matrix $Q(t)$ is needed. There are no direct methods to set up $Q(t)$, so it is defined intuitively. Since the measurements from the IMU are used in (2.58), in the form of ${}^{b}a$ and ${}^{b}\omega$, their errors can be used in $Q(t)$. As this matrix is the noise covariance matrix of $X(t)$, it is an $m * m$ matrix where m is the length of $X(t)$. If one assumes independence between each axes, and between angles

and the other states, the following $Q(t)$ can be posed:

$$Q(t) = \begin{bmatrix} e_{a_x}^2 \frac{\Delta t^4}{4} & 0 & 0 & e_{a_x}^2 \frac{\Delta t^3}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & e_{a_y}^2 \frac{\Delta t^4}{4} & 0 & 0 & e_{a_y}^2 \frac{\Delta t^3}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & e_{a_z}^2 \frac{\Delta t^4}{4} & 0 & 0 & e_{a_z}^2 \frac{\Delta t^3}{2} & 0 & 0 & 0 \\ e_{a_x}^2 \frac{\Delta t^3}{2} & 0 & 0 & e_{a_x}^2 \Delta t^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & e_{a_y}^2 \frac{\Delta t^3}{2} & 0 & 0 & e_{a_y}^2 \Delta t^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & e_{a_z}^2 \frac{\Delta t^3}{2} & 0 & 0 & e_{a_z}^2 \Delta t^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e_{\omega_x}^2 \Delta t^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{\omega_y}^2 \Delta t^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{\omega_z}^2 \Delta t^2 \end{bmatrix} \tag{2.59}$$

where e_a and e_ω are the errors of the accelerometer and gyroscope. The elements of the diagonal are the state errors squared multiplied by a power of Δt to obtain the correct units. The remaining non-zero elements are obtained similarly. Taking the 4^{th} element of the 1^{st} column as an example, e_a is multiplied by $\frac{\Delta t^2}{2}$ to match the units of ${}^n x$ and Δt for ${}^n \dot{x}$. Multiplying the two results by each other yields $e_{a_x}^2 \frac{\Delta t^3}{2}$.

F_k and Q_k are then obtained by discretizing the above model as described in Subsection 2.4.3.

When the measurements are taken from the camera, they can be written in the form of (2.47), using (2.4) and (2.6) to convert between the coordinate

systems:

$$\begin{bmatrix} {}^{c}x \\ {}^{c}y \\ {}^{c}z \\ {}^{i}\phi \\ {}^{i}\theta \\ {}^{i}\psi \end{bmatrix}_{cam} = \begin{bmatrix} {}^{c}R_{n} \begin{bmatrix} {}^{n}x \\ {}^{n}y \\ {}^{n}z \end{bmatrix} \\ {}^{b}\phi - {}^{c}\phi + 90^{o} \\ {}^{b}\theta - {}^{c}\theta \\ {}^{b}\psi - {}^{c}\psi \end{bmatrix} + r_{k,cam} \tag{2.60}$$

The corresponding $H_{k,cam}$ can be found through (2.54) to get the following result:

$$H_{k,cam} = \begin{bmatrix} {}^{c}R_{n}I_{3*3} & \mathbf{0}_{3*6} \\ \mathbf{0}_{3*6} & I_{3*3} \end{bmatrix} \tag{2.61}$$

As for $r_{k,cam}$ in (2.60), its covariance $R_{k,cam}$ must be defined. If all measurements are assumed independent from each other, then $R_{k,cam}$ becomes a diagonal matrix where each term on the diagonal corresponds to the uncertainties, defined in (2.17) and (2.16), of the measurements squared:

$$R_{k,cam} = diag\left(\begin{bmatrix} \Delta^{n}x^{2} & \Delta^{n}y^{2} & \Delta^{n}z^{2} & \Delta^{n}\phi^{2} & \Delta^{n}\theta^{2} & \Delta^{n}\psi^{2} \end{bmatrix}\right) \tag{2.62}$$

When the GPS is used for the measurements, from (2.29) and (2.32), its

corresponding Z_k becomes:

$$\begin{bmatrix} {}^{n}x \\ {}^{n}y \\ {}^{n}z \\ {}^{n}\dot{x} \\ {}^{n}\dot{y} \\ {}^{b}\psi \end{bmatrix}_{GPS} = \begin{bmatrix} {}^{n}x \\ {}^{n}y \\ {}^{n}z \\ {}^{n}v_h \begin{bmatrix} \cos {}^{b}\psi \\ \sin {}^{b}\psi \end{bmatrix} \\ {}^{b}\psi \end{bmatrix} + r_{k,GPS} \tag{2.63}$$

The associated $H_{k,GPS}$ is:

$$H_{k,cam} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos {}^{b}\psi & 0 & 0 & 0 & 0 & -{}^{n}v_h \sin {}^{b}\psi \\ 0 & 0 & 0 & \sin {}^{b}\psi & 0 & 0 & 0 & 0 & {}^{n}v_h \cos {}^{b}\psi \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.64}$$

and $R_{k,GPS}$:

$$R_{k,GPS} = diag\left(\begin{bmatrix} e_{^{n}x}^2 & e_{^{n}y}^2 & e_{^{n}z}^2 & e_{^{n}\dot{x}} & e_{^{n}\dot{y}} & \Delta^{b}\psi \end{bmatrix}\right) \tag{2.65}$$

where $e_{^{n}x}$, $e_{^{n}y}$, $e_{^{n}z}$, $e_{^{n}\dot{x}}$ and $e_{^{n}\dot{y}}$ are the position and velocity errors of the GPS. $\Delta^{b}\psi$ the uncertainty of ${}^{b}\psi$ found with (2.30).

The last matrix that requires definition is the covariance P_k. Since it is recalculated at each time step in the EKF algorithm, only P_0 must be determined beforehand. Assuming that the initial states are independent from one another, P_0 can be defined as a diagonal matrix where each element is the error squared of its respective known initial state:

$$P_0 = diag\left(\begin{bmatrix} e_{n_x}^2 & e_{n_y}^2 & e_{n_z}^2 & e_{n_{\dot{x}}}^2 & e_{n_{\dot{y}}}^2 & e_{n_{\dot{z}}}^2 & e_{n_\phi}^2 & e_{n_\theta}^2 & e_{n_\psi}^2 \end{bmatrix}\right) \tag{2.66}$$

Chapter 3

Experimental Platform

This chapter explains the equipment, MatLab program and procedure used to perform the tests.

3.1 Equipment

The UAV used for this thesis is shown in Figure 3.1. The envelope is 2 m long and the gondola can move forward or backward on a rail. The gondola controller is a Nanowii using an ATmega32u4 microcontroller and integrated MPU-6050 IMU with 1 kHz accelerometer and 8 kHz gyroscope. Other hardware include a wt41 bluegiga Bluetooth module with a range of up to 800 m, a NEO-6 GPS module with 1 Hz refresh rate and 2.5 m horizontal accuracy and three motors: two for the propellers and one for the movement on the rail. The blimp is powered using a rechargeable Li-Po 7.4 V battery. The first and second versions of the gondola can be seen on Figure 3.2.

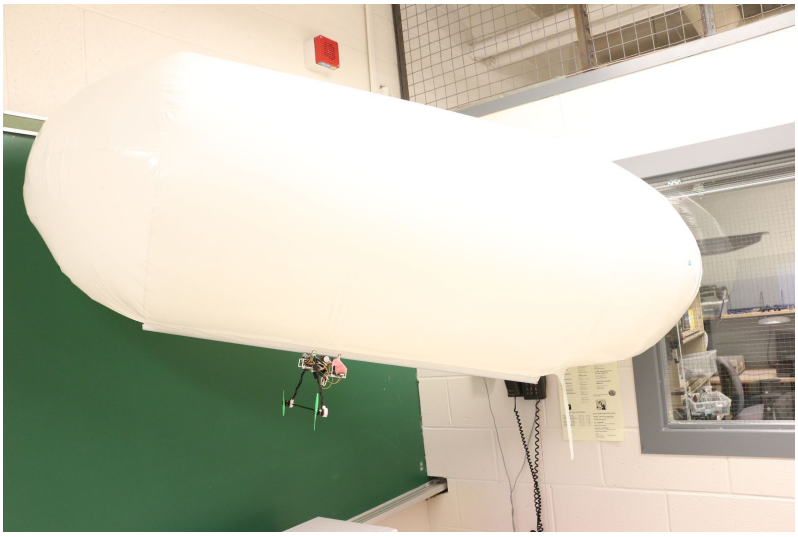


Figure 3.1: Blimp envelope and gondola.

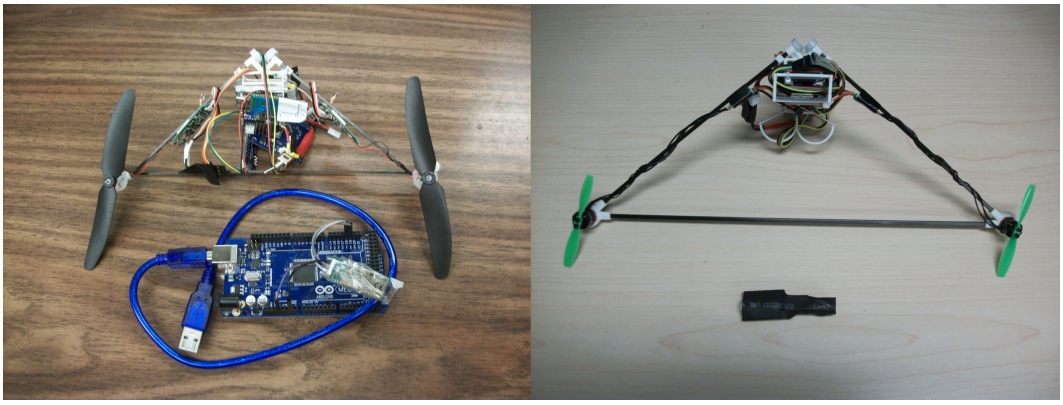


Figure 3.2: First (left) and second (right) versions of the blimp gondola and Bluetooth receiver.

The ground station is composed of a computer, a wt41 bluegiga Bluetooth module and an IR camera. The computer is a Dell desktop computer with 8 GB RAM and Windows 7 OS running MatLab 2015b. The IR camera is a Basler ace acA1300-60gmNIR with gigE interface, $1282 * 1026$ pixel resolution and maximum frame rate of 60 Hz. It is equipped with a Kowa LM12JCM lens with fixed focal length of 12 mm and 1.4 f-number. The aperture and focus rings of the lens are manual.

The equipment used to evaluate the proposed method is composed of a black panel with three embedded IR LEDs, forming a 60 cm sided equilateral triangle. The assembly is powered by four AA bateries. The IR LEDs have an emission angle of 6^o and 1100 mW/sr radiant intensity at 100 mA. The black panel was used for its simplicity and ensuring contrast between the LED illumination and background. The panel can be seen in Figure 3.3 with the gondola on top, to prevent damage when the panel is placed on the ground, and the IR LEDs highlighted by red circles.

3.2 Graphical User Interface

Figure 3.4 shows the graphical user interface (GUI) used for the experiment. Using a USB cable, the blimp is connected by entering the correct COM port and baud rate in the "Serial Port" box. The blimp can also be connected wirelessly by using Realterm, a terminal software used to see data sent to and from the Nanowii [47]. Realterm's GUI is seen in Figure 3.5.

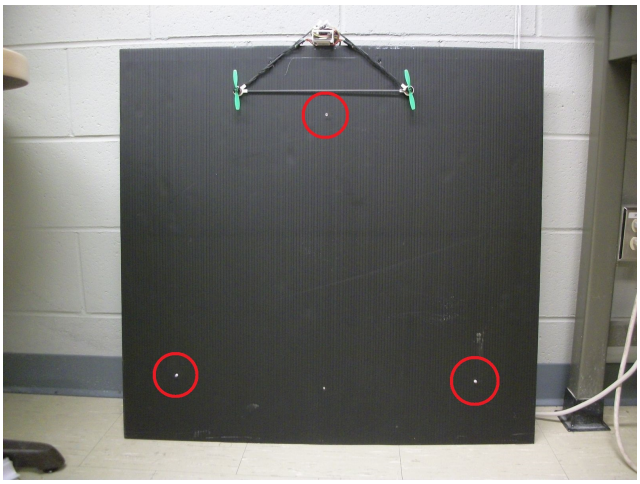


Figure 3.3: Black panel with IR LEDs.

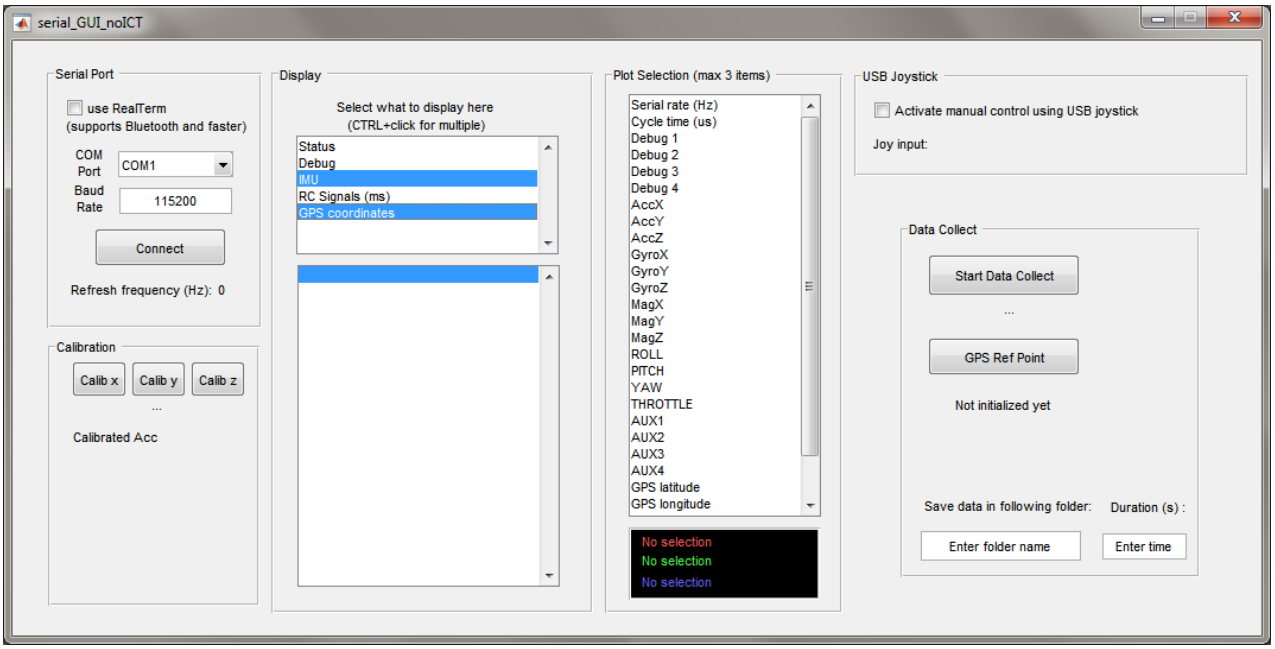


Figure 3.4: Blimp GUI.

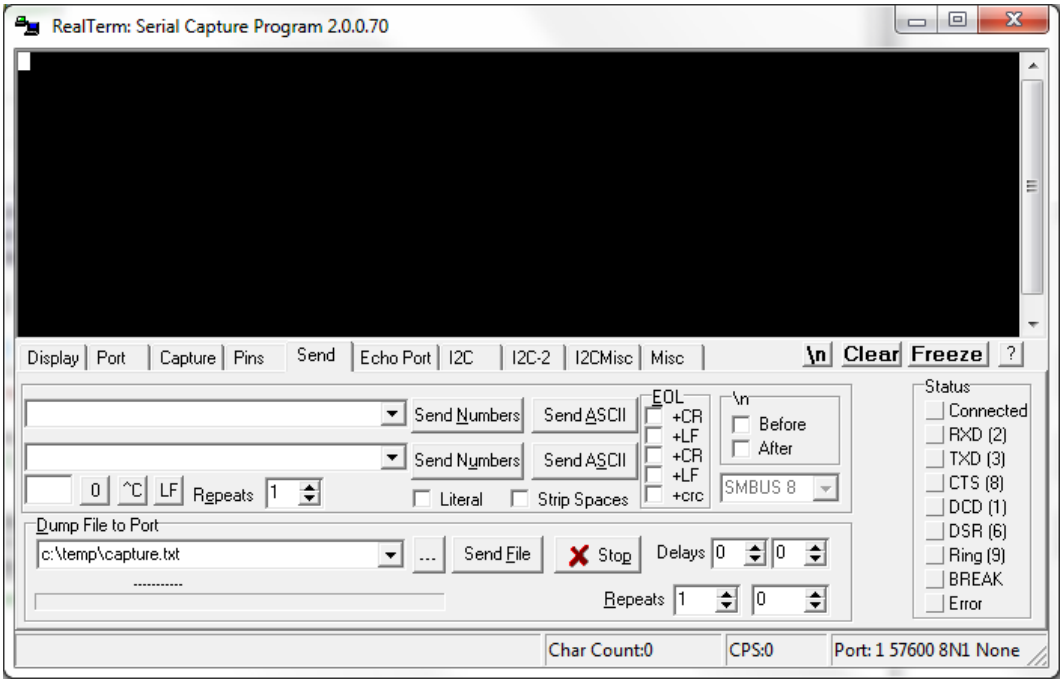


Figure 3.5: RealTerm GUI.

When the blimp is connected to the MatLab GUI, the raw IMU and GPS data obtained via RealTerm can be displayed in the bottom listbox inside the "Display" box. Before logging data for a test, the IMU axes must be calibrated separately, as explained in Subsection 2.3.1, from the buttons inside the "Calibrate" box. Calibrated IMU data in linear acceleration and angular velocity units are displayed here afterwards.

The GPS coordinates of the ground station can be initialized in the "Data Collect" box. Similarly to the IMU calibration, the coordinates are the mean of data taken over a small amount of time. Then, after specifying a folder name and duration, tests can be performed by clicking the button labeled "Start Data Collect", which starts sensor and time data logging. The other features, "Plot Selection" and "USB joystick", in the GUI that weren't discussed here

are not used in this thesis.

Starting a test simultaneously opens a second instance of MatLab which connects the IR camera and starts taking and saving pictures with time elapsed until the test duration has been reached. The main function in the GUI gathers data in an indefinite loop. If the camera is used in the main function, it slows down the IMU data gathering, which has a higher frequency than the camera, as MatLab can only evaluate lines of code sequentially. Opening a second instance of MatLab allows to perform the two actions simultaneously.

The procedure can be summarized with the flow chart in Figure 3.6.

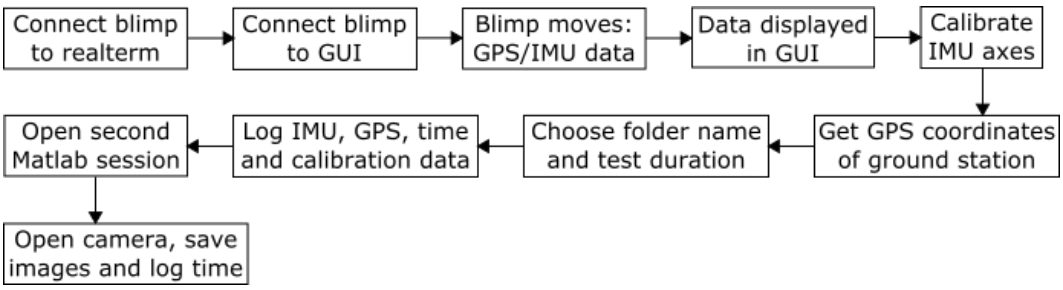


Figure 3.6: Experiment flow chart.

3.3 Data Analysis

Analysis is done in two steps: image processing and then with the EKF. The image processing steps were discussed in Section 2.2 and are summarized in the flow chart of Figure 3.7. Once the PnP solutions of each image are registered, the position, orientation and error can be calculated for each set of solutions. Then, going through the IMU and GPS data sequentially, ${}^{b}\phi$ and

${}^b\theta$ are estimated. As soon as ${}^b\psi$ can be estimated from GPS measurements as in (2.29), the EKF can start. A camera image is considered as measurement for the EKF when the time elapsed of an IMU data is greater than that of the image. As for the sets of solution for each image, the chosen set has calculated angles closest to ${}^b\phi$, ${}^b\theta$ and ${}^b\psi$ estimated from IMU and GPS. This sequence of operations is summarized in Figure 3.8.

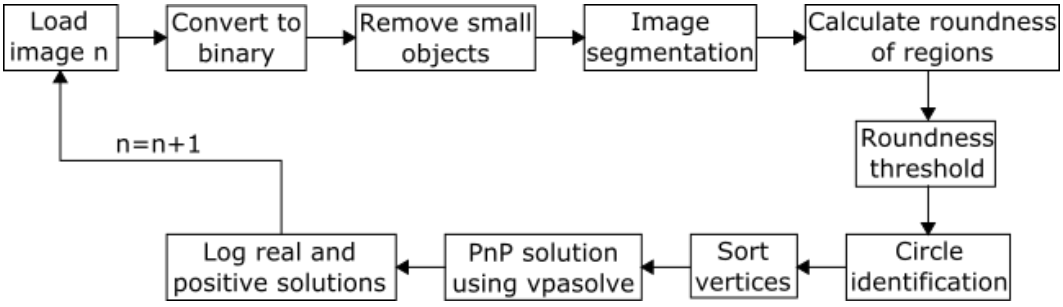


Figure 3.7: Image processing flow chart.

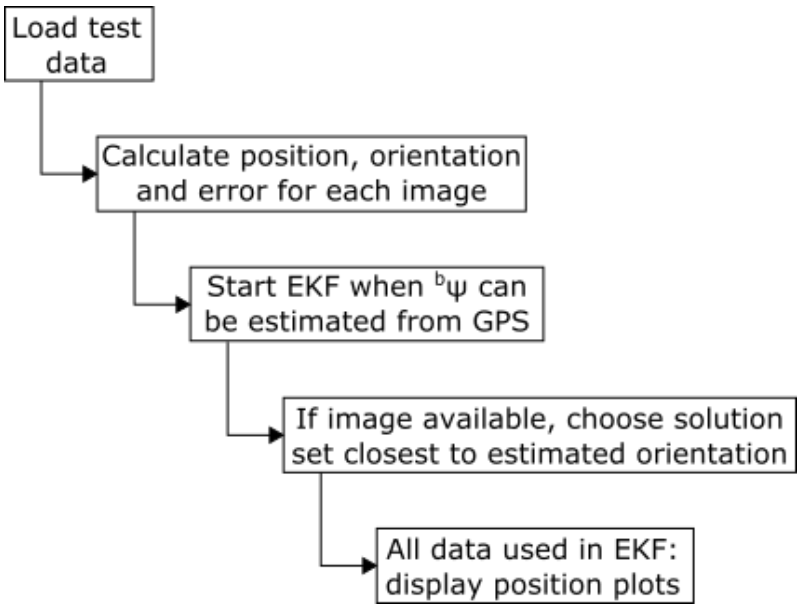


Figure 3.8: Extended Kalman Filter flow chart.

Chapter 4

Experiments

This chapter first presents pre-tests used to validate the theory and give initial observations for the actual tests. The results and analysis of these tests are subsequently shown.

4.1 Pre-Test Vision Calibration

Pre-experiments were conducted to determine the validity of the PnP algorithm estimation. By knowing the triangle pattern dimensions and the position of the LEDs with respect to the camera, the position estimation error could be measured. Two pre-tests were performed at short and long ranges, 26.5 cm and 33 m respectively, where the distances are measured from the sensor of the camera. These pre-tests were performed with the LED pattern perpendicular to the camera's x axis. For this reason, the real and positive PnP algorithm outputs are very similar: any can be used. In the case where

the LED pattern would be at an angle with respect to the camera's sensor, the output solutions would be different. During tests with the blimp gondola, the chosen solution would be the one resulting in angles (2.15) closest to the ones estimated from the IMU, (2.27), and GPS, (2.29).

At short range, the LEDs were fitted on a breadboard and placed on a desk at different θ angles, as seen in Figure 4.1. The two short range trial results can be seen in Table 4.1 along with the relative error for each estimation, where the measured value is taken as the actual value. It can be observed that the estimation in the ${}^{c}x$ axis does not change with ${}^{b}\phi$.

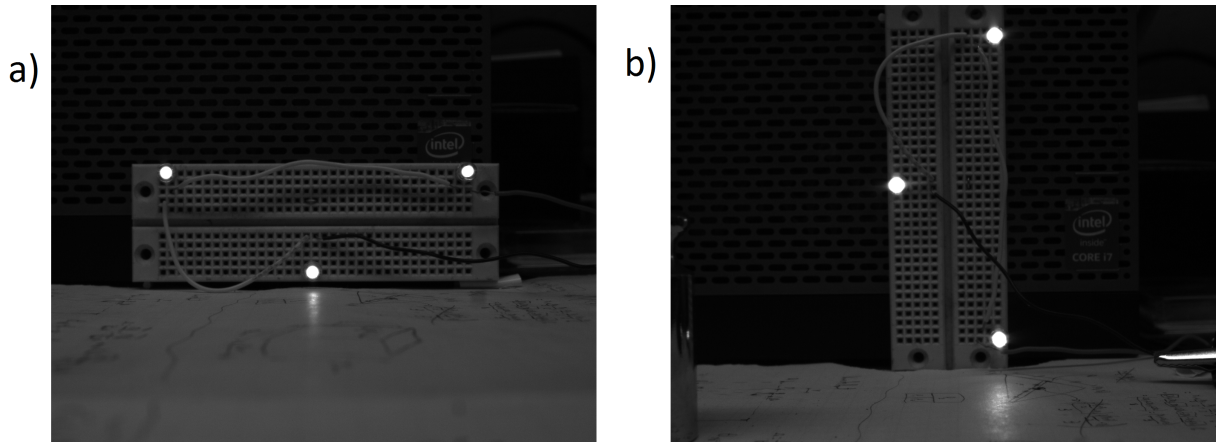


Figure 4.1: P3P short range: a) trial 1, b) trial 2.

At long range, the black panel was used, for a larger triangle pattern, in two hallways of different lengths as seen in Figure 4.2. The results can be seen in Table 4.2.

Similar observations can be made for the short and long range pre-tests. In general, the estimated positions are in the measured range, but there are errors up to 10.6%, 310.0% and 125.0% for each axes. For the ${}^{c}y$ and ${}^{c}z$ axes,

	Trial 1			Trial 2		
	Measured	Estimated	Relative error	Measured	Estimated	Relative error
${}^{c}x_1$	265.0	259.0	2.3%	265.0	258.0	2.6%
${}^{c}y_1$	-18.5	-13.5	27.0%	6.0	9.9	65.0%
${}^{c}z_1$	0.0	-3.1	310.0%	-13.5	-12.6	6.7%
${}^{c}x_2$	265.0	239.0	9.8%	265.0	262.0	1.1%
${}^{c}y_2$	-9.0	-12.6	40.0%	-47.0	-50.7	7.9%
${}^{c}z_2$	41.8	35.8	14.4%	13.5	14.0	3.7%
${}^{c}x_3$	265.0	265.0	0.0%	265.0	237.0	10.6%
${}^{c}y_3$	9.0	13.5	50.0%	-36.5	-28.9	20.8%
${}^{c}z_3$	-41.8	-43.4	3.8%	13.5	13.8	2.2%

Table 4.1: Short range pre-test results (in mm)

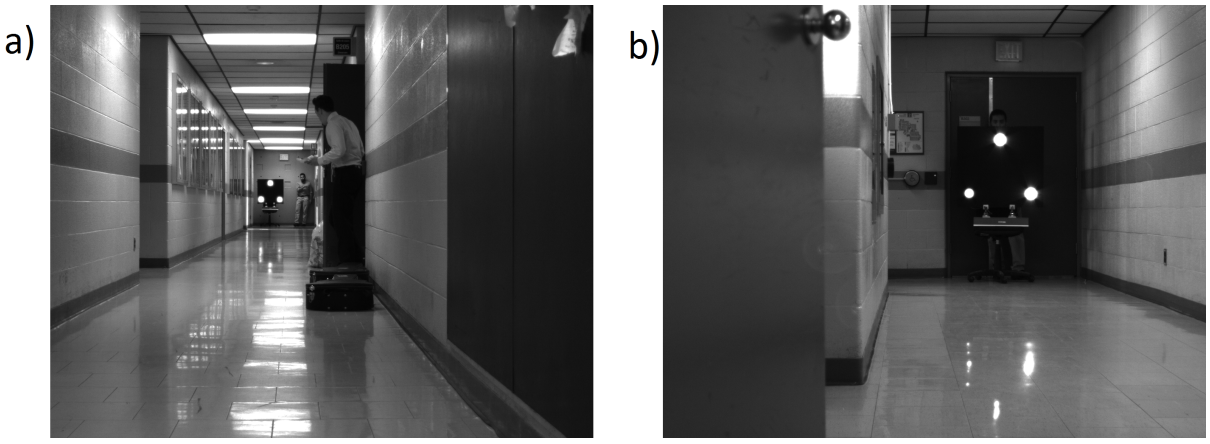


Figure 4.2: P3P long range: a) trial 1, b) trial 2.

	Trial 1			Trial 2		
	Measured	Estimated	Relative error	Measured	Estimated	Relative error
${}^{c}x_1$	32893.0	30420.0	7.5%	9938.0	9350.0	5.9%
${}^{c}y_1$	192.1	629.0	227.4%	192.1	270.0	40.6%
${}^{c}z_1$	-1195.5	-1938.0	62.1%	954.5	341.2	64.3%
${}^{c}x_2$	32893.0	30360.0	7.7%	9938.0	9230.0	7.1%
${}^{c}y_2$	712.3	1144.0	60.6%	712.3	781.5	9.7%
${}^{c}z_2$	-895.5	-1637.0	82.8%	1254.5	629.8	49.8%
${}^{c}x_3$	32893.0	30330.0	7.8%	9938.0	9180.0	7.6%
${}^{c}y_3$	192.1	624.0	224.8%	192.1	259.3	35.0%
${}^{c}z_3$	-595.5	-1340.0	125.0%	1554.5	922.2	40.7%

Table 4.2: Long range pre-test results (in mm)

the large errors can be explained by camera misalignment. This is supported by the fact that the dimensions of the triangle pattern are kept with the estimations. For instance, in Table 4.2, trial 1, the difference between ${}^{c}z_1$ and ${}^{c}z_3$ is approximately 60 cm for both the measured and estimated positions.

For the ${}^{c}x$ axis, the estimated values are always lower than the measured values, as seen in Tables 4.1 and 4.2. The camera misalignment has a lesser impact on the x axis since the cosine of a small angle approaches 1. Then, the image processing step may not be exact. If the image coordinates, found with the "imfindcircles" function, of the circles' centers change, the outputs would also differ. For instance, if the distance between the circles' centers increase on the image, the pattern will appear larger, thus closer. Altering the shape of the circles, using the dilation operation for example, could change the centers coordinates. Using Figure 4.2 a), the effect off the structuring element size for the dilation operation on ${}^{c}x$ of the triangle's centroid has been evaluated in Figure 4.3 by varying the size from 1 to 20 pixels.

Figure 4.3 shows that there is variance in the estimated ${}^{c}x$ from the structuring element size between 3 and 14 pixels. Under the size of 3 pixels, circle recognition failed. Starting at the size of 15 pixels, there is a sharp decrease in the estimated ${}^{c}x$: the circles become so large that they merge together and appear bigger, thus closer. In this thesis, the pixel size was fixed at 10 pixels to neglect the effect of changing the structuring element size.

Since there is still a difference between the measurements and estimations

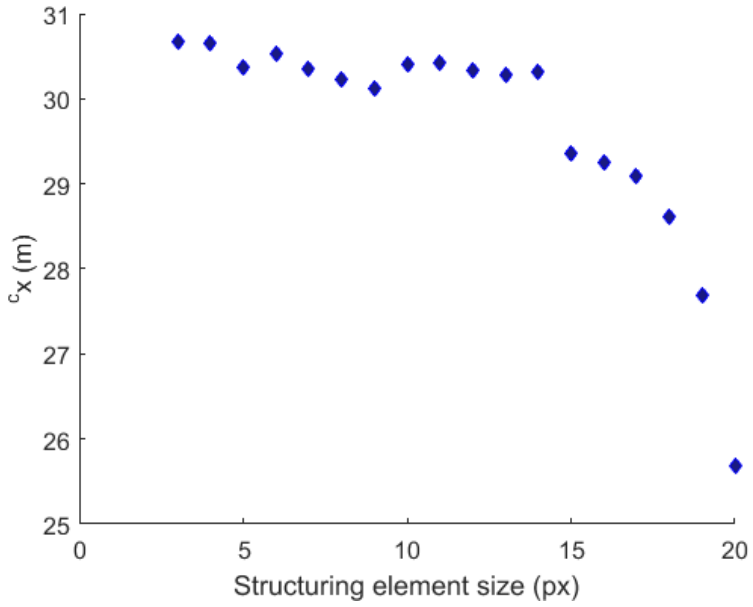


Figure 4.3: Effect of structuring element size on ${}^{c}x$

of ${}^{c}x$, the estimations were calibrated with pre-tests made using the black panel in the same hallway than for the long range pre-test. The panel was moved across the hallway at regular intervals and ${}^{c}x$ was estimated for a hundred image frames at each location. The intervals were equal to three times the length of the floor tiles, or 91.5 cm. From this data, it was possible to plot the average absolute error $\epsilon_{{}^{c}x}$ and standard deviation $\sigma_{{}^{c}x}$ for each distance, as seen in Figure 4.4 and 4.5.

An exponential and second order polynomial curves were fitted on the plots. Their respective equations are the following:

$$\epsilon_{{}^{c}x} = 10.93e^{-0.4247{}^{c}x} + 7.845e^{-0.001921{}^{c}x} \tag{4.1}$$

$$\sigma_{{}^{c}x} = (9.80e^{-5}){}^{c}x^{2} + 0.00143{}^{c}x - 0.00367 \tag{4.2}$$

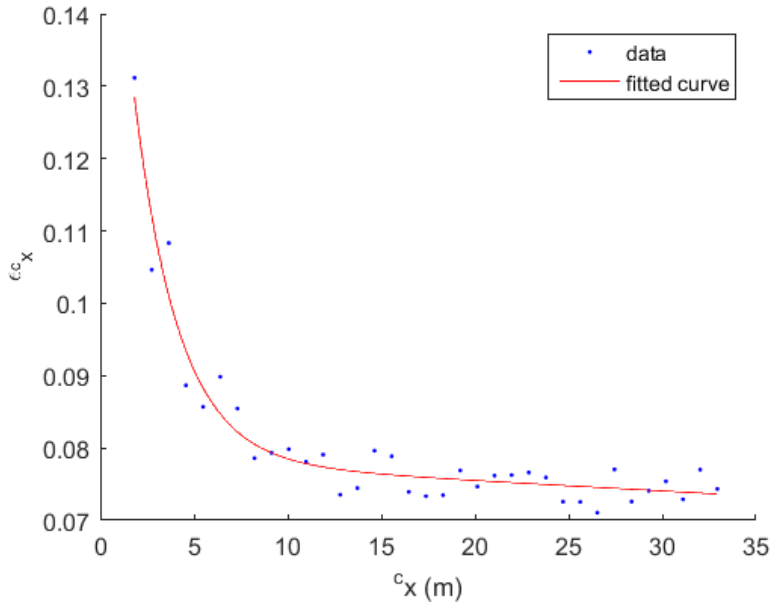


Figure 4.4: Average absolute ${}^cx$ error

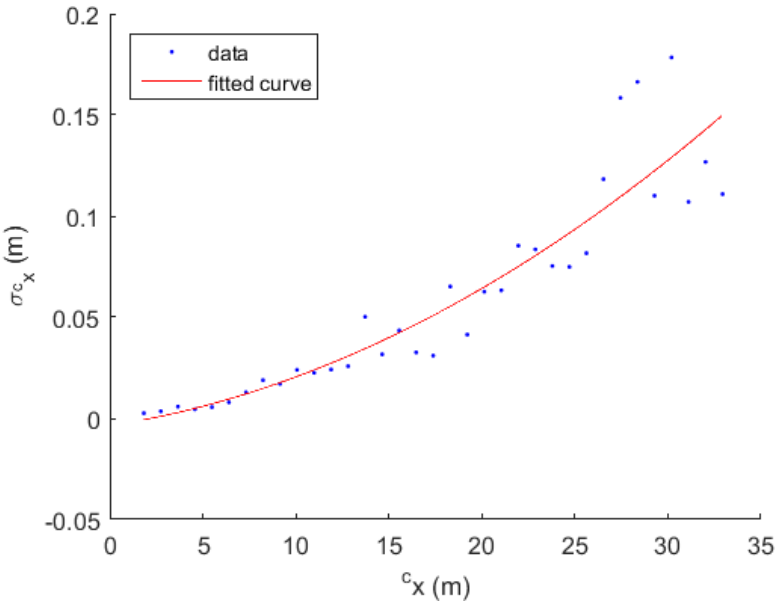


Figure 4.5: Standard deviation of ${}^cx$ measurements

For the actual tests, the ${}^{c}x$ estimation can be rectified by adding the error from (4.1). As for (4.2), it is used as the uncertainty $\Delta {}^{c}x_p$ found in (2.16).

The effect of lens distortion was also studied during these pre-tests. It was possible to compensate for the distortion using an application from the MatLab Image Processing and Computer Vision toolbox. Since the effect was found to be minimal on the resulting images, the effect of lens distortion was neglected for the actual tests.

4.2 Pre-Test Observations

Pre-tests using IMU, GPS and camera were performed inside a building. While this allowed to verify the sensor data acquisition, it also helped identifying problems that could arise during tests. When the blimp was kept stationary, to identify the coordinates of the ground station for instance, the GPS readings drifted. Moreover, the GPS coordinates were found to be off by about 500 m [48]. The source of this problem was most likely interference of the GPS signals caused by the building. Secondly, the blimp microcontroller would sometimes disconnect from the GUI, caused by batteries with low voltage. Thirdly, the IMU and GPS readings were not be updated while the blimp was still connected. It was found that, using a bluetooth receiver with 10 m range, there could be interference with the signal if a person stood in the receiver's line of sight. Lastly, the camera could suddenly disconnect, potentially due to continuous usage leading to overheating. This could be prevented through the

use of an external fan or performing the tests in a cold weather.

A different LED assembly, which better simulated the front of the blimp's envelope than the black panel, was used for a pre-test performed outside. It was a 70 cm diameter exercise balloon with three IR LEDs in front of black cardboard, taped on the balloon's circumference, seen on Figure 4.6. The LEDs were powered individually using two CR2016 Lithium $3V$ batteries.

Figure 4.6: Balloon with IR LEDs.

With this assembly, it was difficult to properly align the LEDs with respect to the camera. In fact, at least one LED was often not visible on the camera image display. This assembly also allowed to see the effects of sunlight on image processing: not only did the LEDs appeared very dim, the sun glare on the balloon affected the LED detection algorithm by being too close to an LED, reducing the roundness of its illumination, or by appearing as a fourth circle. For these reasons, subsequent tests were performed with the black panel after

sunset. A discussion of possible implementation will be provided in Section 5.2.

4.3 Park Test

Because the blimp's controls were not fully defined at the time and to prevent damaging the blimp's components, the tests were performed while carrying the UAV and LED assembly. Simulating the descent could then be done by walking down a hill. The first test was conducted in park Lemoyne, Gatineau, as seen in Figure 4.7. This park was chosen for its large open space and significant slope. The ground station location and farthest distance used for the test are circled in red.

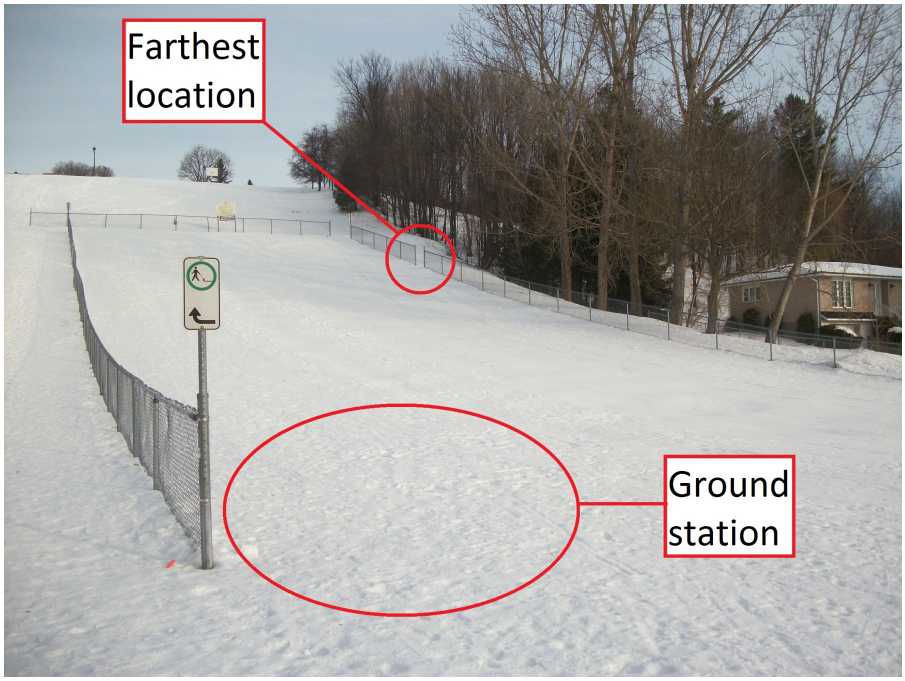


Figure 4.7: Park test location

Figure 4.8 shows a picture of the ground station taken during the park test.

Figure 4.8: Ground station during park test

Figure 4.9 shows the descent with the black panel during a park test. The image processing result is displayed on the ground station screen.

Multiple tests were performed at this location by varying the path. General remarks for the park tests are the following:

- The position and orientation are plotted for each axes. On each plot, four sets of data are displayed together for comparison: camera measurements Z_{cam} with corresponding EKF estimation X_{cam}, and camera measurements Z_{gps} with corresponding EKF estimation X_{gps}. However, there is no Z_{gps} available for ${}^{b}\phi$ and ${}^{b}\theta$.

- To simplify calculations, the measurements from each sensor, camera,

Figure 4.9: Park test descent in progress

IMU and GPS, were all taken at the same frequency of 3.22 Hz that could be provided by the camera. This value is low, compared to the maximum camera frame rate of 60 Hz, due to running the Matlab GUI and image processing simultaneously.

- Only one instance of Matlab is used to perform the tests.

- Because Z_{gps} was updated at approximately $0.99Hz$, a lower frequency than that of Z_{cam}, the measurement update step in the EKF using Z_{gps} was only performed on every new Z_{gps} rather than every time step.

- The matrix Q_k for both measurement sources was multiplied by a factor of 1000, chosen by trial and error. This was done to increase the weight of measurements since the values in the R_k were large compared to that

of Q_k. Not doing so resulted in the state estimates drifting away from the measurements in some cases.

- The two sample tests presented in this section where chosen as they respectively display large translation and rotation. Two more results are presented in Appendix A.

The gondola's position along the x, y and z axes plotted on Figures 4.10, 4.11 and 4.12 are of a sample test labeled #Direct2, a descent straight towards the camera, stopping around the 42 s mark.

- On each axis, the EKF estimation using camera, X_{cam}, is found to be very consistent with its measurements as it almost completely overlaps Z_{cam}.

- The EKF estimation using GPS, X_{gps}, also follows its measurements Z_{gps} closely, but less so than for the camera. Two sources might explain this situation. The first is that Z_{gps} have a larger R_k, thus greater belief in the dynamic equations. The second is that the GPS measurements take longer to update than the camera, thus less data to rectify the estimate.

- Compared to Figures 4.10 and 4.11, The GPS data sets along the ${}^n z$ axis on Figure 4.12 appear closer to a staircase function since the GPS can only measure height to the nearest meter.

- For each axes, it can be observed that the position estimated from the camera and GPS data sets follow the same trend, but with a relatively

constant offset between them. This may come from the GPS coordinates of the ground station which were not exact. In fact, it was found that the GPS coordinates drifted while remaining in a stationary position. This could be due to it being in direct contact with the gondola, creating intereference in the signal.

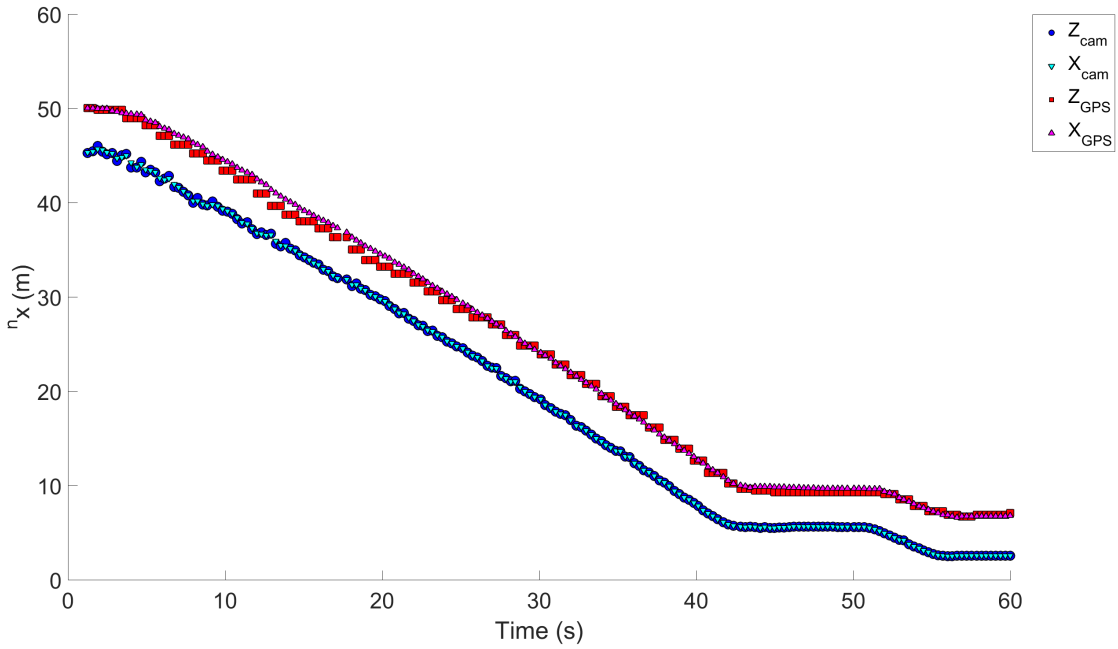


Figure 4.10: Park test #Direct2, ${}^{n}x$

The blimp's orientation for #Direct2, ${}^{b}\phi$, ${}^{b}\theta$ and ${}^{b}\psi$, are plotted on Figures 4.13, 4.14 and 4.15.

- On these three plots, the variance of Z_{cam} is very noticeable. The angles obtained with the camera are more sensitive to small changes between camera frames as opposed to position, which is taken as the centroid of the LED pattern.

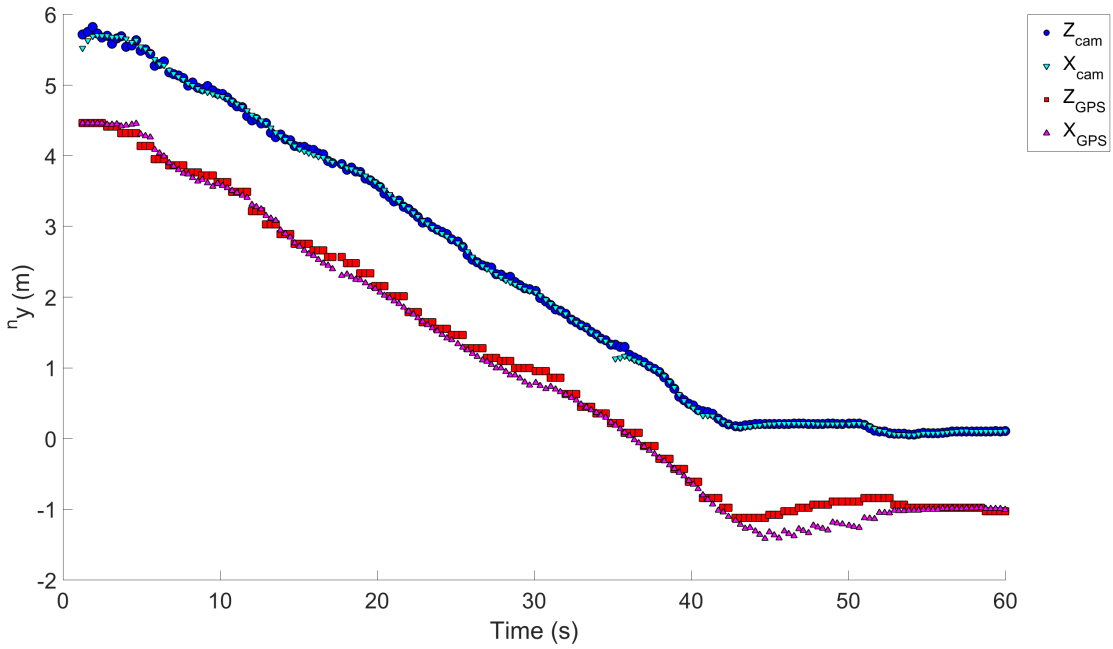


Figure 4.11: Park test #Direct2, ${}^{n}y$

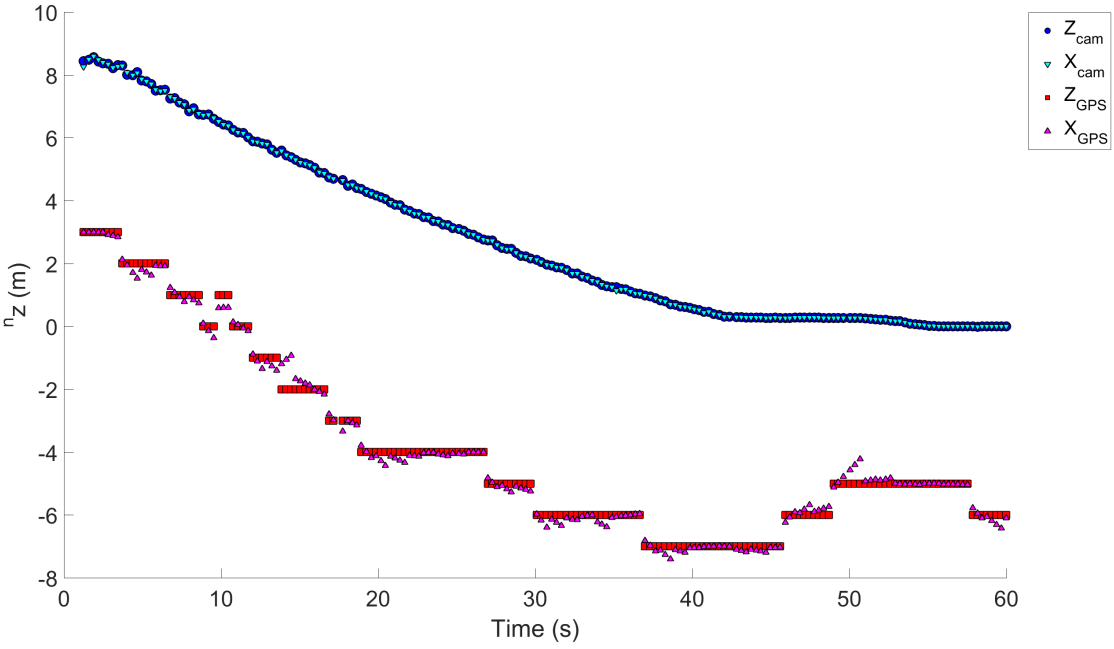


Figure 4.12: Park test #Direct2, ${}^{n}z$

- On Figure 4.14, the Z_{cam} data set appears to move quickly between 25^o and 0^o before the 30 s mark. This error may come from the PnP algorithm, where different solutions may be chosen between consecutive time steps, leading to discrepancies in the data.

- Regarding Z_{cam} on Figure 4.15, it follows the behavior of Z_{gps} since the ${}^b\psi$ measurement from the camera depends on the GPS measurements to determine whether the blimp is approaching the camera or not.

- For each angle of X_{GPS}, they have small variance compared to X_{cam} and appear to be constant. This behavior can be explained by the following two observations: the absence of measurements to rectify ${}^b\phi$ and ${}^b\theta$, and the very large uncertainty $\Delta{}^b\psi$ preventing ${}^b\psi$ to follow the measurements correctly.

- On Figure 4.15, the measurements vary between $\pm 180^o$ after 42 s where the descent stopped and the blimp remained mostly idle. Three causes can explain this issue: the drifting GPS measurements, ${}^b\psi$ being very close to 180^o throughout the test and using the "atan2" operator instead of "arctan" in (2.29), to get values between $\pm 180^o$.

For the test labeled #Angle_Sweep3, the blimp was rotated back and forth in every direction to verify the performance of the EKF algorithm on the orientation. The blimp's position for this test is plotted on Figures 4.16, 4.17 and 4.18.

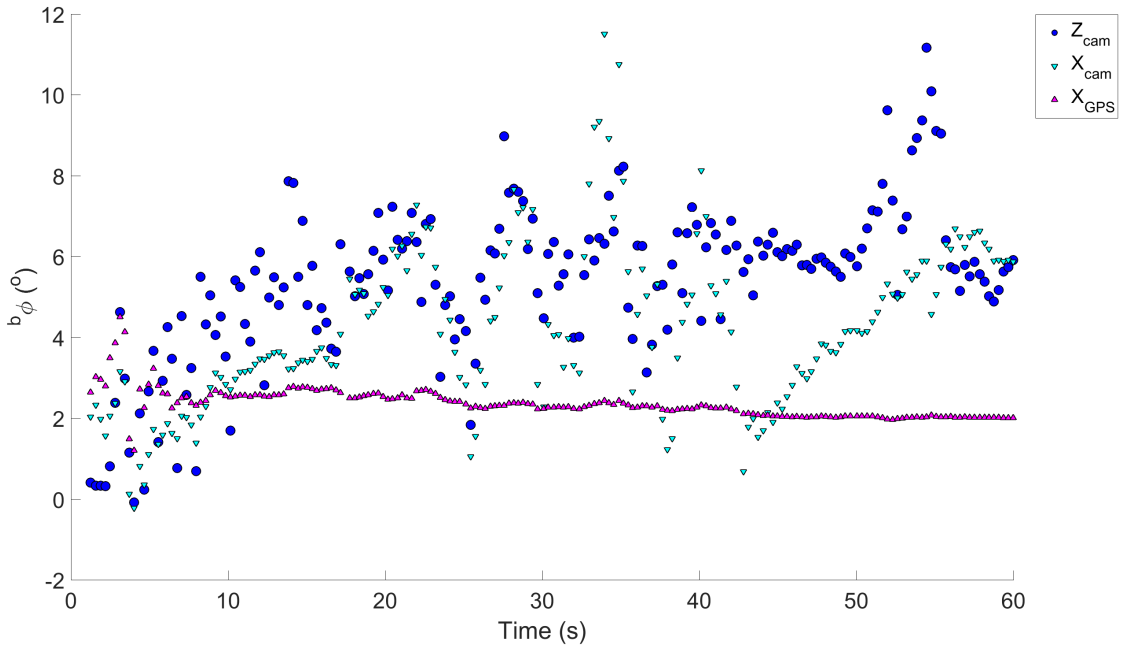


Figure 4.13: Park test #Direct2, ${}^{b}\phi$

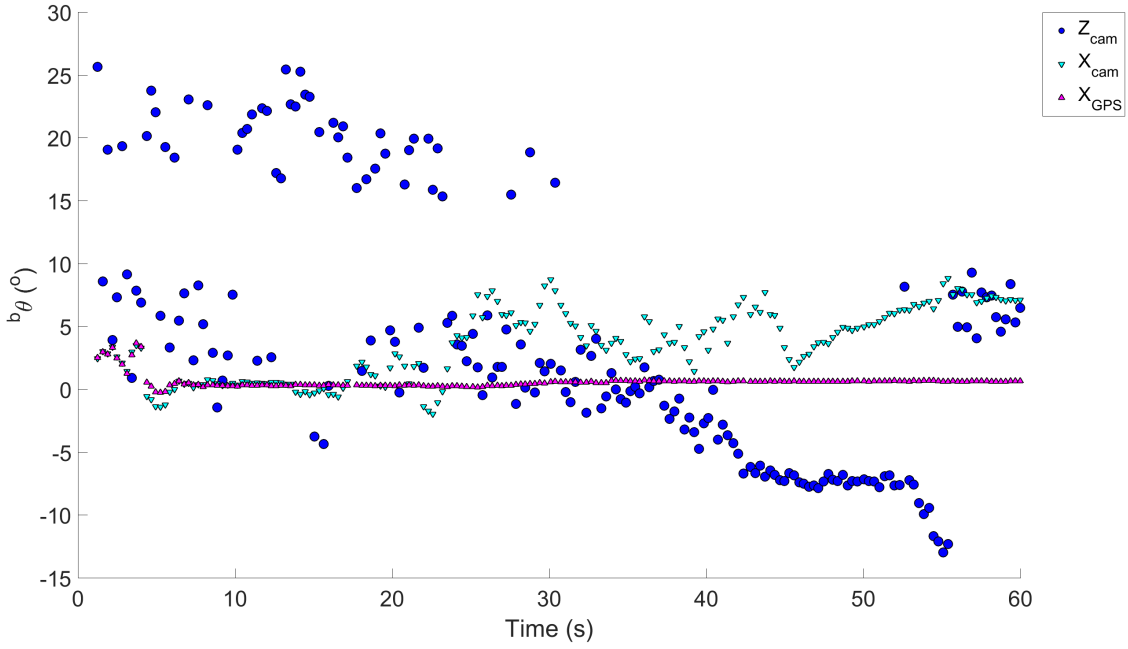


Figure 4.14: Park test #Direct2, ${}^{b}\theta$

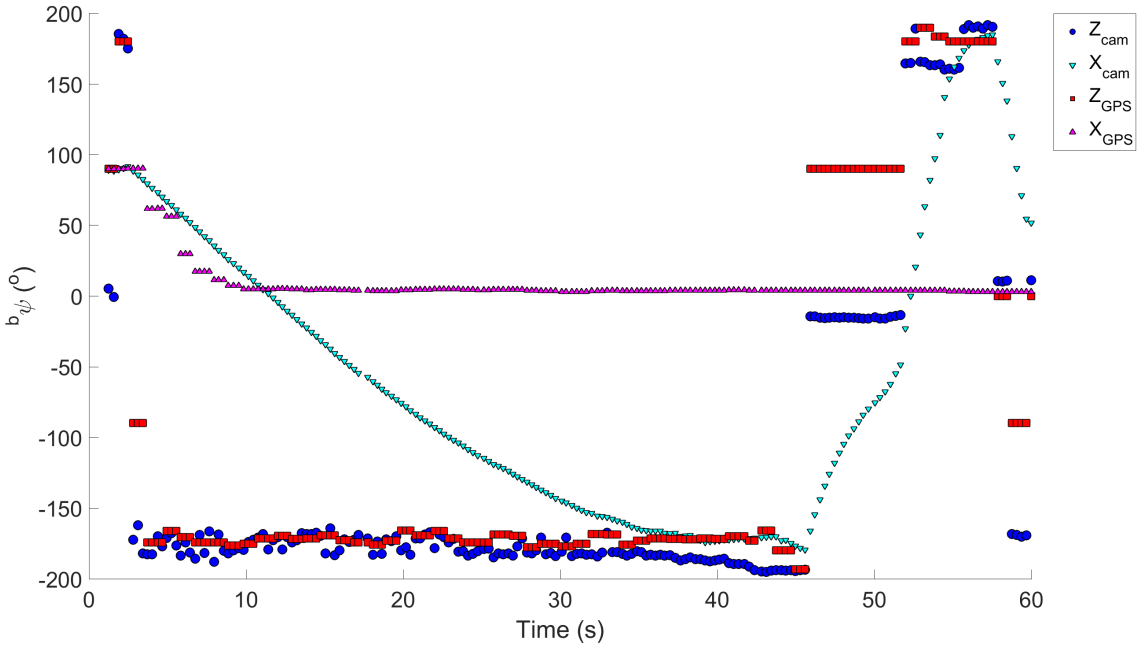


Figure 4.15: Park test #Direct2, $^b\psi$

- Similarly to test #Direct2, there is a large bias between the camera and GPS data sets for position, due to the drifting GPS coordinates. In fact, a variation of 4 m apears in the GPS data sets for the z axis on Figure 4.18 even though there were no apparent vertical movements during the tests.

- On Figure 4.17, oscillations along the y axis are present since the rotation of the LED pattern was not about its center of gravity. The larger period of oscillation in X_{cam} compared to that of X_{GPS} is most likely due to the infrequent Z_{GPS} update.

The orientation plots of #Angle_Sweep3 in Figures 4.19, **??** and 4.21 present more consistent results than for #Direct2 as the angles are less noisy,

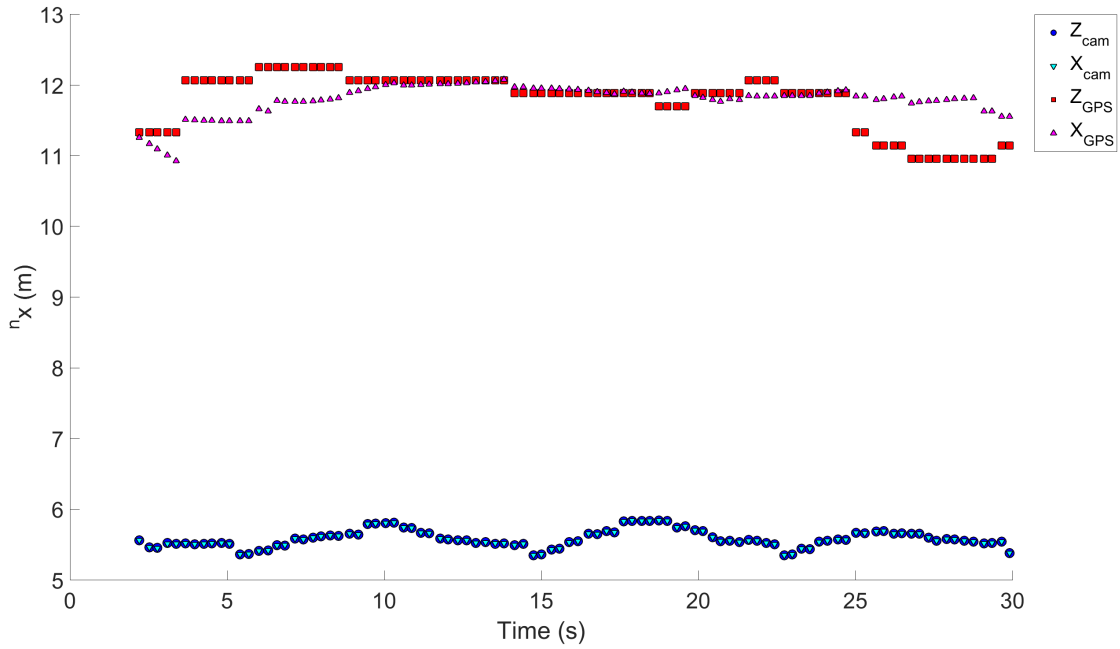


Figure 4.16: Park test #Angle_Sweep3, ${}^{n}x$

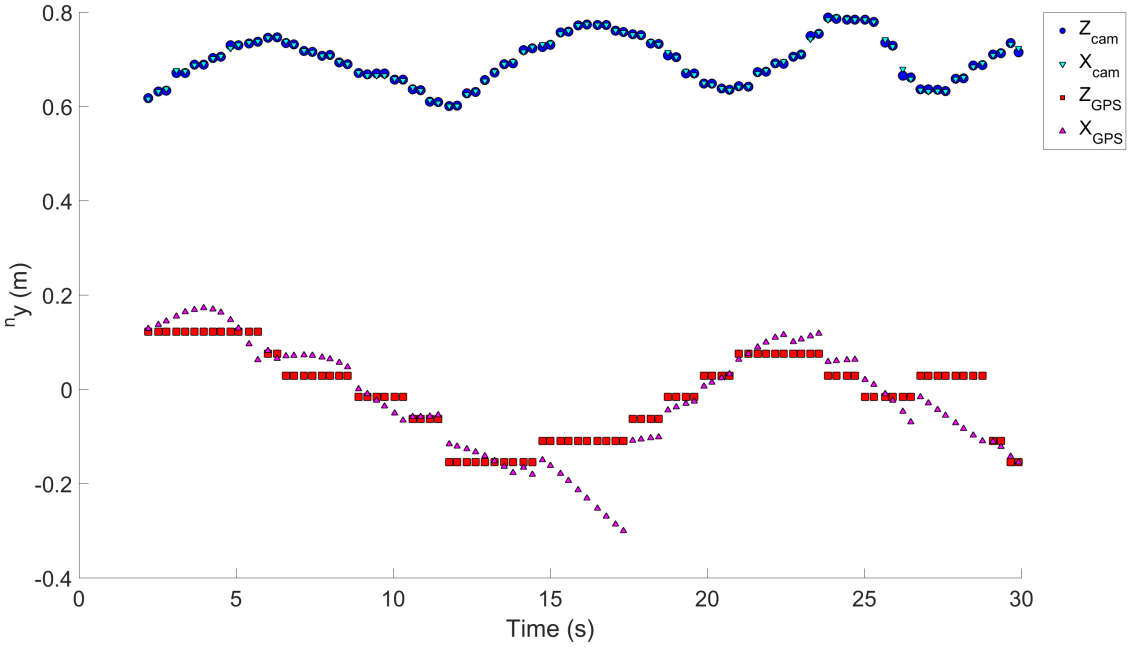


Figure 4.17: Park test #Angle_Sweep3, ${}^{n}y$

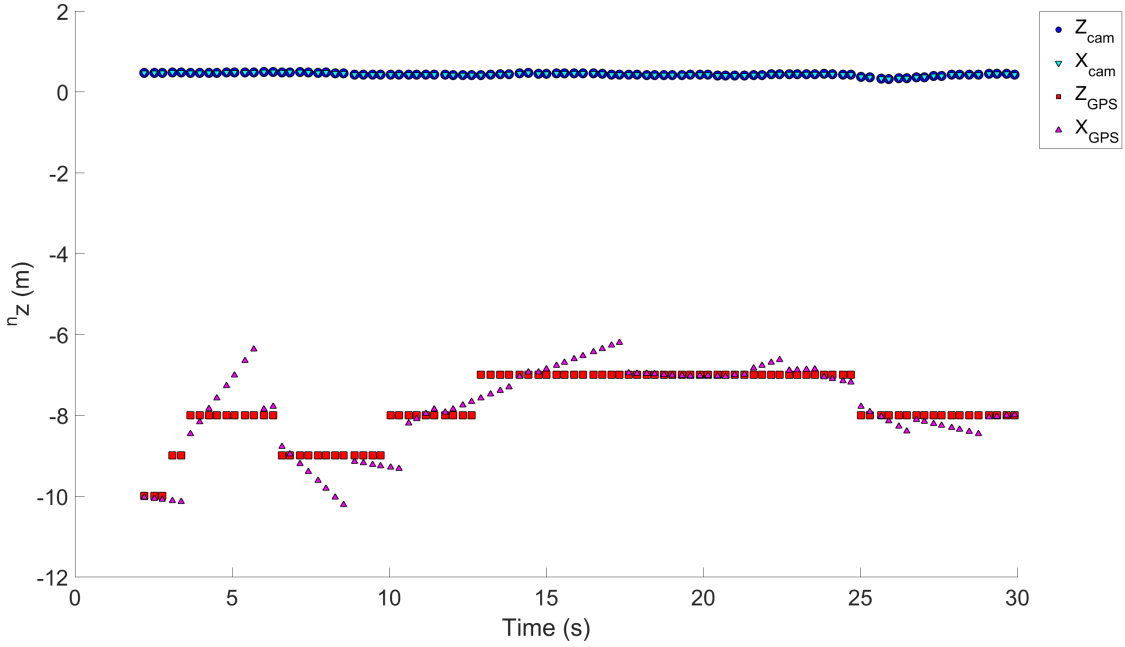


Figure 4.18: Park test #Angle_Sweep3, ${}^{n}z$

for the camera data sets, and less constant, for the GPS data sets.

- In Figures 4.19 and 4.20, the oscillations are present for Z_{cam}, X_{cam} and X_{GPS}. There is an offset between the camera and GPS data sets, however. This could either indicate a certain offset between the IMU axes and the black panel or an error during the calibration of the IMU.

- In Figure 4.20, X_{cam} appears to follow more loosely the measurements Z_{cam}. This could be explained by the choosing the wrong solution from the PnP algorithm, since there appears to be more Z_{cam} data in the upper half of the plot.

- As for Figure 4.21, the behavior is similar to that of Figure 4.15, with the measurements moving between $\pm 180^{o}$ even though the rotation about

the z axis was not significant during this test.

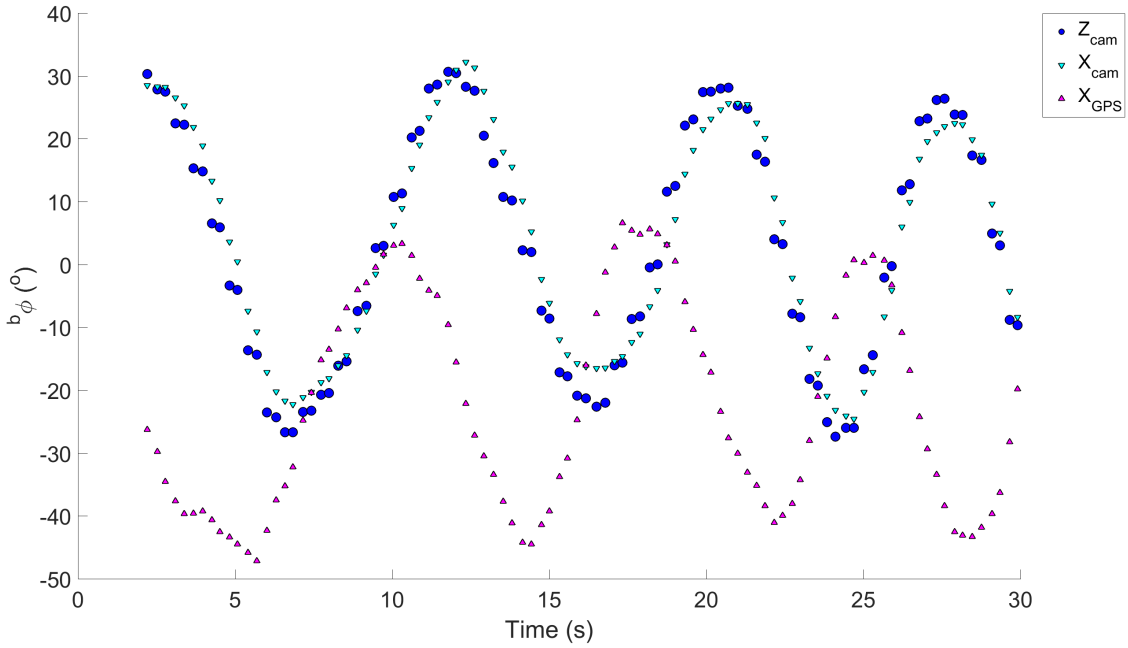


Figure 4.19: Park test #Angle_Sweep3, $^b\phi$

4.4 Building Test

Following the park test, a secondary test was performed in the stairwell of the University of Ottawa engineering building. In this test, two different time steps were used: one for the IMU and one for the camera, to evaluate this effect on the EKF output. The test location is shown in Figure 4.22. This location was chosen to keep the ground station inside the building while the camera was outside, pointed towards the top of the stairs.

The general comments on the building tests are the following:

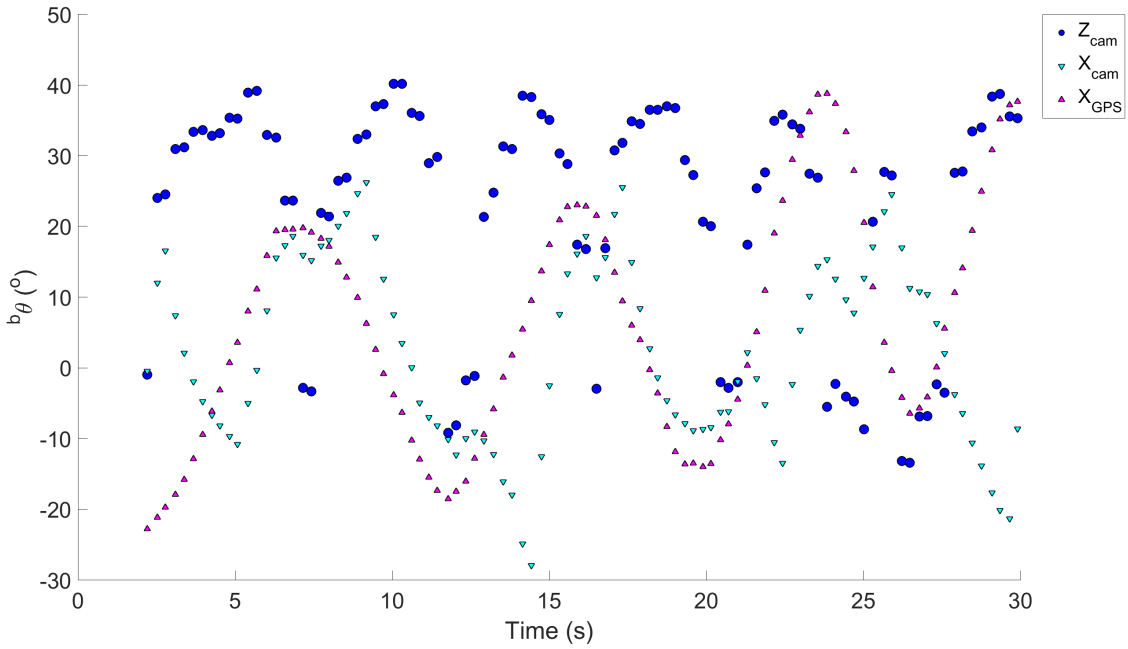


Figure 4.20: Park test #Angle_Sweep3, ${}^{b}\theta$

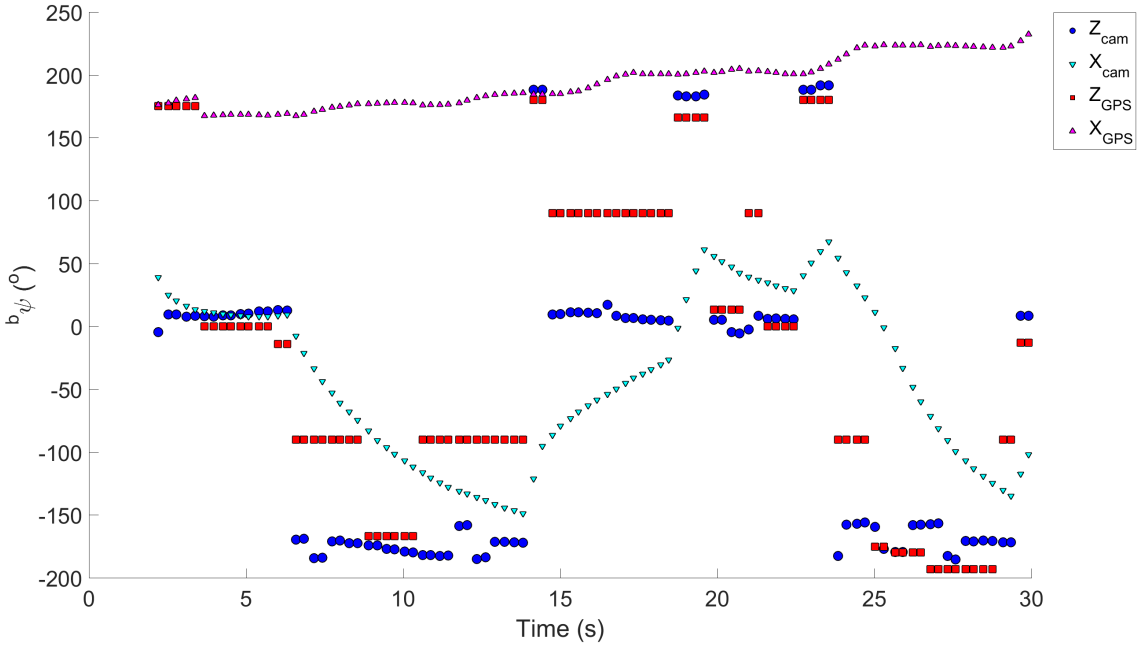


Figure 4.21: Park test #Angle_Sweep3, ${}^{b}\psi$

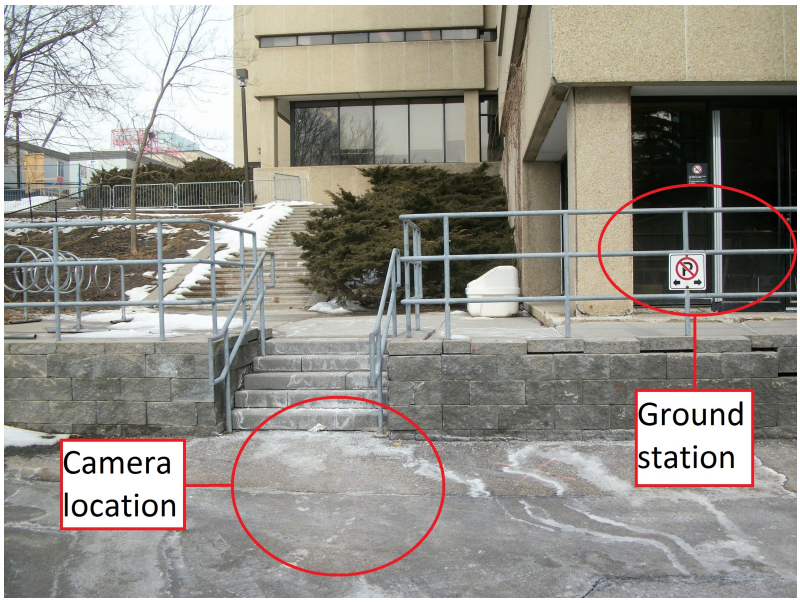


Figure 4.22: Park test location

- The tests performed on this location consisted of walking up and down the stairs while holding the blimp, due to a more limited path. While moving away from the camera is not representative of a landing procedure, the data obtained is still useful.

- The update rate of the IMU was approximately 11.3 Hz and that of the camera, 2.72 Hz. The GPS frequency was the same than that of the park tests.

- To compensate for the smaller time interval of the accelerometer and gyroscope data, the Q_k matrix was once again multiplied by a factor of 1000 and the elements corresponding to the angle process noise, by an additional factor of 1000.

- Two building test results can be found in Appendix A.

The test labeled #Direct9 consisted of walking up and down the stairs. The position plots are found on Figures 4.23, 4.24 and 4.25 and the observations are the following:

- The camera EKF state estimation X_{cam} once again follows closely the associated measurements Z_{cam}.
- From the data obtained, the camera failed to identify the LED pattern several times, notably around the 25 and 45 s marks. On Figure 4.24, the estimation for ${}^{n}y$ around the 45 s mark simply used the dynamic equations until a new measurement was provided at 47 s, slightly rectifying the estimate. Thus, failing to recognize the LED pattern for a few seconds does not greatly affect the estimation.
- At the same time on Figure 4.25, the slope of X_{cam} suddenly changes from negative to positive after the new measurement is provided. This may point towards a problem with the IMU calibration or a too weak effect of the dynamic equations on the state estimation without measurements.
- The state estimation using GPS X_{GPS} appears to follow less closely the measurements Z_{GPS}. This may be due to the building affecting the GPS measurements or the smaller time interval of the IMU measurements, leading to a smaller Q_k thus even less belief in the measurements.
- Similarly to the park tests, there is a gap between the camera and GPS data sets due to the drifting GPS measurement for the ground station,

or camera in this case. On Figure 4.25, this gap is not sufficient to explain the difference between the camera and GPS data sets: while a difference in height of approximately 3 m was covered during the tests, that distance is estimated at 11 m using the GPS measurements. Plus, it estimates that the blimp was moving down for the first 27 s while it was moving up.

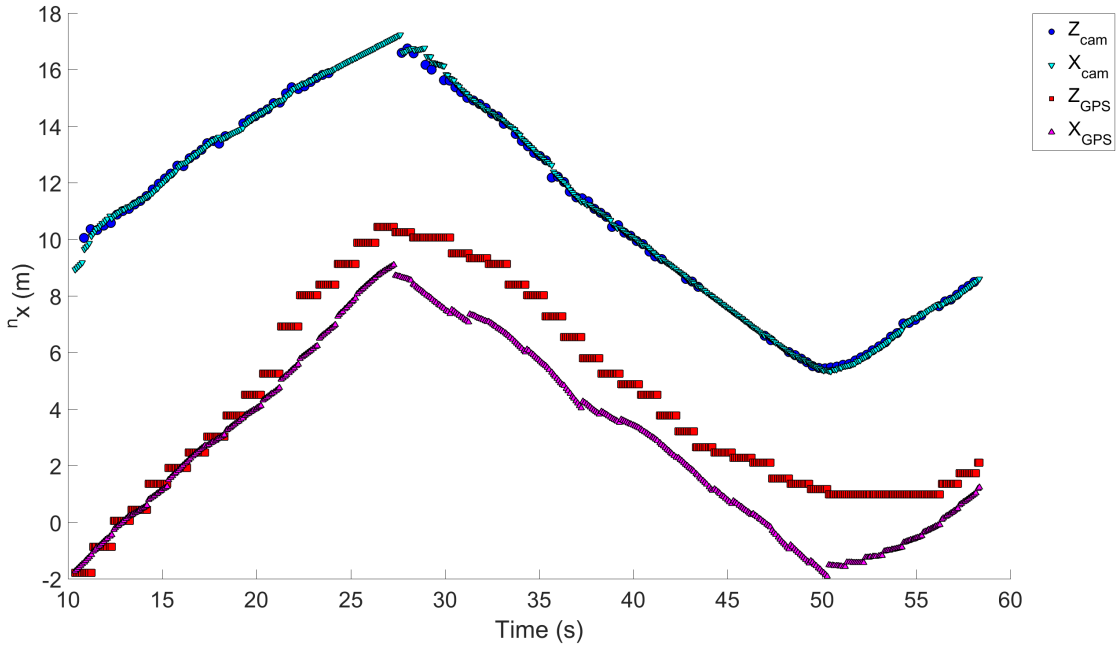


Figure 4.23: Building test #Direct9, ${}^{n}x$

The orientation estimation obtained in Figures 4.26, 4.27 and 4.28 showed similar behavior from that of the park tests. One new observation can be made on Figure 4.28 around the 27 s mark, where the top of the staircase was reached and the descent began, causing a sudden transition from about -60^{o} to 160^{o} on Z_{cam}. The estimation X_{cam} takes approximately 5 s to perform

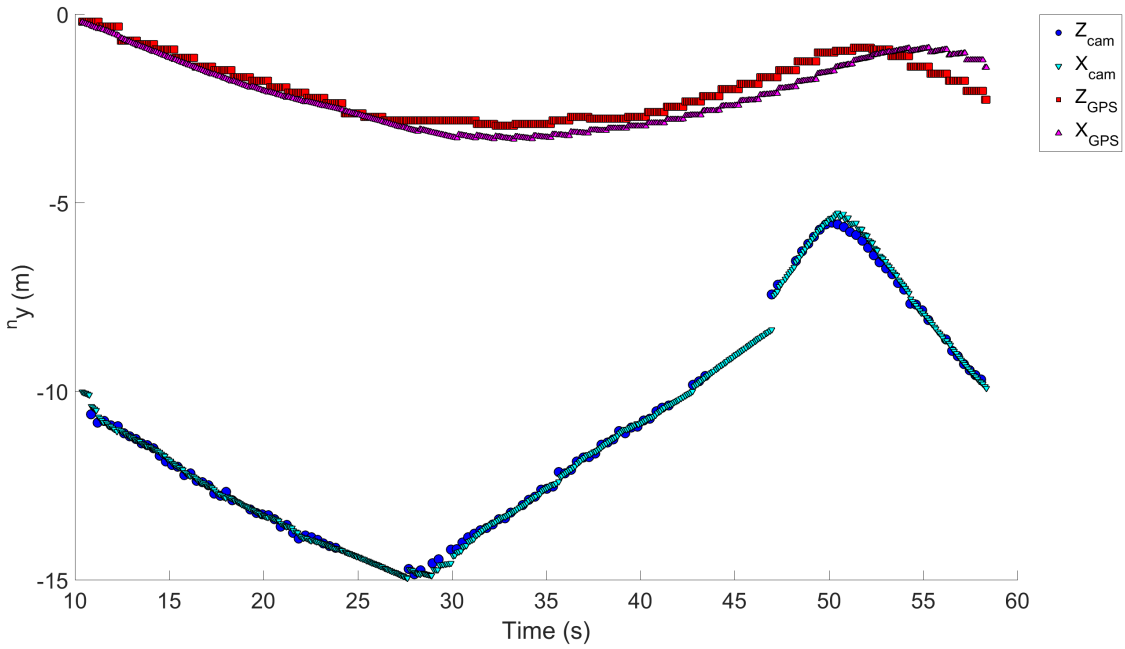


Figure 4.24: Building test #Direct9, ${}^{n}y$

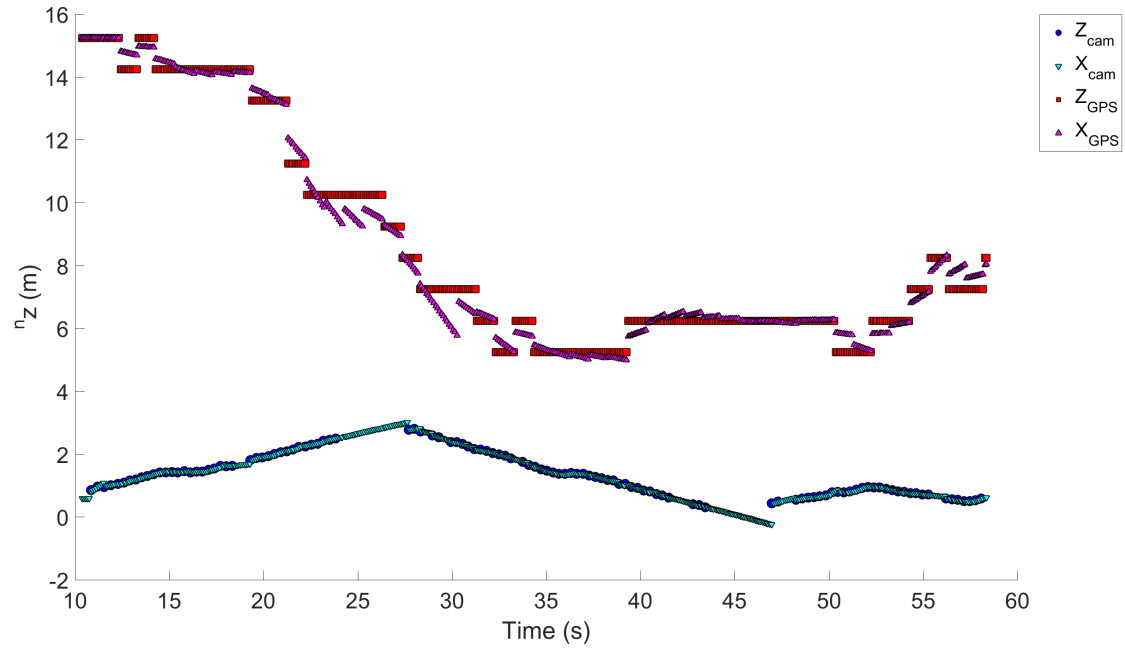


Figure 4.25: Building test #Direct9, ${}^{n}z$

the transition, as if the blimp was slowly turning around. This behavior is expected of the EKF as a sudden change in X_{cam} would indicate a too large belief in the measurements.

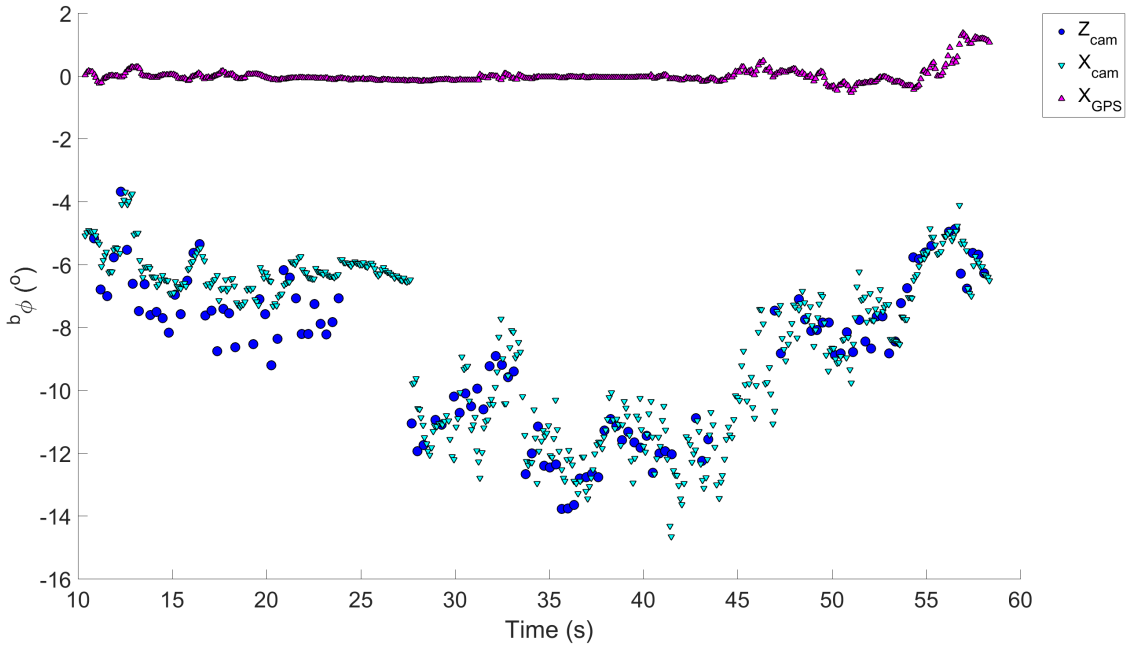


Figure 4.26: Building test #Direct9, ${}^{b}\phi$

To complement the results obtained in subsection 4.3, the last building test presented is of the #Direct17 test where the blimp was rotated about one axis at a time while its position did not change. As for the park test #Angle_Sweep3, the behavior seen on the position plots of Figures 4.31, 4.30 and 4.29 are similar. The drift in GPS measurements is especially noteworthy: 5 m for the x axis, 1 m for the y axis and 14 m for the z axis.

The corresponding orientation plots are shown in Figures 4.32 to 4.34. The three body angles of the blimp were rotated separately in the following order:

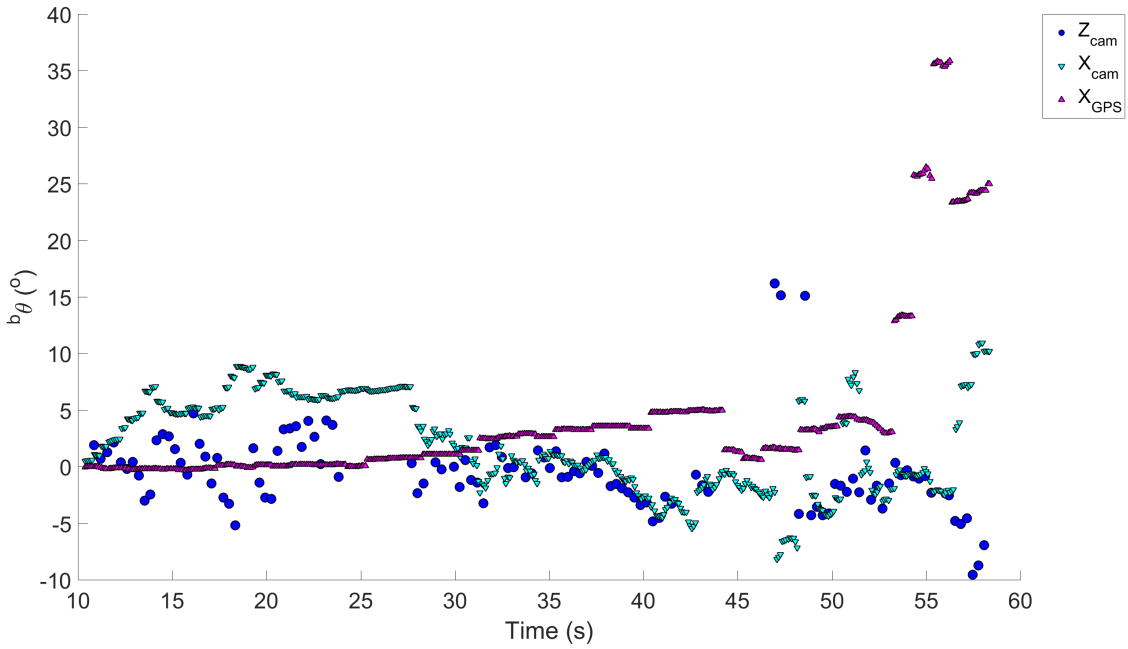


Figure 4.27: Building test #Direct9, ${}^{b}\theta$

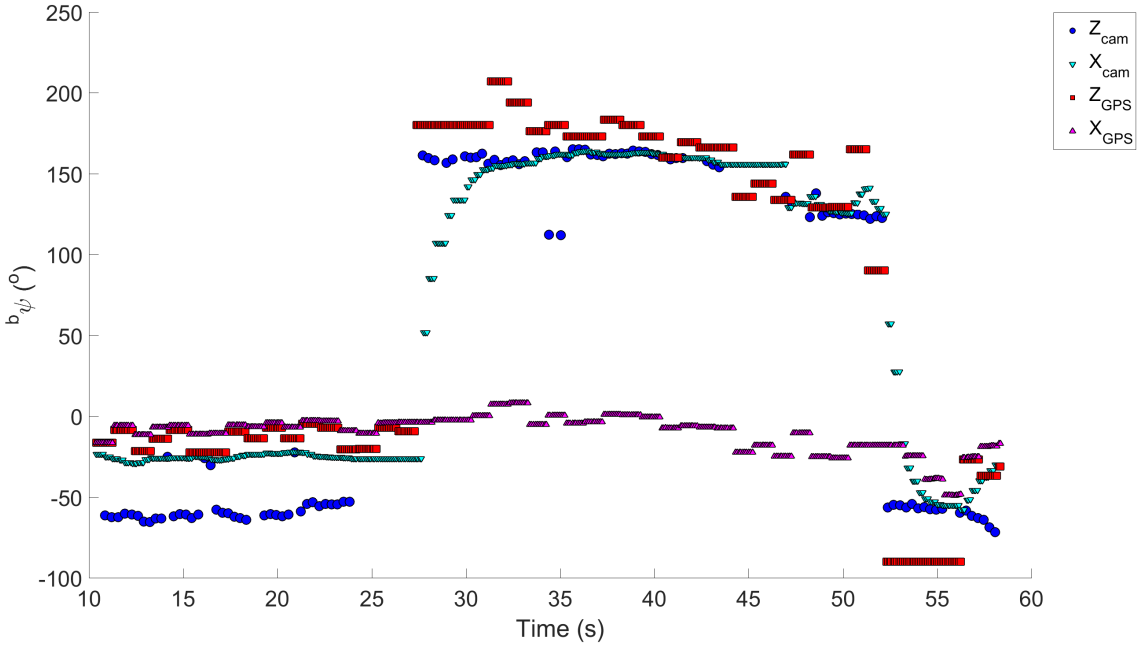


Figure 4.28: Building test #Direct9, ${}^{b}\psi$

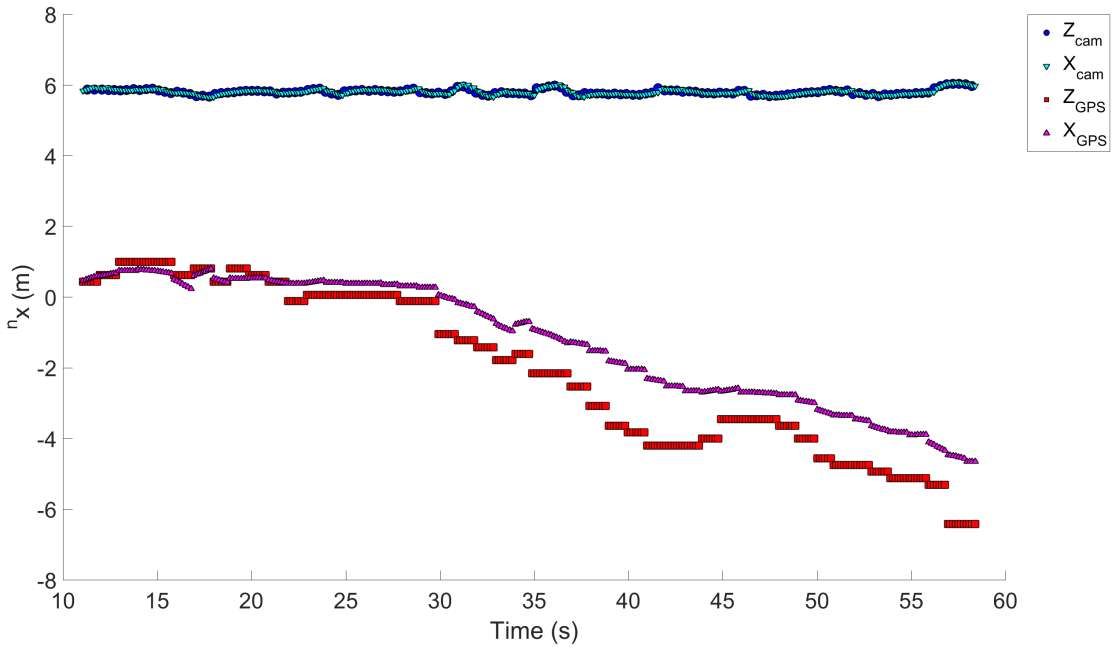


Figure 4.29: Building test #Direct17, ${}^{n}x$

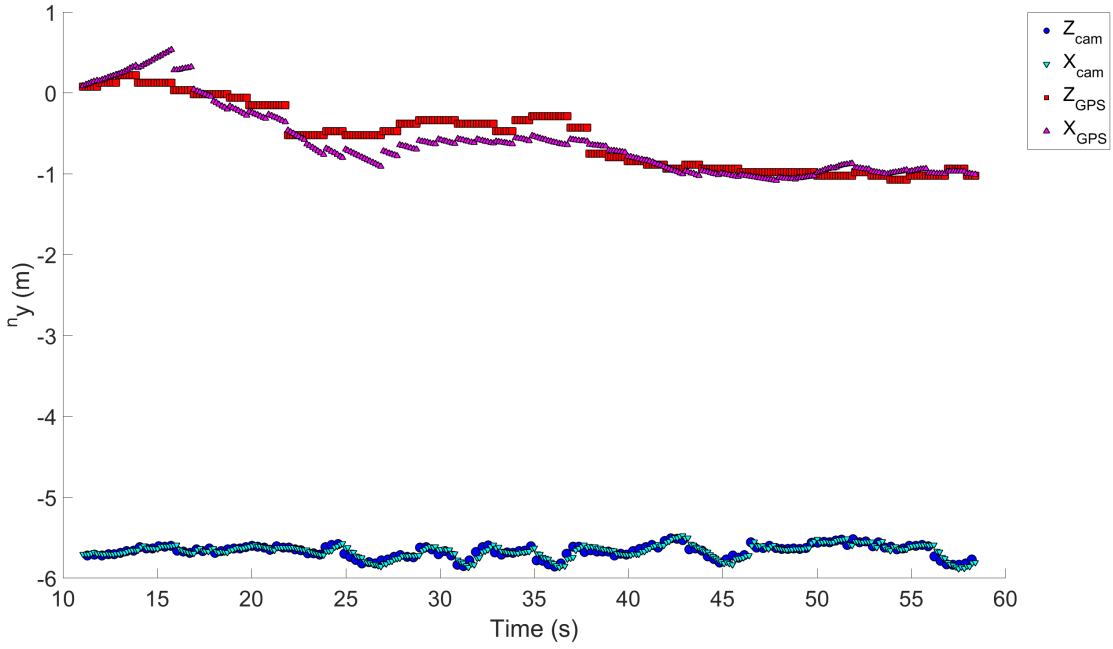


Figure 4.30: Building test #Direct17, ${}^{n}y$

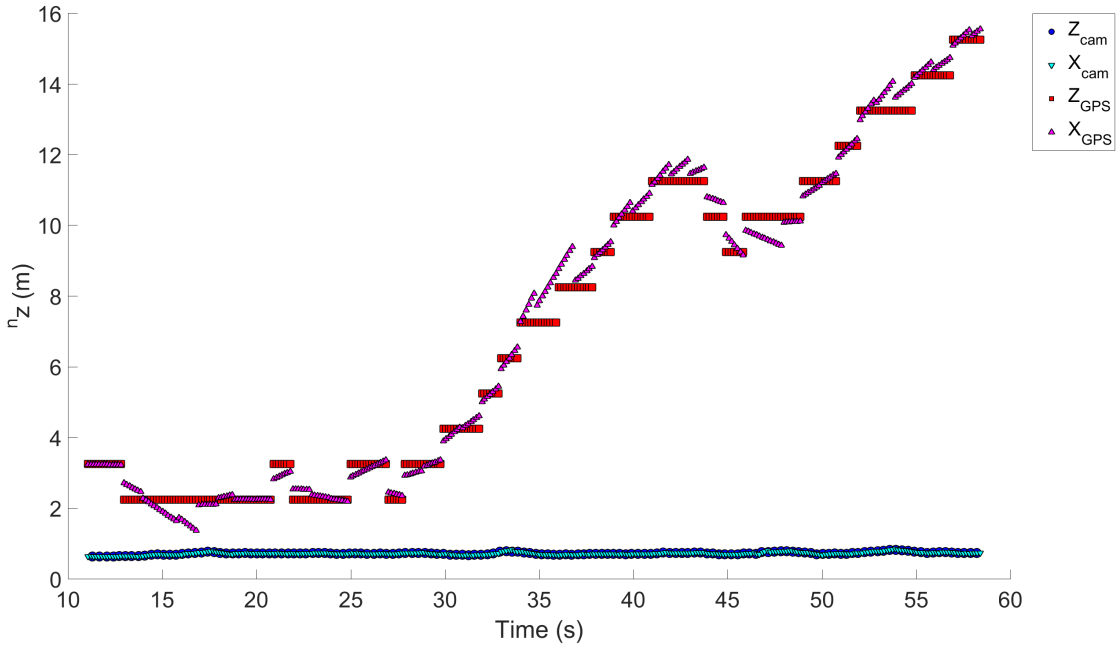


Figure 4.31: Building test #Direct17, ${}^{n}z$

${}^{b}\phi$, ${}^{b}\psi$, ${}^{b}\theta$, ${}^{b}\psi$, ${}^{b}\phi$ and ${}^{b}\theta$

- On Figure 4.32, X_{cam} is seen to follow closely the measurements Z_{cam}. After the 17.5 s mark, the slope of the measurement Z_{cam} becomes negative. At that precise moment, the EKF estimation X_{cam} has a positive slope in between measurements. The cause of this contradiction is unclear.

- On Figure 4.32, the problem of choosing the correct solution of the PnP algorithm is obvious at the 56 s mark, where X_{cam} and Z_{cam} are completely opposed to each other. This is most likely due to ${}^{b}\psi$ which plays a large role in finding the correct PnP solution. In Figure 4.34, Z_{cam} displays large variations not representative of the test.

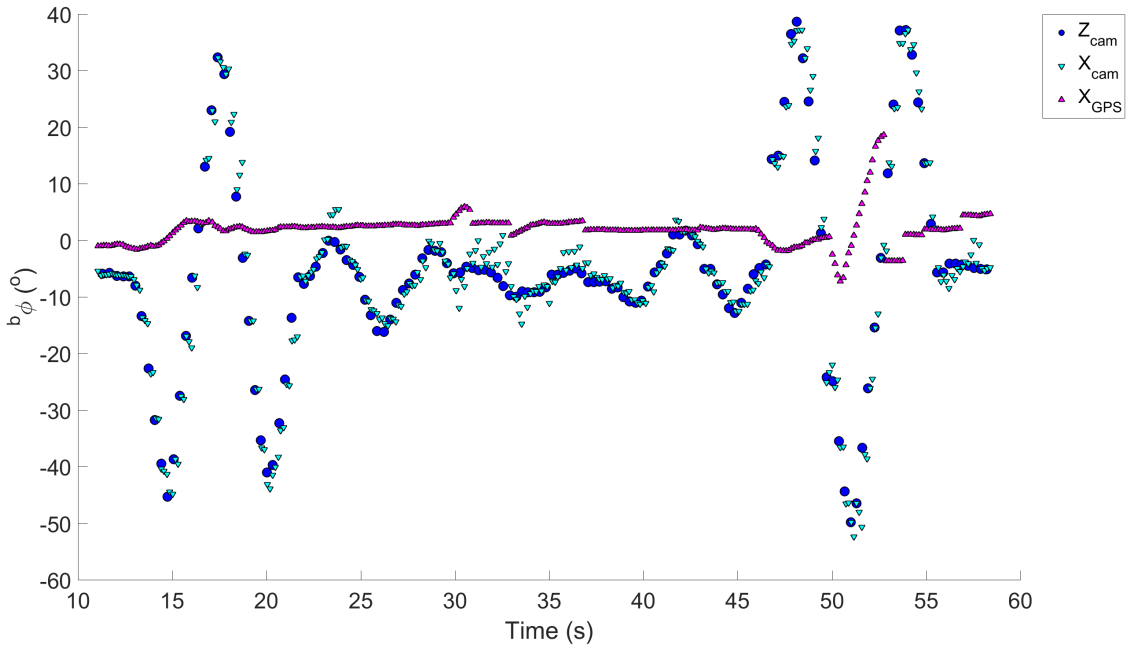


Figure 4.32: Building test #Direct17, ${}^{b}\phi$

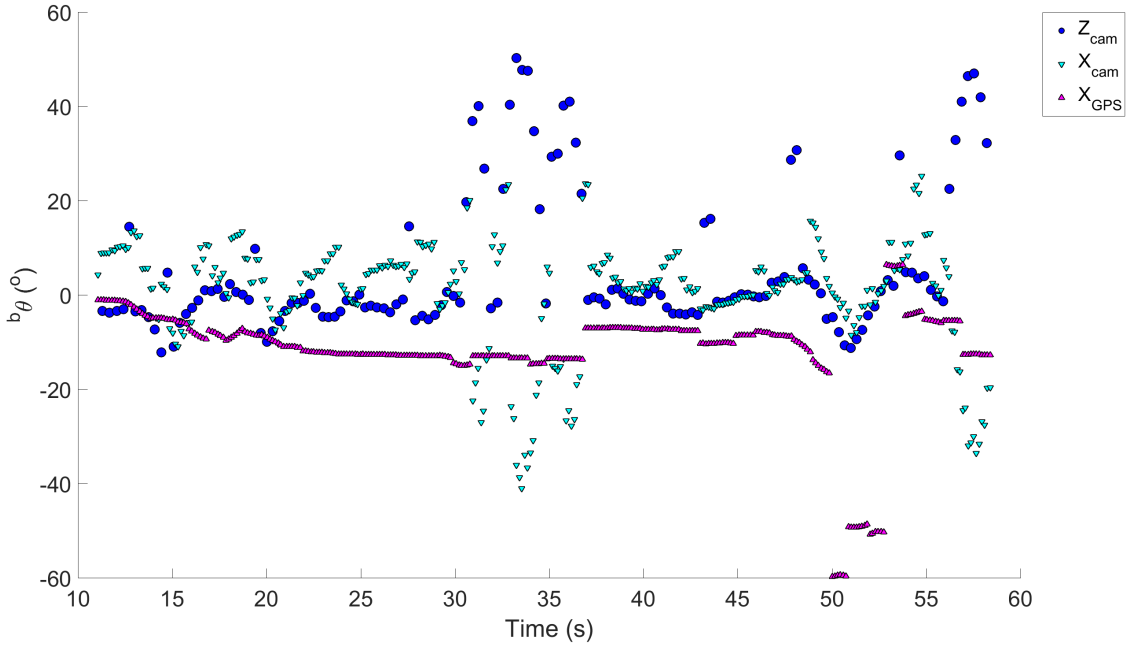


Figure 4.33: Building test #Direct17, ${}^{b}\theta$

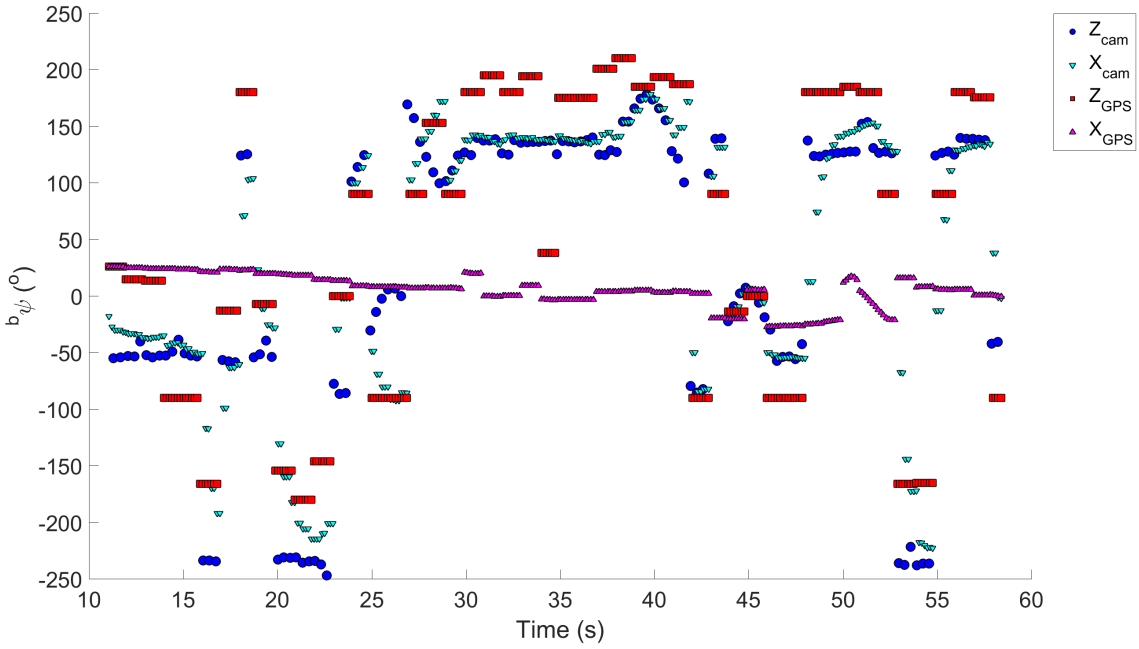


Figure 4.34: Building test #Direct17, $^b\psi$

From the results of the building tests, it was observed that the effect of a smaller time interval for an increase in IMU measurements added more X_{GPS} data in between measurements, thus a slightly better precision. While it could be argued that a blimp does not need so much precision in estimating its pose as compared to a faster moving quadcopter UAV, it is still important for landing which requires precise state estimation. No other significant changes appeared in the overall behavior of the EKF.

Chapter 5

Conclusion and Recommendations

5.1 Key Findings

This thesis presented a method to estimate the orientation and position of a blimp UAV with respect to an origin. The method uses a triangular pattern of IR LEDs, recognized by a grounded IR camera. Measurements from the camera, coming from a PnP algorithm, and an IMU on the blimp are fused with an EKF to obtain the pose estimation. By comparing the state estimation with that of a GPS and IMU measurements fused with an EKF, it was found that both give similar results for horizontal positioning, but the vertical positioning from the camera measurements is much more precise than that of the GPS, making it more useful for landing maneuvers. However, the proposed method

still relies on GPS to estimate the orientation.

5.2 Recommendations

The vision system needs to be more reliable. The LEDs could be replaced by clusters of LEDs or IR spotlights to provide better radiant intensity and angle of view. This would allow the pattern to be visible during daylight as well as countering the single LED misalignment with respect to the camera. A more distinct shape for the LED clusters or spotlights, such as adding concentric IR circles around them similarly to the pattern in [10], could be introduced for better recognition. While a triangular pattern of LEDs can still be used alongside the PnP algorithm, a round model, akin to the balloon showed in Section 4.2, should be used to better simulate the shape of the blimp's envelope.

To help detecting the blimp when out of view, the camera could be installed on two servomotors for pan and tilt. Controls could be added to the servomotors so that once the camera detects the LED pattern, it will attempt to move to keep the LED pattern in the middle of the images. In the case where the blimp would not be facing the camera, due to it turning around for instance, an additional detection system could be used. This could take the form of navigational red and green lights on the side of the envelope. Of course, this would require a visible light camera instead of IR.

For the PnP algorithm to be of proper use, it is necessary to determine the correct solution from its output. The actual method consisted of comparing

the angles estimated from the camera to the ones estimated with the IMU and GPS. If this method is kept, the angle ${}^{b}\psi$ requires a better approximation than that obtained with the GPS. This can be done with the use of a magnetometer.

To reduce the potential errors from the camera's angles ${}^{c}\phi$, ${}^{c}\theta$ and ${}^{c}\psi$, a sensor suite consisting of an IMU and magnetometer could be installed on the camera. This would provide more precise angles, reduce the errors from (2.22) and increase the belief in measurements for the EKF.

The effect of temperature on the IMU measurements and a more in-depth calibration method could be used to improve their accuracy and precision. If the blimp is to fly, the effect of Earth's rotation could be added to the dynamic equations.

The test results showed similar behavior between the EKF estimations with the camera and with the GPS for the x and y axes, but showed a relatively constant bias between the two sensors. The GPS coordinate of the ground station could be acquired using a more advanced GPS method as described in Subsection 1.4.1. For more accurate results from the GPS module onboard the blimp, it could simply be better isolated from the gondola and Bluetooth module to reduce potential signal interference. It could even be interesting to investigate the use of pressure sensors as a mean to replace the GPS altitude measurement for the EKF [49].

If the communicating link between the two Bluetooth modules breaks, The IMU and GPS data won't be transmitted to the ground station until the link

is reestablished by manual operation. To prevent this problem from occuring, a more robust communication system should be employed such as having a secondary communicating link. Alternatively, the program on the blimp's microcontroller could be adapted to provide a basic landing sequence using available on-board sensors.

Lastly, a more advanced dynamic model of the blimp, taking into account external perturbations such as wind, should be used such as the one developed by Reckoskie [50]. If more non-linearities are introduced from this model, it may be preferable to investigate employing a more adapted sensor fusion method rather than keeping the EKF.

References

[1] E. Lanteigne, W. Gueaieb, D. Robillard, and S. Recoskie, "Unmanned airship design with sliding ballast: Modeling and experimental validation," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on.* IEEE, 2016, pp. 1246–1253.

[2] RobotShop, "Robotshop home page," [http://www.robotshop.com/], 2016.

[3] O. Shakernia, R. Vidal, C. S. Sharp, Y. Ma, and S. Sastry, "Multiple view motion estimation and control for landing an unmanned aerial vehicle," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3. IEEE, 2002, pp. 2793–2798.

[4] J. Wendel, O. Meister, C. Schlaile, and G. F. Trommer, "An integrated gps/mems-imu navigation system for an autonomous helicopter," *Aerospace Science and Technology*, vol. 10, no. 6, pp. 527–533, 2006.

[5] A. Cho, J. Kim, S. Lee, S. Choi, B. Lee, B. Kim, N. Park, D. Kim, and C. Kee, "Fully automatic taxiing, takeoff and landing of a uav using a

single-antenna gps receiver only," in *Control, Automation and Systems, 2007. ICCAS'07. International Conference on.* IEEE, 2007, pp. 821–825.

[6] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Visually guided landing of an unmanned aerial vehicle," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 3, pp. 371–380, 2003.

[7] S. Lee and J.-B. Song, "Mobile robot localization using infrared light reflecting landmarks," in *Control, Automation and Systems, 2007. ICCAS'07. International Conference on.* IEEE, 2007, pp. 674–677.

[8] Z.-F. Yang and W.-H. Tsai, "Using parallel line information for vision-based landmark location estimation and an application to automatic helicopter landing," *Robotics and Computer-Integrated Manufacturing*, vol. 14, no. 4, pp. 297–306, 1998.

[9] Y. Bi and H. Duan, "Implementation of autonomous visual tracking and landing for a low-cost quadrotor," *Optik-International Journal for Light and Electron Optics*, vol. 124, no. 18, pp. 3296–3300, 2013.

[10] S. Lange, N. Sünderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments," in *Advanced Robotics, 2009. ICAR 2009. International Conference on.* IEEE, 2009, pp. 1–6.

[11] FLIR, "Flir home page," [http://www.flir.ca/home/], 2016.

[12] G. Xu, Y. Zhang, S. Ji, Y. Cheng, and Y. Tian, "Research on computer vision-based for uav autonomous landing on a ship," *Pattern Recognition Letters*, vol. 30, no. 6, pp. 600–605, 2009.

[13] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle," *Journal of intelligent & robotic systems*, vol. 61, no. 1-4, pp. 221–238, 2011.

[14] O. A. Yakimenko, I. I. Kaminer, W. J. Lentz, and P. Ghyzel, "Unmanned aircraft navigation for shipboard landing using infrared vision," DTIC Document, Tech. Rep., 2002.

[15] J. Banks, M. Pachano, L. Thompson, and D. Hanny, *RFID Applied*. John Wiley & Sons, Inc., 2007.

[16] M. Kim and N. Y. Chong, "Rfid-based mobile robot guidance to a stationary target," *Mechatronics*, vol. 17, no. 4, pp. 217–229, 2007.

[17] S. Park and S. Hashimoto, "Autonomous mobile robot navigation using passive rfid in indoor environment," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 7, pp. 2366–2373, 2009.

[18] SensorWiki, "Sensorwiki - ultrasound," [http://www.sensorwiki.org/doku.php/sensors/ultrasound], 2016.

[19] S. S. Ghidary, T. Tani, T. Takamori, and M. Hattori, "A new home robot positioning system (hrps) using ir switched multi ultrasonic sensors," in

Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on, vol. 4. IEEE, 1999, pp. 737–741.

[20] T. Mwakibinga and J. Lee, "Altitude control of an unmanned aerial vehicle using a binaural bat echolocation system," RITE Technical Reports, Tech. Rep., 2005.

[21] E. Orhan, "Bayesian inference: Particle filtering," [https://www.bcs.rochester.edu/people/robbie/jacobslab/cheat_sheet/particle-filtering.pdf], 2012.

[22] F. Caballero, L. Merino, I. Maza, and A. Ollero, "A particle filtering method for wireless sensor network localization with an aerial robot beacon," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 596–601.

[23] B. Ludington, E. Johnson, and G. Vachtsevanos, "Augmenting uav autonomy," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 63–71, 2006.

[24] R. G. Brown and P. Y. C. Hwang, *Introduction to random signals and applied Kalman filtering : with MATLAB exercises*. John Wiley & Sons, Inc., 2012.

[25] J. Wang, M. Garratt, A. Lambert, J. J. Wang, S. Han, and D. Sinclair, "Integration of gps/ins/vision sensors to navigate unmanned aerial vehi-

cles," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, pp. 963–970, 2008.

[26] T. J. Ross, *Fuzzy Logic with Engineering Applications*. John Wiley & Sons, Inc., 2004.

[27] X. Le, "Fire detection robot using type-2 fuzzy logic sensor fusion," Ph.D. dissertation, Université d'Ottawa/University of Ottawa, 2015.

[28] H.-H. Lin, C.-C. Tsai, and J.-C. Hsu, "Ultrasonic localization and pose tracking of an autonomous mobile robot via fuzzy adaptive extended information filtering," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 9, pp. 2024–2034, 2008.

[29] G. W. Ng, *Intelligent Systems - Fusion, Tracking and Control*. Research Studies Press Ltd, 2003.

[30] J. Racz and A. Dubrawski, "Artificial neural network for mobile robot topological localization," *Robotics and autonomous systems*, vol. 16, no. 1, pp. 73–80, 1995.

[31] B. H. Kaygisiz, A. M. Erkmen, and I. Erkmen, "Gps/ins enhancement using neural networks for autonomous ground vehicle applications," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2003, pp. 3763–3768.

[32] R. Prasad and M. Ruggieri, *Applied Satellite Navigation Using GPS, GALILEO, and Augmentation Systems*. Artech House, 2005.

[33] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, and D. Nakazawa, "Introduction," in *Autonomous Flying Robots*. Springer, 2010, pp. 1–29.

[34] S. International, "Automated driving," [https://www.sae.org/misc/pdfs/automated_driving.pdf], 2014.

[35] Vectornav, "Inertial measurement units and inertial navigation," [http://www.vectornav.com/support/library/imu-and-ins], 2016.

[36] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.

[37] C. S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 1720–1727.

[38] M. Kim, H. W. Kim, and N. Y. Chong, "Automated robot docking using direction sensing rfid," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 4588–4593.

[39] D. Dardari, N. Decarli, A. Guerra, and F. Guidi, "The future of ultra-wideband localization in rfid," in *RFID (RFID), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–7.

[40] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[41] H. Ku, "Notes on the use of propagation of error formulas," 1966.

[42] V. Kumar, "Integration of inertial navigation system and global positioning system using kalman filtering," Ph.D. dissertation, INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY MUMBAI, 2004.

[43] P. Aggarwal, Z. Syed, X. Niu, and N. El-Sheimy, "A standard testing and calibration procedure for low cost mems inertial sensors and units," *The Journal of Navigation*, vol. 61, no. 2, pp. 323–336, 2008.

[44] C. Network, "Length of a degree of latitude and longitude calculator," [http://www.csgnetwork.com/degreelenllavcalc.html], 2011.

[45] J. D. Barton, "Fundamentals of small unmanned aircraft flight," *Johns Hopkins APL technical digest*, vol. 31, no. 2, pp. 132–149, 2012.

[46] G. Welch and G. Bishop, "An introduction to the kalman filter," *Proceedings of the Siggraph Course, Los Angeles*, 2001.

[47] Realterm, "Terminal software," [https://realterm.sourceforge.io/], 2016.

[48] Google, "Google maps - gps coordinates, latitude and longitude," [https://www.gps-coordinates.net/], 2017.

[49] G. Conte and P. Doherty, "An integrated uav navigation system based on aerial image matching," in *Aerospace Conference, 2008 IEEE*. IEEE, 2008, pp. 1–10.

[50] S. Recoskie, "Autonomous hybrid powered long ranged airship for surveillance and guidance," Ph.D. dissertation, Université d'Ottawa/University of Ottawa, 2014.

APPENDICES

Appendix A

Test Results

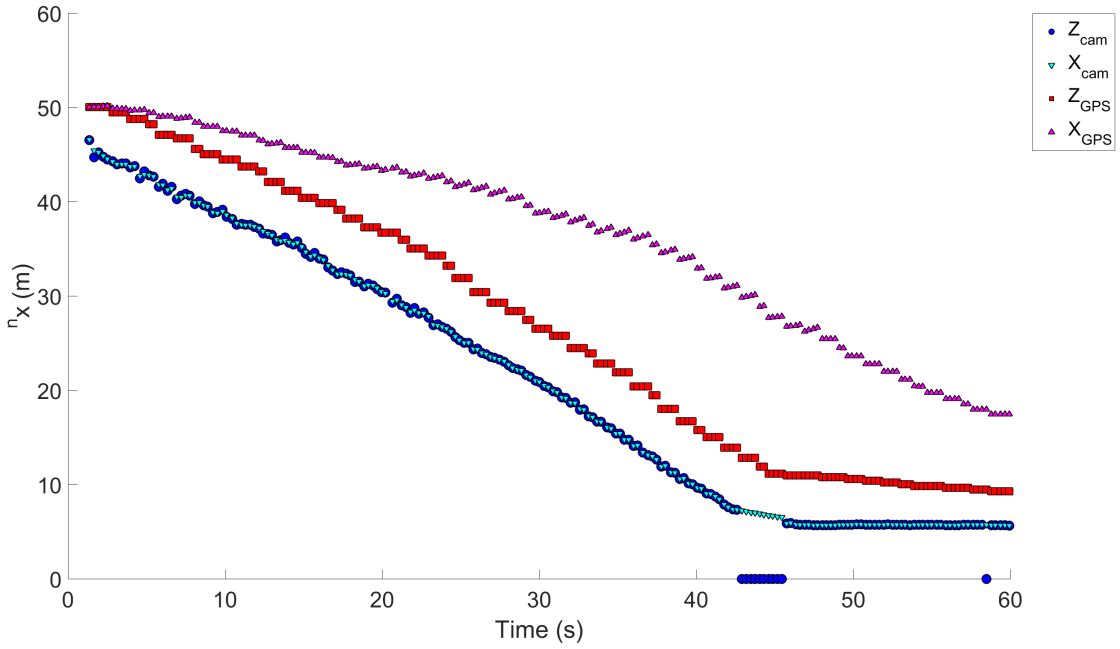


Figure A.1: Park test #Slalom3, ${}^{n}x$

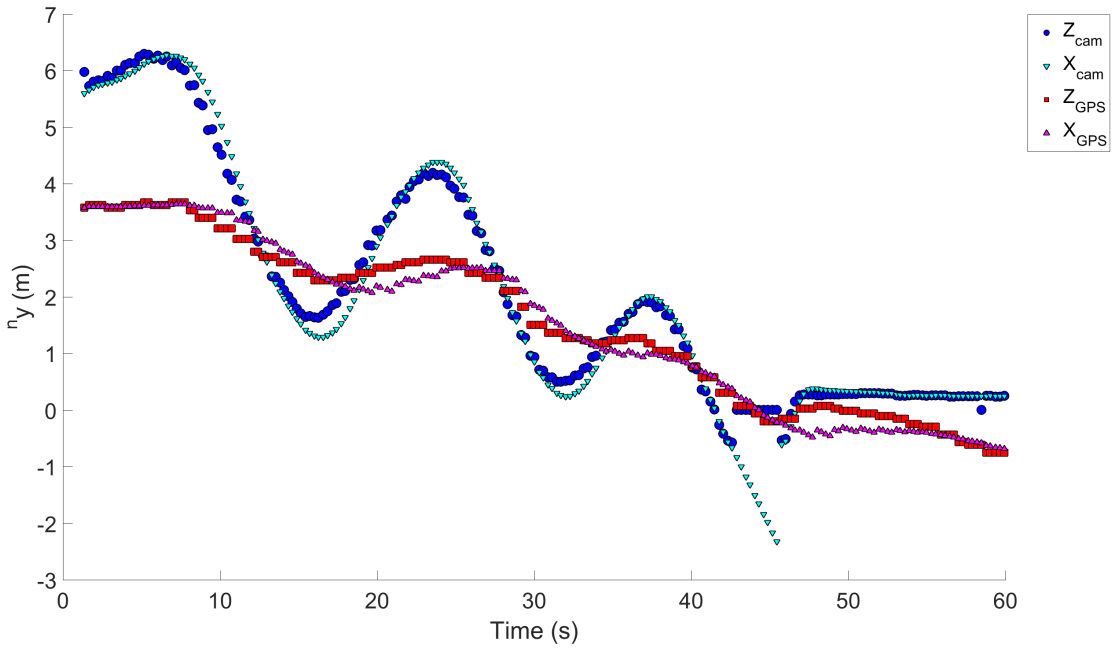


Figure A.2: Park test #Slalom3, ${}^{n}y$

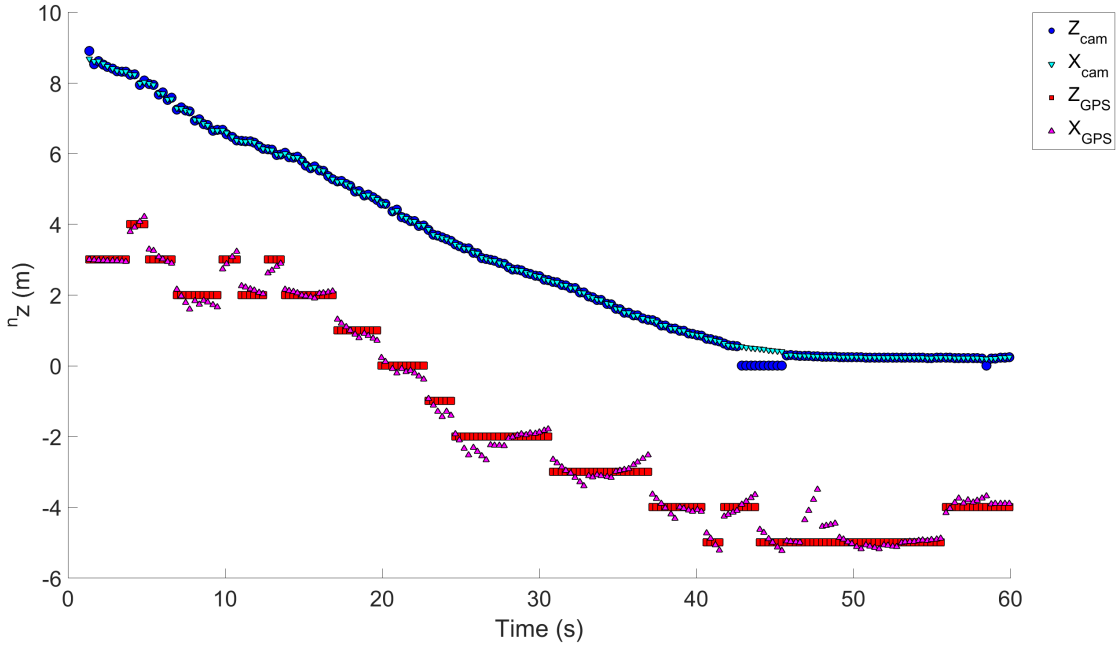


Figure A.3: Park test #Slalom3, ${}^{n}z$

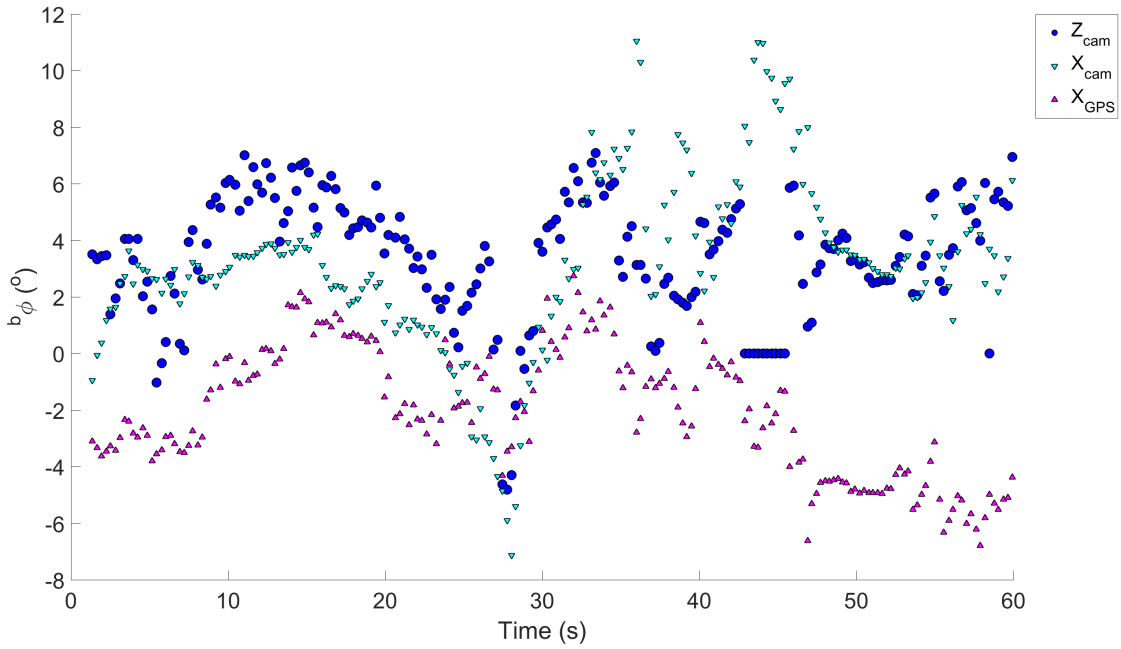


Figure A.4: Park test #Slalom3, ${}^{b}\phi$

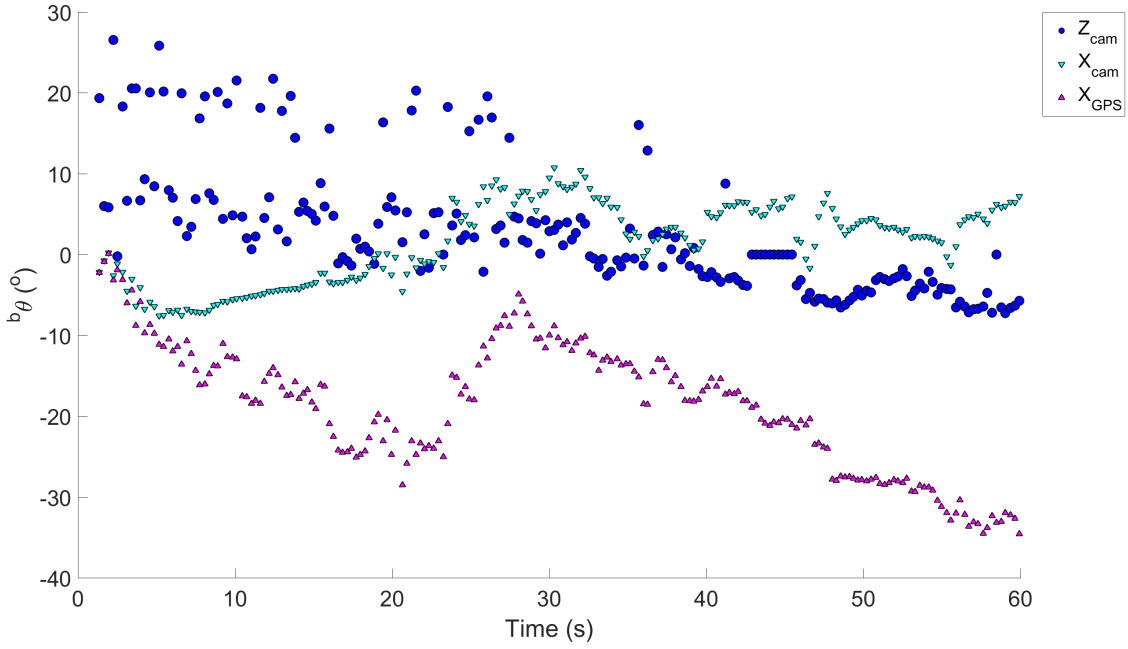


Figure A.5: Park test #Slalom3, ${}^{b}\theta$

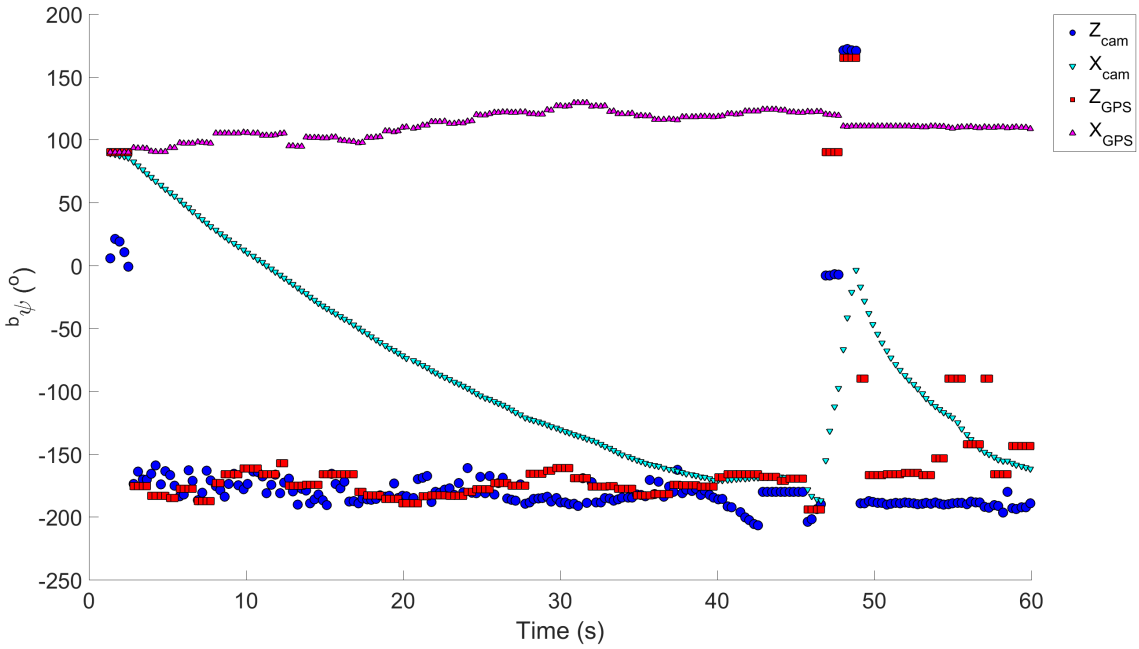


Figure A.6: Park test #Slalom3, ${}^{b}\psi$

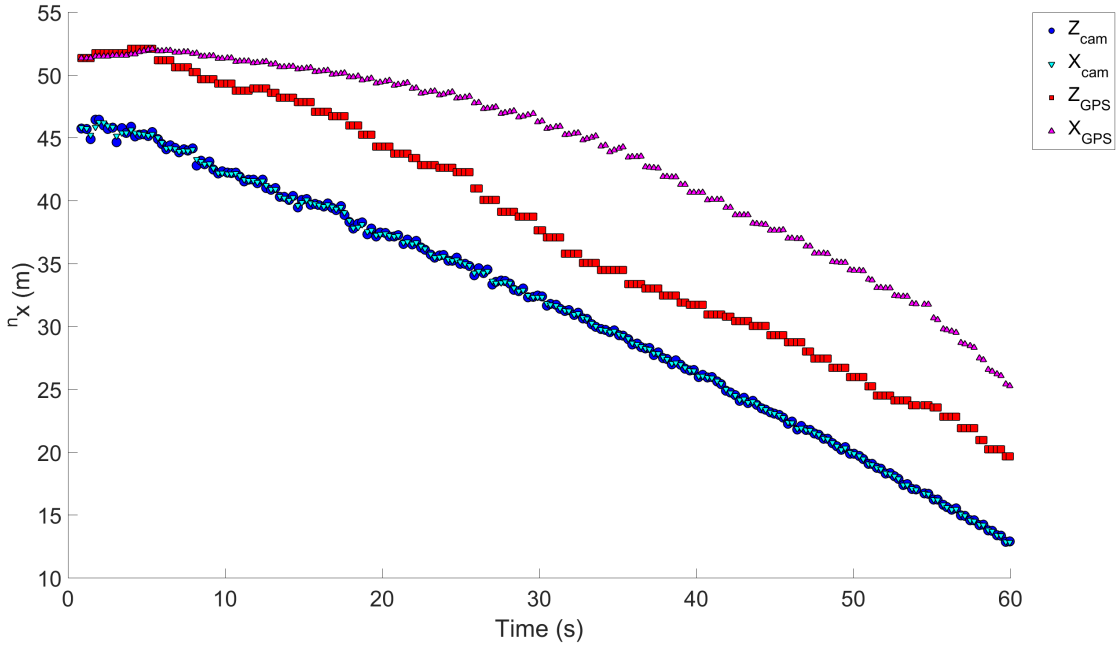


Figure A.7: Park test #Wind2, ${}^{n}x$

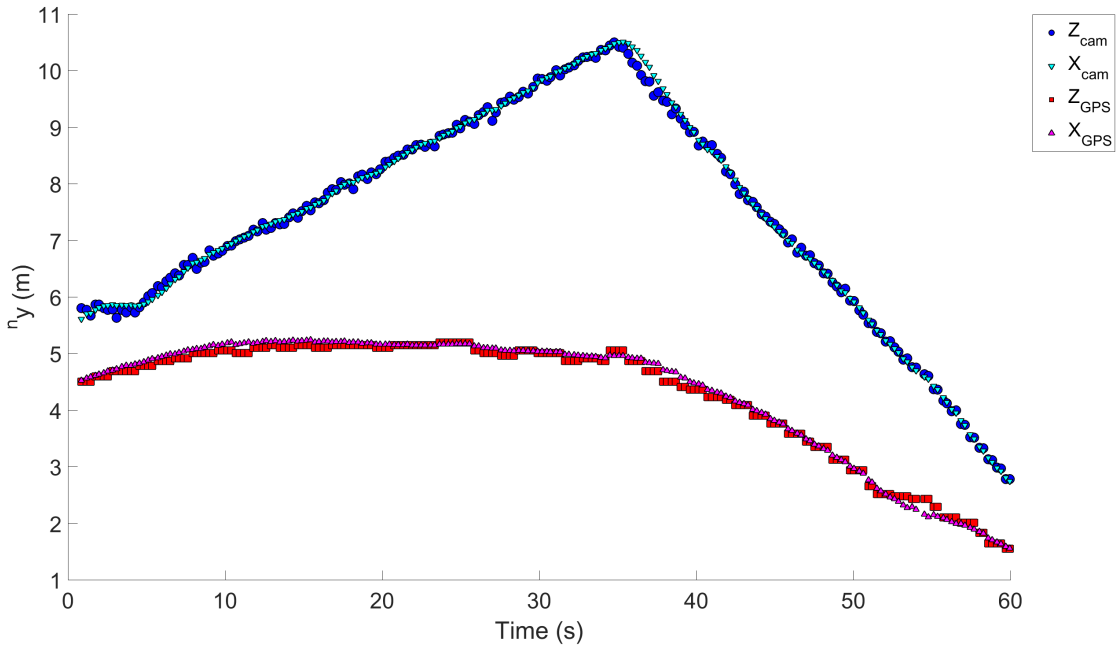


Figure A.8: Park test #Wind2, ${}^{n}y$

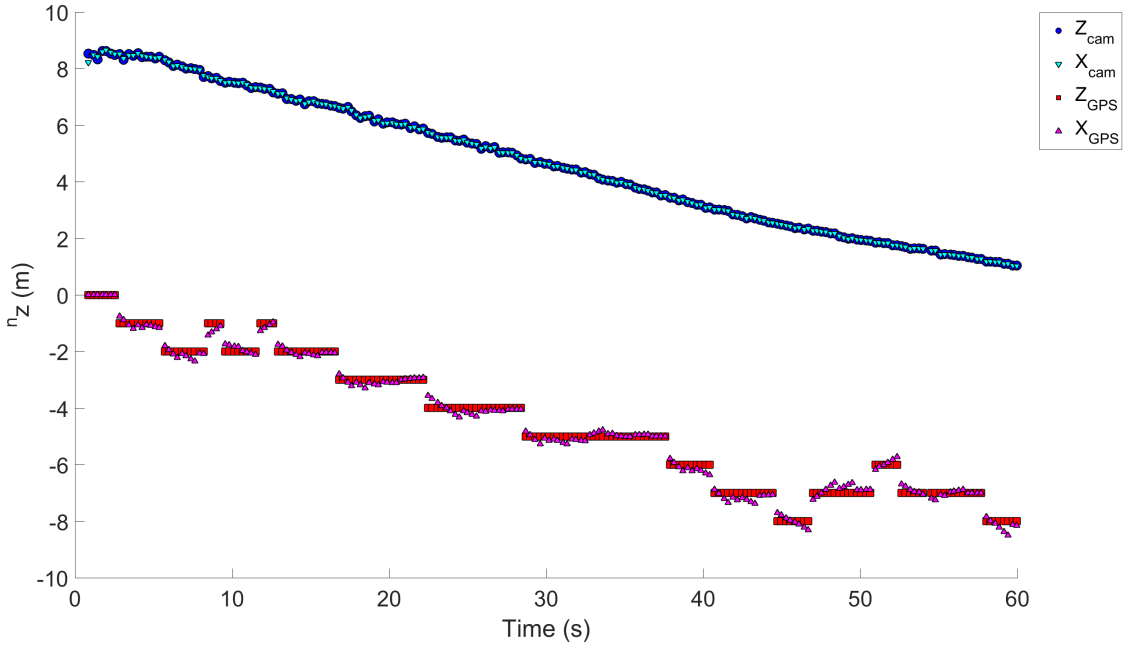


Figure A.9: Park test #Wind2, ${}^{n}z$

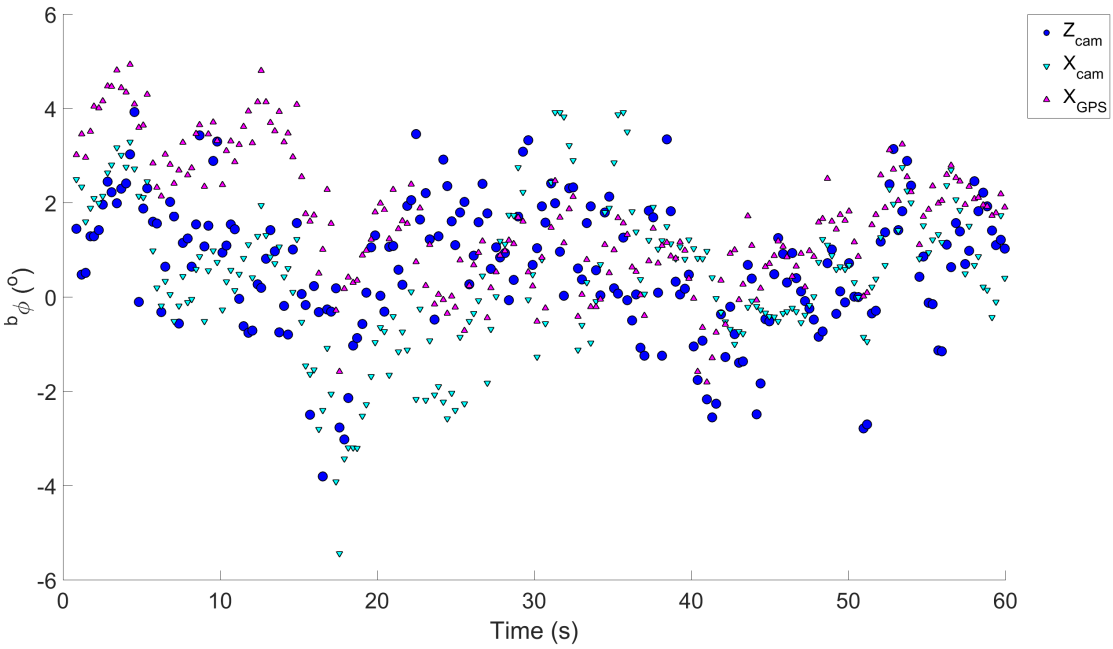


Figure A.10: Park test #Wind2, ${}^{b}\phi$

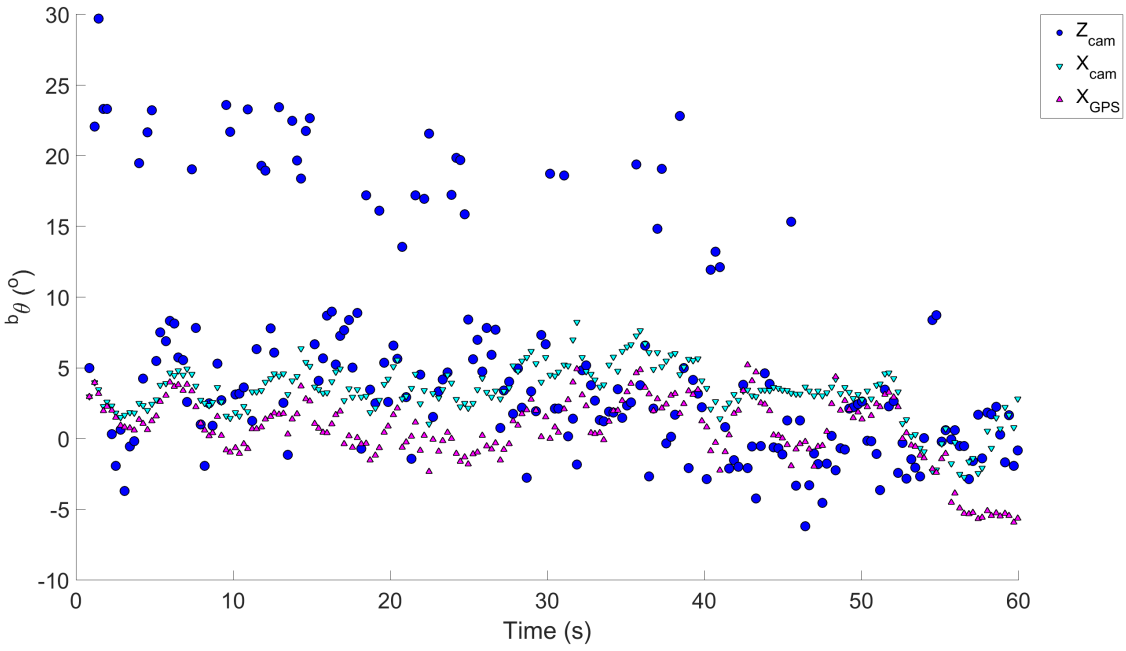


Figure A.11: Park test #Wind2, ${}^{b}\theta$

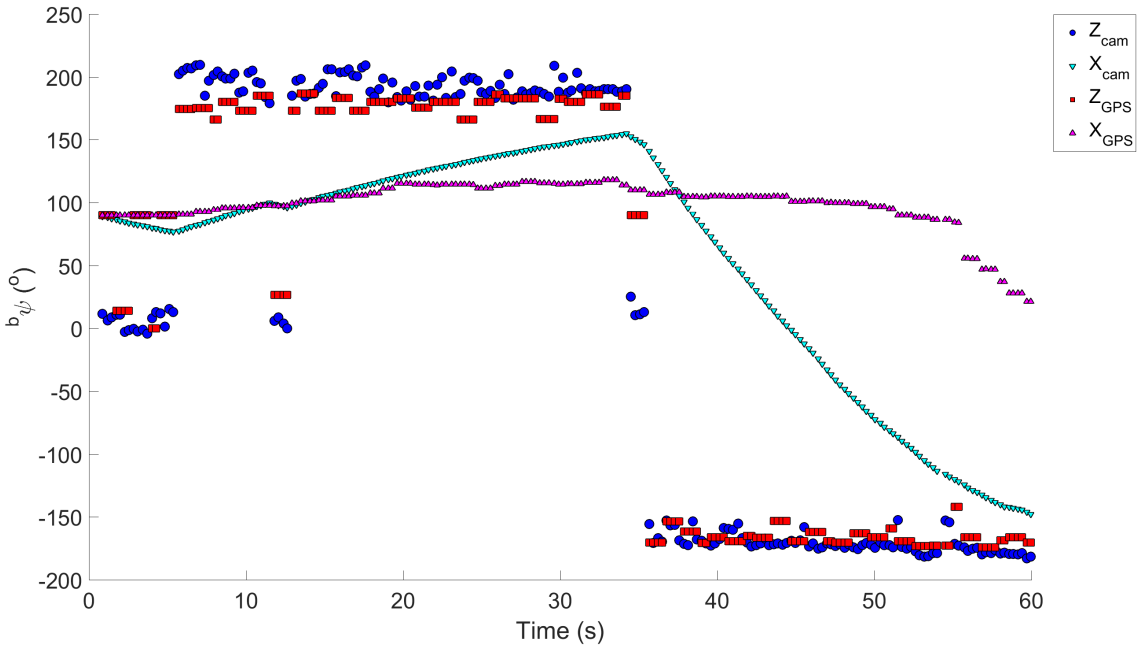


Figure A.12: Park test #Wind2, ${}^{b}\psi$

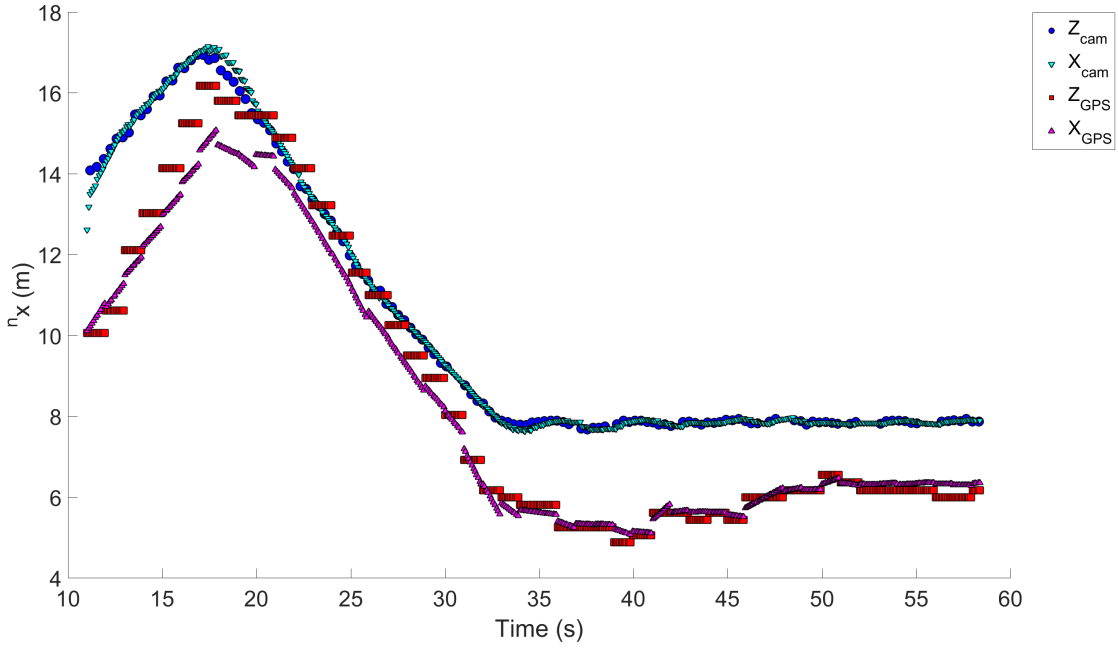


Figure A.13: Building test #Direct6, ${}^{n}x$

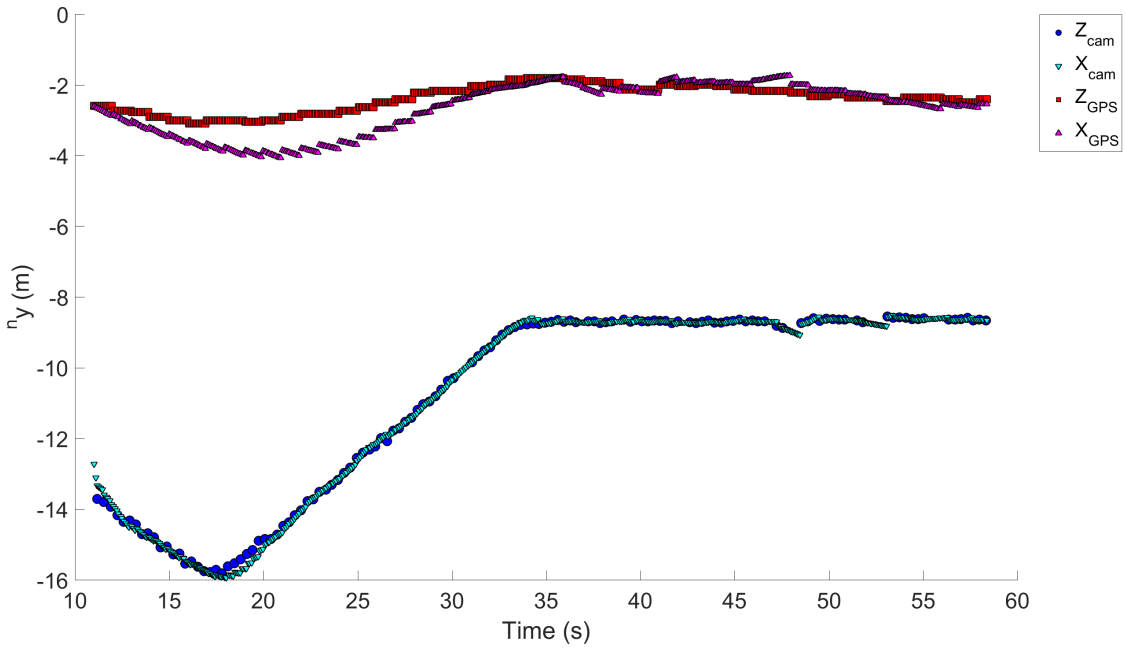


Figure A.14: Building test #Direct6, ${}^{n}y$

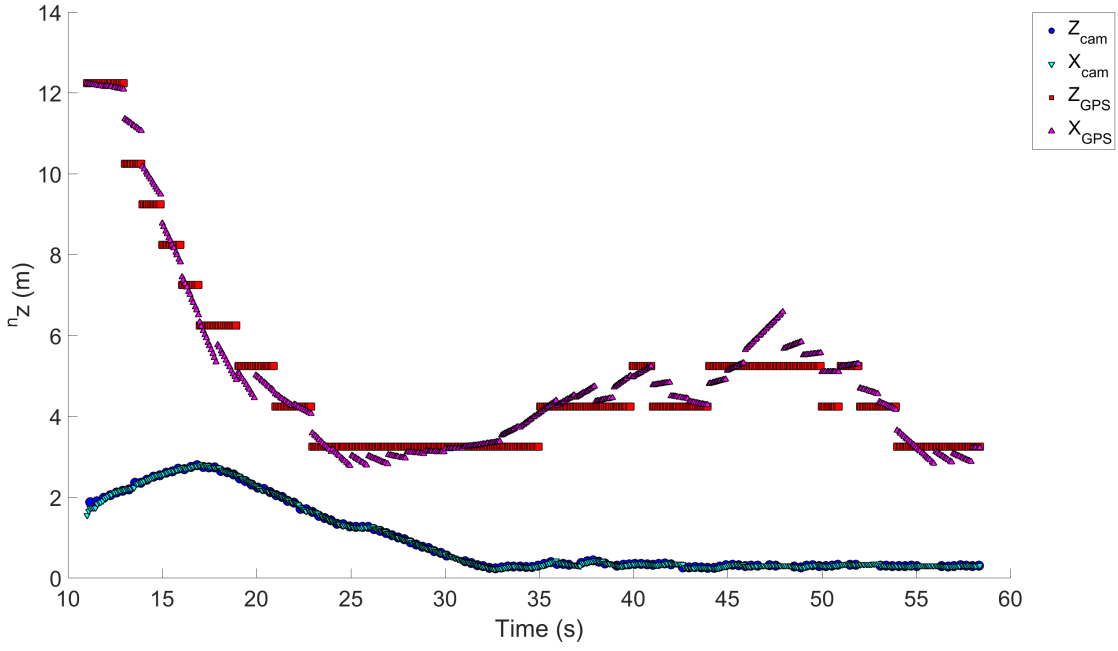


Figure A.15: Building test #Direct6, ${}^{n}z$

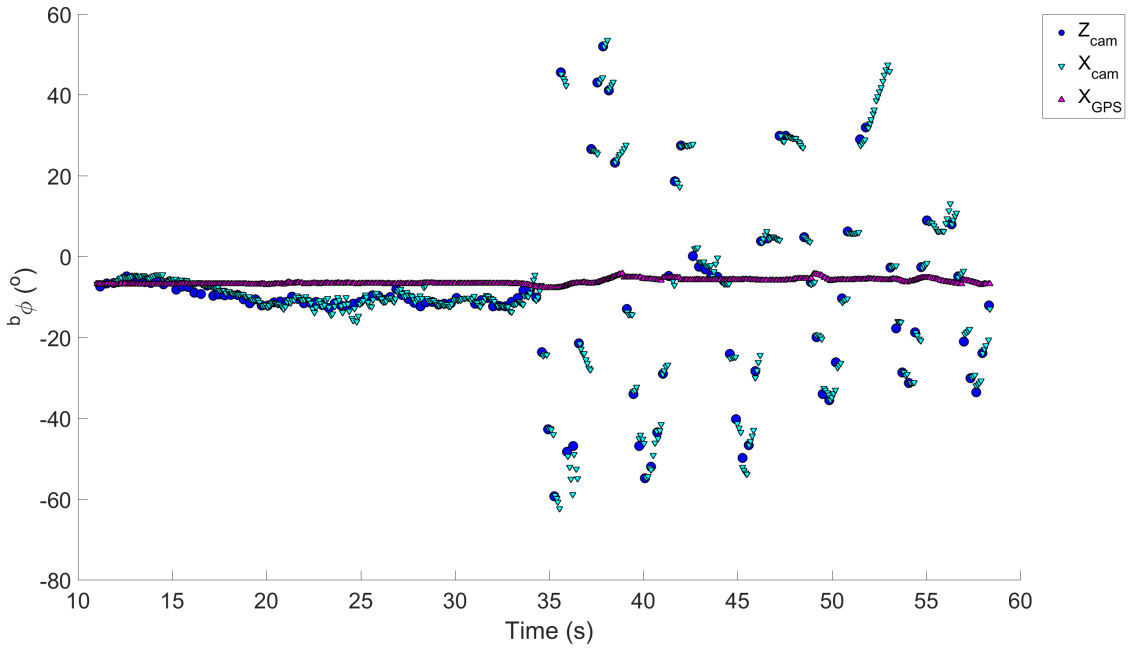


Figure A.16: Building test #Direct6, ${}^{b}\phi$

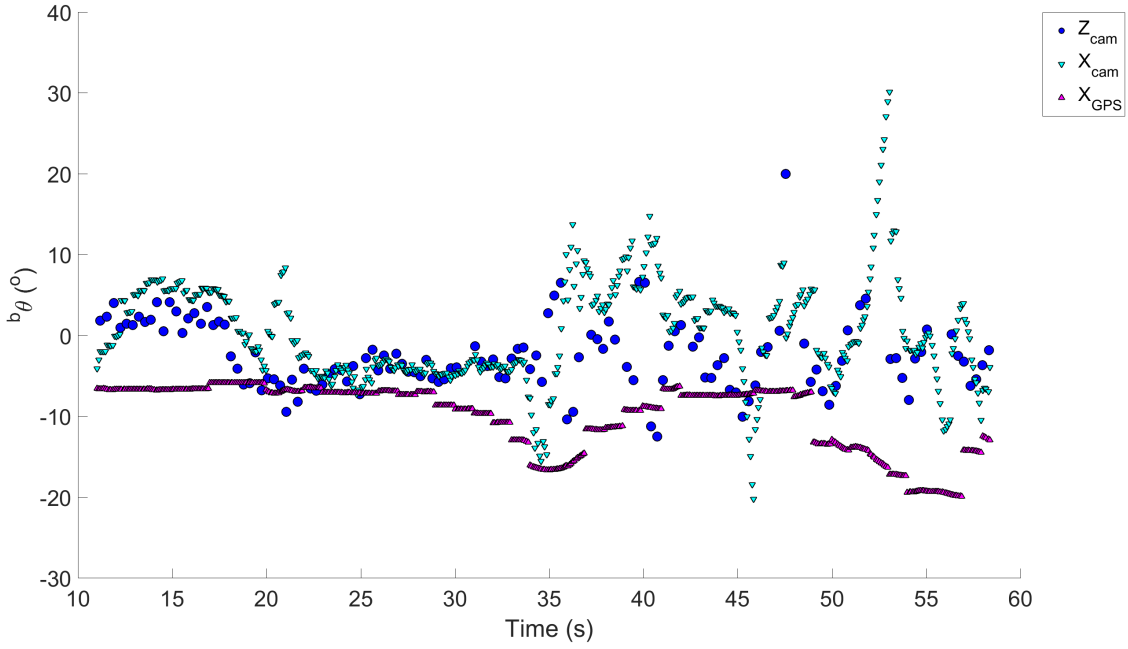


Figure A.17: Building test #Direct6, ${}^{b}\theta$

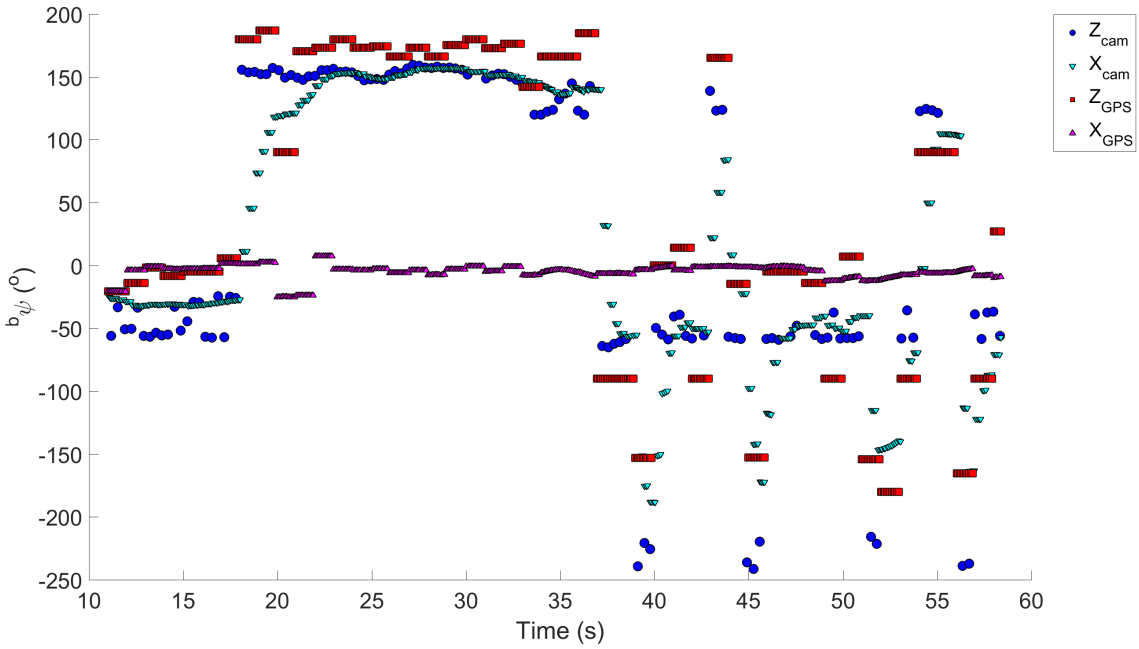


Figure A.18: Building test #Direct6, ${}^{b}\psi$

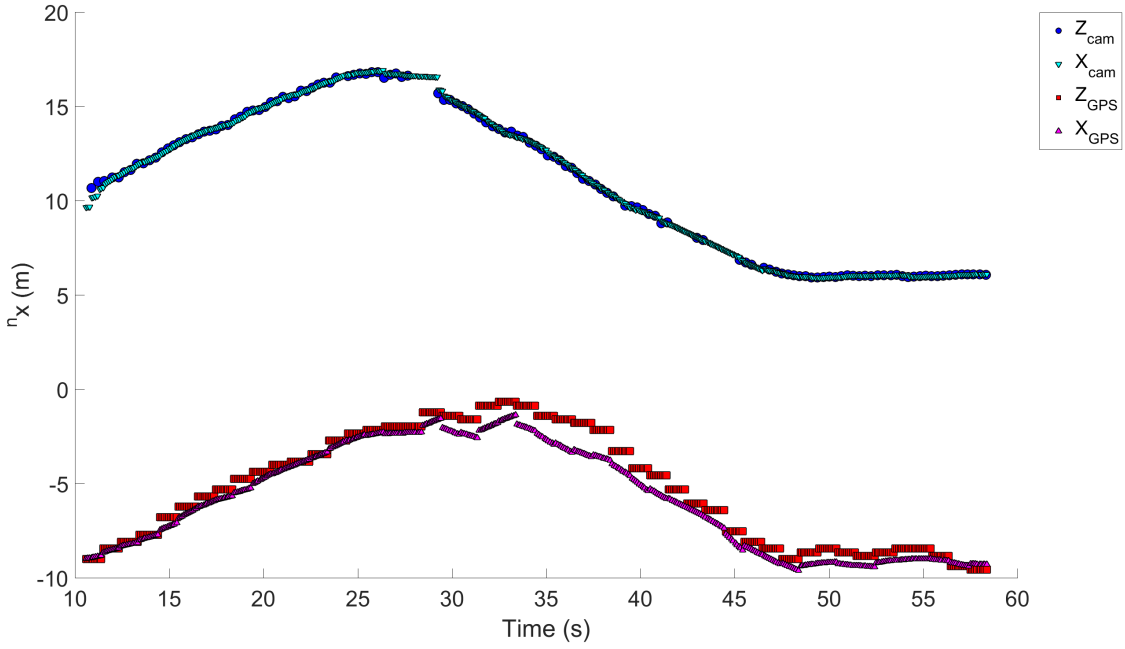


Figure A.19: Building test #Direct10, ${}^{n}x$

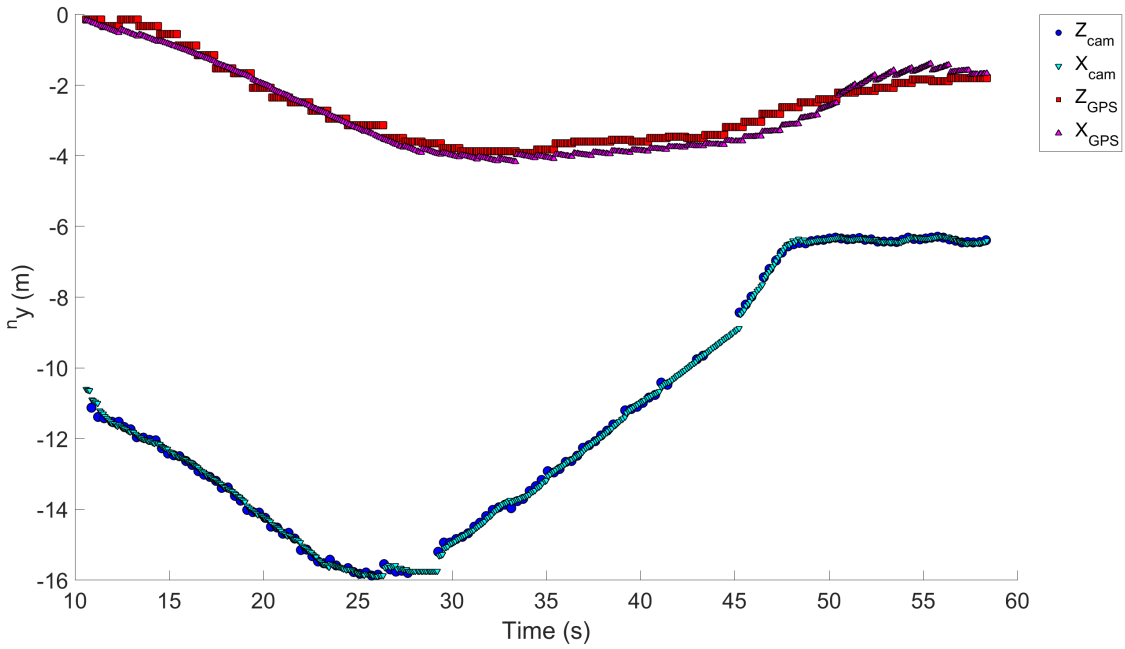


Figure A.20: Building test #Direct10, ${}^{n}y$

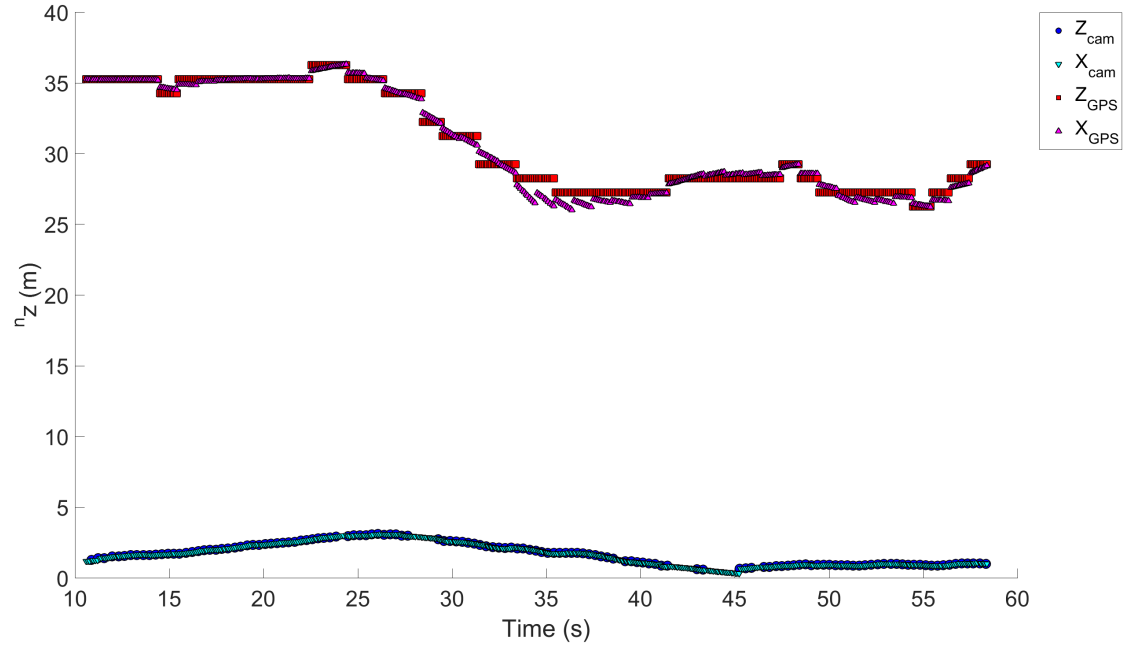


Figure A.21: Building test #Direct10, ${}^{n}z$

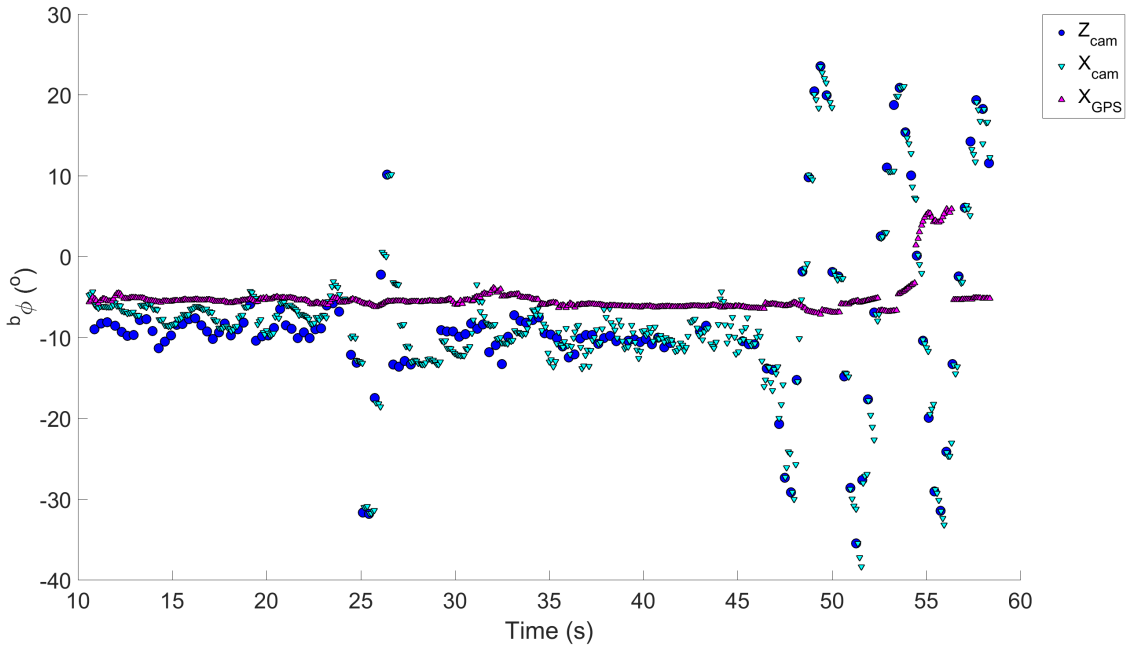


Figure A.22: Building test #Direct10, ${}^{b}\phi$

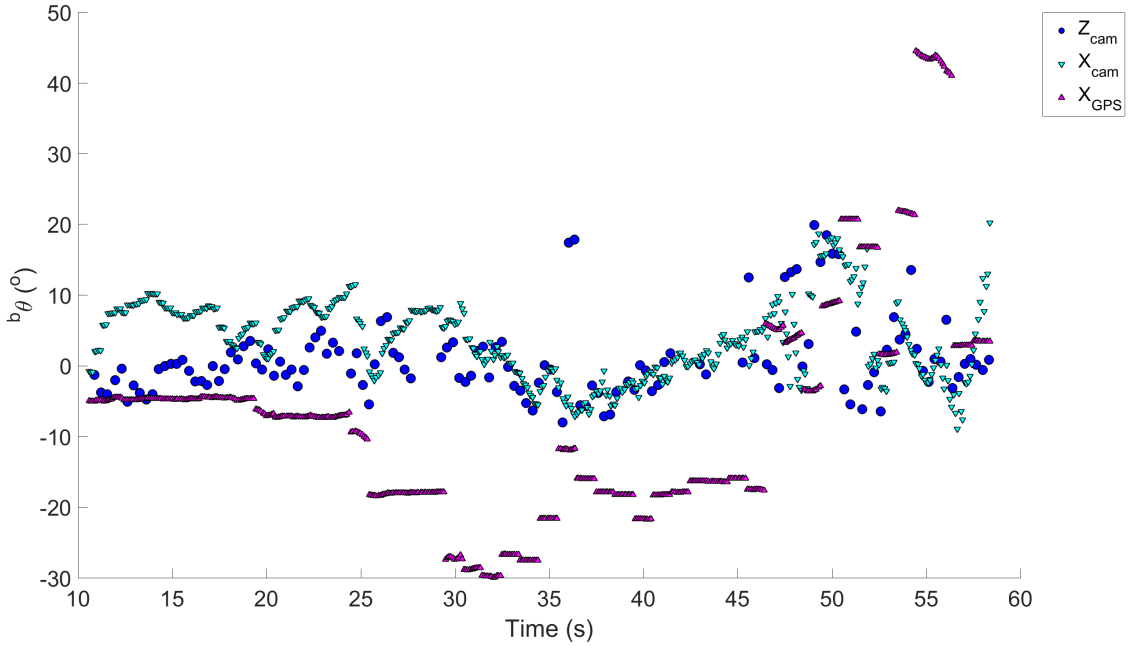


Figure A.23: Building test #Direct10, ${}^{b}\theta$

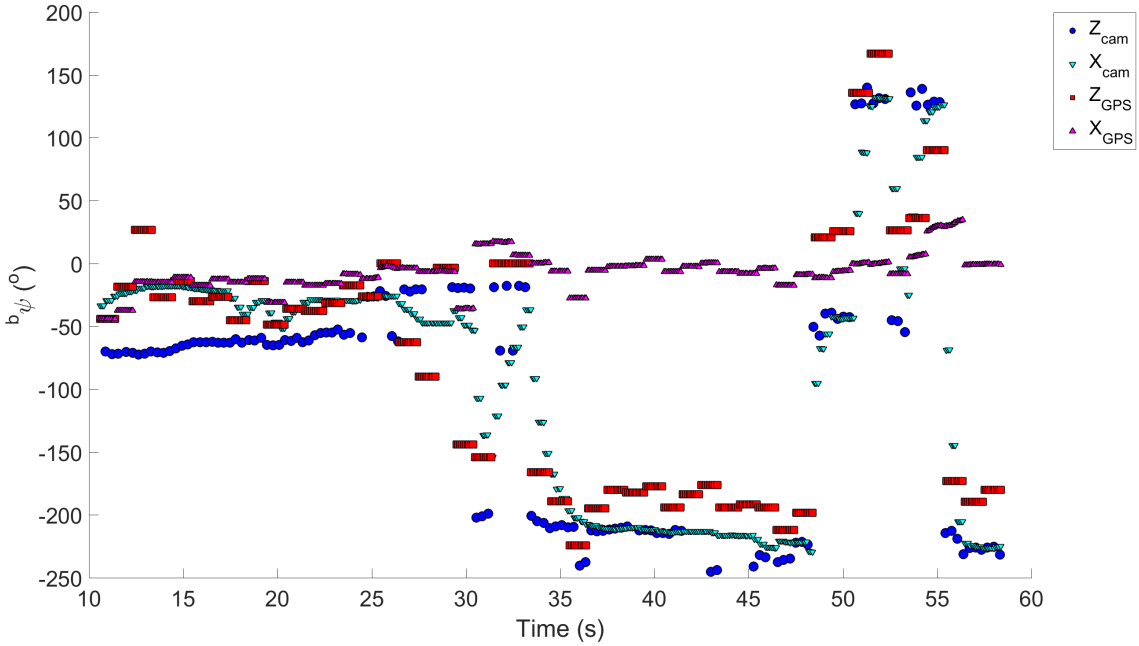


Figure A.24: Building test #Direct10, $^{b}\psi$