Vision-based Estimation and Control of Airdrop Vehicles for Aerial Deployment of Sensor Networks

by

Hyungil Ahn

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

Author ....
Department of Mechanical Engineering
May 7, 2004

Certified by ....
John J. Deyst
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by ....
Haruhiko Asada
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by ....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

Vision-based Estimation and Control of Airdrop Vehicles for Aerial Deployment of Sensor Networks

by

Hyungil Ahn

Submitted to the Department of Mechanical Engineering
on May 7, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Recent experiences indicate there are limitations to the surveillance capability of high-value long-distance standoff ISR (Intelligence, Surveillance, Reconnaissance) platforms such as the latest observation satellites and unmanned aircraft. The high-tech systems in the platforms are often influenced by weather conditions or defeated by low-tech adversaries such as foliage, caves and concealment. These limitations indicate that close-in, on-the-ground sensor networks are essential parts for powerful ISR capability.

However, it is hard to place ground sensors at planned specific locations. For example, certain sensor nodes may be required to be located on rooftops or tree canopies. Therefore, it is critical to deliver the sensors with high precision and reliability.

The current UAV (Unmanned Aerial Vehicle) technologies may give a key solution to sensor deployment because they have shown the capability for delivering a variety of payloads. The delivery utilizing UAVs may also minimize cost and complexity which are critical factors in real missions where sensor networks may consist of hundreds of sensors.

In this thesis, A way of utilizing UAVs for precision delivery of sensor nodes is suggested, a prototype of a precision airdrop vehicle (PDV) is suggested, and a vision-based estimation and control system design, using both an IMU (Inertial Measurement Unit) and a vision system to achieve precision airdrop deployment is described. The design was also tested in a hardware-in-the-loop-simulation (HILSIM) framework. Good performance and reliability of the design were demonstrated in HILSIM tests.

From HILSIM tests, it was shown that the PDV launched from a carrier UAV at a speed of about 20 m/sec and at an altitude of about 350 meters, could be guided to a target point with a maximum error of 5 meters, if the launching position could be estimated to within a maximum error of 50 meters from the ideal launching position. This implies that the exact target position and the exact launching position are not necessary for precision node delivery.

Thesis Supervisor: John J. Deyst
Title: Professor of Aeronautics and Astronautics

Thesis Supervisor: Haruhiko Asada
Title: Professor of Mechanical Engineering

Acknowledgments

First of all, I wish to record my deep gratitude to my advisor Prof. John Deyst for giving me the opportunity to do researches in this project, and for his full support and consistent guidance.

I am sincerely grateful to Sean George in Draper Laboratory, for his many constructive inputs and great help. I am also grateful to Dr. Brent D. Appleby, Draper Laboratory, for introducing the project to MIT from Draper Laboratory.

I would like to express my deep gratitude to Prof. Harry Asada for showing a deep interest in the research and providing me with kindly comments on my thesis.

I am deeply indebted to the project team members. I wish to thank Sanghyuk Park, who gave me a lot of insights into the problems and significant comments. With the help of his great skill in leading the team and designing the airdrop vehicle, I was able to focus on my own research part. I also thank Joshua Torgerson for his great help. He designed the parachute deployment mechanism, which was one of the most critical components in the airdrop vehicles. Moreover, he was always with me during a lot of flight tests and drop tests. I am also grateful to Damien Jourdan for always having a warm heart for me and giving me helpful advice. Thank Alexander Omelchenko and Damian Toohey for their technical advice and good friendship. Thank Anand Srinivas for always giving me nice smiles.

I am thankful to Sung-il, Yeun-woo, and Jae-kyu for their genuine friendship. I had a lot of fun with them for last years at MIT. I also thank Kyoung-soo for his concern for me and priceless advice. I would also like to express my sincere gratitude to Dong-hoon, Hong-rock, Won-jae, Jin-yeop, Young-myoung, Jeong-dae, Young-ho, Seong-min, Soo-kyum, Ki-hoon, Hyung-ki, and Hyun-seok for their consistent friendship with me, even though they are in Korea.

I wish to express my respect for Prof. Youdan Kim at Seoul National University in Korea. He gave me a great deal of support and encouragement.

I am deeply grateful to my family in Korea. My mother and father continuously gave me endless support and love. My sister and brother-in-law, Ha-nul and Ki-hak showed hearty interest in my work all the time. Ku-bin, my cute nephew, always pleased me. I would also like to show my deep gratitude to my mother-in-law, my father-in-law, and my brother-in-law Min-kyu for their full support and uniform concern for me.

Last but by no means least, I am most grateful to my lovely wife, Yun-jung for her patience and tolerance. She always encouraged me whenever I was disappointed in my research results. With her invaluable support and thoughtful consideration, I was able to accomplish the work. I love you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Overview

This chapter first describes the background and objective of the research on precision airdrop for aerial deployment of sensor networks. It then illustrates our approach and the challenges of the project. Finally, it gives a brief outline of the thesis.

1.1 Background and Objective

Recent experiences indicate there are limitations to the surveillance capability of high-value long-distance standoff ISR (Intelligence, Surveillance, Reconnaissance) platforms. For example, although the latest observation satellites and unmanned aircraft have elaborate high-tech surveillance capabilities such as high-resolution vision systems, these technologies are often influenced by weather conditions or defeated by low-tech adversaries such as foliage, concealment, and caves, as shown in Figure 1-1. These limitations provide motivation for the use of close-in, on-the-ground sensor networks as indispensable tools in the powerful ISR mission.

Focused on finding a solution to these limitations, research on aerial deployment of sensor networks from standoff distances has been done under the the MIT Draper Partnership Program (MDPP). The goals of the MDPP are:

- To utilize the current UAV (Unmanned Aerial Vehicle) technologies to develop a system capable of deploying an ISR sensor network over an area of interest.
- To demonstrate new ISR capability to achieve both wide-area, long-range sensing and close-in sensing in complex terrain, all from a safe standoff distance.

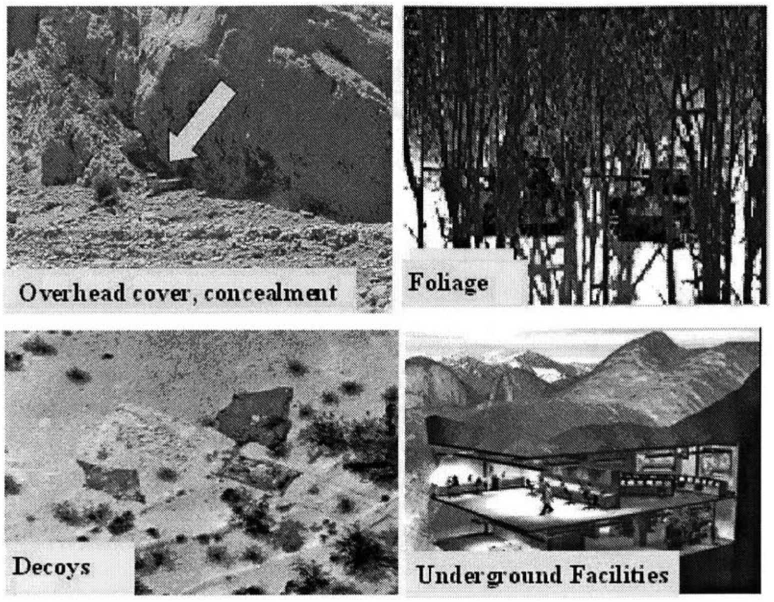


Figure 1-1: Low-tech Adversaries: Concealment, foliage, decoy, and underground facility

- To identify new concepts and key enabling technologies for insertion of sensors, vehicles, and communication networks that can be deployed from UAVs.

- To prototype a selected set of system components to demonstrate key capabilities.

Although well-planned sensor networks on the ground are essential parts for effective ISR capability, it is often hard to place the ground sensors at specific locations. For example, certain sensor nodes could be required to be delivered to small areas such as rooftops or tree canopies. Therefore, it is critical to deliver sensors with high precision and reliability.

The current UAV technologies may provide a key solution to sensor deployment because they have shown the capability to deliver a variety of payloads. The UAVs can also minimize cost and complexity, which are critical factors in real missions where sensor networks may consist of hundreds of individual nodes.

The objective of this thesis is to suggest how to utilize UAVs for the accurate delivery of sensor nodes, to present a prototype of a precision airdrop vehicle (PDV), and to design a vision-based estimation and control system to achieve precision placement.

1.2 Approach and Challenge

Several issues were considered in designing a precision airdrop vehicle (PDV).

- It should be mechanically simple to design yet capable of precision control. It is also expected to be able to maximize accuracy and reliability of precision drop at minimal cost and complexity.
- It ought to ensure minimal modification to existing UAV payload interfaces. It was initially assumed that the Predator UAV, shown in Figure 1-2, would be the nominal carrier UAV. Thus, the mechanical design should be able to take advantage of the Predator's existing payload interfaces.
- Both a low and high-g delivery platform, each with suitable control capability, is needed for various kinds of purposes.

As a result of these considerations, both a ballistic-type and a parafoil-type platform (Figure 1-3) were analyzed. These are relatively simple mechanical designs and they can be packaged so that they will be compatible with the Predator's wing mounted dispensers (Figure 1-2). Ballistic-type platforms were developed for high-g delivery, while parafoil-type platforms were devised for low-g delivery.

Parafoil-type platforms are very useful when a low descent rate is needed to safely deliver a payload to the ground. They are also desirable when a mission requires a drop from relatively low altitude or maneuvering is important. Because parafoil-type vehicles can more easily glide all the way to the target, they are capable of reaching a much larger target area than ballistic type vehicles. However, ballistic-type platforms are preferred in a mission which calls for high accuracy and where the payload is capable of surviving high-g impact.

In our approach, both ballistic-type and parafoil-type vehicles utilize vision systems for their state estimation and control. A vision system is composed of a low-cost camera, wireless NTSC video communication link, and a ground station. As a first stage, the ground station program allows a human operator to track the target image by a mouse device, and then transmits the tracked target pixel positions to the PDV on-board computer. In combination with inertial sensor data and magnetic sensor measurements, the vision-based target position measurements are used as inputs to an extended Kalman filtering algorithm.

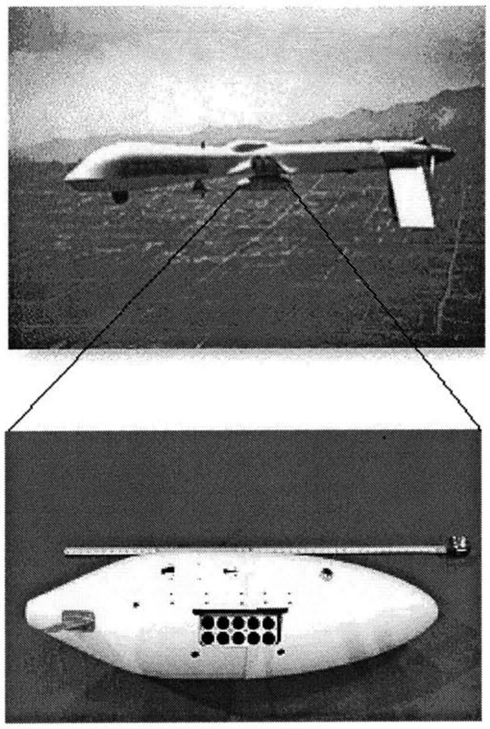


Figure 1-2: Predator and Flare Dispenser Tubes

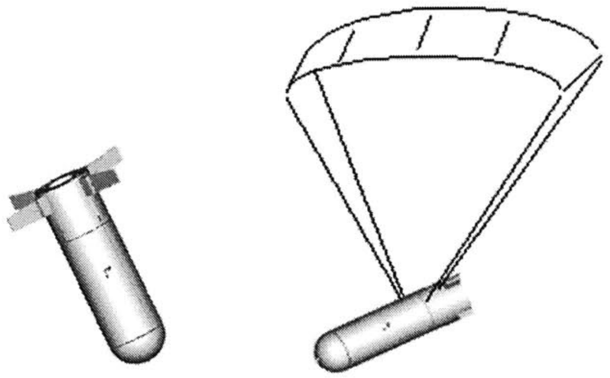


Figure 1-3: Ballistic-Type and Parafoil-Type

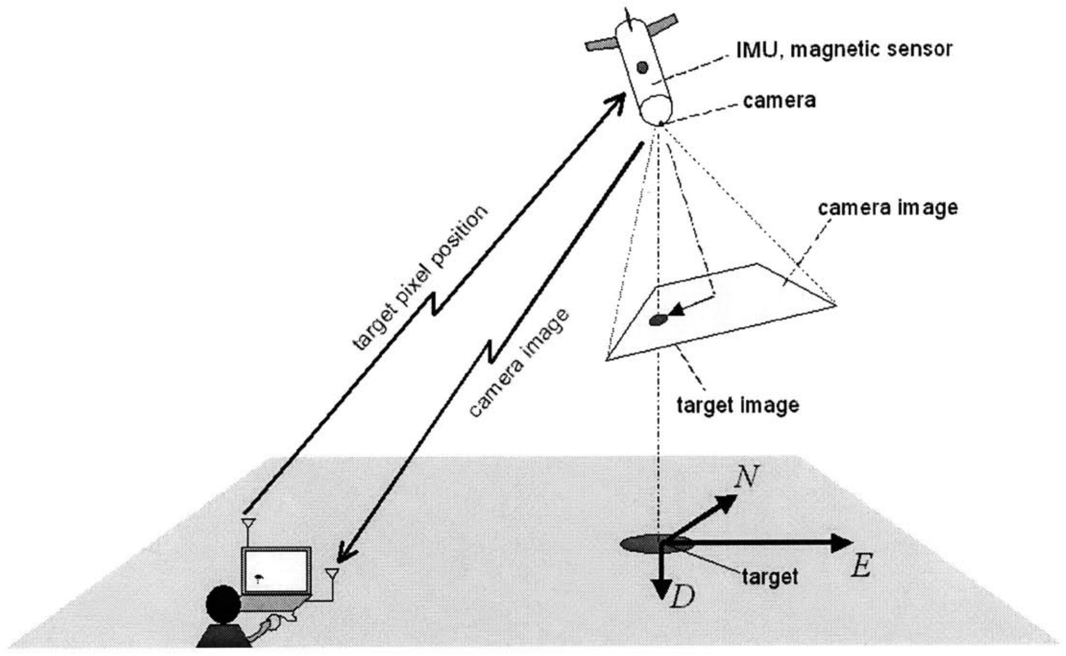


Figure 1-4: Concept of Vision System

In addition, the tracked target pixel positions are also used for the target pixel position controller. Eventually, it is anticipated that a human operator's target tracking can be replaced by an image processing module. Figure 1-4 describes the concept of our vision system.

Note that GPS (Global Positioning System) sensors are not used for our setup. The reasons are:

- The accurate GPS devices are expensive.
- The exact GPS coordinates of our target are often unknown. Moreover, it is difficult to obtain the required accuracy of 5 meters by means of GPS sensors alone.
- It may be necessary to modify or adapt the target location as a result of the operator's close-up view of the situation, as the PDV approaches the target.

This thesis focuses on ballistic-type PDVs. The research on parafoil-type PDVs are described in Damian Toohey's thesis [22]. Unless otherwise mentioned, PDVs mean ballistic-type PDVs from here in this thesis. For test purposes, the following mission scenario was devised.

- The PDV is dropped from a carrier UAV, in level flight at an altitude of about 350 meters. When launching the PDV, the carrier's speed is about 20 m/sec. The launching position can be estimated to within a maximum error of 50 meters from the ideal launching position (calculated in the condition of no wind).

- The vision system can see the target point in no greater than 3 seconds after the launch, and both vision-based state estimation and target pixel position controller start working then.

- Servo-actuated control fins control the attitude of the PDV, and can be used to guide the PDV to the target location.

- A parachute is released at an altitude of about 30 meters, to ensure that components can be reused.

- The required accuracy (maximum error) is 5 meters.

Many technical challenges were overcome to devise a precision airdrop system. Among them are:

- Mechanical design for the PDV to endure forces during the mission

- Robust parachute mechanism design

- Development of an avionics system design that includes accurate sensing and servo system, and a reliable power system

- High-bandwidth wireless network design for transmission/reception of the camera video, vision-based measurement data, and command signals

- Vision system design to allow a human operator to track the target image precisely

- Vision-based controller design to stabilize the camera image and to direct the PDV to the target

- Accurate vision-based estimation design

1.3 Thesis Outline

In this thesis, a vision-based estimation and control design using both an IMU (Inertial Measurement Unit) and a vision system to achieve precision state estimation and feedback control are presented. The design was tested in a hardware-in-the-loop-simulation (HILSIM) framework. The contents of the remaining chapters are as follows:

Chapter 2. Experimental Setup This chapter describes the experimental setup required for the precision airdrop framework which includes the precision airdrop vehicle (PDV), its avionics components, and the ground station (GS). It also illustrates the hardware-in-the-loop-simulation (HILSIM) framework and additional components for the HILSIM. It details all of the computer programs developed for the real airdrop framework and the HILSIM.

Chapter 3. Dynamics Modeling and Software Simulation This chapter describes the dynamic modeling of the PDV and briefly illustrates software simulation. It defines dynamic models used for analysis, which include the PDV rigid body equations of motion, the aerodynamic force and moment models, the servo models, the sensor models, and the wind models. It details a software simulation which contains all of the components required for the simulation of the precision airdrop platform.

Chapter 4. Estimation and Controller Design This chapter describes the state estimation algorithm, which is an extended Kalman filter that utilizes inertial sensor data, magnetic sensor measurements, and vision-based measurements. The chapter also illustrates control approaches for precision airdrop. Pitch/yaw dampers and vision-based target pixel position controllers are designed for directional control, and an axial controller is designed for axial control.

Chapter 5. Test Results This chapter details the software simulation results and the HILSIM results for several initial conditions.

Chapter 6. Conclusion This chapter summarizes the thesis and suggests future work.

Chapter 2

Experimental Setup

The precision airdrop system is essentially composed of two components: the precision airdrop vehicle (PDV) component, the ground station (GS) component. These two components interact through multiple types of wireless communication.

The precision airdrop vehicle (PDV) guidance and control system makes use of several avionics components as well as software for state estimation and control. The ground station (GS) has a very high-performance CPU and a frame grabber card for processing the received NTSC video. The GS software allows a human operator to see the camera image and track the target pixel location to guide the PDV.

Multiple types of wireless communication are employed. First, a 2.4 GHz 802.11b wireless network supporting the IPX (Internetwork packet exchange) protocol was used for communicating flight data and tracked target pixel locations between the PDV and the GS. Second, a 2.4 GHz wireless NTSC(National Television System Committee) video link served to transmit NTSC video from the vehicle camera to the GS. Third, an RF link between the RC receiver and the RC transmitter was used to control the power on-and-off of the avionics, the airdrop mechanism, and the parachute deployment mechanism. However, since frequency ranges or channels of these links were all different from each other, RF interference was not a problem.

Several additional programs, such as a flight dynamics simulation (FDS) and a camera vision simulation (CVS), were required for a hardware-in-the-loop simulation (HILSIM), which used the same avionics components and software as real airdrop tests.

This chapter will first describe the experimental setup needed for the real precision airdrop framework which includes the PDV, its avionics components, and the GS. It will then illustrate the HILSIM framework and additional components for the HILSIM in detail. Finally, it will explain all the developed computer programs such as the PDV program, the GS program, the FDS program, and the CVS program.

2.1 Precision Airdrop Vehicle

A guided, fin-stabilized projectile was chosen as our ballistic PDV prototype system (Figure 2-1). Several mechanical design requirements were:

- Enough inner space to contain a payload, including all of the avionics components
- A replaceable nose section to cushion the impact when landing on the ground and to secure the camera sensor and other avionics
- A tail section with servo-actuated fins
- A reliable parachute deployment mechanism
- A lug on the vehicle body to allow the PDV to be mounted to the test-bed aircraft before the airdrop
- light weight, simple frame for easy construction

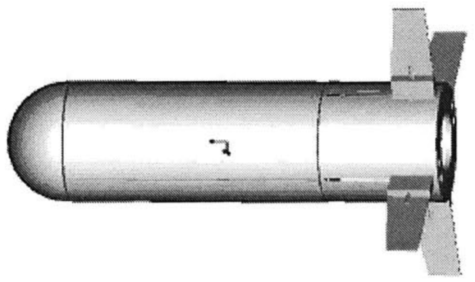


Figure 2-1: Ballistic Precision Airdrop Vehicle (PDV)

A carbon fiber core with balsa ribs, a cylindrical balsa shell, and a styrofoam spherical nose cone were used for manufacturing the PDV (Figure 2-2). The total weight of the PDV

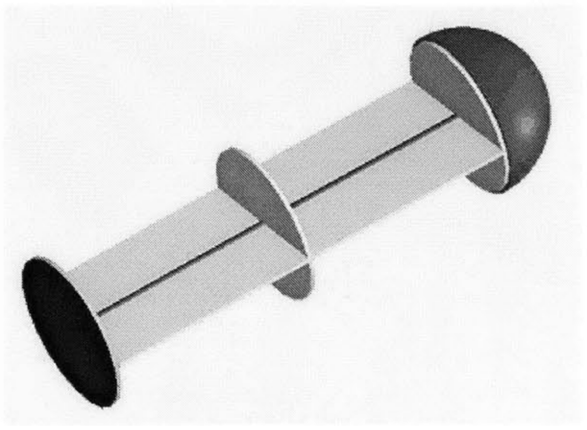


Figure 2-2: Core Structure of PDV

containing all the avionics components was 1.76 kg and the terminal velocity was assumed to be about 60 m/sec. Aerodynamic coefficients were estimated based on software analysis and wind tunnel tests of a prototype model.

The control unit in the tail section was equipped with four servo-actuated fins. The motions for directional control were generated by collective rotations of the control fins, while axial control utilized differential modes of the control fins. Axial control prevents the PDV from rotating about the vehicle's longitudinal axis so as to ensure a stabilized camera image. The PDV structure and the servo mechanism were mainly developed by Sanghyuk Park and the project team.

A parachute was used for reducing the terminal velocity when the PDV hits the ground. The parachute deployment mechanism shown in Figure 2-3 was designed to contain the parachute and release it near an altitude of 30 meters. A servo motor opened the mechanism and was remotely controlled by an human operator on the ground. The joint between the vehicle body and the parachute was chosen to endure the shock when inflating the parachute. A streamer was used to extract the parachute at release. This parachute deployment mechanism was developed by Joshua Torgerson and the project team.

2.2 Test-bed Aircraft

Our test-bed aircraft is a pusher RC aircraft which was able to support up to 40 lb of payload and has a 16 ft wingspan as shown in Figure 2-4. The PDV release mechanism in

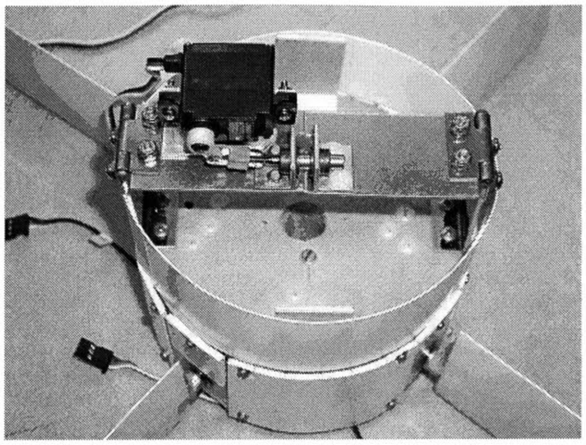


Figure 2-3: Parachute Deployment Mechanism of PDV

Figure 2-5 was designed to mount the PDV under the wing before the launch and release the PDV by remote control. Joshua Torgerson and the project team developed this release mechanism. In real airdrop tests, the launch was usually performed at an altitude of about 350 meters, which is approximately the highest altitude at which the pilot was able to safely control the test-bed RC aircraft.

2.3 Avionics and Ground Station

Various avionics components were utilized in our precision airdrop system. The precision airdrop vehicle (PDV) avionics is composed of a PC/104+ embedded computer, low-cost IMU inertial instruments, a three-axis magnetic sensor, a servo controller and servos, a low-cost camera with a 2.4 GHz wireless NTSC video transmitter, an 802.11b wireless network card, an RC receiver, a power system, and a parachute deployment mechanism.

A very high-performance portable computer served as the ground station (GS). The ground station had an 802.11b wireless network card, a wireless NTSC video receiver, a high gain antenna for 2.4 GHz RF, and a frame grabber card.

2.3.1 Precision Airdrop Avionics

Figure 2-6 shows a schematic of our precision airdrop avionics, including both the PDV avionics and the GS. Several types of wireless communications are used between these two components.

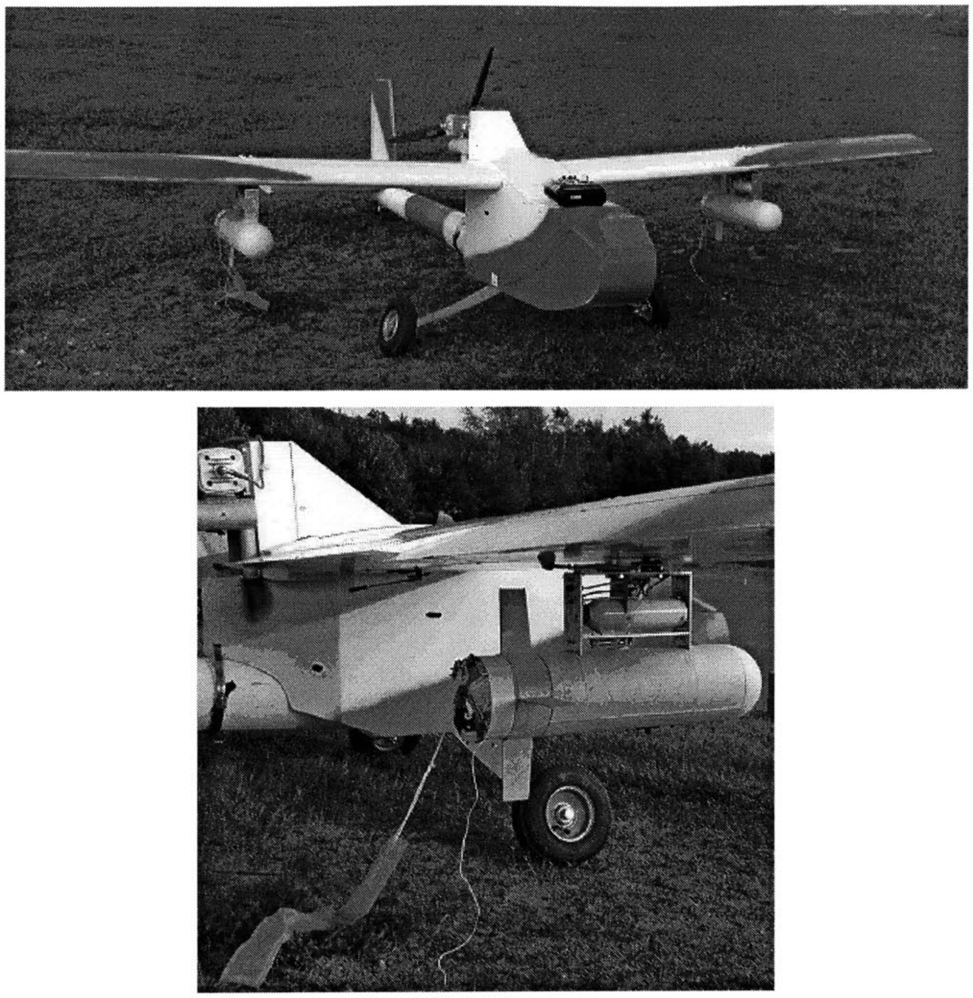


Figure 2-4: Test-bed Aircraft

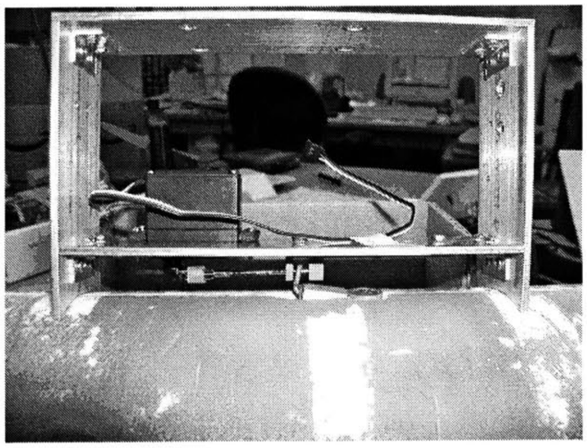


Figure 2-5: Release Mechanism of PDV

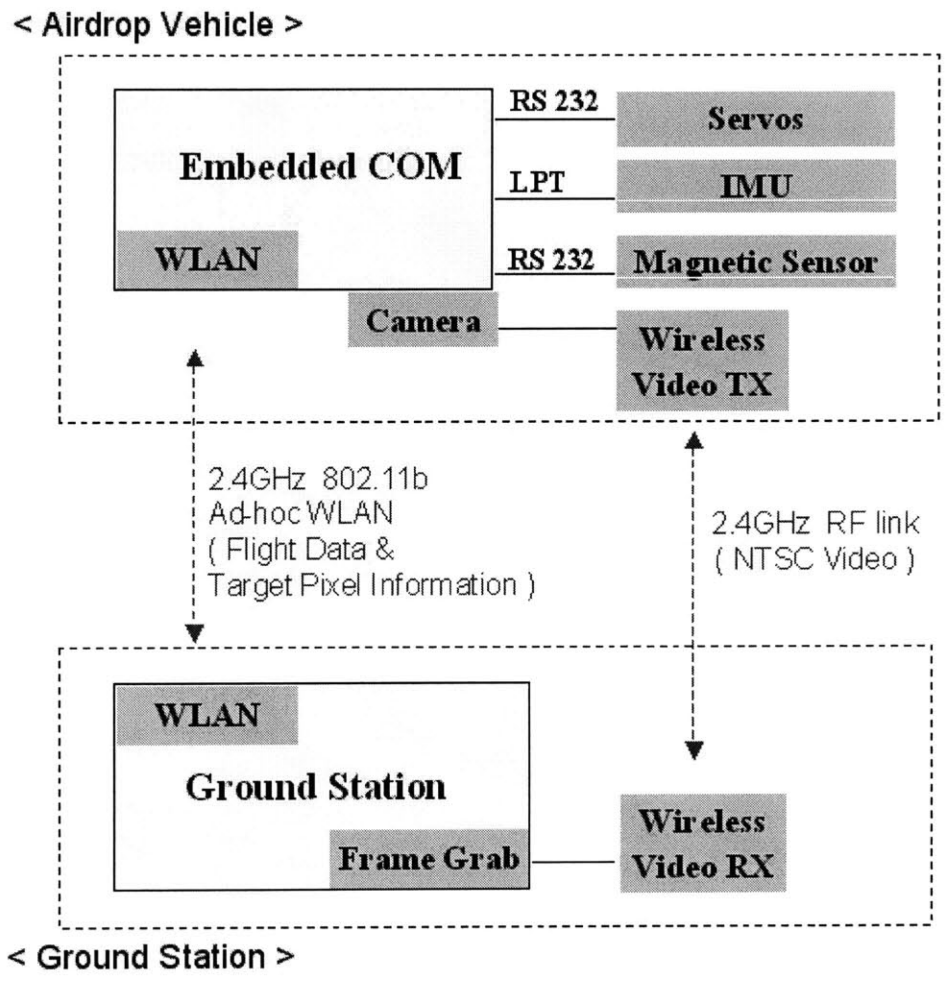


Figure 2-6: Precision Airdrop Avionics

The PDV has the PC/104+ embedded computer which runs the PDV software and is interfaced with all the PDV avionics parts other than the low-cost camera and the wireless NTSC video transmitter. The servo-actuated fins are controlled through a servo controller, which is connected to the on-board computer via an RS-232 serial link. The low-cost IMU is made up of an A/D converter, signal conditioning parts and low-cost inertial sensors including three-axis accelerometers and rate gyros, and is interfaced to the on-board computer via a parallel link. The three-axis magnetic sensor is also connected to the on-board computer via an RS-232 serial link. The 2.4 GHz 802.11b wireless network card is interfaced to the on-board computer via a PCMCIA slot. Flight data and tracked target pixel location data can be sent to, or received from the GS, respectively, via the 802.11b wireless network, using the IPX protocol. The camera and the wireless NTSC video transmitter are separate from the on-board computer, and the wireless NTSC video from the camera is transmitted through another 2.4 GHz wireless link to the GS.

The ground station (GS) requires very high-performance for processing the received camera images in timely fashion. The 2.4 GHz 802.11b wireless network card, which supports the IPX protocol, is interfaced to the GS computer via a PCI slot. The tracked target pixel location data are sent to the PDV through the 802.11b wireless network, and the flight data is also received from the PDV through the same network. The frame grabber card for processing the received NTSC video is also interfaced to the GS via another PCI slot. The wireless NTSC video receiver with a 2.4 GHz high gain directional patch antenna, for extending the video transmission range, is connected to the frame grabber card via a certain SMC to BNC cable. The GS program has the capability of showing the camera image and sending the target pixel locations, tracked by an human operator, to the PDV.

2.3.2 PDV Computer

The PDV computer runs the vehicle state estimation and control program and interfaces with the other avionics components. In addition, the PDV program executes the timer interrupt loop for state estimation and control at a frequency of 100 Hz.

The 686CORE & BASE PC/104+ embedded computer from CompuLab [1] was used as our vehicle on-board computer, because it has all the components needed to run real-time operating systems and it provides various ports for standard peripherals. The embedded computer is composed of the 686CORE module and the 686BASE PC/104+ single board computer.

The 686CORE module is a tiny (56 mm x 68 mm) computer, designed to serve as a building block in embedded applications. The module has NS Geode 266MHz CPU, 2Mbyte NOR flash disk, 64Mbyte NAND flash disk, and 128Mbyte SDRAM. For embedded applications, the 686CORE provides a 32-bit PCI bus, general purpose I/O lines and many other essential functions. The 686CORE module can be turned into a PC/104+ single board computer by integrating it with the 686BASE.

The 686BASE is a standard PC/104+ compliant, single board computer (96 mm x 91 mm x 15 mm). It uses a 686CORE module to implement most of the provided functions and also implements several additional important functions on-board. The 686BASE contains PC/104+ expansion connectors which opens it to a wide range of standard peripheral cards. It has a parallel port, four serial ports (RS-232/RS-485/RS-422), three USB ports, two 10/100BaseT Ethernet ports. Furthermore, the 686BASE contains a PCMCIA / Card Bus controller and slots. Off-the-shelf PCMCIA modules can extend the system with capabilities such as a larger solid state disk, modem, and wireless LAN.

The flash disk built into the 686CORE behaves exactly like a regular hard disk drive. Either NOR or NAND or both can be used as an on-board flash disk. MS-DOS 6.0 was used as the operating system and Borland C++ 3.1 was mainly used as the software development environment, where timer interrupts, RS-232 serial communication, parallel communication, and RAM-disk can be easily applicable.

2.3.3 IMU (Inertial Measurement Unit)

For the PDV, the low-cost IMU was developed with the use of a printed circuit board design. The IMU consists of low-pass filters for signal conditioning, a 3-axis accelerometer, rate gyros, an ADC (Analog-to-Digital Converter), and a power regulator. The IMU sampling frequency is 100 Hz, because every 0.01 seconds the PDV program which uses 100 Hz timer interrupt asks for IMU data. The IMU data are input to our extended Kalman filter, which is described in Chapter 4.

Rate Gyros

The ADXRS150 from Analog Devices [2] was used for measuring angular rate. It is a low-cost rate gyro which is integrated with all of the required electronics on a single chip. It is also a very tiny chip which is available in a space-saving 32-pin Ball Grid Array(BGA) surface-mount package measuring a mere 7 mm x 7 mm x 3 mm.

The device has two polysilicon sensing structures, each containing a dither frame which is electrostatically driven to resonance. A rotation about the z axis, normal to the plane of the chip, produces a Coriolis force which displaces the sensing structures perpendicular to the vibratory motion. This Coriolis motion is detected by a series of capacitive pickoff structures on the edges of the sensing structures. The resulting signal is amplified and demodulated to produce the rate signal output.

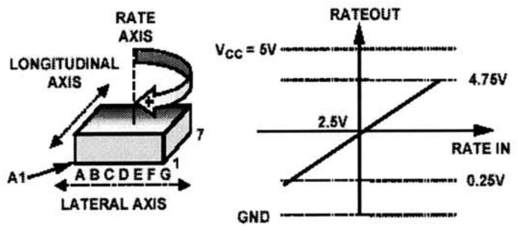


Figure 2-7: ADXRS150 Rate Sensitive Axis

The ADXRS150 is used as the PDV z-axis rate-sensing device, also called yaw-rate sensor. The output signal is a voltage proportional to angular rate about the axis normal to the top surface of that package: a positive output voltage for clockwise rotation about the axis normal to the package top, as described in Figure 2-7. A single external resistor can be used for lowering the scale factor, and an external capacitor is used for setting the bandwidth. The important features of the ADXRS150 are as follows:

· Dynamic Range	:	+/- 150 deg/sec
· Initial Sensitivity @ 25 ^oC	:	11.25 to 13.75 mV/deg/sec, 12.5 mV/deg/sec Typical
· Initial Null	:	2.50 V
· Null Drift Over Temperature	:	+/- 300 mV
· Rate Noise Density @ 25 ^oC	:	0.05 deg/sec/$\sqrt{}$ Hz
· Maximum Bandwidth(user selectable)	:	500 Hz
· Power Supply	:	5.00 V and 6.0 mA
· Operating Temperature Range	:	-40 ^oC to +85 ^oC
· Size	:	7 mm x 7 mm x 3 mm
· Weight	:	< 1 gram

Three ADXRS150 rate gyros are used for the IMU to measure three-axis angular rates.

Three-axis Accelerometer

The Crossbow CXL04LP3 three-axis accelerometer [3], as shown in Figure 2-8, was used for measuring linear accelerations. It is a low-cost, small, and light weight sensor available in the range of +/- 4 g. The sensing element is a silicon micro-machined capacitive beam. The capacitive beam is held in force balance for full scale non-linearity of less than 0.2 %.

Figure 2-8: Crossbow CXL04LP3 3-axis Accelerometer

The features of the CXL04LP3 are:

· Input Range	:	+/- 4 g
· Zero g Drift	:	+/- 0.2 g
· Sensitivity	:	500 +/- 25 mV/g
· Non-Linearity	:	+/- 0.2 %
· Noise	:	10 mg rms
· Bandwidth	:	DC - 100 Hz
· Power Supply	:	5.00 V and 5.0 mA/axis
· Operating Temperature Range	:	-40 ^oC to +85 ^oC
· Size	:	1.90 cm x 4.76 cm x 2.54 cm
· Weight	:	46 gram

The three-axis accelerometer measures the acceleration of the PDV that results from forces other than gravity, and resolves it in to its predefined 3-axis frame, which can be easily related to the body-fixed axis frame. In other words, the outputs are proportional to the 3-axis components of the vector sum of the forces other than gravity, such as aerodynamic forces.

A/D Converter

For converting analog sensor data into digital data available to the PDV computer, the MAX147BCPP 12-bit resolution analog-to-digital converter (ADC) was used [4]. It combines an 8-channel multiplexer, high-bandwidth track/hold, and a serial interface with high conversion speed and ultra-low power consumption. It operates from a single +2.7V to +5.25V supply, and its analog inputs are software configurable for unipolar/bipolar and single-ended/differential operation. The MAX147BCPP works with an external reference, and uses either the internal clock or an external serial-interface clock to perform successive-approximation analog-to-digital conversions. In unipolar mode, an analog input signal from 0 V to VREF can be converted into a 12 bit digital output; in bipolar mode, the signal can range from -VREF/2 to +VREF/2. In single ended-mode, input signal voltages are referred to COM; in differential mode, the voltage difference between two channels is measured. Single-ended six channel (three channels for the three-axis accelerometer, three channels for three rate gyros) under unipolar inputs were used for our purpose. Some important features are:

· Resolution	:	12 bits
· External Clock Frequency	:	0.1 MHz Min, 2.0 MHz Max
· Conversion Time	:	less than 0.012 msec for each channel

For the external clock mode conversion, we used the parallel (LPT1) link of the PDV computer. The parallel port has a total of 12 digital outputs and 5 digital inputs accessed via 3 consecutive 8 bit ports in the processor's I/O space. Figure 2-9 shows a 25-way female D-type connector.

- 8 output pins accessed via the DATA (0x0378) Port
- 5 input pins (one inverted, /S7) accessed via the STATUS (0x0379) Port
- 4 output pins (three inverted, /C3, /C1, /C0) accessed via the CONTROL (0x037A) Port
- The remaining 8 pins are grounded

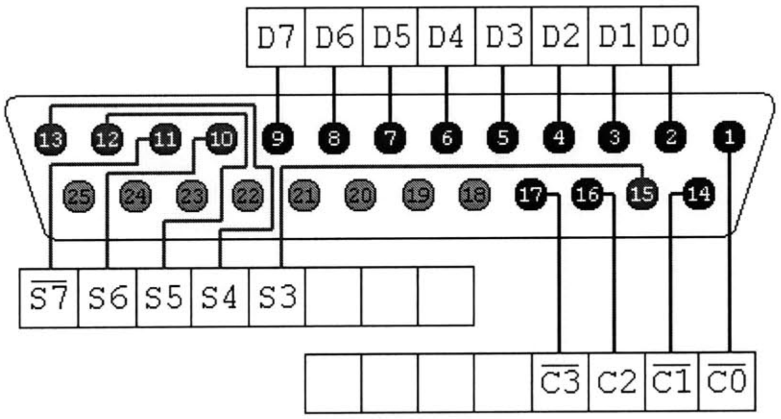


Figure 2-9: 25-Way Female D-Type Connector (Parallel Port)

The enhanced parallel port (EPP) mode was used for the parallel link. S5, S6, D4, D5, D6, and GND pins are utilized as DOUT, SSTRB, DIN, /CS, SCLK, and two GNDs (AGND and DGND) in the MAX147BCPP, respectively. However, actually the PDV computer has a unique parallel connector other than a 25-way female D-type connector. Thus, in order to know the correct pin locations in the connector, referring to the user manual for the PDV computer is required.

In the Borland C++ 3.1 environment, the *outportb* function is used for setting the pins to high (logic 1) or low (logic 0) in parallel or simultaneously. Similarly, the *inportb* function is used for reading the values (1 or 0) of pins in parallel. With the use of the parallel link of the PDV computer and the two functions, the driver codes to read accerelation and rate gyro data from ADC were implemented.

A conversion is started by clocking a control byte into DIN. With /CS low, each rising edge on SCLK clocks a bit from DIN into the MAX147's internal shift register. After /CS falls, the first arriving logic 1 bit defines the MSB of the control byte. The following shows the control-byte for each channel in the single-ended and unipolar mode:

Channel 1 : 0x8F (10001111)
Channel 2 : 0xCF (11001111)
Channel 3 : 0x9F (10011111)
Channel 4 : 0xDF (11011111)
Channel 5 : 0xAF (10101111)
Channel 6 : 0xEF (11101111)

In unipolar input mode, the output is straight binary. Data is clocked out at the falling edge of SCLK in MSB-first format. In addition, we easily found that the 1 volt analog input corresponds to 819 ADC output since the resolution is 12 bits, thus the acceleration and the angular rate can be given as:

$$\begin{aligned} acceleration &= -g \times 2 \times (ADCoutput - 2047)/ADCconst \\ angularrate &= (ADCoutput - 2047)/ADCconst/RATEconst \end{aligned} \tag{2.1}$$

where $ADCconst = 819$ /volt, $RATEconst = 0.0125$ volt/deg/sec.

The sampling frequency of each channel is 100 Hz, since inertial sensor data is supposed to be processed every 0.01 seconds for state estimation in the PDV software which also makes use of 100 Hz timer interrupt.

Signal Conditioning

Signal conditioning is a basic component of all the measurement devices. It converts incoming measurements into a form acceptable to digitization hardware. The filtering process blocks unwanted signal frequencies arising from external noise sources from the incoming

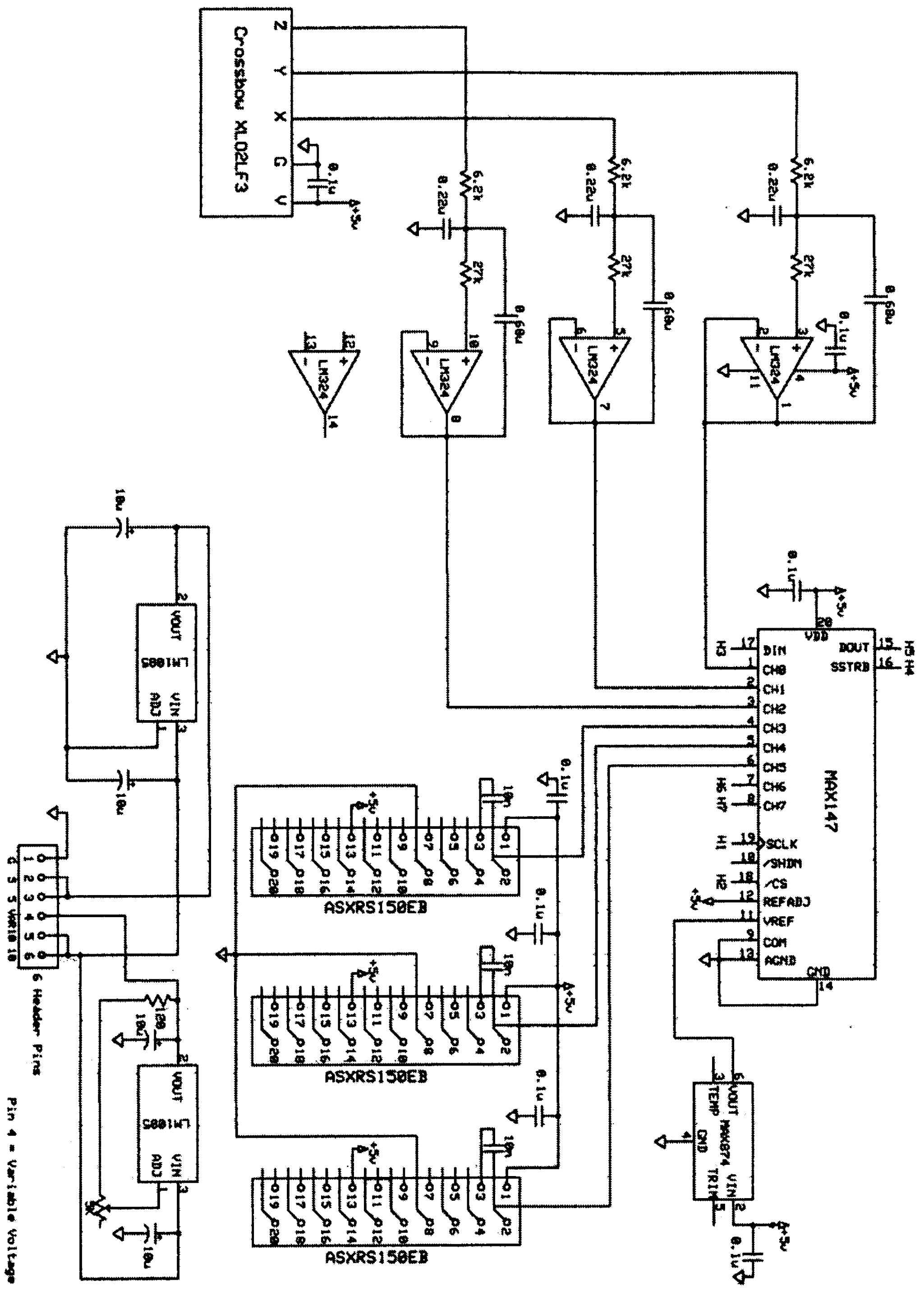


Figure 2-10: IMU Circuit Diagram

signals. Proper filtering also prevents aliasing, where higher frequency components of a signal appear as lower frequency components. One way of avoiding the problem of aliasing is to apply a low-pass filter to the signal, prior to the sampling stage, to remove any frequency components above the Nyquist frequency (half the sampling frequency).

Our IMU is a printed circuit board which has low-pass filters for signal conditioning, an accelerometer (CXL04LP3), rate gyros (ADXRS150), an ADC (MAX147), and a power regulator. Since the accelerometer bandwidth is 100 Hz and the rate gyro bandwidth is up to 500 Hz (user selectable), these sensors provide response characteristics more than adequate to capture the PDV dynamics, whose fast dynamics frequency is near 3.2 Hz. Note, however, that the wider the bandwidth of the sensor the more we must be concerned with eliminating sensor response to higher-than-desired frequencies. Also the sensor bandwidth is larger than half the sampling frequency of 100 Hz. These two facts mean that low-pass filters prior to the ADC are required for high frequency noise rejection and anti-aliasing. The rate gyro contains an internal low-pass filter so the bandwidth can be limited by an external capacitor. In order to filter the accelerometer output signal, active low-pass filters are implemented with the use of LM324 OP-AMP's. The low-pass filter's upper cutoff frequency was chosen at 30 Hz, which is less than half the sampling frequency. Figure 2-10 shows the IMU circuit diagram.

Power Regulator

The IMU has a power regulator for the purpose of supplying regulated voltage to accelerometers, rate gyros, ADC, and OP-AMP's. All the components in the IMU require power at 5 V. The power regulator takes 7.2 V from six Ni-Mh AA 1.2 V batteries in series, as an input source, and regulates the input to a 5 V output.

2.3.4 Power System

The power budget for the avionics components in the PDV is shown in Table 2.1. Six Ni-Mh AA 1.2V batteries in series, which generate 7.2 V, are used for the PDV computer and the IMU. The 5 V regulator is placed to regulate the input of 7.2 V to 5 V output which is needed in the PDV computer. The voltage source of 14.4 V, which comes from three Ni-Cd 4-cell 4.8 V batteries in series, supplies the power to the servo controller and servos, the camera, the cooling fan, and the NTSC video transmitter. The voltage output from the

12 V regulator is used for the camera, the cooling fan, and the NTSC video transmitter which all need 12 V input. In addition, the adjustable regulator is arranged for the servo controller and servos which need 6 V input. The power system guarantees about 30 minutes of running time, which was adequate for test purposes. The RC receiver and relay switches were used for remotely powering all the other avionics components while the vehicle is in the air. Figure 2-11 shows the power system circuit diagram.

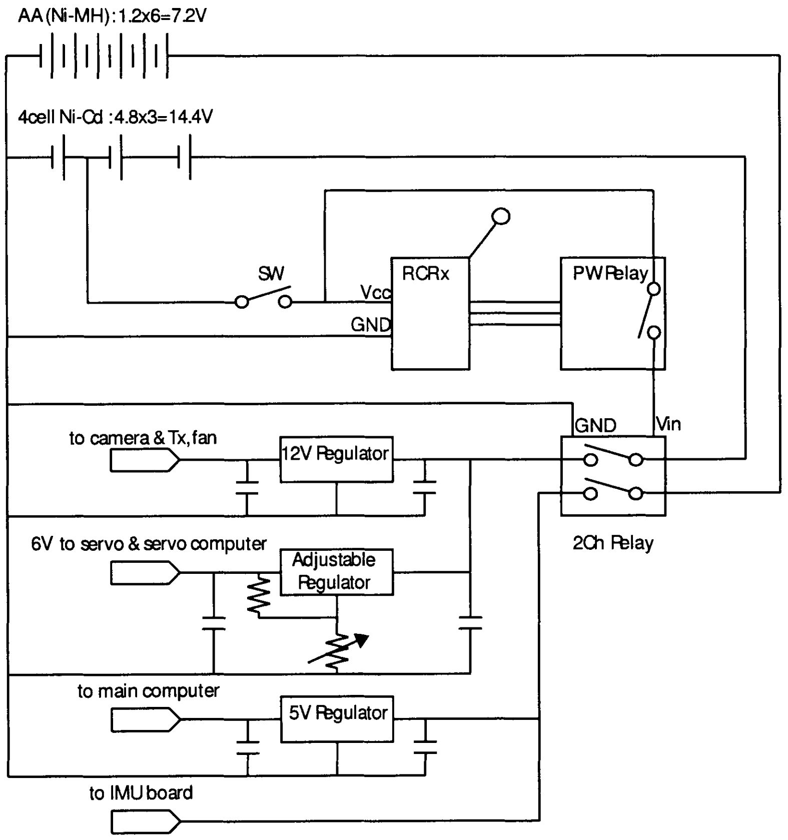


Figure 2-11: Power System Circuit Diagram

2.3.5 Three-axis Magnetic Sensor

Magnetic sensor measurements are used for improving the state estimates of our extended Kalman filter, as described in Chapter 4. The Honeywell HMR2300 three-axis smart digital magnetic sensor [6] detects the strength and direction of the magnetic field and communicates the x, y, and z components directly to a computer. Three independent bridges are

Components	Voltage [V]	Current [A]	Power [W]
On-board Computer	5	2.00	10.00
Cooling Fan	12	0.08	0.96
Servo Controller & Servos	6	0.40	2.40
Camera	12	0.05	0.60
NTSC Video Transmitter	12	0.12	1.44
IMU	7.2		

Table 2.1: Power Budget for Avionics Components

oriented to sense the x, y, and z axes of the magnetic field. The bridge outputs are then converted to a 16-bit digital value by means of an internal delta-sigma A/D converter. The sample rate can be varied from 10 samples per second (sps) to 154 sps. Each sample contains an x, y, and z reading and can be outputted in either 16-bit signed binary or binary coded decimal (BCD) ASCII characters. Some important features are as follow:

Field Range	:	+/-2 Gauss
Resolution	:	67 μGauss
Weight	:	28 grams (board only), 98 grams (in aluminum enclosure)
Output Rate Selectable	:	10 to 154 sps
Three-Axis Digital Output	:	BSD ASCII (28 bytes for every reading) or 16 bit Signed Binary (7 bytes for every reading)
RS-232 or RS 485 Serial Output	:	9,600 or 19,200 baud
Operating Temperature	:	-40 $^oC \sim$ 85 ^oC
Supply Voltage	:	6.5 V $\sim$ 15 V
Supply Current	:	27 mA Typical, 35 mA Max

The data were serially output to the PDV computer by means of the RS-232 serial link at 19,200 baud. The 16 bit signed binary format was used since it was more efficient for a computer to interpret and only 7 bytes were transmitted every reading. The output sample rate was 100 sps in the continuous reading mode.

Transmission time for 19,200 baud are about 0.5 msec/byte. In the RS-232 serial link, signals are sent and received simultaneously (full duplex). One signal goes from the computer (Tx, pin #3 in a DB9 socket) to the HMR (Rx, pin #3) and the other from the HMR (Tx, pin #2) to the computer (Rx, pin #2 in a DB9 socket). The HMR2300 driver codes which includes the interface functions for the RS-232 serial link and the decoding functions of 16 bit signed binary format was also developed for the PDV system.

2.3.6 Servo System

The Pontech SV203B/C servo motor controller [7] was used for controlling four servo-actuated fins. The SV203B/C is a Microchip PIC16C73 microcontroller based servo motor controller board. It accepts RS232 serial data from the PDV computer and outputs a PWM (pulse width modulated) signal to control the RC servo motors. The C code to control servos was developed with the use of the RS232 serial link between the PDV computer and the servo controller. The features of the SV203B/C are as follows:

· Controls 1 to 8 servos per board, 8-bit resolution

· Interface to PC through RS232 serial port (2400 to 19.2K baud, 9600bps default)

· 5 Ch, 8-bit A/D input port reading 0 $\sim$ 5 volts

· Dimensions: 1.4 in x 1.7 in

· Servo Connectors: 3 pin sip. Futaba J-type connectors.

· Power supply: 4.8 V to 6.0 V

By assigning a value from 0 to 255 to each servo, each PWM for that servo channel is generated, for example, 0 = No PWM, 1 = 0.6 ms pulse width, 127 = 1.5 ms pulse width, and 255 = 2.4 ms pulse width. In order to find the one-byte (0$\sim$255) values which were transmitted from the PDV computer to the SV203B/C via the serial link to control servos, the angular calibration was made. In our servo-actuated fins, 110, 127 and 144 in the byte data mean -10 deg, 0 deg and 10 deg, respectively, in control fin angle deflections:

$$servo = (17/10.0)(57.3 \times angle) + 127 \tag{2.2}$$

where *servo* is the one-byte(0$\sim$255) value to control a servo-actuated fin. In addition, *servo* is 255 if *servo* is larger than 255, and *servo* is 0 if *servo* is smaller than 0.

The Hitec HS-81MG servos [15] were used as our servo motors. The specifications are follows:

Weight	:	19.0 g
Dimensions	:	30 mm x 12 mm x 30 mm
Torque	:	2.6 kg-cm @4.8 V, 3.0 kg-cm @ 6.0 V
Speed	:	0.11 sec/60 deg @ 4.8 V, 0.09 sec/60 deg @ 6.0 V
Bearing Type	:	None
Motor	:	3 Pole Ferrite

The low bandwidth of about a 10 Hz (for controlling four servos) was a weak point of the combination of a SV203B/C servo controller and four HS-81MG servos since our timer interrupt loop for state estimation and control was executed at a frequency of 100 Hz.

2.3.7 RC Receiver and transmitter

A Futaba FP-R148DP 8 channel PCM receiver was turned out to be robust in the presence of electric and magnetic noise from other avionics components. In addition, a Futaba FP-8UAFS PCM1024 8 channel transmitter was used with a Futaba FP-TP-FM 72MHz RF module. The RC receiver and transmitter link was utilized for remote power turn on-and-off of the avionics, and as a discrete controller for airdrop and parachute deployment.

2.3.8 Vision-based System

The vision-based system was devised in such a way that the measured values of target pixel locations on the camera images in the ground station could be used for vision-based measurement updates in our estimation algorithm. This vision-based system is divided into the PDV part and the GS part. The 2.4 GHz wireless link was used for the NTSC video transmission. The center frequency of 2.470 GHz for the video signal was applied to both transmitter and receiver. This choice of the center frequency for the video signal was appropriate since the video signal did not interfere with the frequency range used for the 802.11b wireless network, which has the center frequency of 2.412 GHz and the bandwidth of about 30 MHz.

Precision Airdrop Vehicle Components:

- Camera

The PC-53XS from Supercircuits [9] served as the camera sensor. It is a very small and low-cost color video camera. The horizontal and vertical field of view angles are 74 degrees and 59 degrees, respectively, and the aspect ratio of view is 4:3. The PC-53XS weighs only 9 grams. Output is standard NTSC video, with digital signal processing automatically controlling the white balance and electronic shutter. The specifications are as follows:

Image Sensor	:	1/3 inches color CMOS
Picture Element	:	512(H) × 492(V), 251,904 Pixels
Scanning System	:	525 lines interlaced
Resolution	:	350 TV lines
Video Out	:	1 Vp-p composite video 75Ω NTSC
Lens	:	4.3 mm, field of view 74 degrees
Minimum illumination	:	7.0 lux, F 1.2 3200K
Power Consumption	:	DC 11-13 V, 50 mA
Dimensions	:	0.64 in × 0.64 in × 1.05 in
Weight	:	9 grams

- 2.4 GHz NTSC video transmitter

The MPX-2400 from Polaris [10] was used as the NTSC video transmitter. The MPX-2400 is a mini lightweight 2.4 GHz wireless 4CH Video and stereo audio transmitter module, and can be easily integrated with a camera. The specifications are as follows:

Frequency	:	2.4 GHz
Channel Frequencies (GHz)	:	Ch1 - 2.413, Ch2 - 2.432, Ch3 - 2.450, Ch4 - 2.470
Video Input	:	1 Vp-p composite video 75Ω NTSC
Output Impedance	:	50Ω Typical
Output Port	:	Integral dipole antenna or SMA connector
Antenna Connector	:	SMA Male
Pin Assignment	:	Red - Power, Black - GND, Yellow - Video(NTSC)
Power Requirements	:	12 V DC, 120 mA
Operating Temperature	:	0 ^oC to 45 ^oC
Dimensions	:	.47 in × 1.5 in × 2.65 in
Product Weight	:	36 grams

Ground Station Components:

- 2.4 GHz NTSC video receiver & high gain directional patch antenna

The PI-366R from Polaris was used as the 2.4 GHz wireless 4CH NTSC video receiver. It is equipped with a professional SMA connector on its antenna port. This allows for the use of optional components such as directional antennas and inline amplifiers. The features are as follows:

Frequency	:	2.4 GHz
Channel Frequencies (GHz)	:	Ch1 - 2.413, Ch2 - 2.432, Ch3 - 2.450, Ch4 - 2.468
Video Out	:	1 Vp-p composite video 75Ω NTSC
Power Requirements	:	9 V DC, 400 mA
Antenna Connector	:	SMA Male
Dimensions	:	4.8 in × 2.36 in × 1.28 in
Weight	:	508 grams

The 2.4 GHz 18dB gain directional patch antenna was equipped with the PI-366R NTSC video receiver to extend the video transmission range. The N-Type Male to SMA Female cable for high frequency was used for connecting the antenna to the receiver. The specifications are as follows:

Type of Antenna	:	Patch
Frequency	:	2.3 ∼ 2.4 GHz
Gain	:	18 dB
Antenna Connector	:	N-Type Female
Supplied Cable	:	N-Type Male to SMA Female 50Ω impedance cable
Compatible Receivers	:	PI-366R
H Plane Beam	:	15 deg
E Plane Beam	:	15 deg
VSWR	:	1.5 dBi
Survival Wind	:	110 mph
Impedance	:	50Ω
Polarization	:	Vertical
Dimensions	:	2 in × 15.7 in × 15.7 in
Weight	:	3 Lbs.

By means of the configuration of the MPX-2400 video transmitter and the PI-366R video receiver with 18 dB gain antenna, the clear camera images on the ground station were achieved in the range of about 400 meters.

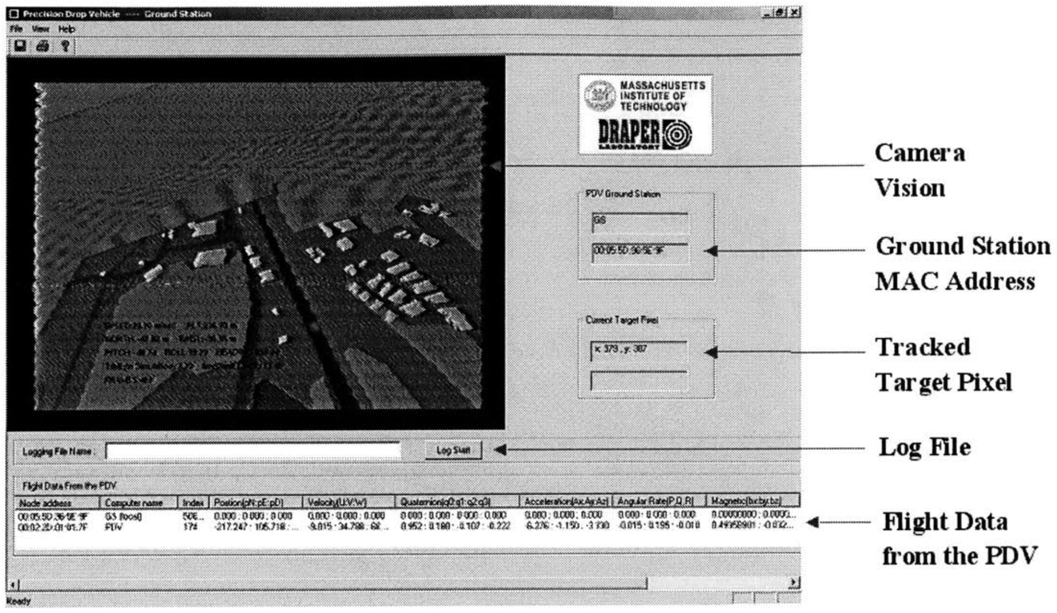
- Frame Grabber Card

  Sensoray Model 611 frame grabber PCI card was used for processing the received NTSC video [5]. The frame grabber converts video camera signals into digital data required for computer image processing and display. Up to four cameras may be connected to the board: four with composite video or three with composite video and one with an S-video output. The frame grabber can transfer 30 frames per second with the supplied software. It supports a variety of digital formats including RGB24 and Y8 that are directly compatible with Windows bitmaps. The frame grabber's signal processor uses two flash analog-to-digital converters; one for chrominance and one for luminance. Each A/D provides two times oversampling of the camera signal. Oversampling, combined with analog anti-aliasing filters, significantly reduces the overall sensitivity to unwanted aliasing signals and enhances the frame grabber's digital filtering. Camera Connections Separate BNC-style and DIN connectors are used for preserving the high frequency components of the video signals. The software development kit includes a 32-bit DLL and C-language APIs that allow capturing and saving color and monochrome images. The frame grabber card was connected to the PI-366R NTSC video receiver via a certain SMC to BNC cable. The specifications are as follows:

Capture Rate	:	30 frames/sec (NTSC, RS-170, CCIR), 25 frames/sec (PAL, SECAM)
A/D Resolution	:	8-bits for luminance, 8-bits for chrominance
FIFO Size	:	560 bytes (280 pixels)
Output Resolution	:	768 x 576 (PAL, SECAM), 640 x 480 (NTSC, RS-170)
BUS	:	PCI

- GS Software GUI (Graphical User Interface)

  The target tracking view of the GS software is the region where a human operator keeps tracking the target image by means of the end of the mouse arrow. Figure 2-12 shows the target tracking view in the ground station program during the HILSIM. The pixel values of the target image, which are tracked or continually designated



GS

00:05:5D:36:5E:9F

x: 379 , y: 387

Flight Data From the PDV

Node address	Computer name	Index	Position(pN:pE:pD)	Velocity(U:V:W)	Quaternion(q0:q1:q2:q3)	Acceleration(Ax:Ay:Az)	Angular Rate(P,Q,R)	Magnetic(bx:by:bz)
00:05:5D:36:5E:9F	GS (host)	506...	0.000 : 0.000 : 0.000	0.000 : 0.000 : 0.000	0.000 : 0.000 : 0.000 : 0.000	0.000 ; 0.000 ; 0.000	0.000 : 0.000 : 0.000	0.00000000 : 0.0000...
00:02:2D:01:01:7F	PDV	174	-217.247 : 105.718 : ...	-9.015 : 34.798 : 68...	0.952 : 0.180 : -0.107 : -0.222	-6.276 : -1.150 : -3.330	-0.015 : 0.195 : -0.010	0.49958901 : -0.032...

Figure 2-12: Ground Station Software GUI (Graphical User Interface)

by the mouse device, are transmitted to the PDV computer at a frequency of 40 Hz through the 802.11b wireless network. More detailed description is given in the software section.

2.3.9 802.11b Wireless Network

The 2.4 GHz 802.11b wireless network supporting the IPX protocol was used for communicating flight data and tracked target pixel locations between the PDV and the GS. A channel of center frequency at 2.412 GHz (channel 1) was assigned to the 802.11b network. As was previously described, the bandwidth of the 802.11b network is about 30 MHz, and 2.470 GHz was assigned for the center frequency of the NTSC video link. Thus, there was no interference between two wireless links.

The 802.11b wireless networks are divided into two types: ad-hoc mode networks and infrastructure mode networks. In the ad-hoc mode, data in the network is transferred to and from wireless network adapters connected to computers. An ad-hoc network is a kind of peer-to-peer network. There are some benefits of an ad-hoc network: it is simple to set up, inexpensive, and fast. Throughput rates between two wireless network adapters are twice as fast as when an access point is used. However, the communication range of the wireless network can be increased by adding an access point. Access points can be used for the infrastructure mode.

For our experimental setup, both modes were considered. The ad-hoc mode was first applied to the HILSIM because of its simplicity. However, for the real precision airdrop framework the infrastructure mode was used because the communication range in the mode was much longer than that in the ad-hoc mode. The use of an access point, with two 6 dB gain antennas, gave a communication range of over 400 meters. However, as far as our experimental setup is concerned, there is no significant difference between two modes except the communication range and the setup in hardware.

The IPX (Internetwork packet exchange) protocol was chosen as our base protocol for 802.11b wireless communication. It is a user datagram protocol (UDP) and a routable protocol. However, it is not a session-based protocol and there is no guarantee or verification of successful delivery. Because it is faster than TCP/IP and easier to implement on MS-DOS 6.0 operating system, our communication protocol (for flight data and tracked target pixel locations, or commands from the GS) was defined and implemented on the IPX protocol. In our experimental setup, the network number of 00000000 and the frame type of Ethernet 802.3 were applied to both the PDV computer and the GS computer. More details are described in the software part.

PDV Components:

- 2.4 GHz 802.11b Wireless LAN PCMCIA card

  The Lucent's WaveLan 2.4 GHz 802.11b Wireless LAN PCMCIA card [14] was used as the 802.11b wireless network card in the PDV side. The firmware update using the recent Lucent's firmware update program was required for supporting both wireless network modes: ad-hoc (IBSS) and infrastructure modes. When the firmware was updated, the card supported both wireless network modes for MS-DOS 6.0 which was the operating system of the PDV computer. For the PDV computer part, the mode (ad-hoc or infrastructure) and channel used for the network are specified in the network configuration (NET.CFG) file. In addition, the DOS 16 bit ODI driver (WVLAN43.COM) for the WaveLan card was loaded in the boot time.

Ground Station Components:

- 2.4 GHz 802.11b Wireless LAN PCI card

The D-Link DWL-520 802.11b wireless LAN PCI card [12] served as the 802.11b wireless network card in the GS side. The DWL-520 can automatically connect to any specified ad-hoc or infrastructure wireless network as soon as it is active.

- 2.4 GHz 802.11b Access point & two 6 dB gain directional patch antennas

  The Linksys WAP54G access point [13] was used as an access point in the infrastructure mode. Although the WAP54G is a 802.11g (Data rates up to 54Mbps – 5 times faster than 802.11b) access point, it is also able to interoperate with 802.11b networks at data rates up to 11Mbps. It has two detachable dipole antennas. In order to extend the communication range, the two detachable dipole antennas were replaced with two 6 dB gain directional patch antennas. By means of the infrastructure mode in this configuration, a communication range of over 400 meters was achieved.

2.3.10 Ground Station (GS) Computer

The Commander-AT2 model from Sterling Computer [11] was used as the GS computer. The Commander-AT2 is a lunchbox computer offering a SBC/Backplane arrangement setup. This unit offers an 8 slot passive backplane with multiple full length PCI slots. The features are as follows:

Motherboards	:	SBC/Backplane
Processors	:	Pentium 4
Slots	:	8 Slots
Display Size	:	14.1 in
Resolution	:	1024 x 768
Dimensions	:	15.7 in x 11.5 in x 8.8 in
Case Weight	:	19.5 lbs

The GS computer has an 802.11b wireless network PCI card and a frame grabber PCI card in the slots. The operating system of the GS was MS Windows 2000 Professional, and the main software development environment was MS Visual C++ 6.0.

2.4 Hardware-in-the-loop-Simulation

A hardware-in-the-loop-simulation (HILSIM) is a very powerful tool for verifying and debugging real-time embedded systems which operate with real-world inputs and outputs. A

HILSIM contains all the possible avionics components and software in the loop and uses the same I/O signal lines as real systems. It allows us to analyze the faults or delays in our avionics components. In general, a HILSIM can be divided into two parts: real-time systems and hardware-in-the-loop (HIL) simulators. The real-time systems refer to all of the avionics components and software in a HILSIM, and the HIL simulators in a HILSIM are responsible for receiving output signals of the real-time systems, simulating the real-world, and generating the input signals for the real-time systems.

In our HILSIM setup, the real-time systems are: (1) the PDV software and avionics components except sensors (such as the camera sensor, inertial sensors and the magnetic sensor), (2) the ground station software and hardware including the frame grabber, the 802.11b wirless network card, and the NTSC video receiver. The HIL simulators in our HILSIM are the flight dynamics simulator and the camera vision simulator. The flight dynamics simulation (FDS) software and the camera vision simulation (CVS) software were also run on separate computers.

2.4.1 Hardware-in-the-loop-Simulation Framework

Figure 2-13 shows our HILSIM schematic. The PDV part and the GS part in the precision airdrop schematic shown in Figure 2-6 are also used for the HILSIM framework. However, as was previously mentioned, the HIL simulators such as the flight dynamics simulation (FDS) and the camera vision simulation (CVS) are required for the HILSIM.

The PDV is capable of estimating the state variables with the use of sensor data and target pixel locations, and controlling the servo-actuated fins.

The flight dynamics simulator inputs servo fin deflections from potentiometers and propagates the vehicle nonlinear dynamic equations to compute all the state variables for simulating the real-world. FDS replicates the IMU and magnetic sensor data inputs required from the PDV sensor suite.

In order to create the camera image, a separate laptop computer with NTSC video output capability was used as the camera vision simulator (CVS). It has 3D map information. It takes the position and attitude information from the simulation computer as input signals, and generates a realistic simulated image. The vision images are then transmitted to the ground station via NTSC video link.

In the ground station GUI, a human operator can identify and track the target image.

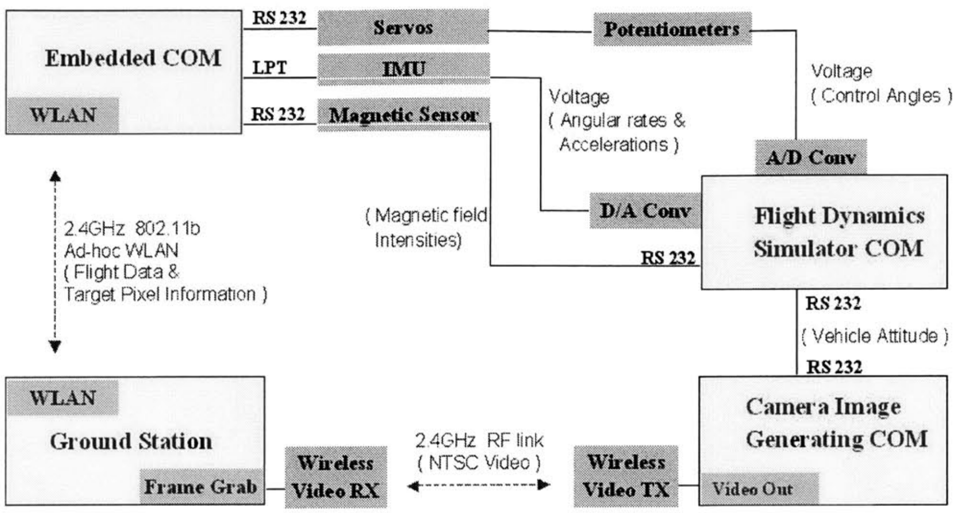

Embedded COM
WLAN
RS 232
LPT
RS 232
Servos
IMU
Magnetic Sensor
Potentiometers
Voltage (Control Angles)
Voltage (Angular rates & Accelerations)
A/D Conv
D/A Conv
RS 232
(Magnetic field Intensities)
Flight Dynamics Simulator COM
2.4GHz 802.11b Ad-hoc WLAN (Flight Data & Target Pixel Information)
RS 232
(Vehicle Attitude)
RS 232
WLAN
Ground Station
Frame Grab
Wireless Video RX
2.4GHz RF link (NTSC Video)
Wireless Video TX
Camera Image Generating COM
Video Out

Figure 2-13: Hardware-In-the-Loop-Simulation (HILSIM) Schematic

The tracked target pixel locations are then transmitted to the PDV computer via 802.11b wireless network to close the loop.

Several additional data communications were performed in the HILSIM: (1) The magnetic sensor data simulated in the flight dynamics simulator were transmitted to the PDV through an RS-232 serial link with 19,200 baud, at a frequency of 100 Hz. (2) The IMU data simulated in the flight dynamics simulator were converted into analog values by means of DAC, and then transmitted to the IMU board in the PDV. (3) Potentiometers measure servo fin deflections, and then ADC converts the analog values into digital values which can be used for the FDS computer. (4) The vehicle attitude and positions in the flight dynamics simulator are transmitted to the camera vision simulator via an RS-232 serial link with 115,200 baud, at a frequency of 100 Hz.

2.4.2 Flight Dynamics Simulation (FDS) Computer

A desktop computer which has two serial ports for RS-232, a DAC (Digital-to-Analog Converter) card and an ADC (Analog-to-Digital Converter) card served as the FDS computer. It runs FDS software which executes a timer interrupt loop every 0.01 seconds. MS-DOS 6.0 was used as the operating system, and Borland C++ 3.1 was mainly used as software development environment.

2.4.3 Camera Vision Simulation (CVS) Computer

A laptop computer with NTSC video output capability and a serial port for RS-232 was used as the CVS computer. The operating system was MS Windows XP Professional, and the main software development environment was MS Visual C++ 6.0. The OpenGL graphic library and the OpenGL Utility Toolkit (GLUT) for Win 32 are used for generating graphics for the camera image.

2.5 Computer Software

In the precision airdrop framework two software programs play key roles: the PDV software and the GS software. However, in the HILSIM framework two additional software programs are required for two HIL simulators: the FDS software and the CVS software. The PDV software and the FDS software are based on MS-DOS 6.0 which supports timer interrupts, whereas the GS program and the CVS program each are based on MS Windows 2000 and XP.

2.5.1 PDV Software

The PDV software is mainly composed of five parts: state estimation, control, flight data logging, sensor data acquisition, and target data acquisition. In the main function of the PDV program, the following steps are sequentially executed:

Step 1. Open a new memory file for logging flight data

Step 2. Intialize the estimated state vector and the error covariance matrix

Step 3. Initialize an RS-232 serial link (COM2, 9,600 bps) for the servo controller

Step 4. Initialize an RS-232 serial link (COM1, 19.2Kbps) for the magnetic sensor

Step 5. Initialize the IPX 802.11b link for communication with the GS

Step 6. Initialize a timer interrupt (100 Hz clock)

Step 7. Enter the 'while' loop

Step 7.1. Send the control fin angle commands to the servo controller

Step 7.2. Check if the signal notifying of the mission completion has come from the GS. If it has come, escape the 'while' loop. Otherwise, go to **Step 7.1**

By means of the timer interrupt (100 Hz clock), the loop is executed every 0.01 seconds. Note that the 100 Hz periodic execution of the timer interrupt loop is quite separate from the 'while' loop of the main function. In the while loop at Step 7, a servo controlling function (Step 7.1) is used that sends the control fin angle commands to the servo controller via an RS-232 serial link (COM2, 9600 baud). Note that this function was not able to be inserted in the timer interrupt loop since our servo system had a low bandwidth of about 10 Hz, as was previously mentioned in the servo system section. However, the control-related codes still calculate the control fin angle commands at a frequency of 100 Hz, and are processed after the state estimation codes have processed in the timer interrupt loop.

In the timer interrupt loop, the time-critical tasks for state estimation and control are performed. It is therefore required to finish all these time-critical tasks in the time interval of 0.01 seconds. The following time-critical steps are processed in order:

Step 1. Check if the signal notifying of the start of state estimation (2 seconds before the airdrop) has come from the GS. If it has come, go to **Step 1.1** Otherwise, go to **Step 2**

Step 1.1. If new vision-based measurement data (target pixel location tracked by a GS operator and sent from the GS at a frequency of 40 Hz) are available, obtain the vision-based measurement data

Step 1.2. Obtain new IMU data

Step 1.3. Obtain new magnetic sensor measurement data

Step 1.4. Update estimated states and error covariance matrix (the extended Kalman filtering step)

Step 1.5. If after the airdrop, calculate the axial control angle commands and go to **Step 1.5.1** Otherwise, go to **Step 1.6**

Step 1.5.1. If after 3.5 seconds after the airdrop, calculate the directional control angle commands

Step 1.6. Write flight data every 0.05 seconds (20 Hz) on the memory file for logging

Step 2. Send flight data every 0.2 seconds (5 Hz) to the GS for logging in the network mode

After the signal notifying of the mission completion has come, the following termination steps are processed:

Step 8. Close a timer interrupt

Step 9. Close the IPX 802.11b link for communication with the GS

Step 10. Close an RS-232 serial link (COM1, 19.2Kbps) for the magnetic sensor

Step 11. Close an RS-232 serial link (COM2, 9,600 bps) for the servo controller

Step 12. Close a new memory file for logging flight data

The closed memory file is configured to be copied to the flash disk which acts like a hard disk drive for the PDV computer right after the PDV program is terminated.

State Estimation

A predefined notification signal, that makes the PDV program start the state estimation codes in the timer interrupt loop, is sent from the GS by a GS operator via the 802.11b wireless network. In our experimental setup, the notification signal is supposed to be sent from the GS at 2 seconds before the airdrop. The purpose of this is to apply sensor data (IMU and magnetic sensor data) during the level flight to our state estimation. However, it should be noted that vision-based estimation using the target pixel location measurements is set to become effective 3 seconds after the airdrop. This is because there is a delay between the airdrop and when the target becomes visible in the PDV camera. The target pixel locations tracked on the GS are sent to the PDV at a frequency of 40 Hz through the

802.11b wireless network. This allows the PDV software to incorporate the vision-based measurements into the state estimates every 0.025 seconds. More details of state estimation are described in Chapter 4.

Control

The directional control is composed of two parts: the pitch/yaw damper and the vision-based target pixel position control. Like vision-based estimation, the directional control is set to start working at 3.5 seconds after the airdrop. The reason that the time is set to 3.5 seconds, which is slightly larger than the start time of vision-based estimation is: first, the directional control is largely dependent on the vision-based measurements, and second, it allows a smooth transition to be applied to the vision-based target pixel position control.

The axial control, however, starts right after the airdrop since it is not based on tracked target pixel locations and the stabilized axial motion is quite necessary to assure the stable camera image. Refer to Chapter 4 for more detailed description of directional and axial controllers.

Flight Data Logging

The flight log data includes the estimated states, sensor data, and the vision-based measurement data. The PDV flight data are logged in two modes: the RAM-disk mode and the networking mode.

In the RAM-disk mode, the flight log data such as the estimated states, sensor data, and the vision-based measurement data are logged in a memory (RAM) file every 0.05 seconds or at a frequency of 20 Hz. After a mission of the PDV is completed, the PDV program is supposed to be terminated by a specific signal sent from the GS by a GS operator, and then the memory file is closed and copied to the flash disk which acts like a hard disk drive for the PDV computer.

In the networking mode, the flight log data are sent to the GS via the IPX 802.11b wireless network and written into the pre-specified log file on the GS every 0.2 seconds or at a frequency of 5 Hz. For the configuration of the IPX network, the socket number 5050 was used as both source and destination socket numbers. The MAC (Media Access Control) address of the GS was specified as the destination address. Note that this socket number 5050 was also used for the corresponding communication part of the GS.

Sensor Data Acquisition

As was previously mentioned, IMU data are achieved via the PDV computer's parallel link to the A/D converter (ADC) of the IMU. A total of 6 channels of the ADC were used for 3-axis accelerations and 3-axis angular rates. A function for IMU data acquisition was implemented and it allowed 6 channel values to be recorded at a time. This IMU-related function is executed every 0.01 seconds or whenever the timer interrupt loop is executed. The IMU data is utilized for the time-propagation of the state estimates and the error covariance matrix in our extended Kalman filter, which is described in Chapter 4.

In our experimental setup, an RS-232 serial link (COM1, 19,200 baud) was used for obtaining the magnetic sensor data, and the data transfer rate of this link was fast enough to get the data in the 100 sps (samples per second) continuous reading mode with 16 bit signed binary format. In this continuous reading mode, the magnetic sensor actively keeps sending the measured data at a frequency of 100 Hz via the serial link. It enables use of a new magnetic sensor data measurement for our extended Kalman filter every 0.01 seconds. The use of magnetic sensor measurements for the extended Kalman filter is also described in Chapter 4. A function which gives 3-axis magnetic field intensities was developed, and it was run before the measurement update in the timer interrupt loop.

Tracked Target Data Acquisition

The tracked target pixel locations are sent through the IPX protocol-based 802.11b wireless network. For the configuration of the IPX network, the socket number 5050 was specified in the codes of communication part. If a data packet from the GS arrives at the IPX socket (5050), the predefined reception event handler is called. Note that since the GS sends the vision-based measurement data at a frequency of 40 Hz, the reception event handler is called every 0.025 seconds. In addition, the reception event handler obtains the vision-based measurement data from the received data packet. This vision-based measurement data are used for Step 1.1. of the timer interrupt loop.

2.5.2 Ground Station (GS) Software

The ground station software is based on Visual C++/MFC (Microsoft Foundation Class Library) and is composed of five regions as previously shown in Figure 2-12 : camera vision view, ground station MAC (Media Access Control) address view, tracked target pixel location view, log file specification region, and flight data view.

The camera vision view has a horizontal resolution of 640 pixels and a vertical resolution of 480 pixels. The frame grabber converts NTSC video camera signals into digital data, and the supported frame grabber APIs make it possible to generate 30 frames per second on the camera vision view. The MFC *SetTimer* function is used for installing a system timer, and the callback function that processes cyclic timer messages (*WM_TIMER*) executes the FrameGrabber function which is implemented to grab a frame and request its display. The FrameGrabber function makes use of 2 image buffers: while an image is being acquired into one buffer, the image acquired into the alternative buffer is displayed. This continues in succession.

The camera vision view also serves as the target tracking view where a human operator tracks the target image by means of a mouse interface. The MFC *OnMouseMove* function is used for getting the horizontal and vertical pixel values of the target point on the camera vision view. The *OnMouseMove* function is called whenever the mouse cursor moves. It also takes the tracked target pixel location as its arguments.

The ground station MAC (Media Access Control) address is used for the codes of the IPX network initialization. Note that the GS MAC address is also used as the destination address in the PDV program.

In order to permit sending of broadcast messages or bind the IPX datagram socket to the broadcast address which is defined as *FFFFFFFFFFFF*, the *setsockopt* function is called with the *SO_BROADCAST* option. The *WSAAsyncSelect* function is called with the network event option of *FD_READ*. This changes the socket into the non-blocking asynchronous operating mode. In the asynchronous mode, the WinSock DLL sends a message to the program whenever the event occurs. The flight data from the PDV are read in the message-based manner. The *FD_READ* option enables the program to receive notification of readiness for reading. When flight data has come, the *OnWinsock* callback function is called by a message and the *recvfrom* function reads the data. The received flight data are also shown in the flight data view. In addition, the *sendto* function is used for sending the tracked target pixel locations to the PDV. The *sendto* function is periodically called by a windows timer of 40 Hz. By means of the non-blocking asynchronous operating mode, the GS program is capable of tracking the target image by the mouse arrow and simultaneously communicating with the PDV to send the tracked target pixel locations and receive the flight data.

The log file for writing the PDV flight data can be specified or automatically named when the log start button is pressed. Pressing the log stop button stops writing the PDV flight data and closes the log file.

In addition, the GS program is capable of sending signals notifying the PDV program to start the state estimation or complete the mission by pressing predefined keys on the GS keyboard. The notification signal to make the PDV program start the state estimation is supposed to be sent at 2 seconds before the airdrop.

2.5.3 Flight Dynamics Simulation (FDS) Software

The FDS software is required for the HILSIM. Like the PDV software, the FDS software is also based on MS-DOS 6.0 which is easily able to use the timer interrupt loop, RS-232 serial communication, and RAM-disk. By means of a timer interrupt(100 Hz clock), the timer interrupt loop is executed every 0.01 seconds. The following time-critical tasks are performed in the timer interrupt loop.

Step 1. Achieve the servo fin deflection values from ADC

Step 2. Propagate the vehicle nonlinear dynamic equations and computing state variables, with the use of the fourth-order Runge-Kutta numerical integration method

Step 3. Simulate IMU data and magnetic sensor data

Step 4. Send the simulated IMU data to the PDV computer via DAC

Step 5. Send the simulated magnetic sensor data to the PDV computer via an RS-232 serial link (COM2, 19.2Kbps)

Step 6. Send the velocity, position and attitude state variables to the camera vision simulation (CVS) computer via an RS-232 serial link (COM1, 115.2Kbps)

Step 7. Write the states in every 0.2 seconds (5 Hz) on the memory file for logging

All the time-critical tasks are supposed to be processed in the time interval of 0.01 seconds. For the purpose of simulating the digital output of the HMR2300 magnetic sensor, the encoding functions of 16 bit signed binary format were also implemented. In addition, in order to send the necessary state variables to the CVS computer in a more robust fashion, a special encoding algorithm was devised which use a predefined one-byte delimiter. The rules of the encoding algorithm are as follows:

1. Insert a predefined one-byte delimiter between the transmitted float variables.
2. If a float state variable (which is composed of four bytes) includes the same byte as the delimiter, the byte in the variable is sent one more time. If several bytes in a float state variable are equal to the delimiter, every same byte to the delimiter is sent one more time.
3. Send the encoded float stream in a byte-by-byte manner.

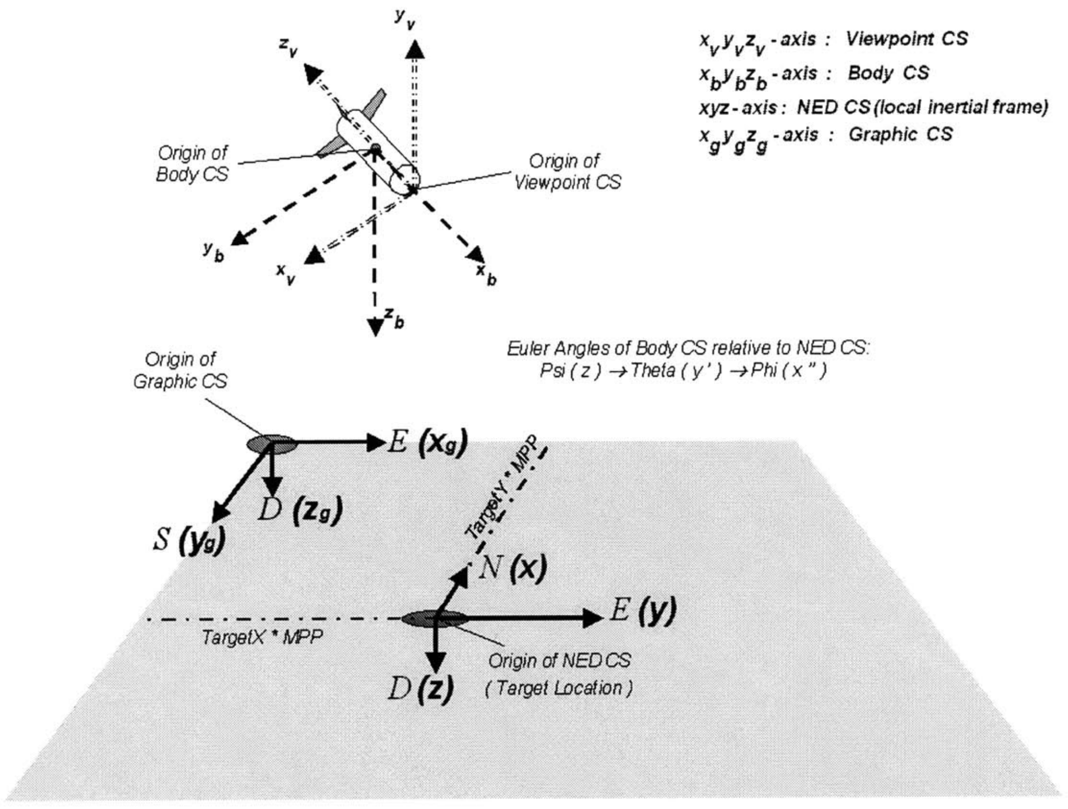
To be in accord with the PDV program which is supposed to start the state estimation at 2 seconds before the airdrop, when the FDS program is started, it simulates level flight during the first 5 seconds and then it simulates airdrop. More details of the flight dynamics simulation and modeling are described in Chapter 3.

2.5.4 Camera Vision Simulation (CVS) Software

The CVS software completes two functions: Receiving the velocity, position and attitude state variables from the flight dynamics simulation (FDS) computer, and generating the camera image corresponding to the instantaneous position and attitude states of the PDV. Additionally, The NTSC video output capability is supported by the CVS computer itself, so it is not of concern to the CVS software.

Because the CVS software is based on MS Windows XP, it includes the Win32-version RS-232 thread procedure which is responsible for reading the serial port and checking status events. The thread procedure creates two overlapped I/O structures, one for read events and another for status events. The thread using overlapped I/O structures makes it possible to independently generate the camera image while simultaneously receiving the state variables via an RS-232 serial link from the FDS computer. In addition, the byte-by-byte received data stream are decoded into float state variables by using the adverse rules of encoding rules mentioned in the FDS program part.

The OpenGL graphic library and the OpenGL Utility Toolkit (GLUT) for Win32 are used for generating the camera images from the received states of the PDV. The CVS software has the virtual 3D graphic models of the flying field where the PDV drop tests are done. The 3D models are represented relative to a predefined graphic coordinate system. There are four coordinate systems used for the CVS software, as shown in Figure 2-14:



$x_v y_v z_v$-axis : Viewpoint CS
$x_b y_b z_b$-axis : Body CS
xyz-axis : NED CS (local inertial frame)
$x_g y_g z_g$-axis : Graphic CS
y_v
z_v
x_v
y_b
x_b
z_b
Origin of Body CS
Origin of Viewpoint CS
Euler Angles of Body CS relative to NED CS: Psi (z) → Theta (y') → Phi (x")
Origin of Graphic CS
E (x_g)
S (y_g)
D (z_g)
TargetY * MPP
TargetX * MPP
N (x)
E (y)
D (z)
Origin of NED CS (Target Location)

Figure 2-14: Four Coordinate Systems: Viewpoint CS, Body CS, NED CS, and Graphic CS

1. **NED CS** The NED (North-East-Down) coordinate system whose origin is the target position

2. **Graphic CS** The graphic (world) coordinate system which is an ESD (East-South-Down) coordinate system and whose origin is (MPP×TargetY, -MPP×TargetX, 0)

in the NED coordinate system, where MPP (Meter Per Pixel) is defined as 1.25 meter/pixel, TargetX is 300 pixels, and TargetY is 250 pixels

3. **Body CS** The vehicle body-fixed axis coordinate system

4. **Viewpoint CS** The viewpoint (camera) coordinate system, where its origin is the viewpoint (camera) position and the direction of the viewpoint (camera) is pointed down its negative z-axis

Note that, in the OpenGL, the viewpoint (camera) is at the origin of the viewpoint coordinate system, looking in the negative z-axis and displaying plane center along the z-axis. In our case, if the camera is at the origin of the vehicle body-fixed axis coordinate system, the viewpoint coordinate system can be transformed into the vehicle body-axis coordinates by means of rotation transformations.

Since the virtual 3D graphic models are represented relative to the graphic coordinates system, we need to transform graphic coordinates (which are also called world coordinates) to viewpoint coordinates (which are also called camera coordinates). In the OpenGL, the transformation is accomplished through the MODELVIEW matrix. However, the OpenGL's approach using the viewing transformations is equivalent to fixing the camera at a specific location/orientaion and transforming the world such that the camera sees the world in the right way. In other words, it moves the world, not the camera. Thus, after the *glMatrixMode (GL_MODELVIEW)* function is called, which indicates how any new transformations will affect the MODELVIEW matrix, it is necessary to construct the MODELVIEW matrix through which the viewpoint (camera) coordinate system is to be transformed into the graphic (world) coordinate system. After the viewing transformations are performed, the models are drawn relative to the graphic coordinate system.

The following is the steps taken to construct the MODELVIEW matrix, which transforms the viewpoint coordinate system into the graphic coordinate system.

Step 1. (Transformation from the Viewpoint CS to the Body CS)

Step 1.1. Rotate 90 deg about the y-axis of the Viewpoint CS.

Step 1.2. Rotate 90 deg about the x-axis of the transformed CS.

Step 1.3. Translate the origin of the transformed CS to the origin of the Body CS. Now the transformed CS is the Body CS.

glRotatef (90.0, 0.0, 1.0, 0.0);

glRotatef (90.0, 1.0, 0.0, 0.0);

glTranslate (-R_{camera}, 0.0, 0.0);

where R_{camera} is the distance between the origin of the Camera CS and the origin of the Body CS.

Step 2. (Transformation from the Body CS to the NED CS)

Step 2.1. Undo the euler angles (state[PSI], state[THETA], state[PHI]) of the Body CS relative to the NED CS: first, -state[PHI] deg about the x-axis of the Body CS, second, -state[THETA] deg about the y-axis of the transformed CS, third, -state[PSI] deg about the z-axis of the transformed CS. Now the transformed CS is oriented with the NED CS, but still has the origin on the body-axis.

Step 2.2. Translate the origin of the transformed CS to the origin of NED CS. Now the transformed CS is the NED CS.

glRotatef (-state[PHI], 1.0, 0.0, 0.0);

glRotatef (-state[THETA], 0.0, 1.0, 0.0);

glRotatef (-state[PSI], 0.0, 0.0, 1.0);

glTranslatef (-state[NORTH], -state[EAST], -state[DOWN]);

where (state[PSI], state[THETA], state[PHI]) are the Euler angles of the Body CS relative to the NED CS, and (state[NORTH], state[EAST], state[DOWN]) are the NED positions of the origin of the Body CS.

Step 3. (Transformation from the NED CS to the Graphic CS)

Step 3.1. Translate the origin of NED CS to the origin of the Graphic CS.

Step 3.2. Rotate 90 deg about the z-axis of the transformed CS.

*glTranslatef (TargetY*MPP, -TargetX*MPP, 0.0);*
glRotatef (90.0, 0.0, 0.0, 1.0);

The *glMatrixMode (GL_PROJECTION)* function indicates how any new transformations will affect the projection matrix. The projection matrix is responsible for adding perspective to our scene. This creates a realistic looking scene. The perspective projection is accomplished by the *glLoadIdentity* function and the *gluPerspective* function. After the glLoadIdentity function has been called, the *gluPerspective* function sets up our perspective view for the scene. The following is the specification of the *gluPerspective* function:

$gluPerspective\ (fovy, aspect, zNear, zFar)$

$fovy$ specifies the field of view angle, in degrees, in the y direction (vertical direction). *aspect* specifies the aspect ratio that determines the field of view in the x direction (horizontal direction). The aspect ratio is the ratio of x (width) to y (height). $zNear$ specifies the distance from the viewpoint to the near clipping plane (always positive). $zFar$ specifies the distance from the viewpoint to the far clipping plane (always positive). In our case, the horizontal and vertical field of view angles are 74 degrees and 59 degrees, respectively, and the aspect ratio of view is 4:3. Thus, $fovy$ is 59, aspect is 640/480 = 4/3, zNear = 0.1, and zFar = 5000000.0 .

The GLUT is an event-based system: the user sets a number of callback functions, which will be called by the system when the specified event occurs. The *glutDisplayFunc* function sets the display callback for the current window. The display callback function includes all the camera image generation codes.

Chapter 3

Dynamics Modeling and Software Simulation

This chapter describes the dynamic modeling of the PDV and provides a discussion of the software simulation. The dynamic modeling includes the PDV rigid body equations of motion, aerodynamic force and moment models, servo models, sensor models, and wind models. The software simulation was developed in the MATLAB Simulink environment. In addition, it was done before the hardware-in-the-loop-simulation (HILSIM). It allowed us to design the state estimation and control loops and test our precision airdrop framework without using any avionics hardware.

3.1 Geometric and Inertial Properties

It is assumed that the PDV is axially symmetric. The geometric and inertial properties of the PDV are as follows:

mass	: $m = 1.76kg$
characteristic area	: $S = 0.0113\ m^2$ (the circular section area of the PDV body)
characteristic length	: $b = \bar{c} = 0.12\ m$ (the diameter of the circular section of the PDV body)
moment of inertia	:

$$J_x = 0.0031\ kgm^2 \quad J_y = 0.0369\ kgm^2 \quad J_z = 0.0369\ kgm^2$$
$$J_{xz} = 0.0\ kgm^2 \quad J_{xy} = 0.0\ kgm^2 \quad J_{yz} = 0.0\ kgm^2$$

3.2 Equations of Motion

The flat-Earth, body axes 6 degrees-of-freedom dynamics equations [17, Stevens] are

$$\begin{aligned}
\dot{\mathbf{v}}_B &= \mathbf{a} + C^{-1}\mathbf{g} - \Omega_3 \mathbf{v}_B && (\textit{Force Equations}) \\
\dot{\omega}_B &= -J^{-1}\Omega_3 J\omega_B + J^{-1}\mathbf{T}_B && (\textit{Moment Equations}) \\
\dot{\mathbf{q}} &= -\tfrac{1}{2}\Omega_4 \mathbf{q} && (\textit{Kinematic Equations}) \\
\dot{\mathbf{p}}_L &= C\mathbf{v}_B && (\textit{Navigation Equations})
\end{aligned} \tag{3.1}$$

where

$$\Omega_3 \equiv \begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix}, \quad \Omega_4 \equiv \begin{bmatrix} 0 & P & Q & R \\ -P & 0 & -R & Q \\ -Q & R & 0 & -P \\ -R & -Q & P & 0 \end{bmatrix} \tag{3.2}$$

and J is the inertia matrix

$$J \equiv \begin{bmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{bmatrix} \tag{3.3}$$

Note that $\mathbf{q}$ is the quaternion vector which has four quaternion variables of vehicle attitude, and the constraint of the quaternion vector is $||\mathbf{q}||_2 = 1$. The quaternion parameters are related to the Euler angles as follows:

$$\begin{aligned}
q_0 &= \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\
q_1 &= \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\
q_2 &= \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\
q_3 &= \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2)
\end{aligned} \tag{3.4}$$

The rotational transformation matrix from the body-fixed axis frame to the local inertial frame (NED geographic frame) is given by

$$C = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.5}$$

The attitude angles can be found from elements of the coordinate transformation matrix C as follows:

$$\phi = \tan^{-1}\left\{\frac{C_{32}}{C_{33}}\right\}, \quad \theta = -\sin^{-1}\{C_{31}\}, \quad \psi = \tan^{-1}\left\{\frac{C_{21}}{C_{11}}\right\} \tag{3.6}$$

The state vector in Eq. (3.1) is

$$\begin{aligned} \mathbf{x}^T &= \left[\mathbf{v}_B^T \, \omega_B^T \, \mathbf{q}^T \, \mathbf{p}_L^T \right] \quad \mathit{or} \\ \mathbf{x}^T &= [V_T \; \alpha \; \beta \; P \; Q \; R \; q_0 \; q_1 \; q_2 \; q_3 \; p_N \; p_E \; p_D \;]^T \end{aligned} \tag{3.7}$$

where V_T is the flight speed relative to the local inertial frame, α is the angle of attack, β is the sideslip angle.

The standard form of the vehicle nonlinear dynamics model is:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{3.8}$$

Eq. (3.1) can also be interpreted as the form in Eq. (3.8). Eq. (3.8) is used for simulating the PDV nonlinear dynamics by means of the fourth-order Runge-Kutta numerical integration method. Because of numerical errors, whenever the state vector is propagated, it is required to correct quaternion variables in order to satisfy the quaternion constraint, $||\mathbf{q}||_2 = 1$.

3.2.1 Force and Moment Equations

The force and moment equations in Eq. (3.1) can be written as:

$$\begin{aligned} \dot{V}_T &= \frac{U\dot{U} + V\dot{V} + W\dot{W}}{V_T} \\ \dot{\alpha} &= \frac{U\dot{W} - W\dot{U}}{U^2 + W^2} \\ \dot{\beta} &= \frac{\dot{V}V_T - V\dot{V}_T}{V_T^2 \cos\beta} \\ \dot{P} &= (c_1 R + c_2 P)Q + c_3 \bar{L} + c_4 N \\ \dot{Q} &= c_5 PR - c_6(P^2 - R^2) + c_7 M \\ \dot{R} &= (c_8 P - c_2 R)Q + c_4 \bar{L} + c_9 N \end{aligned}$$

where

$$\begin{aligned} \dot{U} &= RV - QW - g\sin\theta + \frac{1}{m}F_x \\ \dot{V} &= PW - RU + g\cos\theta\sin\phi + \frac{1}{m}F_y \\ \dot{W} &= QU - PV + g\cos\theta\cos\phi + \frac{1}{m}F_z \end{aligned}$$

Note that the PDV has no thrust forces, so F_x, Fy, and Fz are just aerodynamic forces with respect to the body-fixed axis frame. The constants c_1 through c_9 are determined by the inertial properties of the vehicle as follows:

$$\begin{array}{ll}
\Gamma = J_x J_z - J_{xz}^2, & c_1 = \frac{(J_y - J_z)J_z - J_{xz}^2}{\Gamma} \\
c_2 = \frac{(J_x - J_y + J_z)}{\Gamma}, & c_3 = \frac{J_z}{\Gamma} \\
c_4 = \frac{J_{xz}}{\Gamma}, & c_5 = \frac{J_z - J_x}{J_y} \\
c_6 = \frac{J_{xz}}{J_y}, & c_7 = \frac{1}{J_y} \\
c_8 = \frac{J_x(J_x - J_y) + J_{xz}^2}{\Gamma}, & c_9 = \frac{J_x}{\Gamma}
\end{array}$$

The forces and moments with respect to the body-fixed axes are calculated from those with respect to the stability axes as follows:

$$\begin{Bmatrix} F_x \\ F_y \\ F_z \end{Bmatrix} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{Bmatrix} F_{x_s} \\ F_{y_s} \\ F_{z_s} \end{Bmatrix}, \quad \begin{Bmatrix} \bar{L} \\ M \\ N \end{Bmatrix} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{Bmatrix} \bar{L}_s \\ M_s \\ N_s \end{Bmatrix}$$

where

$$\begin{array}{lllllll}
F_{x_s} & = & C_{x_s}\bar{q}S & & \bar{L}_s & = & C_{l_s}\bar{q}Sb \\
F_{y_s} & = & C_{y_s}\bar{q}S & & M_s & = & C_{m_s}\bar{q}S\bar{c} \\
F_{z_s} & = & C_{z_s}\bar{q}S & & N_s & = & C_{n_s}\bar{q}Sb
\end{array}$$

The aerodynamic coefficients about the stability axes are described in Section 3.3.

3.2.2 Kinematic Equations

The kinematic equations in Eq. (3.1) or the quaternion attitude equations can be solved as:

$$\begin{aligned}
\dot{q_1} &= -(1/2)(Pq_1 + Qq_2 + Rq_3) \\
\dot{q_2} &= -(1/2)(-Pq_0 - Rq_2 + Qq_3) \\
\dot{q_3} &= -(1/2)(-Qq_0 + Rq_1 - Pq_3) \\
\dot{q_4} &= -(1/2)(-Rq_0 - Qq_1 + Pq_2)
\end{aligned}$$

where $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$

3.2.3 Navigation Equations

The navigation equations in Eq. (3.1) can be solved as:

$$\dot{p_N} = (q_0^2 + q_1^2 - q_2^2 - q_3^2)U + 2(q_1q_2 - q_0q_3)V + 2(q_1q_3 + q_0q_2)W$$

$$\begin{aligned}
\dot{p_E} &= 2(q_1q_2+q_0q_3)U+(q_0^2-q_1^2+q_2^2-q_3^2)V+2(q_2q_3-q_0q_1)W \\
\dot{p_D} &= 2(q_1q_3-q_0q_2)U+2(q_2q_3+q_0q_1)V+(q_0^2-q_1^2-q_2^2+q_3^2)W
\end{aligned}$$

3.3 Aerodynamic Forces and Moments

The coefficients of aerodynamic forces and moments of the PDV with respect to the stability axes are required in the force and moment equations. These coefficients can be computed by linearly summing up all the components as follows [17, Stevens]:

$$\begin{aligned}
C_{x_s} &= -C_D \\
C_{y_s} &= C_{y_\beta}\beta + C_{y_p}\tfrac{P_s b}{2V_T} + C_{y_r}\tfrac{R_s b}{2V_T} + C_{y_{\delta_{vc}}}\delta_{vc} \\
C_{z_s} &= -C_L \\
C_{l_s} &= C_{l_\beta}\beta + C_{l_p}\tfrac{P_s b}{2V_T} + C_{l_r}\tfrac{R_s b}{2V_T} + C_{l_{\delta_{hd}}}\delta_{hd} + C_{l_{\delta_{vd}}}\delta_{vd} \\
C_{m_s} &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\tfrac{Q_s\bar{c}}{2V_T} + C_L(X_{cg}-X_{ref})\tfrac{1}{\bar{c}} + C_{m_{\delta_{hc}}}\delta_{hc} + C_{m_{\delta_{vd}}}\beta\delta_{vd} \\
&\quad -C_{D_{streamer}}(X_{total}-X_{cg})\tfrac{\alpha}{\bar{c}} \\
C_{n_s} &= C_{n_\beta}\beta + C_{n_p}\tfrac{P_s b}{2V_T} + C_{n_r}\tfrac{R_s b}{2V_T} + C_{y_s}(X_{cg}-X_{ref})\tfrac{1}{\bar{c}} + C_{n_{\delta_{vc}}}\delta vc + C_{n_{\delta_{hd}}}\alpha\delta_{hd} \\
&\quad -C_{D_{streamer}}(X_{total}-X_{cg})\tfrac{\beta}{b}
\end{aligned}$$

where

$$\begin{aligned}
P_s &= P\cos\alpha + R\sin\alpha \\
Q_s &= Q \\
R_s &= -P\sin\alpha + R\cos\alpha
\end{aligned}$$

and X_{cg} (the center of gravity location) is 0.26 meters, X_{ref} (the reference location about which aerodynamic coefficients are obtained) is 0.22 meters, and X_{total} (the end location where the streamer is pivoted) is 0.50 meters. All locations are measured from the tip of the PDV nose. In addition,

$$\begin{aligned}
C_L &= C_{L_0} + C_{L_\alpha}\alpha + C_{L_q}\tfrac{Q_s\bar{c}}{2V_T} + C_{L_{\delta_{hc}}}\delta_{hc} \qquad (Total\ \ Lift\ \ Coefficient) \\
C_D &= C_{D_{base}} + C_{D_{in,\alpha}} + C_{D_{in,fin}} + C_{D_{streamer}} \quad (Total\ \ Drag\ \ Coefficient)
\end{aligned}$$

where

$$\begin{aligned}
C_{D_{in,\alpha}} &= 1.282\ \alpha^2 \qquad\qquad\qquad\qquad (Induced\ \ Drag) \\
C_{D_{in,fin}} &= 0.617\ (\delta_{hc}^2+\delta_{vc}^2) + 0.637\ (\delta_{hd}^2+\delta_{vd}^2) \quad (Fin\ \ Drag)
\end{aligned}$$

When the wind is considered, V_A, α_A, and β_A which are described in Section 3.6 are used as substitutes for V_T, α, and β in calculating the aerodynamic forces and moments.

C_{Lo}	C_{m_o}	$C_{L\alpha}$	C_{m_α}	C_{y_β}	C_{l_β}	C_{n_β}	C_{y_p}	C_{l_p}
0.00	0.00	4.79	−3.60	−4.79	0.00	3.60	0.00	−17.2
C_{n_p}	C_{L_q}	C_{m_q}	C_{y_r}	C_{l_r}	C_{n_r}	$C_{L_{\delta_{hc}}}$	$C_{y_{\delta_{vc}}}$	$C_{l_{\delta_{hd}}}$
0.00	23.5	−33.7	23.5	0.0	−33.7	2.98	−2.98	2.46
$C_{l_{\delta_{vd}}}$	$C_{m_{\delta_{hc}}}$	$C_{n_{\delta_{vc}}}$	$C_{m_{\delta_{vd}}}$	$C_{n_{\delta_{hd}}}$	$C_{D_{base}}$	$C_{D_{streamer}}$		
2.46	−6.14	6.14	−0.6188	−0.6188	0.35	0.15		

Table 3.1: Stability Derivatives and Control Derivatives of the PDV

The aerodynamic coefficients of the PDV were obtained from both the vortex lattice method [24, Anderson] and from wind tunnel tests. Table 3.1 lists the stability derivatives and control derivatives used for the PDV dynamics modeling.

3.4 Servo Dynamics Models

As a result of performance tests of the servo system used for the PDV(the combination of a SV203B/C servo controller and four HS-81MG servos), the servo dynamics from the control surface deflection command to the actual deflection is approximated as a second-order low-pass filter with the undamped natural frequency of 50 rad/sec and the damping ratio of 1. The transfer function of the servo dynamics is as follows:

$$G_{servo}(s) = \frac{50^2}{s^2 + 100s + 50^2} \tag{3.9}$$

3.5 Sensor Models

Inertial sensors Rate gyro readings and accelerometer readings include biases and noises. The noise mainly comes from temperature fluctuations, A/D quantization, and EM noise in wires; they are all modeled as zero-mean gaussian white sequences [25][26]. By measuring outputs of inertial sensors, the following 1σ rms values were found and used for modeling the noises:

$$\sigma_{\nu_{a_x}} = \sqrt{E(\nu_{a_x}^2)} = 0.1\ m/sec^2 \quad \sigma_{\nu_{a_y}} = \sqrt{E(\nu_{a_y}^2)} = 0.1\ m/sec^2 \quad \sigma_{\nu_{a_z}} = \sqrt{E(\nu_{a_z}^2)} = 0.1\ m/sec^2$$
$$\sigma_{\nu_P} = \sqrt{E(\nu_P^2)} = 0.01\ rad/sec \quad \sigma_{\nu_Q} = \sqrt{E(\nu_Q^2)} = 0.01\ rad/sec \quad \sigma_{\nu_R} = \sqrt{E(\nu_R^2)} = 0.01\ rad/sec$$

In addition, it is assumed that the biases of both rate gyros and a 3-axis accelerometer are constant, because the mission time of the PDV is very short. The following constants are used as biases in the simulation:

$$\gamma_{a_x} = 0.1 \;\; m/sec^2 \quad \gamma_{a_y} = 0.1 \;\; m/sec^2 \quad \gamma_{a_z} = 0.1 \;\; m/sec^2$$
$$\gamma_P = 0.02 \;\; rad/sec \quad \gamma_Q = 0.02 \;\; rad/sec \quad \gamma_R = 0.02 \;\; rad/sec$$

Thus, in simulations, the measured accelerations and angular rates can be related to the true accelerations and angular rates as follows:

$$\begin{aligned} a_{x_m} &= a_x + \gamma_{a_x} + \nu_{a_x} \\ a_{y_m} &= a_y + \gamma_{a_y} + \nu_{a_y} \\ a_{z_m} &= a_z + \gamma_{a_z} + \nu_{a_z} \\ P_m &= P + \gamma_P + \nu_P \\ Q_m &= Q + \gamma_Q + \nu_Q \\ R_m &= R + \gamma_R + \nu_R \end{aligned} \tag{3.10}$$

Note that the six biases are considered as states in the state estimation algorithm and they are estimated by the navigation code. The state estimation algorithm is described in Chapter 4. Note also that the accelerometer measures the acceleration which results from the forces other than gravity. Thus, in the PDV flight dynamics simulation, the accelerations a_x, a_y, and a_z can be simply related to the aerodynamic forces F_x, Fy, and Fz represented in the body-fixed axis frame:

$$a_x = F_x/m, \quad a_y = F_y/m, \quad a_z = F_z/m$$

3-Axis Magnetic Sensor The noises in magnetic sensor readings are modeled as zero-mean gaussian white sequences. The magnetic sensor readings can be expressed as:

$$\mathbf{b}_{xyz_{meas}} = C^{-1}\mathbf{b}_{NED} + \nu_{\mathbf{b}} \tag{3.11}$$

This relation is equivalent to:

$$\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_{meas} = \underbrace{\begin{bmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{bmatrix} \begin{bmatrix} b_N \\ b_E \\ b_D \end{bmatrix}}_{\mathbf{h_3(x)}} + \begin{bmatrix} \nu_{b_x} \\ \nu_{b_y} \\ \nu_{b_z} \end{bmatrix}$$

In the Boston area, the intensities of the Earth magnetic field in the NED geographic frame are:

$$\begin{aligned} b_N &= 0.18961 \;\; gauss \\ b_E &= -0.05288 \;\; gauss \\ b_D &= 0.49777 \;\; gauss \end{aligned} \tag{3.12}$$

$\sigma_{\nu_{b_x}} = \sqrt{E(\nu_{b_x}^2)}$, $\sigma_{\nu_{b_y}} = \sqrt{E(\nu_{b_y}^2)}$, and $\sigma_{\nu_{b_z}} = \sqrt{E(\nu_{b_z}^2)}$ are set to 0.005 gauss in simulations.

Target Pixel Position Tracking The target pixel position tracking models are required for the software simulation, because in hardware-in-the-loop-simulation, a human operator should take part in the loop and track the target image by a mouse device. The target pixel position tracking models come from Eq. (4.12) and include zero-mean white measurement sequences of the target pixels:

$$\begin{aligned} pixel_x &= \underbrace{\frac{\{\mathbf{x}_{cam\to tar_C}\}_y}{\{\mathbf{x}_{cam\to tar_C}\}_x} \times \frac{1}{tan\frac{\alpha_x}{2}} \times \frac{N'_x}{2}}_{h_1(x)} + \nu_{pixel_x} \\ pixel_y &= \underbrace{-\frac{\{\mathbf{x}_{cam\to tar_C}\}_z}{\{\mathbf{x}_{cam\to tar_C}\}_x} \times \frac{1}{tan\frac{\alpha_y}{2}} \times \frac{N'_y}{2}}_{h_2(x)} + \nu_{pixel_y} \end{aligned} \tag{3.13}$$

where both $\sigma_{\nu_{pixel_x}} = \sqrt{E(\nu_{pixel_x}^2)}$ and $\sigma_{\nu_{pixel_y}} = \sqrt{E(\nu_{pixel_y}^2)}$ are three pixels in size.

3.6 Wind Models

A first order low pass filter with time constant at 2 seconds served as the shaping filter of the wind disturbance, and up to 5 m/sec of wind speed is usually simulated. When including the local wind that covers much larger areas than the size of the vehicle, the wind velocity components (U_g, V_g, W_g) along the vehicle body-fixed axes cause aerodynamic forces and moments as follows [23, Roskam]:

$$\begin{aligned} V_A &= \sqrt{(U - U_g)^2 + (V - V_g)^2 + (W - W_g)^2} \\ \alpha_A &= \alpha + \alpha_g \\ \beta_A &= \beta + \beta_g \end{aligned}$$

where

$$\alpha_g = -\frac{W_g}{V_A}, \quad \beta_g = \frac{V_g}{V_A}$$

Note that when the wind is considered, V_A, α_A, and β_A are used as substitutes for V_T, α, and β in calculating the aerodynamic forces and moments which are described in Section 3.3.

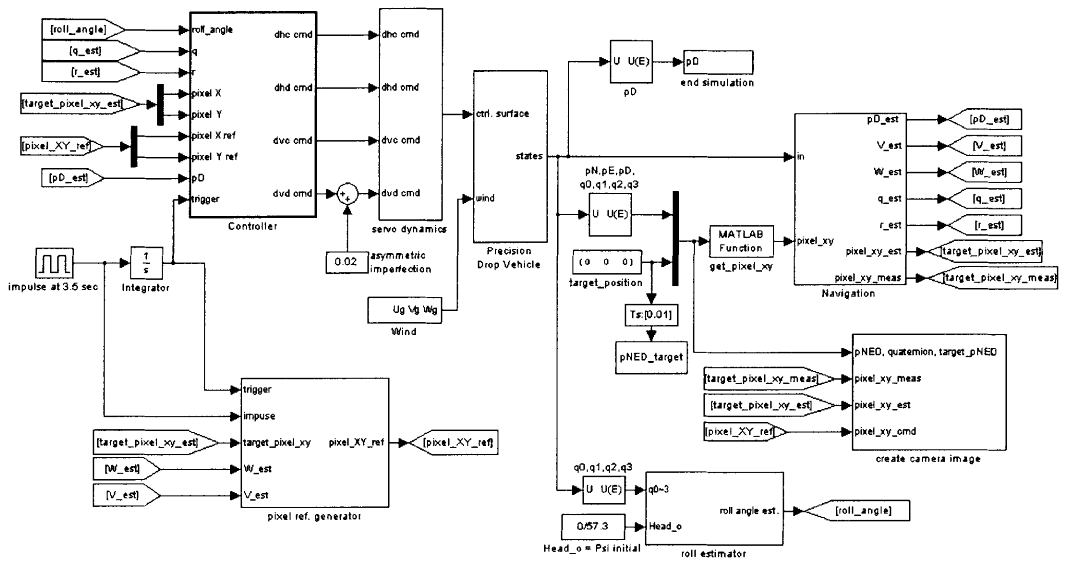


Figure 3-1: Software Simulation Framework

3.7 Software Simulation

Based on this PDV dynamic model, a software simulation was developed in the MATLAB Simulink environment. The software simulation contains all the components required for the simulation of the precision airdrop framework as software forms: PDV dynamics, wind models, servo models, sensor (inertial sensors and a 3-axis magnetic sensor) models, target pixel position tracking models, and state estimation and control. Note that hardware-in-the-loop-simulation (HILSIM) contains all the possible avionics components and software in the loop and uses the same I/O signal lines as real systems. Commonly, a software simulation is developed earlier than a hardware-in-the-loop simulation (HILSIM) and is used for designing the state estimation and control loops.

Figure 3-1 shows the software simulation framework designed in the MATLAB Simulink environment. For the PDV flight dynamics and the state estimation and control loops, several S-function MATLAB files were coded. The Simulink diagrams are described in Appendix B.

Chapter 4

Estimation and Controller Design

In this chapter, we shall first develop the state estimation algorithm, which is an extended Kalman filter based on the data from inertial sensors, magnetic sensor measurements, and the target pixel position measurements from the ground station. We shall then investigate the control approach for the precision airdrop.

The state estimation and the control are tightly coupled for several reasons:

(1) A stabilized camera image is required for the human operator to track the target position and send the meaningful target pixel location.

(2) Vehicle states are estimated from the target position, and these feed into the pitch/yaw dampers and the axial controller.

4.1 State Estimation

The dynamic model of the airdrop vehicle is described by a set of first-order nonlinear differential equations expressed in the state-space form and the measurements for the state estimation are also nonlinear functions of the states. The extended Kalman filter is a commonly used means to calculate state estimates for such a nonlinear case [18, Gelb].

4.1.1 Extended Kalman Filter Setup

In the extended Kalman filter setup for the airdrop vehicle, the state estimates must be time-propagated by integrating the actual nonlinear differential equations at each sampling time step, instead of using the linearized system dynamics matrix or the state transition matrix. However, the error covariance matrix is time-propagated by means of the system dynamics matrix and the same continuous-time error covariance propagation equation to that of the linear case. During the time-propagation of the state estimates and the error covariance matrix, the information from the inertial sensors, such as angular rates and accelerations, is applied.

In the stage of the measurement incorporation into the extended Kalman filtering, the target pixel positions tracked on the ground station and the magnetic sensor measurement data are used for improving the state estimates. Because the measurements are nonlinear functions of states the residual, which can be multiplied by the Kalman gain to improve the state estimate vector, is the difference between the actual measurement and the calculated value of the nonlinear measurement equation. Note that the linearized measurement matrix is not used for calculating the residual.

In order to compute the Kalman gain for the measurement update, it is necessary to linearize the nonlinear measurement equation about the state estimate vector in order to obtain the measurement matrix. Also, the error covariance matrix is updated using the measurement matrix and the computed Kalman gain. Even though, in the extended Kalman filter, the error covariance matrix is approximate [20, Zarchan], it is effective in calculating Kalman gains.

Regarding our extended Kalman filter design, two kinds of filtering algorithms were applied:

- continuous-time propagation of the estimated state vector, continuous-time propagation of the error covariance matrix, and discrete measurement updates by the Joseph formulation [18, Gelb]

- continuous-time propagation of the estimated state vector, discrete-time propagation of the error covariance matrix, and discrete measurement updates by the Carlson's square root filter formulation [16, Carlson]

The former algorithm was used for the software simulation and the latter algorithm was applied to the hardware-in-the-loop-simulation and the PDV computer software.

Dynamics Equations

The dynamic model of the airdrop vehicle, near the surface of the Earth and assuming a flat Earth, can be applied in the following way.

$$\begin{aligned}\dot{\mathbf{p}}_L &= C\mathbf{v}_B && (\textit{Navigation Equations})\\ \dot{\mathbf{v}}_B &= \mathbf{a}+C^{-1}\mathbf{g}-\Omega_3\mathbf{v}_B && (\textit{Force Equations})\\ \dot{\mathbf{q}} &= -\tfrac{1}{2}\Omega_4\mathbf{q} && (\textit{Kinematic Equations})\end{aligned} \tag{4.1}$$

where the local inertial frame is the NED(North-East-Down) geographic frame. Note that $\mathbf{q}$ is the quaternion vector which has four quaternion variables of vehicle attitude, and the constraint of the quaternion vector is $||\mathbf{q}||_2 = 1$.

Next, the acceleration and the angular rates in Eq. (4.1) can be related to accelerometer readings and rate gyro readings with acceleration biases and rate gyro biases as follows.

$$\begin{aligned}\mathbf{a}_m &= \mathbf{a}+\gamma_{\mathbf{a}}+\nu_{\mathbf{a}}\\ \Omega_{3_m} &= \Omega_3+\gamma_{\Omega_3}+\nu_{\Omega_3}\\ \Omega_{4_m} &= \Omega_4+\gamma_{\Omega_4}+\nu_{\Omega_4}\end{aligned} \tag{4.2}$$

Substituting Eq. (4.2) into Eq. (4.1), the following nonlinear filter dynamics in the state-space form can be obtained.

$$\begin{aligned}\dot{\mathbf{p}}_L &= \underbrace{C\mathbf{v}_B}_{\mathbf{f}_1(\mathbf{x})}\\ \dot{\mathbf{v}}_B &= \underbrace{(\mathbf{a}_m-\gamma_{\mathbf{a}})+C^{-1}\mathbf{g}-(\Omega_{3_m}-\gamma_{\Omega_3})\mathbf{v}_B}_{\mathbf{f}_2(\mathbf{x})}+\mathbf{w}_I\\ \dot{\mathbf{q}} &= \underbrace{-\frac{1}{2}(\Omega_{4_m}-\gamma_{\Omega_4})\mathbf{q}}_{\mathbf{f}_3(\mathbf{x})}+\mathbf{w}_{II}\\ \dot{\gamma}_{P,Q,R} &= \underbrace{0}_{\mathbf{f}_4(\mathbf{x})}+\mathbf{w}_{III}\\ \dot{\gamma}_{a_x,a_y,a_z} &= \underbrace{0}_{\mathbf{f}_5(\mathbf{x})}+\mathbf{w}_{IV}\end{aligned} \tag{4.3}$$

The state vector contains 16 elements: three north-east-down position coordinates of the vehicle relative to the NED geographic frame ($\mathbf{p}_L$), three translational velocity coordinates of the vehicle relative to the vehicle body-fixed axis frame ($\mathbf{v}_B$), four quaternion variables of vehicle attitude ($\mathbf{q}$), three rate gyro biases ($\gamma_{P,Q,R}$), and three accelerometer biases (γ_{a_x,a_y,a_z}).

$$\begin{aligned}\mathbf{x}^T &= \left[\mathbf{p}_L^T\ \mathbf{v}_B^T\ \mathbf{q}^T\ \gamma_{P,Q,R}^T\ \gamma_{a_x,a_y,a_z}^T\right]\\ &= \left[x_L\ y_L\ z_L\ u\ v\ w\ q_0\ q_1\ q_2\ q_3\ \gamma_P\ \gamma_Q\ \gamma_R\ \gamma_{a_x}\ \gamma_{a_y}\ \gamma_{a_z}\right]^T\end{aligned} \tag{4.4}$$

Eq. (4.3) can also be interpreted as the standard form of nonlinear filter dynamics model:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{w} \tag{4.5}$$

where $\mathbf{w} = \begin{bmatrix} 0_{1\times3} \ \mathbf{w}_I^T \ \mathbf{w}_{II}^T \ \mathbf{w}_{III}^T \ \mathbf{w}_{IV}^T \end{bmatrix}^T$ is a vector valued zero-mean white process noise described by the process noise matrix Q, which is defined as

$$Q = E(\mathbf{w}\mathbf{w}^T)$$

Measurements

Our extended Kalman filtering scheme uses two kinds of measurement updates: target pixel positions tracked by the ground station and magnetic sensor data.

The measured values of target pixel positions, tracked by the camera images in the ground station software are essential information for vision-based measurement updates in our estimation algorithm. For this, it is required to relate the target pixel location to the vehicle position and attitude, and the target location. From Figure 4-1,

$$\mathbf{x}_{cam\rightarrow tar_L} = \mathbf{x}_{target_L} - \mathbf{p}_L - \mathbf{r}_{cam_L} \tag{4.6}$$

L denotes that a vector is represented relative to the NED local inertial frame. Actually $\mathbf{x}_{target_L}$ is zero in our estimation setup, since the target position is always considered as the origin of our local inertial frame. That is to say, we can always take the NED geographic frame with the origin at the target position as our local inertial frame.

If the camera frame and the vehicle body frame are parallel, then the relation can be written as:

$$\mathbf{x}_{cam\rightarrow tar_C} = C^{-1}(\mathbf{x}_{target_L} - \mathbf{p}_L) - \mathbf{r}_{cam_B} \tag{4.7}$$

The subscript B denotes that a vector is represented relative to the body frame, and the subscript C denotes that a vector is represented relative to the camera frame. Note that if the body frame and camera frame are not parallel, then another rotational transformation matrix from body to camera frames should be multiplied on the right hand side.

In addition, as shown in Figure 4-2, the unit vector in the camera frame is:

$$\hat{e}_C = \frac{\mathbf{x}_{cam\rightarrow tar_C}}{|\mathbf{x}_{cam\rightarrow tar_C}|}$$

Denoting the camera field-of-view angle in the horizontal image axis by α_x, the camera field-of-view angle in the vertical image axis by α_y, the camera image resolution in the

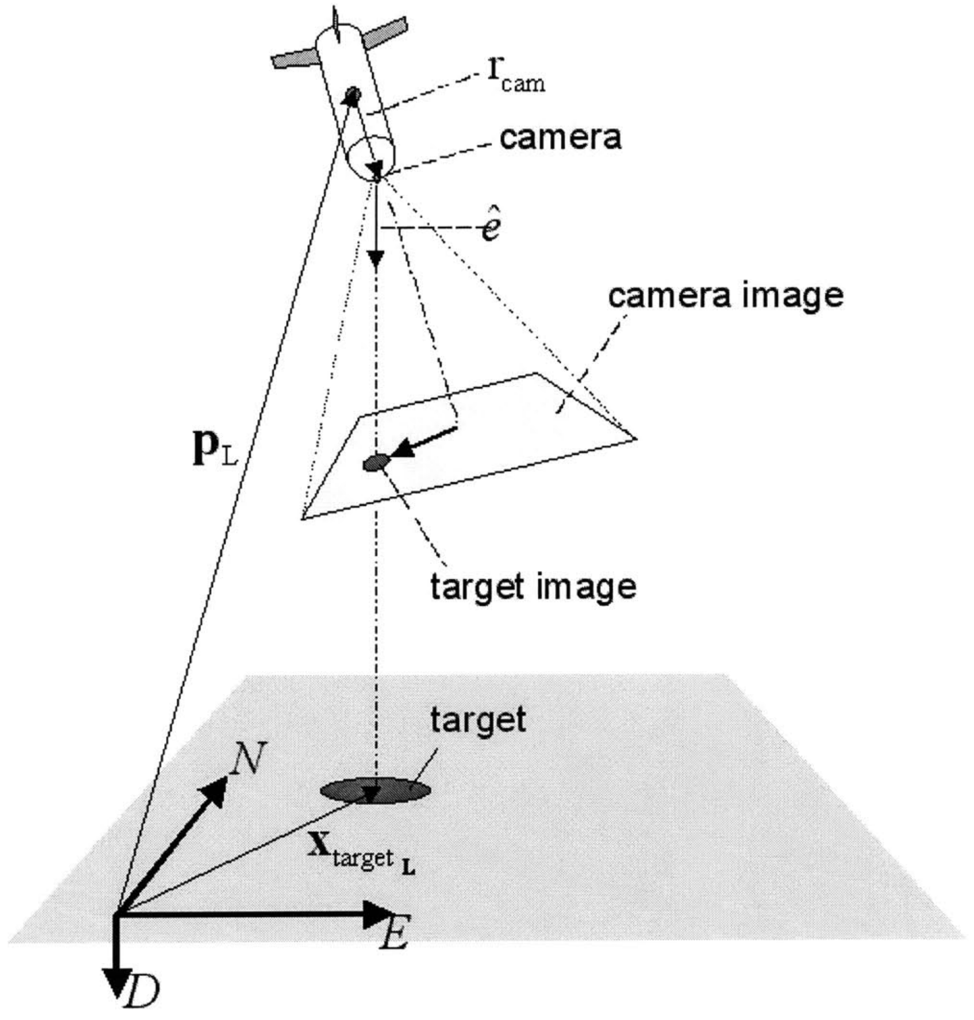


Figure 4-1: NED frame and Relation of Vectors: L denotes that a vector is represented relative to the NED local inertial frame

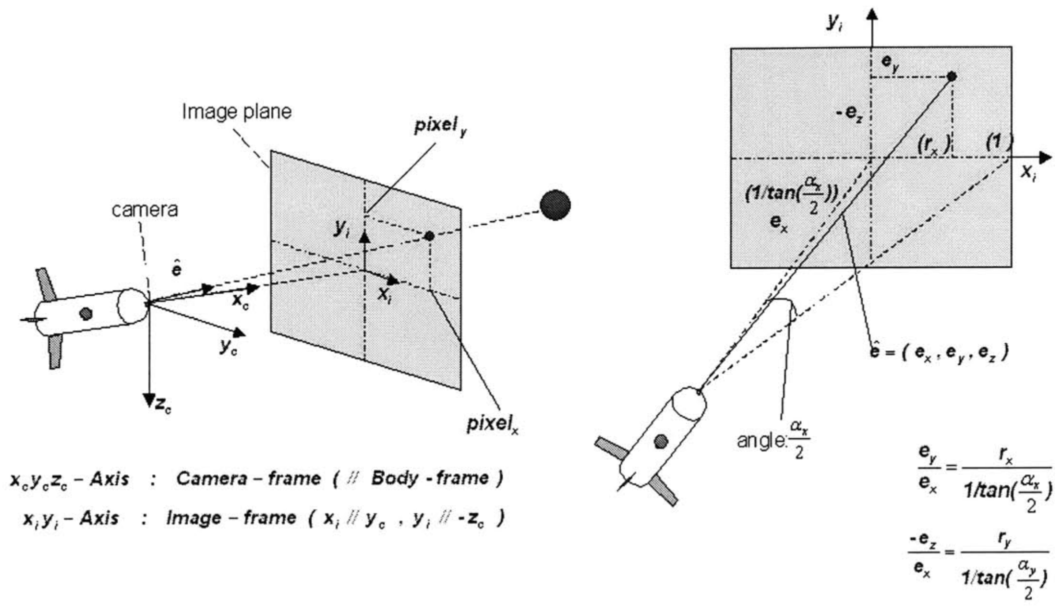


Image plane
pixel$_y$
camera
$\hat{e}$
y_i
x_c
x_i
y_c
z_c
pixel$_x$
$x_c y_c z_c$ - Axis : Camera - frame (// Body - frame)
$x_i y_i$ - Axis : Image - frame (x_i // y_c , y_i // $-z_c$)
y_i
e_y
$-e_z$
(r_x)
(1)
x_i
$(1/tan(\frac{\alpha_x}{2}))$
e_x
$\hat{e} = (e_x, e_y, e_z)$
angle: $\frac{\alpha_x}{2}$
$\frac{e_y}{e_x} = \frac{r_x}{1/tan(\frac{\alpha_x}{2})}$
$\frac{-e_z}{e_x} = \frac{r_y}{1/tan(\frac{\alpha_y}{2})}$

Figure 4-2: Camera frame, Image frame, and Unit Vector of Target Location in Camera Frame

horizontal image axis by N_x, and the camera image resolution in the vertical image axis by N_y, the following relation between the pixel location of the target and the vehicle position and attitude is formed:

$$pixel_x = r_x \times \frac{N_x}{2} \tag{4.8}$$

where r_x is the ratio of the distance from the image center to the projected location of the target image onto the horizontal image axis relative to half the horizontal size of the image:

$$r_x = \frac{\{\hat{e}_C\}_y}{\{\hat{e}_C\}_x} \times \frac{1}{tan\frac{\alpha_x}{2}} = \frac{\{\mathbf{x}_{cam\rightarrow tar_C}\}_y}{\{\mathbf{x}_{cam\rightarrow tar_C}\}_x} \times \frac{1}{tan\frac{\alpha_x}{2}} \tag{4.9}$$

Similarly,

$$pixel_y = r_y \times \frac{N_y}{2} \tag{4.10}$$

where r_y is the ratio of the distance from the image center to the projected location of the target image onto the vertical image axis relative to half the vertical size of the image:

$$r_y = -\frac{\{\hat{e}_C\}_z}{\{\hat{e}_C\}_x} \times \frac{1}{tan\frac{\alpha_y}{2}} = -\frac{\{\mathbf{x}_{cam\rightarrow tar_C}\}_z}{\{\mathbf{x}_{cam\rightarrow tar_C}\}_x} \times \frac{1}{tan\frac{\alpha_y}{2}} \tag{4.11}$$

Note that the ratios r_x and r_y do not depend on the camera image resolutions N_x and N_y, but depend on the camera field-of-view angles α_x and α_y. Moreover, if the resolutions of the image window on the ground station software are N'_x and N'_y, the target pixel location are:

$$pixel_x = r_x \times \frac{N'_x}{2}, \quad pixel_y = r_y \times \frac{N'_y}{2}$$

The low-cost camera used for our hardware-in-the-loop simulation and flight test has:

$$\alpha_x = 74 \;\; deg, \;\; \alpha_y = 59 \;\; deg, \;\; N_x = 640, \;\; N_y = 480$$

and the resolutions of the image window on the ground station software are set to the same values as the camera image resolutions:

$$N'_x = 640, \qquad N'_y = 480$$

Regarding the measured pixel location on the ground station, the measurement noise terms should be added as follows:

$$\begin{aligned} pixel_x &= \underbrace{\frac{\{\mathbf{x}_{cam\rightarrow tar_C}\}y}{\{\mathbf{x}_{cam\rightarrow tar_C}\}x} \times \frac{1}{tan\frac{\alpha_x}{2}} \times \frac{N'_x}{2}}_{h_1(x)} + \nu_{pixel_x} \\ pixel_y &= \underbrace{-\frac{\{\mathbf{x}_{cam\rightarrow tar_C}\}z}{\{\mathbf{x}_{cam\rightarrow tar_C}\}x} \times \frac{1}{tan\frac{\alpha_y}{2}} \times \frac{N'_y}{2}}_{h_2(x)} + \nu_{pixel_y} \end{aligned} \tag{4.12}$$

Since the vision-based measurements are discrete, the nonlinear vision-based measurement equation can be written as:

$$\mathbf{z_{img,k}} = \mathbf{h_{img}}(\mathbf{x_k}) + \mathbf{v_{img,k}} \tag{4.13}$$

where $\mathbf{z_{img,k}} = [pixel_{x,k} \; pixel_{y,k}]^T$, and $\mathbf{v_{img,k}} = \left[\nu_{pixel_{x,k}} \; \nu_{pixel_{y,k}}\right]^T$ is a zero-mean white measurement noise described by the measurement noise covariance matrix R_{img}, which is defined as

$$R_{img} = E(\mathbf{v_{img,k}} \mathbf{v_{img,k}}^T)$$

Now consider magnetic sensor measurements. The measurements from a 3-axis magnetic sensor can be expressed as:

$$\mathbf{b}_{xyz_{meas}} = C^{-1}\mathbf{b}_{NED} + \nu_{\mathbf{b}} \tag{4.14}$$

This relation is equivalent to:

$$\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_{meas} = \underbrace{\begin{bmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{bmatrix} \begin{bmatrix} b_N \\ b_E \\ b_D \end{bmatrix}}_{\mathbf{h_3(x)}} + \begin{bmatrix} \nu_{b_x} \\ \nu_{b_y} \\ \nu_{b_z} \end{bmatrix}$$

The Earth's magentic field intensity is about 0.5 to 0.6 gauss and has a component parallel to the earth's surface that always points toward magnetic north. In the Boston area, the intensities of the Earth magnetic field in the NED geographic frame are:

$$\begin{aligned} b_N &= 0.18961 \ gauss \\ b_E &= -0.05288 \ gauss \\ b_D &= 0.49777 \ gauss \end{aligned} \tag{4.15}$$

Since this magnetic sensor measurements are discrete, the nonlinear magnetic sensor measurement equation can be written as:

$$\mathbf{z_{mag,k}} = \mathbf{h_{mag}}(\mathbf{x_k}) + \mathbf{v_{mag,k}} \tag{4.16}$$

where $\mathbf{z_{mag,k}} = [b_{x,k} \ b_{y,k} \ b_{z,k}]^T$, $\mathbf{v_{mag,k}} = \left[\nu_{b_x,k} \ \nu_{b_y,k} \ \nu_{b_z,k}\right]^T$ is a zero-mean white measurement noise described by the measurement noise covariance matrix R_{mag}, which is defined as

$$R_{mag} = E(\mathbf{v_{mag,k}} \mathbf{v_{mag,k}}^T)$$

Estimated State and Covariance Propagation

In order to compute time-propagation of the error covariance matrix by applying the same continuous-time error covariance propagation equation to that of the linear case, the system dynamics matrix F, which is the first-order approximation to the nonlinear dynamics equations, should first be obtained. The system dynamics matrix F is:

$$\begin{aligned} F &= \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \\ &= \begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{14} & F_{15} \\ F_{21} & F_{22} & F_{23} & F_{24} & F_{25} \\ F_{31} & F_{32} & F_{33} & F_{34} & F_{35} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \end{aligned}$$

where the elements are given in Appendix A.1. Note that the state transition matrix, required for the discrete Riccati equations, can be obtained from the system dynamics matrix. We shall see this when we use the square root filter formulation in discrete time. Note also that the system dynamics matrix or the state transition matrix should not be applied to time-propagation of the state estimates.

The process noise is described as:

$$\mathbf{w} = \begin{bmatrix} 0_{1\times 3} \; \mathbf{w}_I^T \; \mathbf{w}_{II}^T \; \mathbf{w}_{III}^T \; \mathbf{w}_{IV}^T \end{bmatrix}^T$$

where $\mathbf{w}_I$ and $\mathbf{w}_{II}$ are

$$\mathbf{w}_I = -\nu_{\mathbf{a}} + \nu_{\Omega_3}\mathbf{v}_B, \qquad \mathbf{w}_{II} = \frac{1}{2}\nu_{\Omega_4}\mathbf{q} \tag{4.17}$$

The process noise matrix is defined as:

$$\begin{aligned} Q &= E\left[\mathbf{w}\mathbf{w}^T\right] \\ &= \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & E[\mathbf{w}_I\mathbf{w}_I^T] & E[\mathbf{w}_I\mathbf{w}_{II}^T] & 0_{3\times3} & 0_{3\times3} \\ 0_{4\times3} & E[\mathbf{w}_{II}\mathbf{w}_I^T] & E[\mathbf{w}_{II}\mathbf{w}_{II}^T] & 0_{4\times3} & 0_{4\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & E[\mathbf{w}_{III}\mathbf{w}_{III}^T] & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} & E[\mathbf{w}_{IV}\mathbf{w}_{IV}^T] \end{bmatrix} \end{aligned} \tag{4.18}$$

where the elements are given in Appendix A.2. Note that $\mathbf{w}_{III}$ and $\mathbf{w}_{IV}$ are small white process noises used for protecting the Kalman gains from approaching zero and to prevent filter divergence. Moreover, small white noises model the biases of rate gyros and accelerometers as brownian motion and also describe how the bias drifts with time [21].

When it comes to the continuous-time propagation of the estimated state vector $\hat{x}$, the state estimates have to be integrated with the use of the actual nonlinear differential equations instead of using the approximations for the system dynamics. In our setup, the fourth-order Runge-Kutta numerical integration method is applied to the following nonlinear dynamic model of the estimated state vector:

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}(\hat{\mathbf{x}}) \tag{4.19}$$

In addition, the continuous-time propagation of the error covariance matrix P can be computed by integrating the following equation at each time step:

$$\dot{P} = FP + PF^T + Q \tag{4.20}$$

where

$$F = \left(\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\right)_{\mathbf{x}=\hat{\mathbf{x}}}, \qquad Q = Q_{\mathbf{x}=\hat{\mathbf{x}}}$$

As in the case of numerical integration of the estimated state vector, the fourth-order Runge-Kutta numerical integration method was applied here.

Measurement Update

First consider vision-based measurement updates. Target pixel locations tracked by an human operator on the ground station are sent to the PDV computer through an 802.11b wireless channel at 40 Hz frequency. The nonlinear vision-based measurement equation is described as Eq. (4.12) or Eq. (4.13). Thus, the linearized measurement matrix for vision-based measurement updates in the extended Kalman filter is as follows:

$$H_{img} = \frac{\partial \mathbf{h_{img}}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} H_{11} & 0_{1\times3} & H_{13} & 0_{1\times3} & 0_{1\times3} \\ H_{21} & 0_{1\times3} & H_{23} & 0_{1\times3} & 0_{1\times3} \end{bmatrix}$$

where the elements are given in Appendix A.3.

In addition, the measurement noise matrix for vision-based measurement updates is described as:

$$R_{img} = E(\mathbf{v_{img,k}} \mathbf{v_{img,k}}^T) = \begin{bmatrix} \sigma^2_{\nu_{pixel_x}} & 0 \\ 0 & \sigma^2_{\nu_{pixel_y}} \end{bmatrix}$$

In our extended Kalman filter setup, both $\sigma_{\nu_{pixel_x}}$ and $\sigma_{\nu_{pixel_y}}$ are three pixels. These values were determined by considering the filter bandwidth as well as the measurement itself. Note that larger standard deviations of measurement noise tend to result in smaller Kalman gains, that is, decreased filter bandwidth. In other words, if the larger standard deviations of measurement noises are specified in the measurement noise matrix, this tends to increase the time constant of the filter dynamics and delay the response to measurements.

Regarding the magnetic sensor measurements, which are taken at 100 Hz frequency, the nonlinear magnetic sensor measurement equation can be described as Eq. (4.16). Therefore, the corresponding measurement matrix can be linearized as follows:

$$H_{mag} = \frac{\partial \mathbf{h_{mag}}(\mathbf{x})}{\partial \mathbf{q}} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & H_3 & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{4.21}$$

where the elements are given in Appendix A.4. In addition, the measurement noise matrix for vision-based measurement updates is described as:

$$R_{mag} = E(\mathbf{v_{mag,k}} \mathbf{v_{mag,k}}^T) = \begin{bmatrix} \sigma^2_{\nu_{b_x}} & 0 & 0 \\ 0 & \sigma^2_{\nu_{b_y}} & 0 \\ 0 & 0 & \sigma^2_{\nu_{b_z}} \end{bmatrix}$$

$\sigma_{\nu_{b_x}}$, $\sigma_{\nu_{b_y}}$ and $\sigma_{\nu_{b_z}}$ are set to 0.005 gauss in our extended Kalman filter setup.

Both vision-based measurements and magnetic sensor measurements are discrete. The discrete measurement updates are applied to the extended Kalman filtering scheme as follows:

$$\hat{\mathbf{x}}^{+} = \hat{\mathbf{x}} + K(\mathbf{z_k} - \mathbf{h}(\hat{\mathbf{x}})) \tag{4.22}$$

where

$$K = PH^T(HPH^T + R)^{-1}, \qquad H = \left(\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right)_{\mathbf{x}=\hat{\mathbf{x}}} \tag{4.23}$$

and

$$P^{+} = (I - KH)P \tag{4.24}$$

or

$$P^{+} = (I - KH)P(I - KH)^T + KRK^T \tag{4.25}$$

The superscript + refers to values after the measurement updates. Note that Eq. (4.25) suggested by Joseph [18, Gelb] is more numerically robust in maintaining a symmetric positive semi-definite covariance matrix than Eq. (4.24) originally derived by Kalman. However, the Joseph formulation requires notably higher computation cost than the Kalman standard formulation. For this reason, the Joseph formulation was applied to only our software simulation.

4.1.2 Triangular Formulation of the Square Root Filter

Carlson's square root filter maintains the covariance square root matrix in triangular form during both time-propagation and measurement update. The triangular formulation of the square root filter provides a significant improvement in computational speed, and greatly reduces the existing disadvantage in high-order filter designs [16, Carlson]. Taking numerical precision and stability, computational speed, and memory storage into consideration, the Carlson's square root filter was used as the estimation algorithm in our hardware-in-the-loop simulation and the PDV computer software.

Triangular Square Root - Time Propagation

The continuous-time propagation of the estimated state vector in our triangular filtering setup was done in such a way that the estimated state vector was integrated with the use of the actual nonlinear differential equations. The fourth-order Runge-Kutta numerical integration method was applied to Eq. (4.19). This approach is essentially the same as other extended Kalman filtering algorithms.

The triangular formulation of the square root filter starts with the following definition on the error covariance square root S:

$$P = SS^T \tag{4.26}$$

where P is a symmetric positive semi-definite error covariance matrix and S is the unique upper-triangular square root of P by Cholesky decomposition.

When propagating the error covariance matrix in a discrete-time manner, the following discrete-time propagation equation is properly used:

$$P_k = \phi_k \; P_{k-1} \; \phi_k^T \; + \; Q_{d,k} \tag{4.27}$$

where ϕ_k is the state transition matrix and $Q_{d,k}$ is the discrete-time process noise matrix (the subscripts k are referring to t_k).

The square root time update analogous to Eq. (4.27) is derived by means of a matrix RSS(Root-Sum-Square) operation as follows:

$$W = \phi S \tag{4.28}$$

$$S' = (WW^T \; + \; Q_d)^{1/2} \tag{4.29}$$

where S' refers to an updated value (the subscripts k and k-1 are omitted hereafter) and Q_d is the discrete-time process noise matrix. The matrix square root in Eq. (4.28) is computed by Cholesky decomposition in upper-triangular form.

The state transition matrix ϕ can be derived from the system dynamics matrix F in Eq. (4.17), and the discrete-time process noise matrix Q_d can also be found from the continuous-time process noise matrix Q in Eq. (4.18).

$$\phi = e^{FT_s} \simeq I + F \, T_s \tag{4.30}$$

$$Q_d = \int_0^{T_s} e^{F\tau} \, Q \, e^{F^T\tau} dt \simeq \; Q \, T_s \tag{4.31}$$

where T_s is the sampling time which is 0.01 sec in our filtering setup.

Triangular Square Root - Measurement Update

The square root formulation, analogous to the Kalman measurement update algorithm developed by Potter for the case of scalar measurements, is described as follows: in the case of scalar measurements $z, \mathbf{h}^T$ and r replace $\mathbf{z}, H$ and R, respectively, and Eq. (4.24) can be rewritten as:

$$S^+ S^{+T} = S \left[I - \mathbf{f}\mathbf{f}^T / \alpha \right] S^T \tag{4.32}$$

where

$$\begin{aligned} \mathbf{f} &= S^T \mathbf{h} && (4.33) \\ \alpha &= r + \mathbf{f}^T \mathbf{f} = r + \mathbf{h}^T P \mathbf{h} && (4.34) \end{aligned}$$

The triangular square root measurement update algorithm, proposed by Carlson, is described as follows. The upper-triangular square root solution S^+ to Eq. (4.32) can be written as:

$$\begin{aligned} S^+ &= SA && (4.35) \\ A &= \left[I - \mathbf{f}\mathbf{f}^T / \alpha \right]^{1/2} && (4.36) \end{aligned}$$

A is an upper-triangular matrix given by an analytic Cholesky decomposition as follows:

$$A = \mathbf{a}^D - \mathbf{f}^* \mathbf{c}^D \tag{4.37}$$

where $\mathbf{a}^D$ and $\mathbf{c}^D$ denote diagonal matrices having $\mathbf{a}$ and $\mathbf{c}$ as the main diagonals, with elements a_i and c_i computed from successive partial sums of α

$$\begin{aligned} &\alpha_0 = r \\ &\alpha_i = \alpha_{i-1} + f_i^2 \\ &a_i = (\alpha_{i-1}/\alpha_i)^{1/2} \\ &c_i = f_i / (\alpha_{i-1}\alpha_i)^{1/2} \end{aligned} \tag{4.38}$$

Here $\alpha_n \equiv \alpha$ and $i = 1, \cdots, n$. The term $\mathbf{f}^*$ denotes an upper-triangular matrix with columns 1 to n composed of successive partial $\mathbf{f}$ vectors

$$\mathbf{f}^* = [\ \mathbf{0}\ \ \mathbf{f}^{(1)}\ \ldots\ \mathbf{f}^{(n-1)}\ ], \quad \mathbf{f}^{(i)} \equiv [\ \underbrace{f_1 \ \cdots \ f_i}_{i} \ \underbrace{\mathbf{0}}_{n-i}\ ]^T \tag{4.39}$$

Therefore, the updated square root $S^+ = SA$ can be computed by recursive calculation of the successive columns $\mathbf{b_i} \equiv S\mathbf{f}^i$ of $S\mathbf{f}^*$ as follows:

$$\begin{aligned} \mathbf{b}_0 &= \mathbf{0} \\ \mathbf{b}_i &= \mathbf{b}_{i-1} + \mathbf{S}_i f_i \\ \mathbf{S}_i^+ &= \mathbf{S}_i a_i - \mathbf{b}_{i-1} c_i \end{aligned} \tag{4.40}$$

where $\mathbf{S}_i$ denotes the ith column of S, and $i = 1, \cdots, n$. Note that $\mathbf{S}_i$ and $\mathbf{b}_i$ consist of zeroes below the ith element. Moreover, the column vector $\mathbf{b}_n$ is related to the Kalman gain as follows:

$$\mathbf{b}_n = \sum_{i=1}^{n} \mathbf{S}_i f_i = S\mathbf{f} \equiv \mathbf{k}\alpha = \mathbf{k}\alpha_n \tag{4.41}$$

Thus the measurement update of estimate state vector is computed by

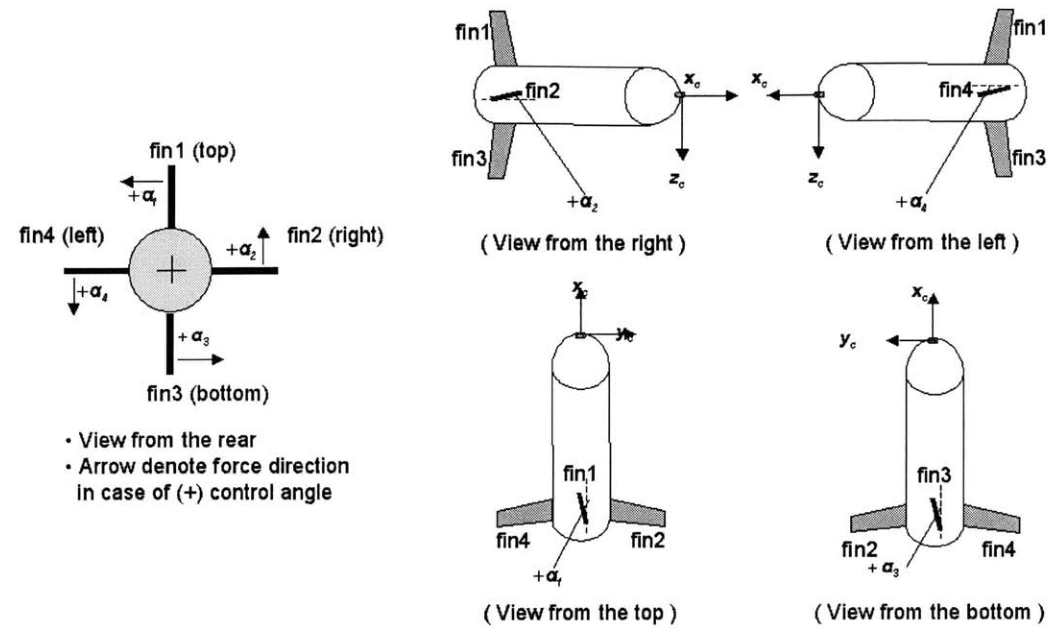
$$\hat{\mathbf{x}}^+ = \hat{\mathbf{x}} + \mathbf{b}_n(z - \mathbf{h}^T\hat{\mathbf{x}})/\alpha_n \tag{4.42}$$

Note that simultaneous measurements can be incorporated one at a time as scalars whenever random errors in the measurements are uncorrelated.

Carlson's triangular square root filter was used as the estimation algorithm in our hardware-in-the-loop simulation and the PDV computer software. At each time step, whenever any measurement data from vision-based measurements or magnetic sensor measurements are available, the scalar measurement update procedure is repetitively applied to each scalar measurement. In addition, it turned out in our simulation that in each time step the magnetic sensor measurement matrix H_{mag} should be constructed once for all b_x, b_y and b_z measurement updates. Each corresponding row vector of $H_{mag,k}$, which is obtained from $\hat{\mathbf{x}}_\mathbf{k}$ or the state estimated vector right after kth time-propagation, should be applied to b_x, b_y and b_z measurement update respectively, instead of reconstructing H_{mag} whenever one scalar measurement update is processed. A similar description also applies to vision-based measurements. In each time step, the vision-based measurement matrix H_{img} is calculated only once for all $pixel_x$ and $pixel_y$ measurement updates.

4.2 Controller Design

Four control fins of the PDV are divided into two sets: two vertical control fins and two horizontal control fins. The two fin sets can be used either collectively or differentially.



$\delta_{vd} = -\frac{1}{2}(\alpha_1 + \alpha_3)$, *vertical differential mode*

$\delta_{vc} = \frac{1}{2}(\alpha_1 - \alpha_3)$, *vertical collective mode*

$\delta_{hd} = -\frac{1}{2}(\alpha_2 + \alpha_4)$, *horizontal differential mode*

$\delta_{hc} = \frac{1}{2}(\alpha_2 - \alpha_4)$, *horizontal collective mode*

(+)δ_{vd}, (+) δ_{hd} : clockwise rotation (view from the rear)
(+) rotation relative to x_c-axis, (+) P

(+) δ_{vc} : left force (view from the rear),
(+) rotation relative to z_c-axis, (+) R

(+) δ_{hc} : upper force (view from the rear),
(-) rotation relative to y_c-axis, (-) Q

Figure 4-3: Definition of Contol Angles, and Collective and Differential Modes

Figure 4-3 shows the definition of the sign of control angles, collective modes and differential modes.

The six degree-of-freedom PDV dynamics are linearized at several altitudes, all with θ = - 90 deg (nose down). The Matlab *linmod* function was used for obtaining the state-space linear model of the dynamics of the nonlinear differential equations, described as a Simulink block diagram. The linearized PDV dynamics results in 3 decoupled dynamic modes: two identical lateral motions due to the axial symmetry of the shape of the PDV, and an axial-rotational motion. Thus, two types of controllers were designed: directional and axial controllers. For the directional control the collective rotations of the horizontal or vertical set of control fins are used. A pitch/yaw damper and a vertical (image y-axis)/horizontal (image x-axis) target pixel position control are separately designed for the directional control. Target pixel position controls use the feedback of the target pixel location from the GS. Also, axial control is required for stable camera vision and precise vision-based measurements. Differential rotations of the two sets of control fins control the axial motion of the PDV, based on roll information from the estimator.

Scaling, which is done by dividing each variable by its maximum expected or allowed change, makes model analysis and controller design much simpler [19, Skogestad]. The maximum scaling numbers used for our controller design are summarized in Table 4.1. They also serve as the performance requirement. For example, it is purposed to control the target pixel location within the maximum error under the disturbance condition up to the maximum wind speed without using more than the maximum deflections of control surfaces.

Variable	Symbol	Maximum
pixel error	e_{pixel_x}, e_{pixel_y}	10
pixel command	$pixel_{x_{cmd}}$, $pixel_{y_{cmd}}$	50
wind disturbance	V_g, W_g	8 m/s
collective ctrl.	δ_{hc}, δ_{vc}	3^o
axial angle error	e_Φ	1^o
axial angle command	Φ_{cmd}	10^o
differential ctrl.	δ_{hd}, δ_{vd}	2^o

Table 4.1: Maximum Scaling Numbers

4.2.1 Directional Controller

The two lateral motions relative to the PDV body-fixed axis frame are the y-axis directional motion and the z-axis directional motion: the z-axis directional motion is described on the the x-z plane of the body-fixed axis frame, and the y-axis directional motion is described on the the x-y plane of the body-fixed axis frame. Assuming that the PDV is axially symmetric, the two lateral motions are identically described and the controllers are the same except for a sign convention. Note that the positive vertical camera image axis (image y-axis) is the opposite of the positive z-axis of the body-fixed axis (body z-axis), and yet the positive horizontal camera image axis (image x-axis) is the same direction as the y-axis of the

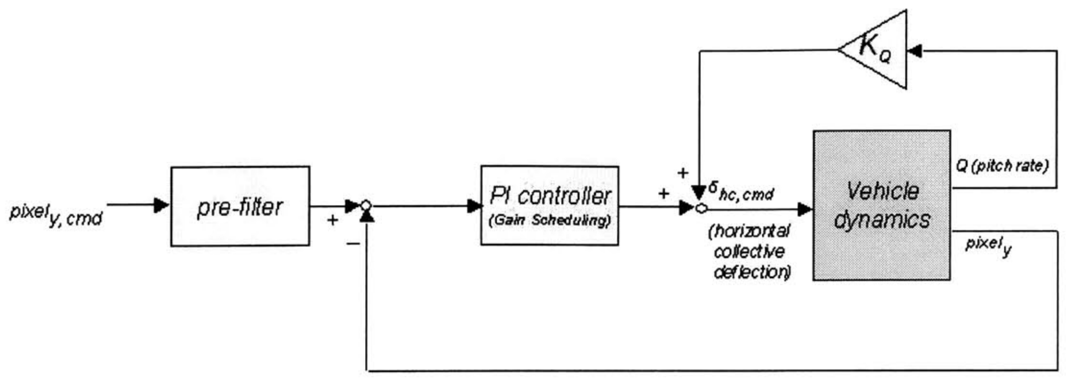


Figure 4-4: Controller Configuration of Directional Motion

body-fixed axis (body y-axis). First consider the z-axis directional motion as an example. The control input is the horizontal collective control surface deflection, δ_{hc}; and the output variables are the pitch rate (q) and the vertical target pixel position ($pixel_y$) which is a target pixel position in the direction of the vertical camera image axis. The controller configuration described in Figure 4-4 shows the combination of the pixel y controller and the pitch damper. Similarly, the controller configuration for the y-axis directional motion is the combination of the the pixel x controller and the yaw damper. These multiloop directional controllers are designed by means of successive loop closures. The inner-loop feedback, which is a pitch/yaw damper (derivative feedback), affects the locations of complex plant poles and increases their damping. It allows the outer-loop gain and the closed-loop bandwidth to be higher. The root-locus technique is used for pitch/yaw dampers (inner-loop) and the loop shaping designs are applied to target pixel position controls (outer-loop).

Pitch/Yaw Damper

As was previously mentioned, the PDV dynamics is numerically linearized at several altitudes (below the altitude of 200 meters), all with a speed V_t = 60 m/s and a pitch angle θ = - 90 deg (nose down). It was found that the trim altitude where the dynamics are linearized doesn't affect the pitch/yaw damper design very much. From the linearized dynamics, a transfer function (δ_{hc}/q), which includes the second-order servo dynamics in Eq. (3.9), is obtained. It indicates that the damping ratio is too small (around 0.1). In order to increase the damping ratio, a pitch damper was designed. The same design applies to the yaw damper. Figure 4-5 shows the root locus design using a simple constant control gain

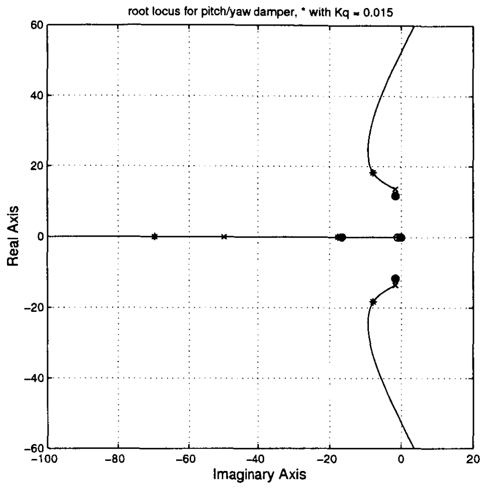


Figure 4-5: Root Locus Design for Pitch/Yaw Damper

K_q. From the pole locations of -7.8389±18.2940j, it can be easily known that the damping ratio can be increased up to 0.4 by means of the control gain of 0.015, and in this case the fast dynamics frequency of the PDV is near 3.2 Hz. Note that the pole location of 50 rad/s which is from the servo dynamics plays an important role. If it is not fast enough, the damping performance can be worse. This means that our servo system speed is critical in achieving good damping.

Vision-based Target Pixel Position Controls

The target pixel position controls are performed by the feedback of target pixel positions from the camera images. Unlike the case of the pitch/yaw damper design, the trim altitudes where the dynamics is linearized are critical in designing the target pixel position controllers. Thus, linearized dynamics models at different operating conditions were used for gain scheduling [28]. The following is the design procedure of the vertical target pixel position controller with a linearized model around an operation condition of the speed of 60 m/s and altitude of 100 meters as an example, with a pitch damper included.

Input-Output Controllability Analysis The input-output controllability analysis is applied to a plant to obtain insight into the inherent limitations on control performance originating in the model itself [19, Skogestad]. In our case, the plant is the linearized dynamic model with a pitch damper as shown in Figure 4-6. Since the variables have

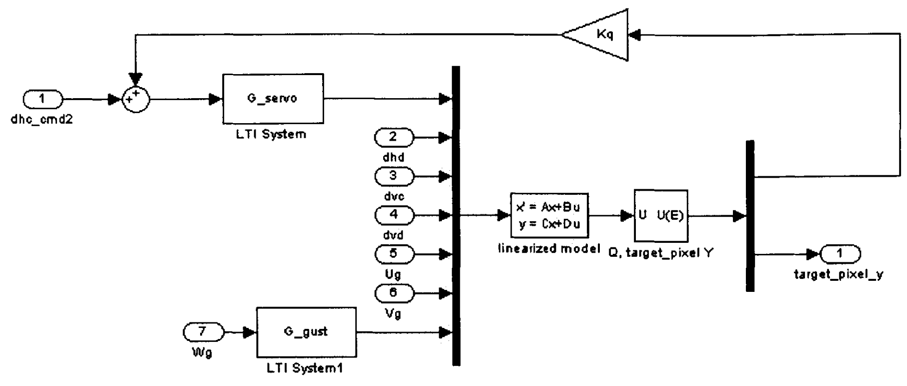


Figure 4-6: Linearized Dynamic Model with a Pitch Damper, for Input-Ouput Controllability Analysis of Directional Control

been scaled, the requirement for acceptable control performance is to keep the control error $|e(\omega)| \leq 1$, for any disturbance $|d(\omega)| \leq 1$ and any reference $|r(\omega)| \leq R$, using an input $|u(\omega)| < 1$. Note that there is no RHP zero or RHP pole in the dynamics, so no limitation is imposed by RHP zeros or RHP poles.

Consider performance requirements imposed by disturbances and reference commands. The effects of the control input $\delta_{hc}(\equiv u)$ and the gust $W_g(\equiv d)$ on the output $pixel_y$, are shown as $|G|$ and $|G_d|$, respectively, in Figure 4-7 along with $R = r_{max}/e_{max} = 5$. The reference command(r) is $pixel_{y,ref}$ in this case and described in the next section. Based on the plot the following points can be made for acceptable control($|e(\omega)| < 1$):

- **Disturbance Rejection** Since $|G_d(j\omega)| > 1$ at some frequency, in other words, the plant is not self-regulating, we need control to obtain faster response speed, and the associated requirement is $w_c > w_d = 3$ rad/s, or with feedback control, $|S(j\omega)| < 1/|G_d(j\omega)|\ \forall\omega$. Note that w_c denotes the gain crossover frequency where $|L(j\omega_c)| = 1$, and w_d is the frequency where $|G_d(j\omega_d)| = 1$.

- **Reference Command Tracking** In order to obtain enough response speed to track reference command($pixel_{y,ref}$) changes, $|S(j\omega)| < 1/R\ \forall\omega \leq \omega_r$, where ω_r is the

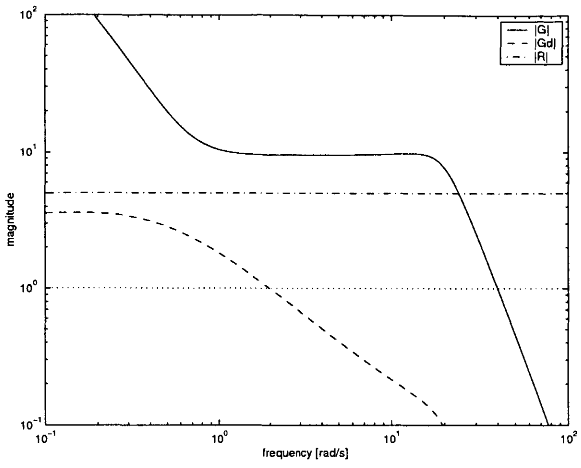


Figure 4-7: Control Input Effect (G) and Disturbance Effect (G_d), and Reference Command (R) for the Scaled Dynamics in Directional Motion

frequency up to which tracking is required.

- **Input Constraint arising from Disturbances** Since $|G(j\omega)| > |G_d(j\omega)|$, input saturation due to disturbances can be avoided.
- **Input Constraint arising from Reference Commands** Since $|G(j\omega)| > R$ at frequencies up to 11.5 rad/s, input saturation is not a problem in this case, where the reference command($pixel_{y,ref}$) changes more slowly than 11.5 rad/s.

Note that the requirement, $|u(\omega)| < 1$ and $|e(\omega)| < 1$ for any $|d(\omega)| < 1$ or any $|r(\omega)| < R$ at any frequency ω, is necessary to avoid input saturation.

Controller Design and Analysis In Figure 4-8, the PI controller provides disturbance rejection and a prefilter is added for command tracking. The corresponding Bode plots for the plant $G(s)$ and the loop transfer function $K(s)G(s)$ are shown in Figure 4-9.

The controller provides a gain crossover frequency of 3.12 rad/s, a phase crossover frequency of 21.23 rad/s, a gain margin of 2.32, and a phase margin of 90.6 degrees.

The resulting sensitivity function $|S| = |(1+KG)^{-1}|$ is shown in Figure 4-10 (a). It shows that the system bandwidth is around 10 rad/s. Note that the performance requirement is satisfied in term of disturbance rejection because $|S| < 1/|G_d|$ at all frequencies. Note also that the fact, $|S| < 1/R$ at frequencies up to around 0.6 rad/s indicates that performance

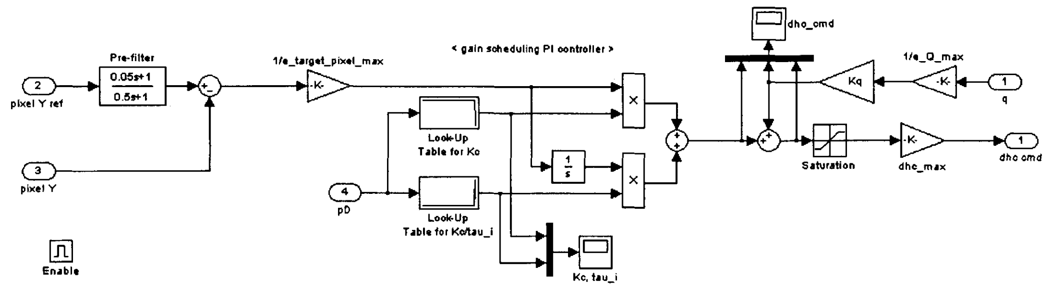


Figure 4-8: Directional Controller Diagram

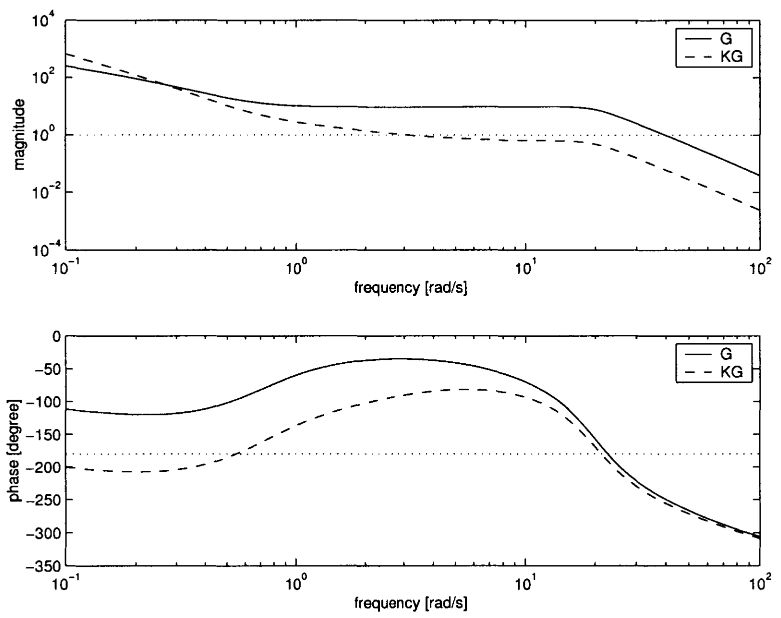
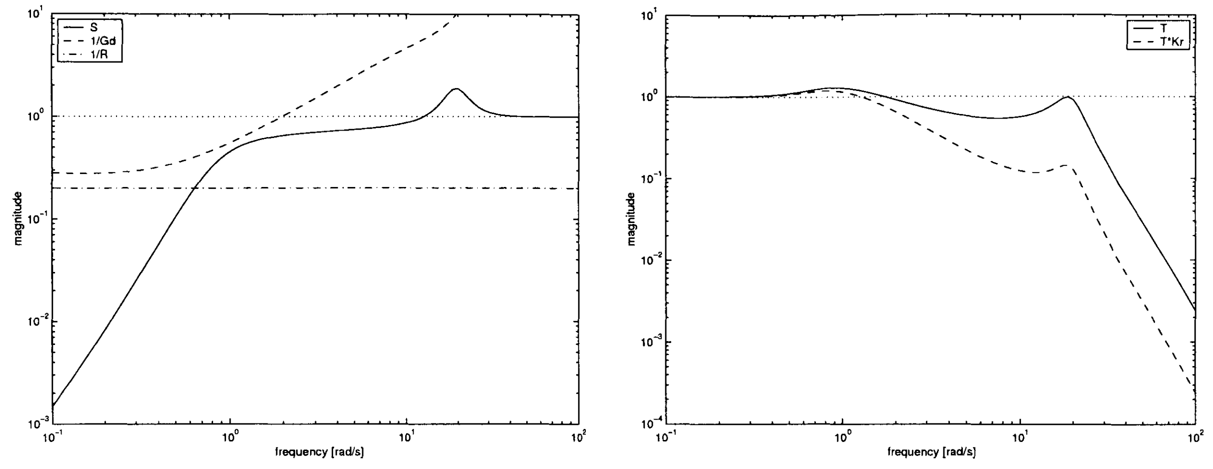


Figure 4-9: Bode Plot for Directional Controller Design

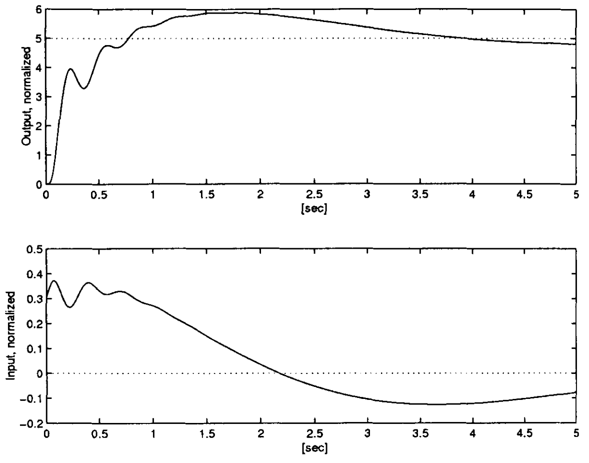


(a) Sensitivity(S) (b) Complementary Sensitivity(T)

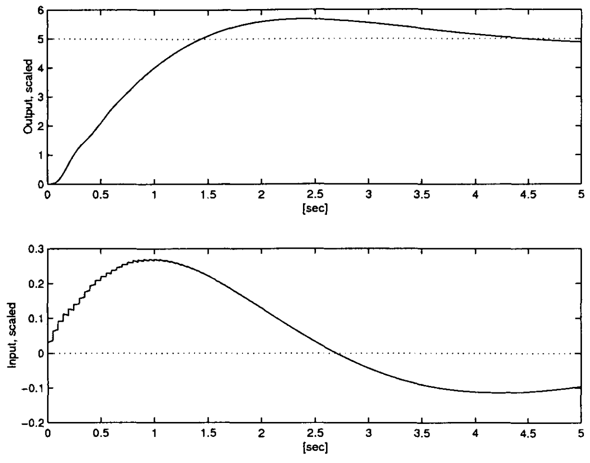
Figure 4-10: Performance Plots for Directional Controller

command tracking can be achieved in this frequency range. However, as is noticed from the complementary sensitivity function T, the reference command needs to be shaped by a prefilter for the purpose of improving the response for command tracking. The lead-lag network $K_r(s) = (0.05s+1)/(0.5s+1)$ is used as a prefilter. Figure 4-11 shows simulations of both command tracking without the prefilter and command tracking with the prefilter in the scaled linear dynamic model. In these plots, the input and the output mean the plant input $\delta_{hc}(\equiv u)$ and the plant output $pixel_y$, respectively. The variables are scaled and $R = r_{max}/e_{max}$ is 5. From Figure 4-11, it is found out that the prefilter $K_r(s) = (0.05s + 1)/(0.5s+1)$ slows down the response a small amount, but it also removes high frequency components and improves the transient response better. It also shows that the input arising from the reference command avoids input saturation because the magnitude of the input $|u(\omega)|$ is less than 1. Note that the reference prefilter gives the control configuration in Figure 4-4 one more degree-of-freedom to improve tracking performance, and the addition of a prefilter doesn't affect disturbance rejection performance because the feedback loop is invariant and the disturbance output isn't affected by the prefilter.

Figure 4-12 shows a simulation of disturbance rejection in the scaled linear dynamic model. The plot shows that the output is maintained within ± 1 boundaries with the maximum wind disturbance while the input $|u(\omega)|$ is not saturated. In other words, the input arising from the disturbance also avoids input saturation.



(a) command tracking without the prefilter



(b) command tracking with the prefilter

Figure 4-11: Command Tracking in Linear Simulation for Directional Controller.

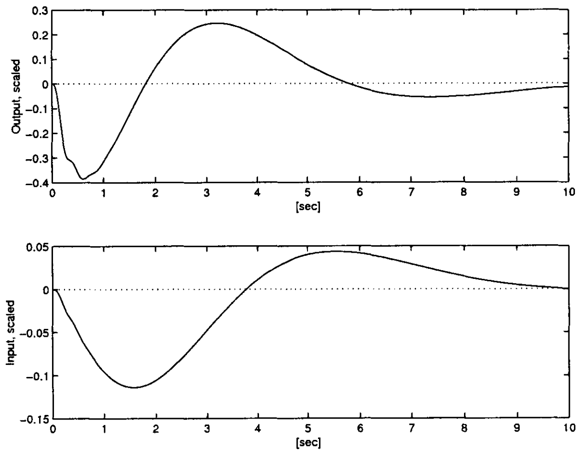


Figure 4-12: Disturbance Rejection in Linear Simulation for Directional Controller.

Gain Scheduling Since the nonlinear PDV dynamics change with the flight conditions, the linearized PDV dynamics needs gain scheduling for the target pixel position controls. It turned out, however, that the dynamics of pixel location do not change very much above the altitude of 50 meters. Table 4.2 summarizes the gain scheduling in the PI controller. The settings for K_c and τ_I are determined by adapting the tuning rules of Ziegler and Nichols [27].

Altitude	K_c	τ_I	PM	GM
50 or higher	0.0616	0.231	87.3	2.3
40 m	0.0462	0.116	76.8	2.6
30 m	0.0346	0.077	69.0	0.4

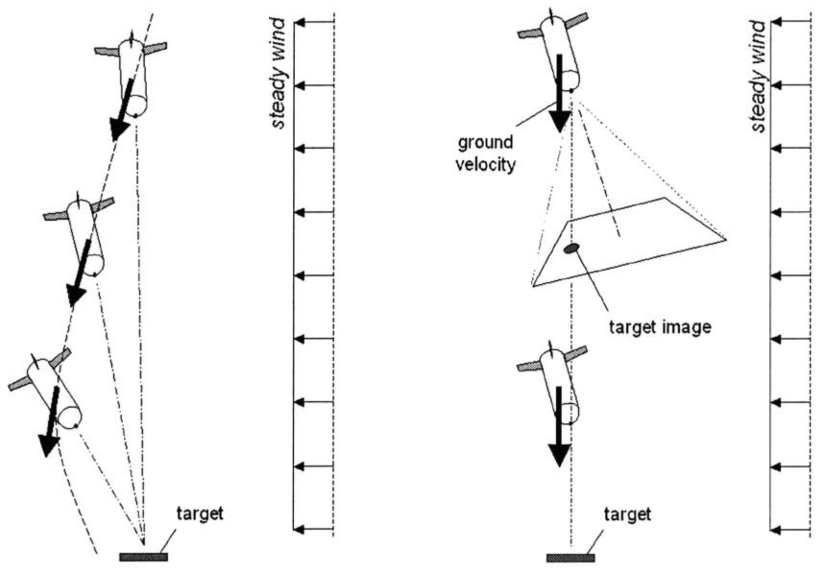
Table 4.2: Gain Scheduling in PI controller : $K_c\left(1+\frac{1}{\tau_I s}\right)$, At the terminal speed of 60 m/s

Reference Commands In this case, a reference command is a vertical axis (image y-axis) target pixel position command, $pixel_{y,ref}$. Ideally, if there is no steady wind, the $pixel_{y,ref}$ is always zero or fixed at the center of the camera image during the airdrop. However, with the zero target pixel position commands, under a steady wind condition, the behavior shown in Figure 4-13 (a) tends to be observed in the simulation. The PDV first drifts with the steady wind, and at some point the sign of the angle of attack changes. It is because the PDV camera (axial) direction always aligns the target direction so the PDV axial direction is not the same as the relative wind direction. After the sign of angle of attack changes, the PDV begin to come back toward the target, but it is insufficient to reach the target.

To prevent this a steady state wind component is estimated with the use of low frequency components of the vehicle body-fixed y and z axis of the vehicle velocity, i.e. V and W. For the vertical (image y-axis or negative body z-axis) pixel position control,

$$W_{g_{steady_{est}}}(s) = \frac{1}{3s+1}W(s) \tag{4.43}$$

and the value of target pixel position command to compensate this effect can be found to



(a) pixel position command = 0 (b) offset pixel position command

Figure 4-13: Steady Wind v.s. Pixel Position Command

be a vertical target offset

$$pixel_{y,ref} = -\frac{N_y/2}{\tan(\alpha_y/2)}\frac{W_{g_{steady_{est}}}}{U_0} \tag{4.44}$$

where the camera image resolution in the vertical image axis $N_y = 480$, the camera field-of-view angle in the vertical image axis $\alpha_y = 59^o$, and the terminal speed $U_0 = 60\ m/s$.

Similarly, for the horizontal (image x-axis or body y-axis) pixel position control,

$$V_{g_{steady_{est}}}(s) = \frac{1}{3s+1}V(s) \tag{4.45}$$

and the value of target pixel position command to compensate this effect can be found to be a horizontal target offset

$$pixel_{x,ref} = \frac{N_x/2}{\tan(\alpha_x/2)}\frac{V_{g_{steady_{est}}}}{U_0} \tag{4.46}$$

where the camera image resolution in the horizontal image axis $N_x = 640$, the camera field-of-view angle in the vertical image axis $\alpha_y = 74^o$, and the terminal speed $U_0 = 60\ m/s$.

With these reference commands, the relative wind direction is close to the vehicle axial direction so the angle of attack is almost zero. Note that the vehicle pitch angle is almost constant and it is not -90 degrees. This is shown in Figure 4-13 (b).

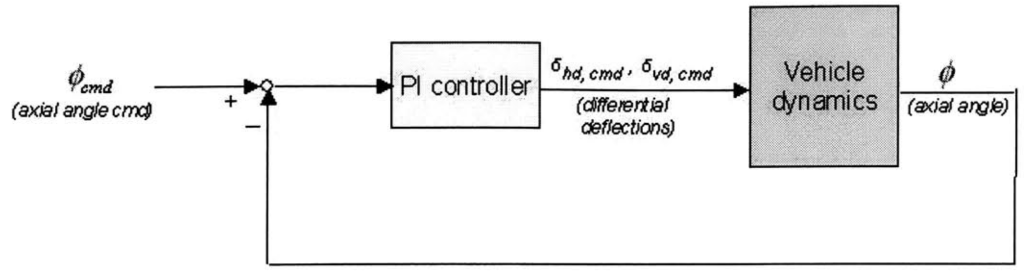


Figure 4-14: Controller Configuration for Axial Motion

Smooth Transition and Anti-Windup During the airdrop, the PDV should not have a sudden attitude change which makes it difficult to track the target image on the GS. By this reasoning, a smooth transition is made when the directional control is first applied. In other words, it is applied to the first control input which is done after several vision-based measurements are used for estimating states.

In addition, an anti-windup is applied to the target pixel position controls because they have a pure integrator in their PI control and may have the integrator saturation effect (integrator windup). For the smooth transition and the anti-windup, the algorithm described in [21, Park] was utilized.

4.2.2 Axial Controller

For stable camera vision and precise vision-based target tracking, the PDV should not rotate about its body x-axis during the airdrop. Thus, in the axial controller, the angle about the body x-axis is controlled to be a constant angle.

The control inputs are a horizontal differential control surface deflection, δ_{hd} and a vertical differential control surface deflection, δ_{vd}. Moreover, it is assumed that the control surfaces work in such a way that δ_{hd} is equal to δ_{vd}. The plant output is the axial angle, ϕ. A wind disturbance model is not included in this dynamics since it is assumed that the PDV is axially symmetric. Also, a wind shear effect is not considered in this small vehicle. Figure 4-14 shows a simple one-degree-of freedom feedback controller for axial angle control.

Linearization A linearized model is extracted around the pitch angle (θ) of -90 degrees(nose down) and the yaw angle (ψ) of zero. With $\theta = -\pi/2$ and $\psi = 0$, the quaternion

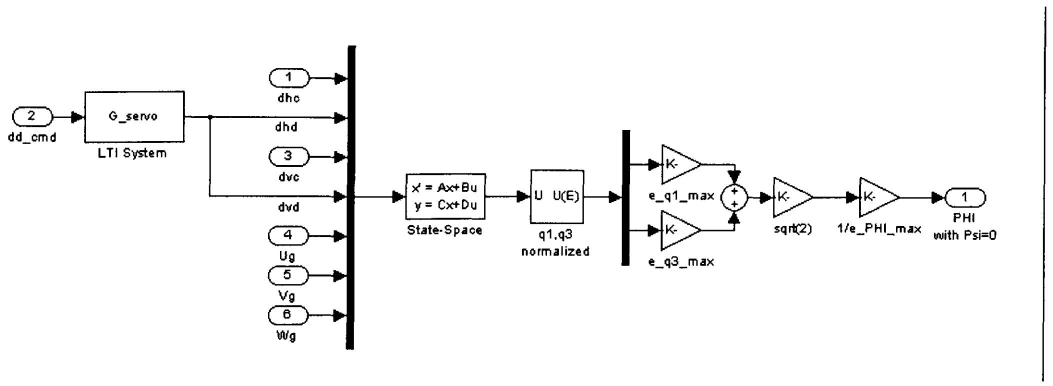


Figure 4-15: Linearized Dynamic Model for Input-Ouput Controllability Analysis of Axial Control

parameters become

$$q_0 = \frac{1}{\sqrt{2}}\cos\frac{\phi}{2},\ q_1 = \frac{1}{\sqrt{2}}\sin\frac{\phi}{2},\ q_2 = -\frac{1}{\sqrt{2}}\cos\frac{\phi}{2},\ q_3 = \frac{1}{\sqrt{2}}\sin\frac{\phi}{2}$$

or

$$\phi = 2\tan^{-1}\left(\frac{(q_1+q_3)/\sqrt{2}}{(q_0-q_2)/\sqrt{2}}\right)$$

Thus, it reduces to the following linearized form:

$$\phi = \sqrt{2}(q_1+q_3)$$

Input-Output Controllability Analysis Figure 4-15 shows the plant for analyzing input-output controllability of axial control. There is no RHP zero or RHP pole in the linearized model, so no limitation is imposed by RHP zeros or RHP poles. The effect of control input (control surface deflections) on the output (axial angle) and the reference command $R = r_{max}/e_{max} = 10^o/1^o = 10$ are shown in Figure 4-16. Note that this is a scaled linear dynamic model. Figure 4-16 indicates that input saturation is avoided if the control bandwidth is chosen in the frequency range below 12 rad/s, because $|G(j\omega)| > R$ at frequencies up to 12 rad/s.

Controller Design and Analysis Figure 4-17 shows the axial controller diagram. A PI controller $K_{axial}(s) = 0.006(1+0.7/s)$ serves as the axial controller. Figure 4-18 shows the loop shaping for the axial controller design. It provides a crossover frequency of 3.4 rad/s, a PM of 59 degrees, and a GM of 9.

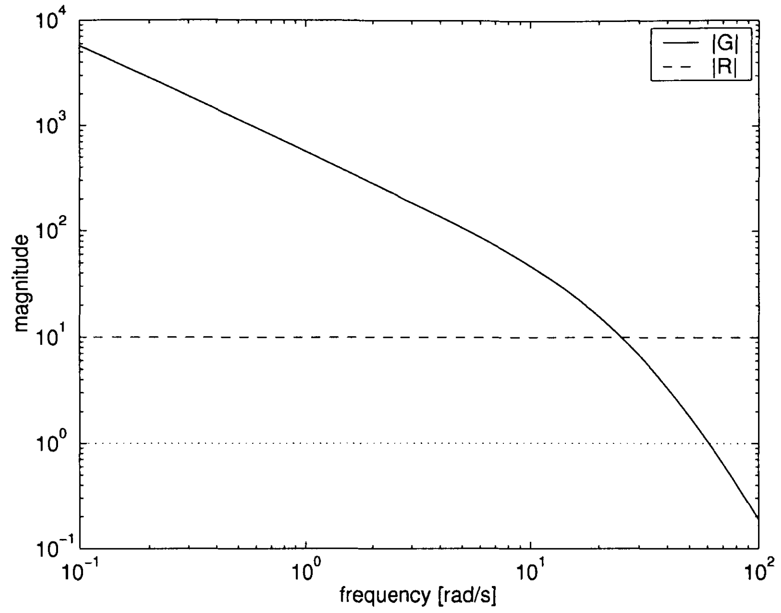


Figure 4-16: Control Input Effect (G) and Reference Command (R) for the Scaled Dynamics in Axial Motion

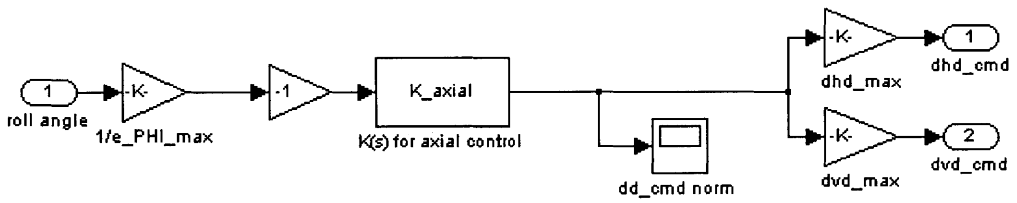


Figure 4-17: Axial Controller Diagram

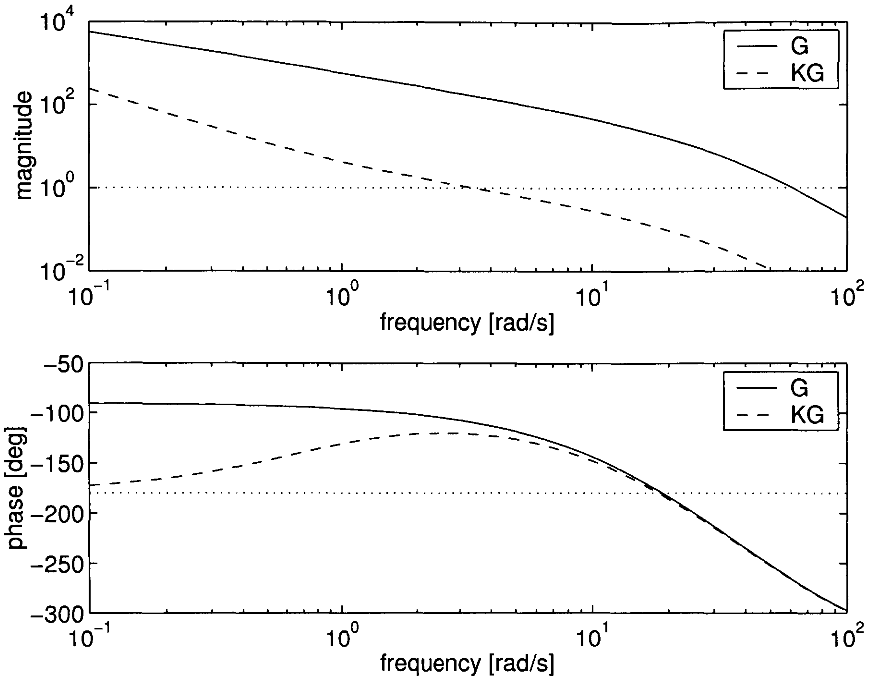


Figure 4-18: Bode Plot for Axial Angle Controller Design

The resulting sensitivity function $|S| = |(1+KG)^{-1}|$ is plotted in Figure 4-19. It shows that performance command tracking can be achieved in the frequencies below around 0.5 rad/s because $|S| < 1/R$ in the frequency range. In other words, the error of the axial angle is maintained within the specified error value of $e_{max} = 1^o$ when the sinusoidal reference commands up to the frequency of 0.5 rad/s with the magnitude of $r_{max} = 10^o$ are applied.

Figure 4-20 shows simulation of reference command tracking for a step reference input in the scaled linear dynamic model. It shows that the input arising from the reference command doesn't saturate because the magnitude of the control input $|u(\omega)|$ is less than 1.

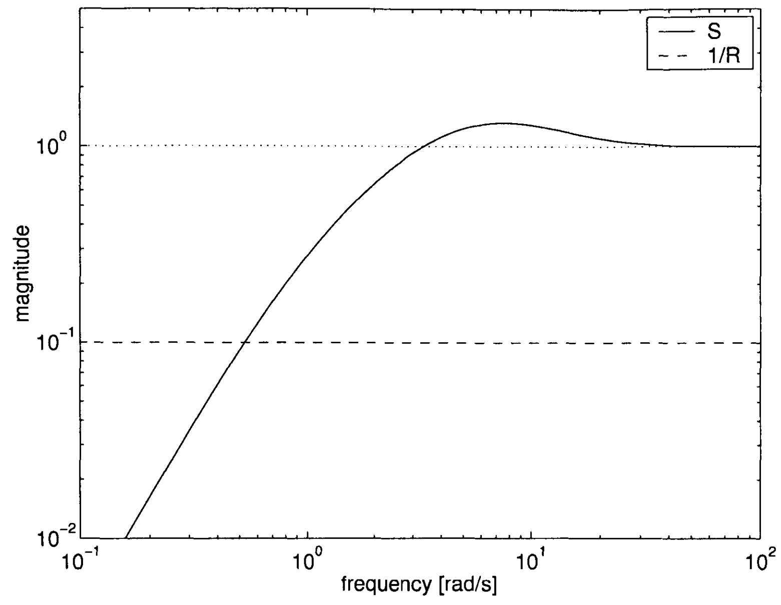


Figure 4-19: Sensitivity for Axial Angle Controller

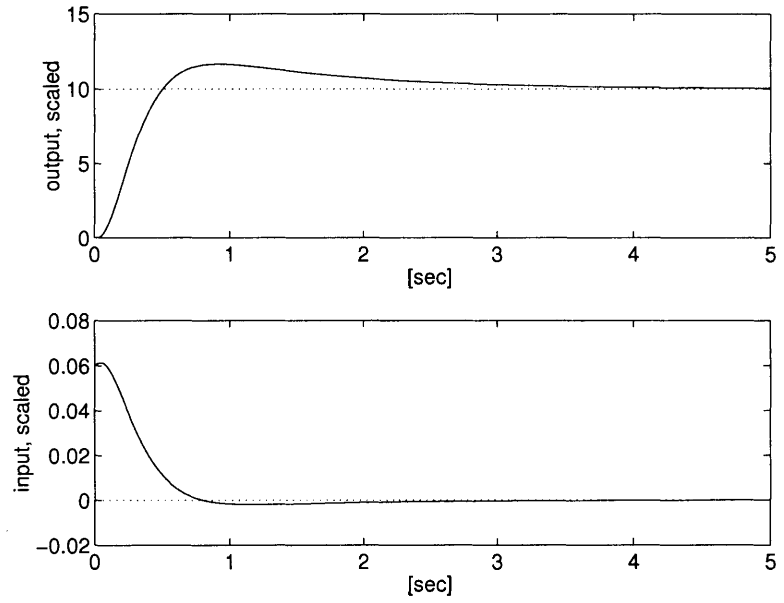


Figure 4-20: Linear Simulation for Axial Angle Control : Command Tracking

Chapter 5

Test Results

This chapter discusses the results of various evaluations of the system using the software simulations described in Section 3.7 and then the results of hardware-in-the-loop simulation (HILSIM) described in Section 2.4.

5.1 Software Simulation Results

A software simulation is usually developed earlier than a HILSIM, and is used for designing the state estimation and control loops. The MATLAB Simulink diagrams of the software simulation, which include the designed state estimation and control diagrams, are described in Appendix B.

The software simulation made use of all of the models described in Chapter 3. It also used the extended Kalman filter described in Section 4.1.1: including propagation of the estimated state vector and propagation of the error covariance matrix by digital integration of the appropriate differential equations, and discrete measurement updates by the Joseph formulation.

In these simulations, the PDV was launched at the altitude of 350 meters with an initial velocity of 20 m/sec. The assumed PDV drop position (the initial position in simulation) was made different from the ideal launching position (which was calculated in the condition of no wind) by 30 meters in both north and east directions, to test the control capabilities.

The initial state vector in the flight dynamics simulation was:

$$\begin{aligned}\mathbf{x}^T &= [V_T,\ \alpha,\ \beta,\ P,\ Q,\ R,\ q_0,\ q_1,\ q_2,\ q_3,\ p_N,\ p_E,\ p_D\ ]^T \\ &= [20,\ 0,\ 0,\quad 0,\ 0,\ 0,\quad 1,\ 0,\ 0,\ 0,\quad -140+30,\quad 0+30,\quad -350\ ]^T\end{aligned}$$

relative to the NED local inertial frame, which has its origin at the target point. It is assumed that the ideal launching position is $(p_N,\ p_E,\ p_D) = (-140,\ 0,\ -350)$.

The initial estimated state vector in the state estimation loop is:

$$\begin{aligned}\mathbf{x}^T &= [p_N,\ p_E,\ p_D,\ U,\ V,\ W,\ q_0,\ q_1,\ q_2,\ q_3,\ \gamma_P,\ \gamma_Q,\ \gamma_R,\ \gamma_{a_x},\ \gamma_{a_y},\ \gamma_{a_z}]^T \\ &= [-140,\ 0,\ -350,\ \ 20,\ 0,\ 0,\ \ 1,\ 0,\ 0,\ 0,\ \ 0,\ 0,\ 0,\ \ 0,\ 0,\ 0\,]^T\end{aligned}$$

relative to the NED local inertial frame, which has its origin at the target point.
The initial error covariance matrix in the state estimation loop is a diagonal matrix:

$$P = diag\left([\sigma^2_{p_N}, \sigma^2_{p_E}, \sigma^2_{p_D}, \sigma^2_U, \sigma^2_V, \sigma^2_W, \sigma^2_{q_0}, \sigma^2_{q_1}, \sigma^2_{q_2}, \sigma^2_{q_3}, \sigma^2_{\gamma_P}, \sigma^2_{\gamma_Q}, \sigma^2_{\gamma_R}, \sigma^2_{\gamma_{a_x}}, \sigma^2_{\gamma_{a_y}}, \sigma^2_{\gamma_{a_z}}]\right)$$

where

$$\begin{array}{llll}\sigma_{p_N} = 30 & \sigma_{p_E} = 30 & \sigma_{p_D} = 30 & \\ \sigma_U = 0.1 & \sigma_V = 0.1 & \sigma_W = 0.1 & \\ \sigma_{q_0} = 0.01 & \sigma_{q_1} = 0.01 & \sigma_{q_2} = 0.01 & \sigma_{q_3} = 0.01 \\ \sigma_{\gamma_P} = 0.02 & \sigma_{\gamma_Q} = 0.02 & \sigma_{\gamma_R} = 0.02 & \\ \sigma_{\gamma_{a_x}} = 0.1 & \sigma_{\gamma_{a_y}} = 0.1 & \sigma_{\gamma_{a_z}} = 0.1 & \end{array}$$

The vision-based estimation was set to initiate at 3 seconds after launch, the directional control was set to initiate at 3.5 seconds after launch, and the axial controller was set to initiate immediately after launch. Also, the vision-based target image measurements occurred every second and were initiated 3 seconds after launch.

Figure 5-1, Figure 5-2, and Figure 5-3 show the result of the case where the PDV was launched at the altitude of 350 m without the 3-axis magnet sensor. Figure 5-4, Figure 5-5, and Figure 5-6 show the result of the case that the PDV was launched at the altitude of 350 m with the 3-axis magnet sensor. By observing all the results, the followings can be concluded:

- The magnetic sensor greatly improves the state estimation and, in particular, has the most significant effect on the quaternion variables or euler angles. This result is quite intuitive because the outputs of the 3-axis magnetic sensor provide a direct update to the vehicle attitude.

- The vision-based target image measurements do not have much effect on the estimation of the altitude p_D, whereas they help the estimation of the horizontal and vertical locations (p_E and p_N) considerably. This result is also quite intuitive because the target image location is mainly decided by p_E and p_N, when the PDV is directed towards the target point and the target image location is near the center of the camera view.

- The biases in pitch and yaw rate gyros (γ_Q and γ_R) are better estimated than the bias in the roll rate gyro (γ_P).
- The biases in the accelerometer ($\gamma_{a_x}, \gamma_{a_y}$ and γ_{a_z}) are not corrected very much by the vision-based measurements.
- The estimation of the target pixel positions is good all the time (even at times between vision-based measurements).

5.2 Hardware-in-the-loop Simulation Results

The HILSIM contains all the possible avionics components and software in the loop and uses the same I/O signal lines as real systems. In the HILSIM, we can include more of the actual errors and delays in our avionics components into the simulation.

Unlike the software simulation, the HILSIM does not use the target pixel position tracking models described in Chapter 3, because a human operator is included in the hardware-in-the-loop.

The extended Kalman filter described in Section 4.1.2 was applied to the HILSIM: continuous-time propagation of the estimated state vector, discrete-time propagation of the error covariance matrix, and discrete measurement updates by Carlson's square root filter formulation.

Note that because the vehicle software used for HILSIM is applied to the real PDV without any change, HILSIM tests real drop tests. In the HILSIM, the state estimation loop starts at two seconds before the launch. It means that the PDV has two seconds of level-flight before it is launched, as in the real drop tests. Thus, the initial estimated state vector in the state estimation loop is:

$$\begin{aligned} \mathbf{x}^T &= [p_N,\ p_E,\ p_D,\ U,\ V,\ W,\ q_0,\ q_1,\ q_2,\ q_3,\ \gamma_P,\ \gamma_Q,\ \gamma_R,\ \gamma_{a_x},\ \gamma_{a_y},\ \gamma_{a_z}]^T \\ &= [-140-40,\ \ 0,\ \ -350,\ \ \ 20,\ 0,\ 0,\ \ \ 1,\ 0,\ 0,\ 0,\ \ \ 0,\ 0,\ 0,\ \ \ 0,\ 0,\ 0\,]^T \end{aligned}$$

relative to the NED local inertial frame which has the origin at the target point. The initial error covariance matrix is the same as the one used for the software simulation.

Figure 5-7, Figure 5-8, Figure 5-9, and Figure 5-10 show the results of the case in which the PDV was launched at an altitude of 380 m without the 3-axis magnet sensor. Figure 5-11, Figure 5-12, Figure 5-13, and Figure 5-14 show the results of the case in which the PDV was launched at the altitude of 380 m with the 3-axis magnet sensor.

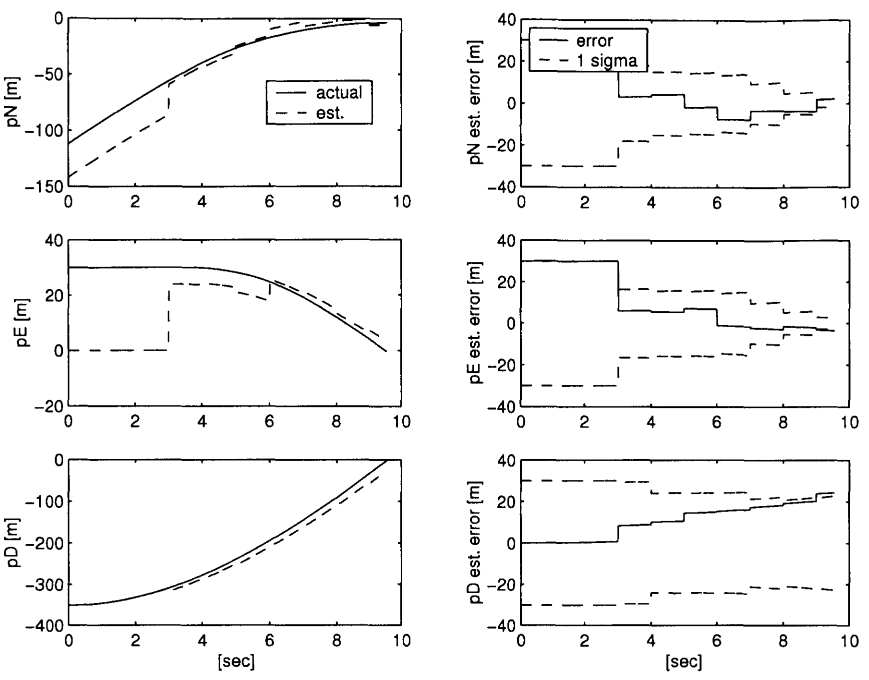
By comparing all the results, the following conclusions can be drawn:

- The estimated state errors in HILSIM are much larger than the ones in software simulation, especially in the case without using the 3-axis magnetic sensor. One main reason for this can be found from the plots of fin deflection angles in Figure 5-10 and Figure 5-14. These plots show that the actual fin deflections are stepped and have a time delay of around 0.2 sec in response to fin deflection commands or control inputs by controllers. This means that the servo motors deflecting fins do not have enough bandwidth to respond to the deflection command inputs.

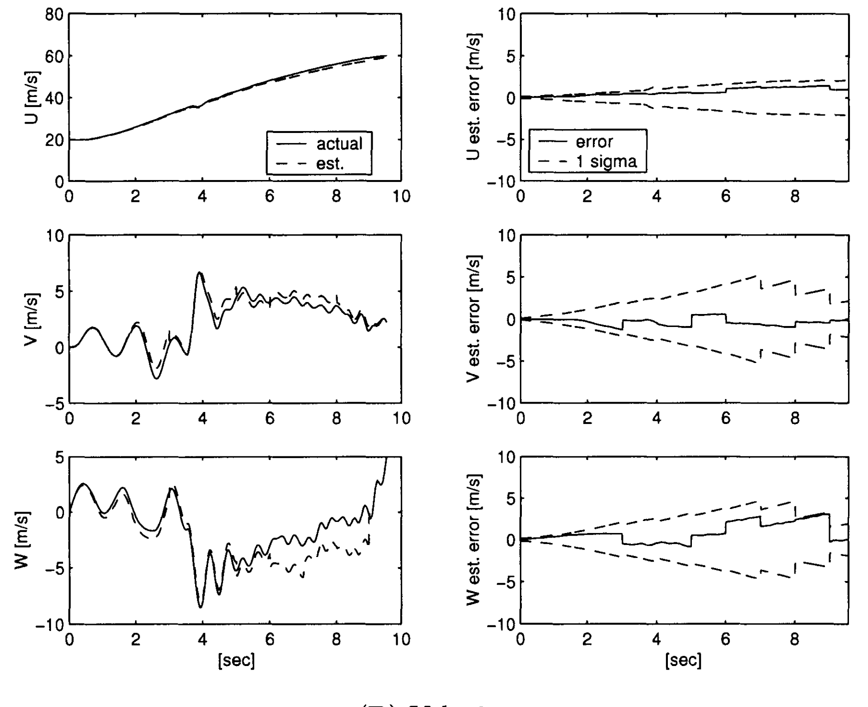
- Like the case of software simulation, the 3-axis magnetic sensor enhances state estimation.

- From the plots of target pixel positions, it is observed that the patten of estimated target pixel positions is very similar to the one of tracked (measured) target pixel positions. Note that the meaningful vision-based target tracking starts at 3 seconds after the launch because the vision-based estimation is set to start working from then. In addition, since a human operator tracked the target image well (in these tests, with a time delay of around 0.1 sec), the estimated target pixel positions are very close to the actual target pixel positions.

- The PDV launched from a carrier UAV at a speed of about 20 m/sec and at an altitude of about 350 meters, could be guided to a target point with a maximum error of 5 meters if the launching position could be estimated to within a maximum error of 50 meters from the ideal launching position.

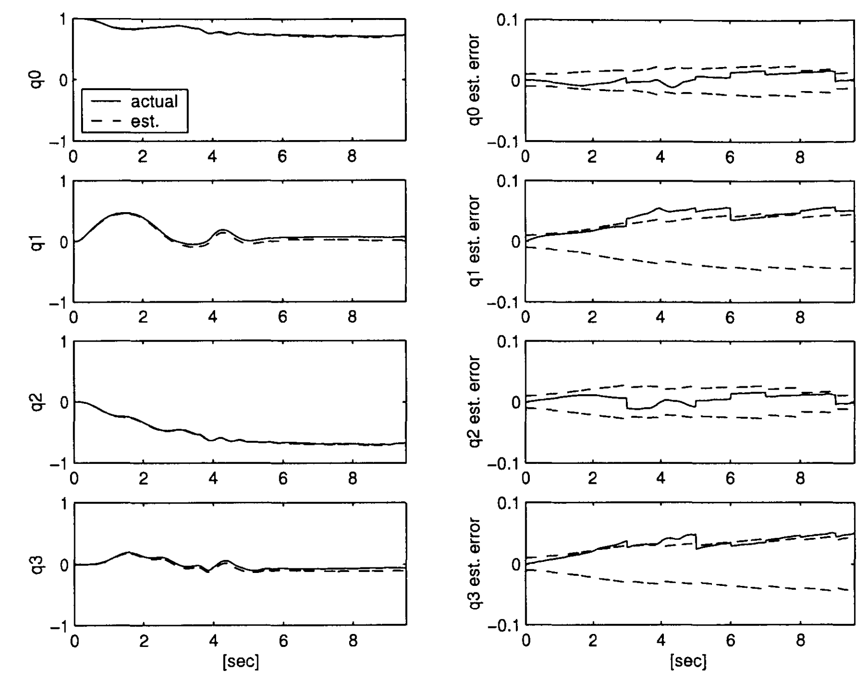


(A) Position

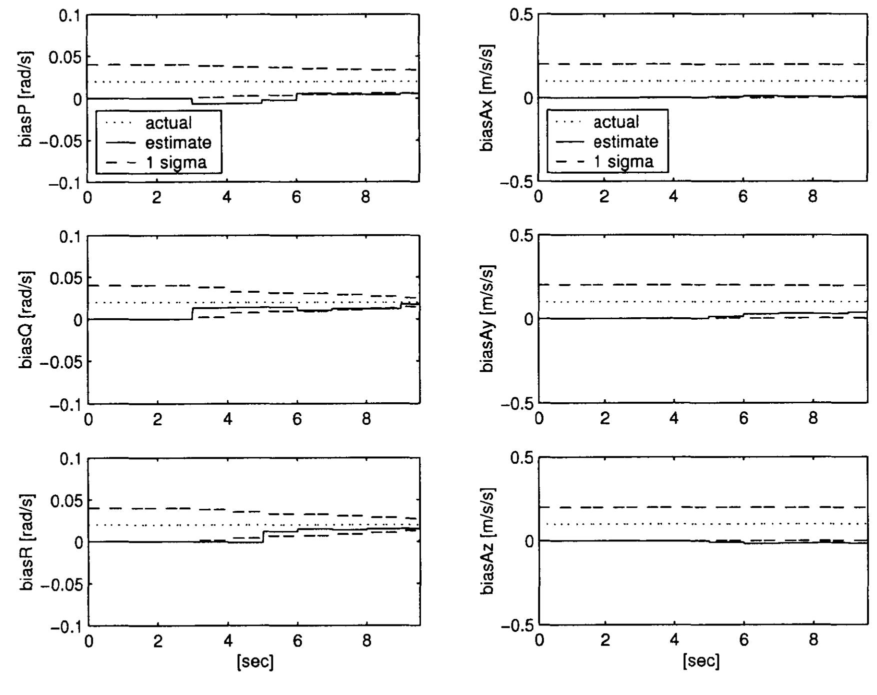


(B) Velocity

Figure 5-1: Software Simulation: drop from 350 m altitude without the 3-axis magnet sensor (Position and Velocity)

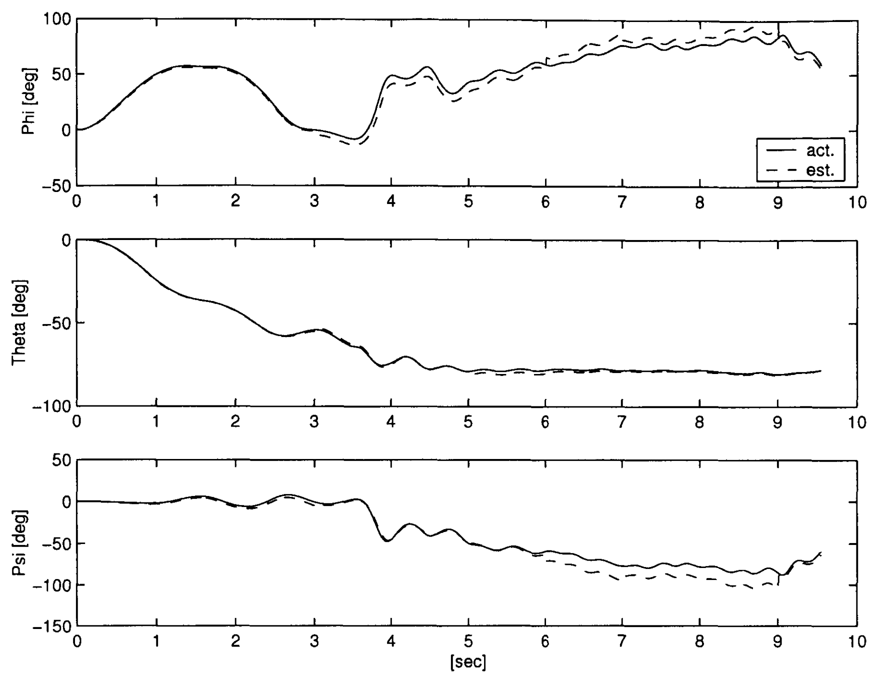


(C) Quaternion

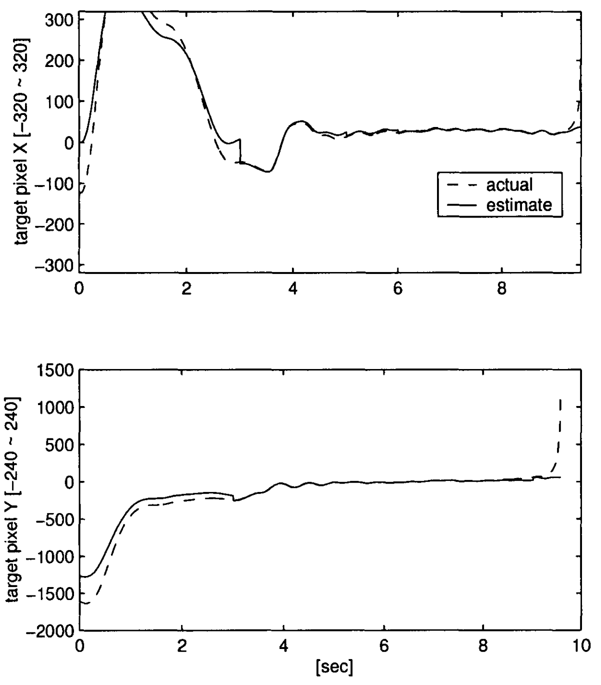


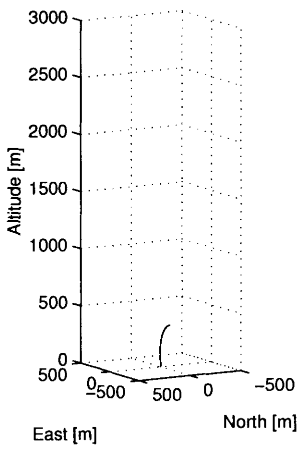
(D) Bias

Figure 5-2: Software Simulation: drop from 350 m altitude without the 3-axis magnet sensor (Quaternion and Bias)



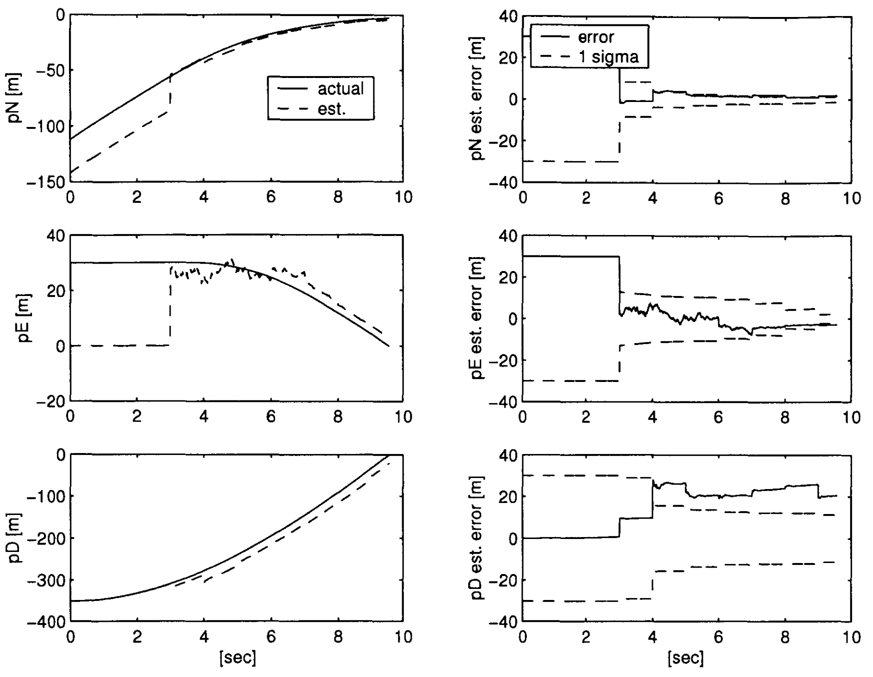
(E) Euler Angles



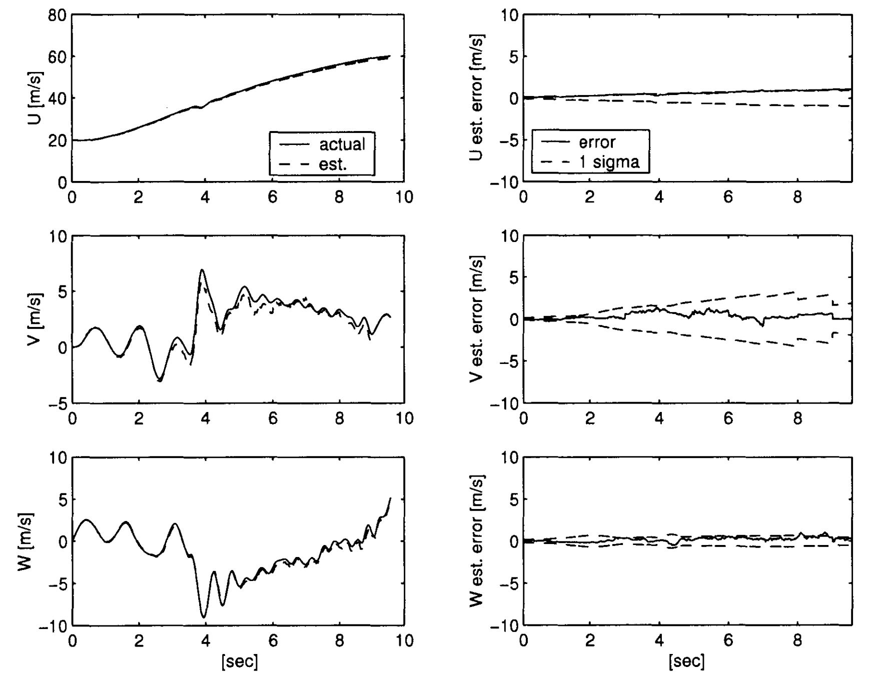


(F) Target Pixel Positions

Figure 5-3: Software Simulation: drop from 350 m altitude without the 3-axis magnet sensor (Euler Angles and Target Pixel Positions)

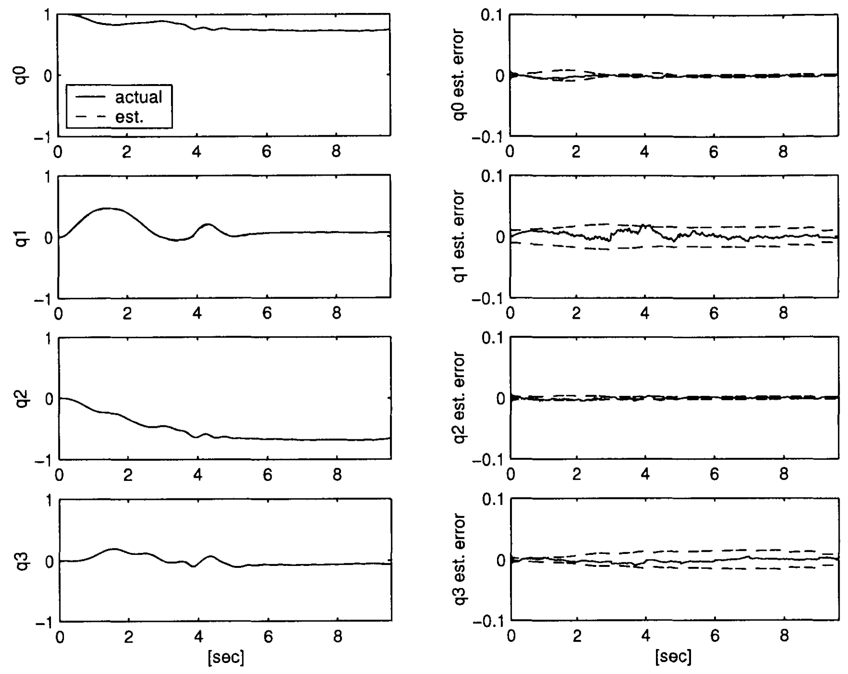


(A) Position

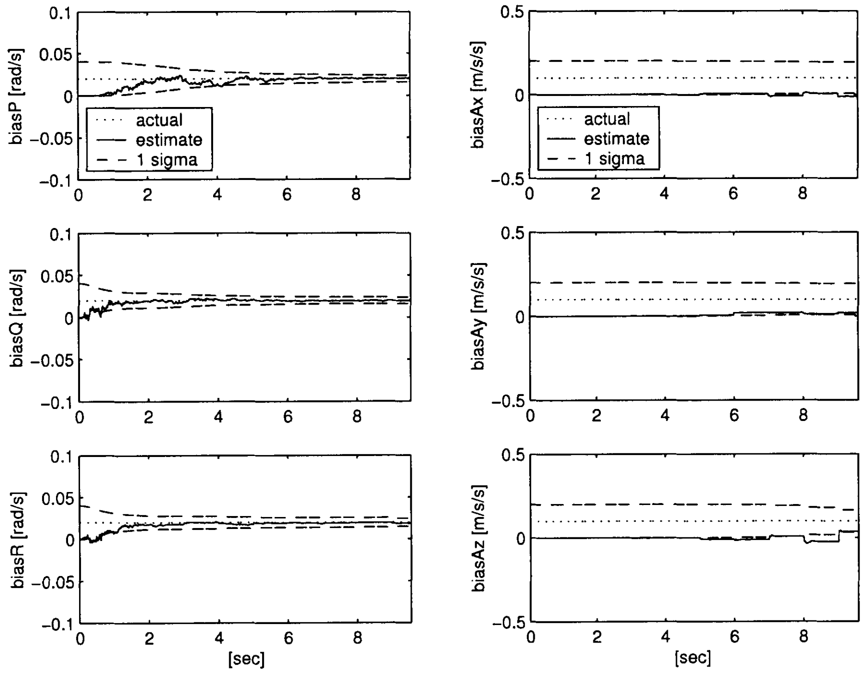


(B) Velocity

Figure 5-4: Software Simulation: drop from 350 m altitude with the 3-axis magnet sensor (Position and Velocity)

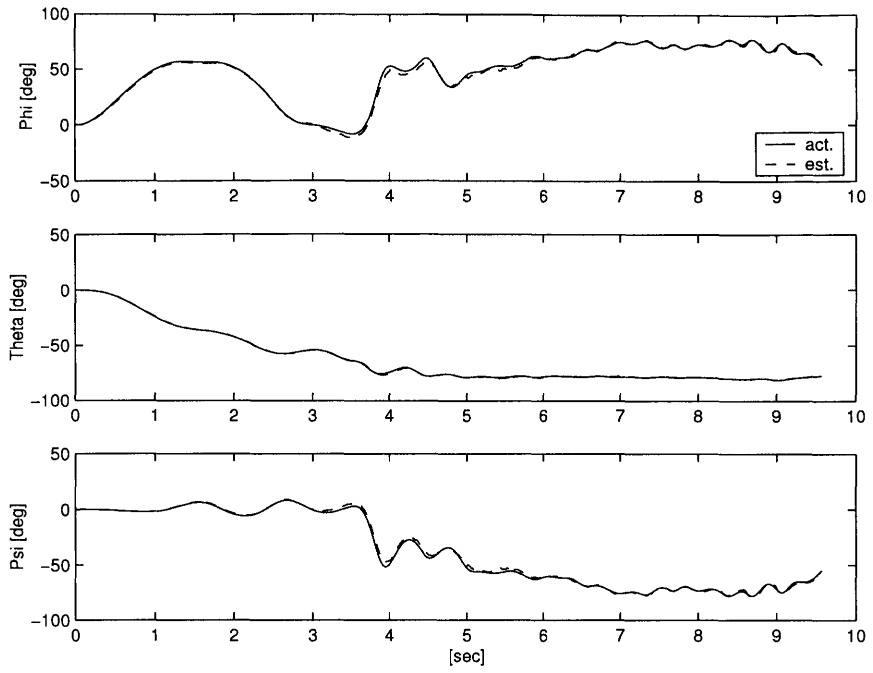


(C) Quaternion

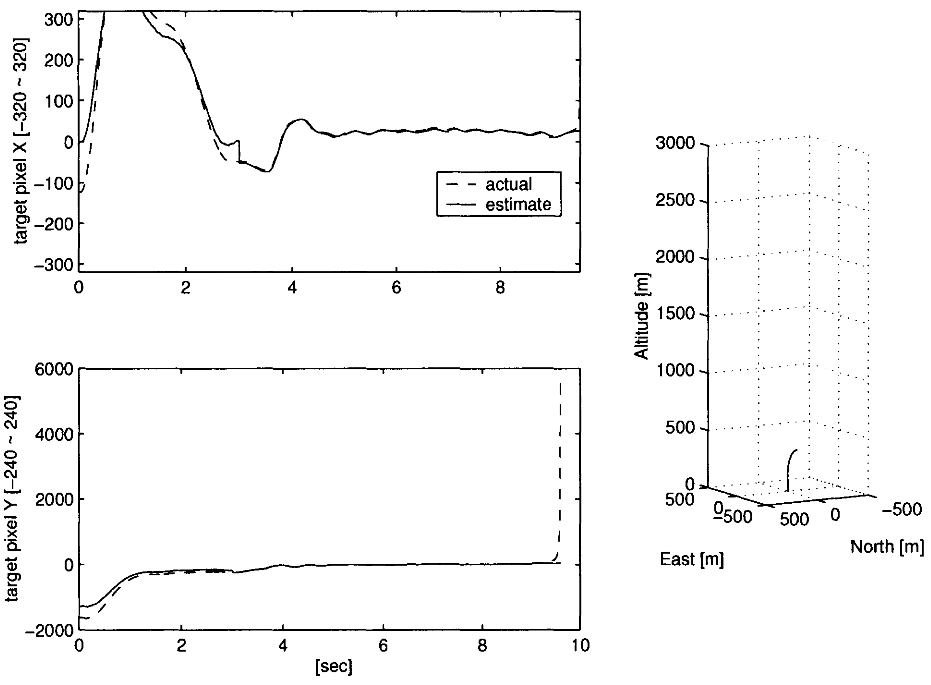


(D) Bias

Figure 5-5: Software Simulation: drop from 350 m altitude with the 3-axis magnet sensor (Quaternion and Bias)

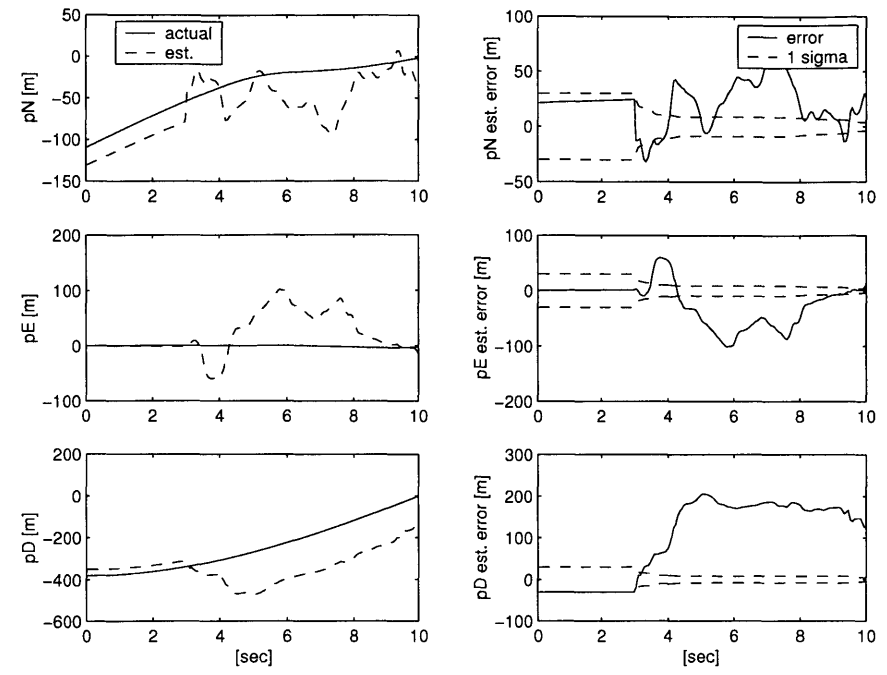


(E) Euler Angles

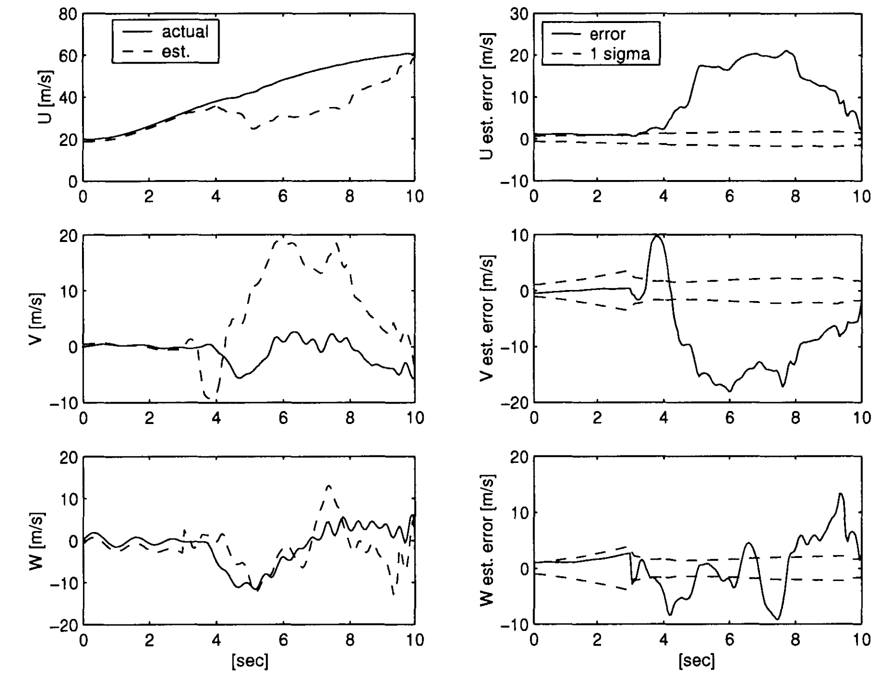


(F) Target Pixel Positions

Figure 5-6: Software Simulation: drop from 350 m altitude with the 3-axis magnet sensor (Euler Angles and Target Pixel Positions)

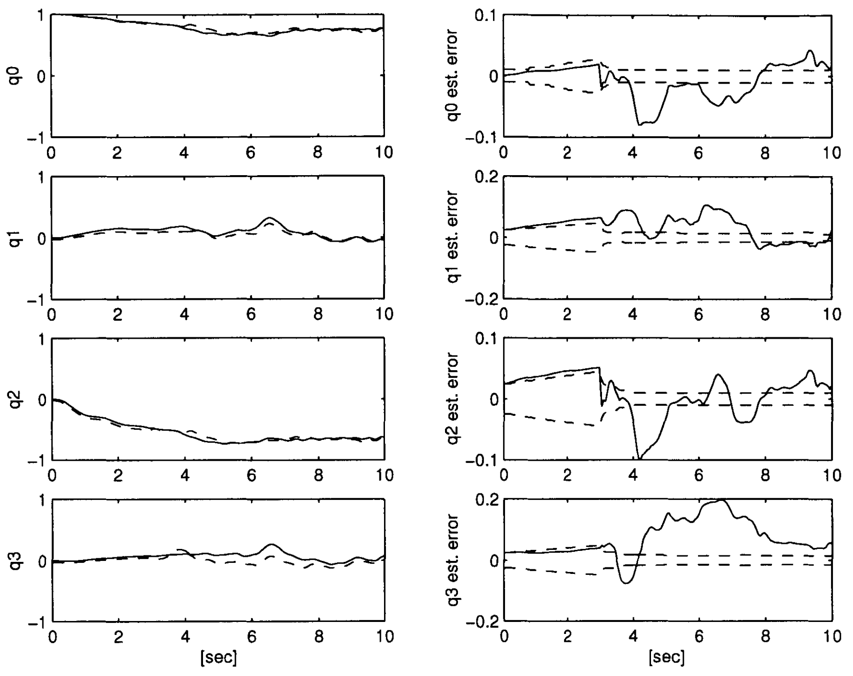


(A) Position

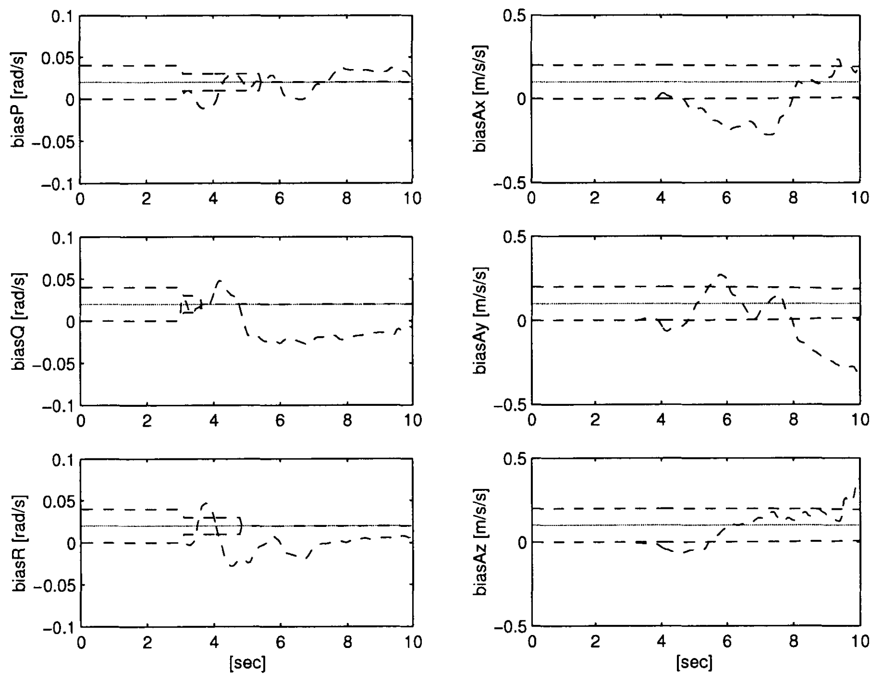


(B) Velocity

Figure 5-7: HILSIM: Drop from 380 m altitude without the 3-axis magnet sensor (Position and Velocity)

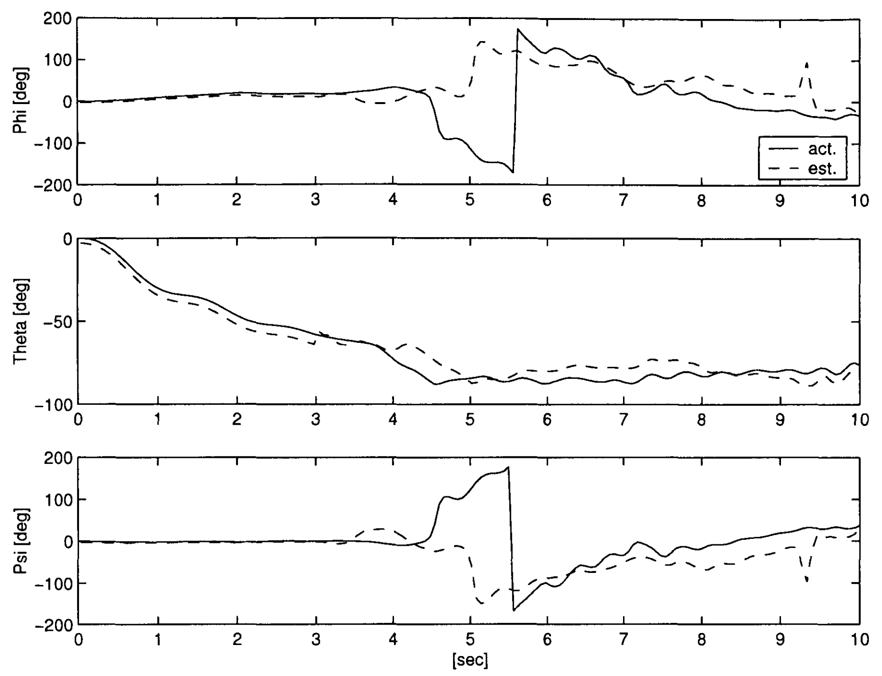


(C) Quaternion

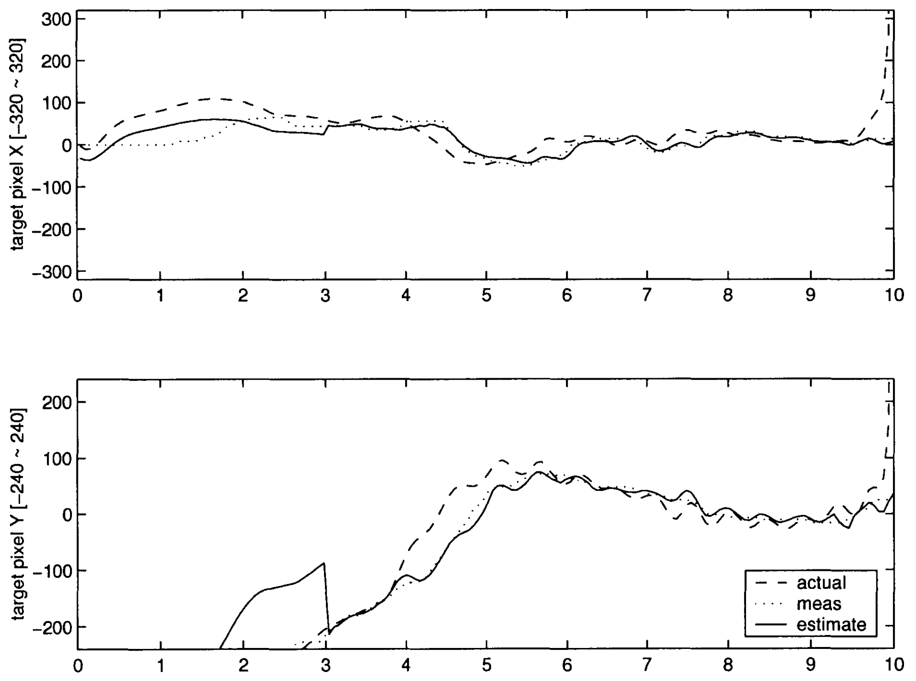


(D) Bias

Figure 5-8: HILSIM: Drop from 380 m altitude without the 3-axis magnet sensor (Quaternion and Bias)

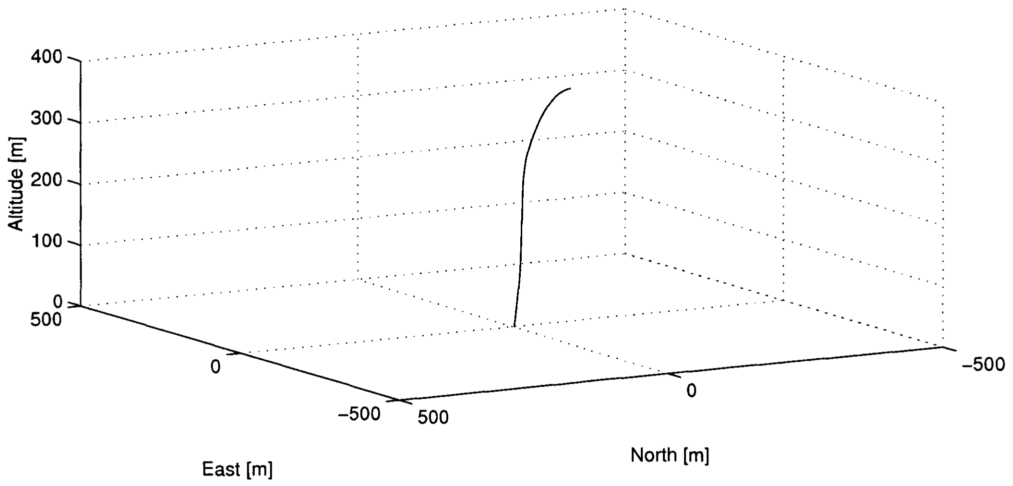


(E) Euler Angles

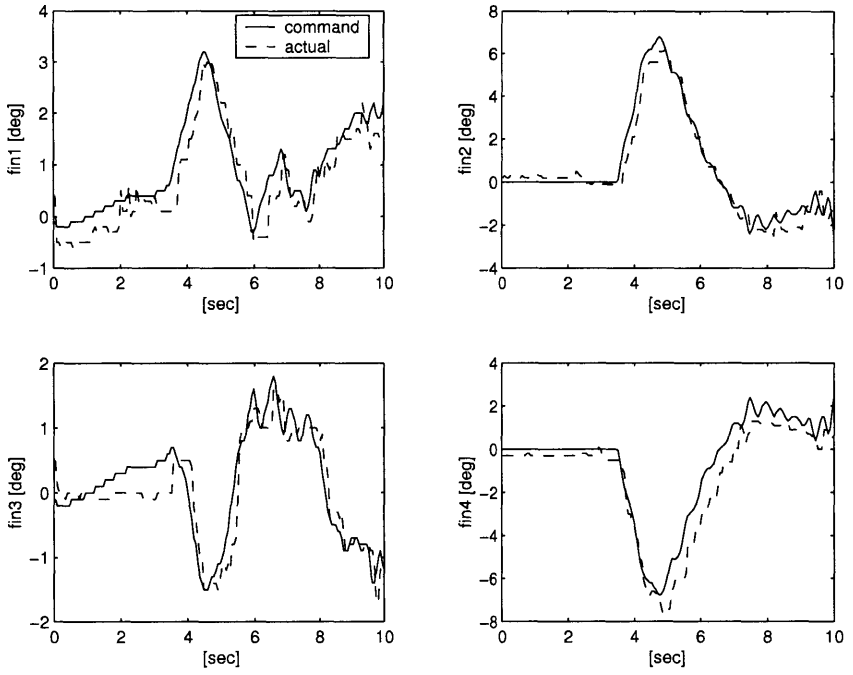


(F) Target Pixel Positions

Figure 5-9: HILSIM: Drop from 380 m altitude without the 3-axis magnet sensor (Euler Angles and Target Pixel Positions)

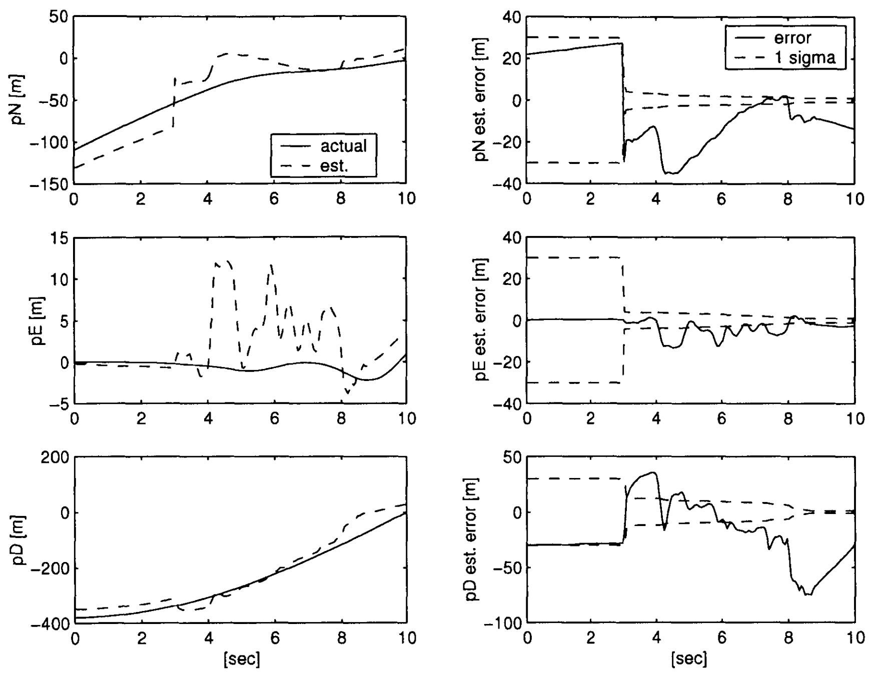


(A) Trajectory

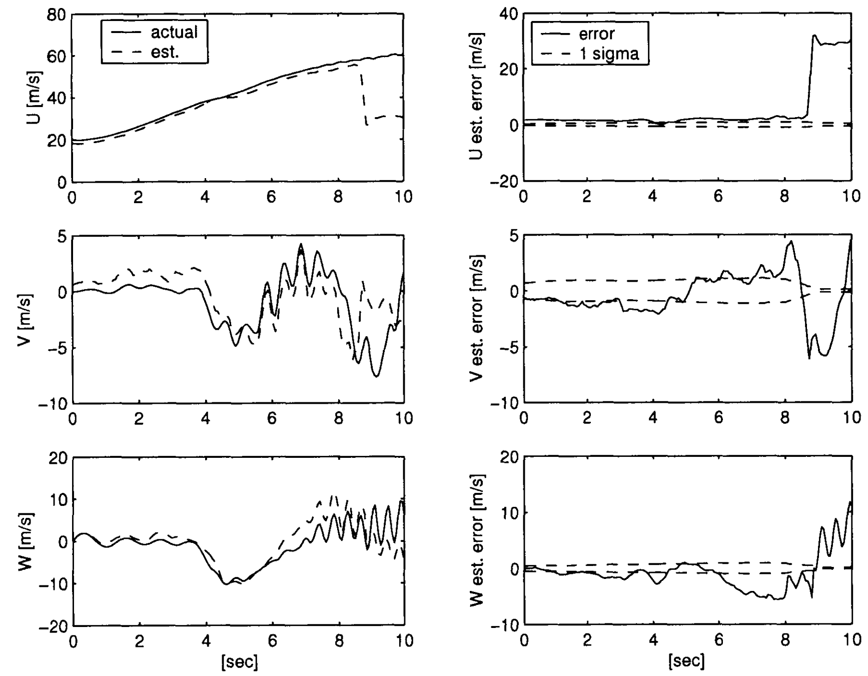


(B) Fin Deflection Angles

Figure 5-10: HILSIM: Drop from 380 m altitude without the 3-axis magnet sensor (Trajectory and Fin Deflection Angles)

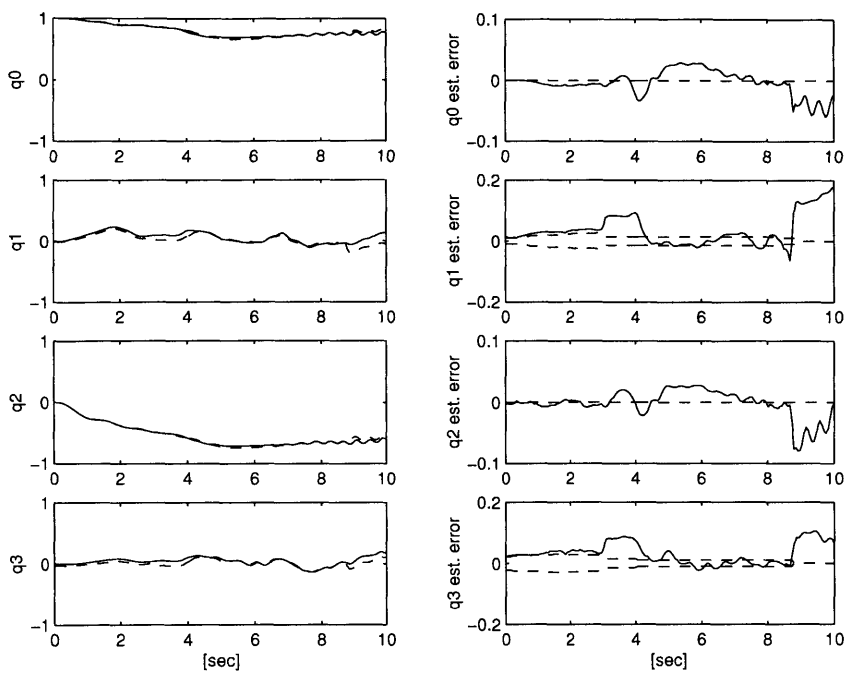


(A) Position

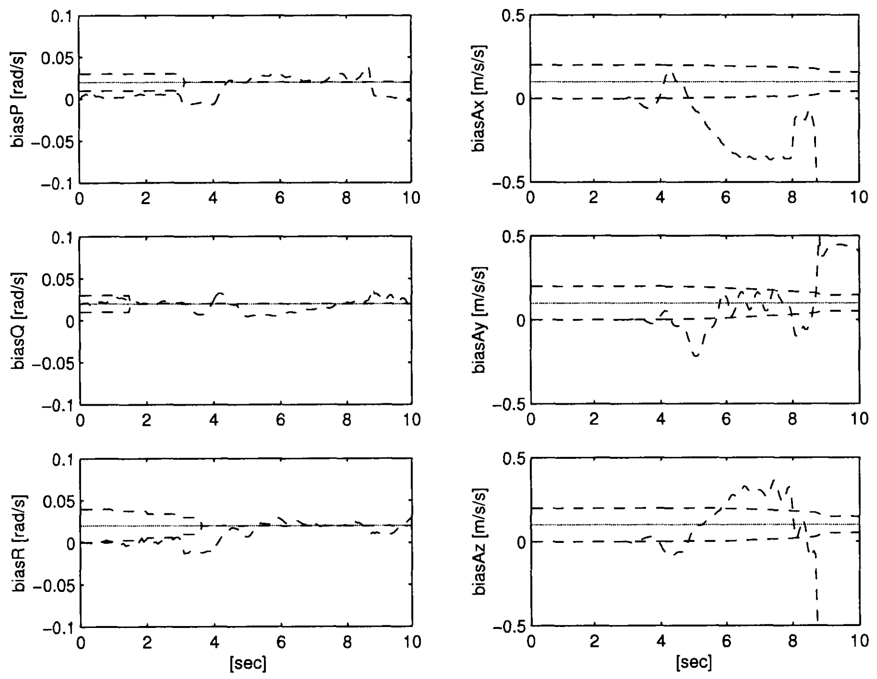


(B) Velocity

Figure 5-11: HILSIM: Drop from 380 m altitude with the 3-axis magnet sensor (Position and Velocity)

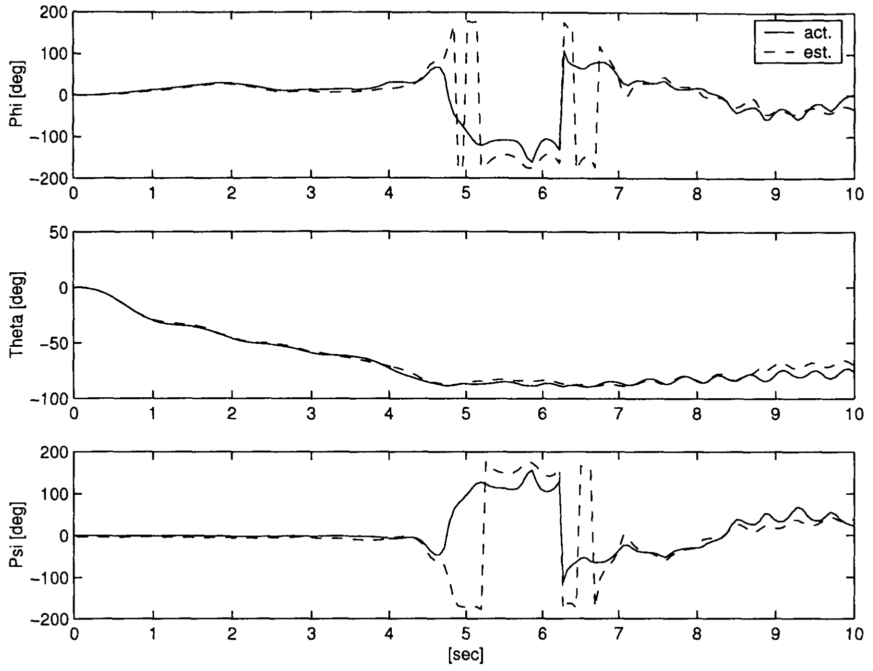


(C) Quaternion

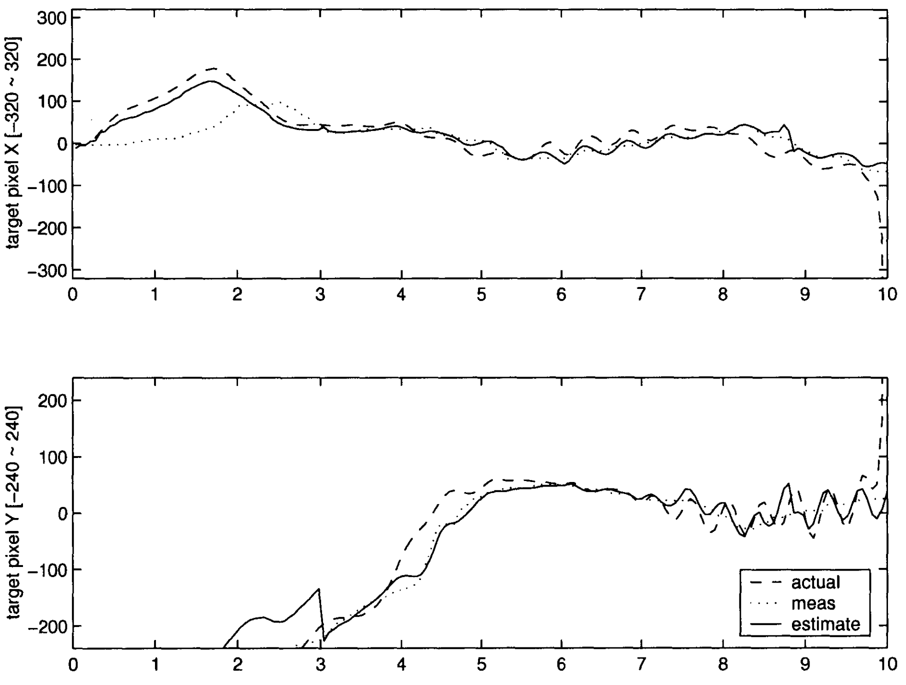


(D) Bias

Figure 5-12: HILSIM: Drop from 380 m altitude with the 3-axis magnet sensor (Quaternion and Bias)

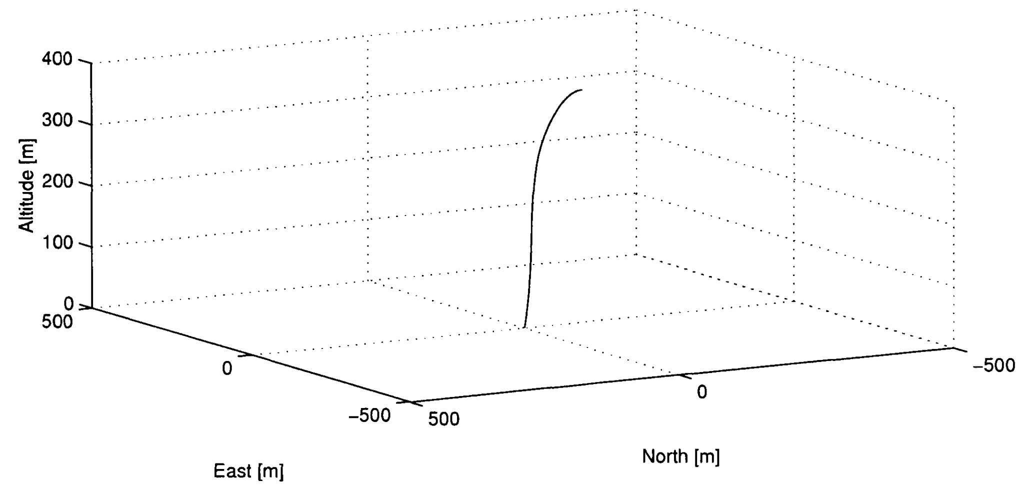


(E) Euler Angles

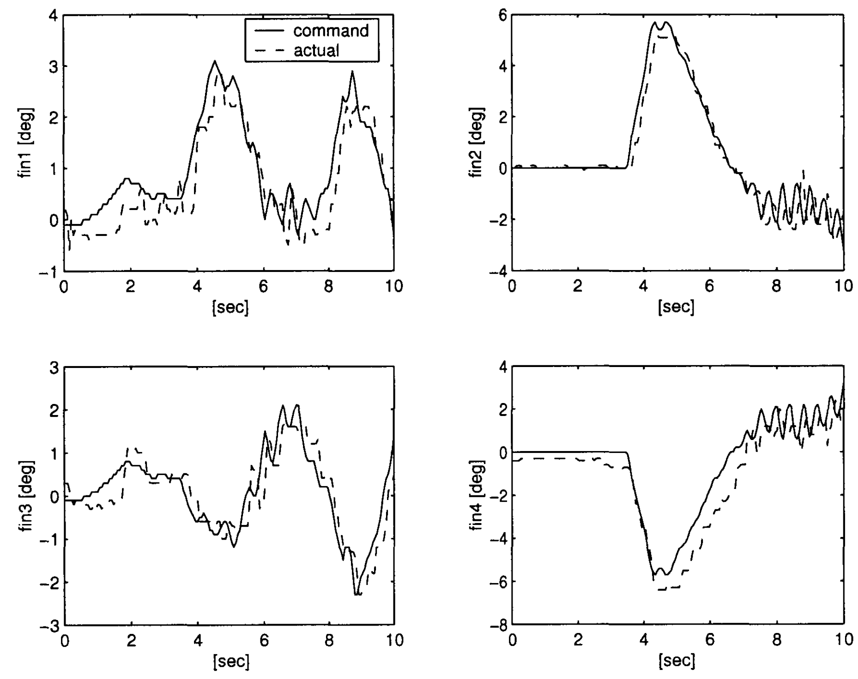


(F) Target Pixel Positions

Figure 5-13: HILSIM: Drop from 380 m altitude with the 3-axis magnet sensor (Euler Angles and Target Pixel Positions)



(A) Trajectory



(B) Fin Deflection Angles

Figure 5-14: HILSIM: Drop from 380 m altitude with the 3-axis magnet sensor (Trajectory and Fin Deflection Angles)

Chapter 6

Conclusions

6.1 Summary

This thesis suggested how to utilize unmanned aerial vehicles for the precision delivery of sensor nodes, presented a ballistic-type precision airdrop vehicle (PDV) and a vision system, and designed a vision-based estimation and control system to achieve precision drop.

The Precision Airdrop Framework is essentially composed of two components: the PDV component, the ground station (GS) component. Multiple types of wireless communication are used to interface these two components.

The Precision Airdrop Vehicle utilized several avionics components and software, both as a means for state estimation and for control of the vehicle trajectory. The GS has a very high-performance CPU and a frame grabber card for processing the received video. The ground station software and display allow a human operator to see the onboard camera images and track the target pixel locations, so as to guide the vehicle to the target.

The Hardware-In-the-Loop-Simulation (HILSIM) is a very powerful tool for verifying and debugging real-time embedded systems which operate with real-world inputs and outputs. The HILSIM setup is composed of the real-time systems and the HIL simulators. The real-time systems are:

- The PDV software and avionics components, except sensors which are emulated

- The ground station software and hardware including the frame grabber, the 802.11b wirless network card, and the video receiver

The HIL simulators are:

- The flight dynamics simulator and the flight dynamics simulation (FDS) software
- The camera vision simulator and the camera vision simulation (CVS) software

The Software Simulation was developed in the MATLAB Simulink environment. The software simulation allows one to design the state estimation and control loops and test the precision airdrop framework without using any avionics hardware.

The Vision-based Estimation incorporated the vision-based target position measurements in an extended Kalman filter with inertial sensor data and magnetic sensor measurements. Using camera images, the ground station allows a human operator to track the target image using a mouse device, and then transmits the tracked target pixel positions to the PDV on-board computer. In the future, an image processing module could be used to replace a human operator's target tracking.

The Extended Kalman Filter utilized two kinds of filtering algorithms:

- propagation of the estimated state vector and the error covariance matrix by digital integration of differential equations, and discrete measurement updates by Joseph formulation
- propagation of the estimated state vector by digital integration, discrete-time propagation of the error covariance matrix, and discrete measurement updates of state estimates by Carlson's square root filter formulation.

The former algorithm was used for the software simulation and the latter algorithm was applied to the hardware-in-the-loop-simulation and the PDV computer software.

The Vision-based Directional Controllers are implemented by the feedback of target pixel positions obtained by human tracking of target images using a computer mouse. The directional controls have pitch/yaw dampers as inner-loops. Linearized dynamics at different operating conditions are used for gain scheduling. A smooth transition algorithm is used when the directional control is first applied. In addition, an anti-windup algorithm is used for preventing the integrator saturation effect arising from the pure integrator in the PI control.

An Axial Controller controls rotation about the body x-axis so as to eliminate roll motion. This controller is required for stable camera vision and precise vision-based target tracking.

The HILSIM tests demonstrated good performance and reliability of the vision-based estimation and control design. The HILSIM tests indicate that the PDV launched from a carrier UAV with the speed of around 20 m/sec at the altitude of around 350 meters can reach the target point with a maximum error of 5 meters, provided that the launching position can be estimated within a maximum error of 50 meters from the ideal launching position. Hence the exact target position and the exact launching position, relative to a certain NED local inertial frame, are not necessary for the precision node delivery.

6.2 Future Work

At the time of writing of this thesis the real precision airdrop test has not yet been performed. However, several drop tests have been performed with surrogate vehicles to check the structural design of the PDV, the communication range of the wireless NTSC video link, the quality of camera images, and the parachute deployment mechanism. The sequential steps of the precision airdrop tests are as follows. (Note that the launching time of the PDV is regarded as 0 second):

Step 1. [-5 sec] The carrier with the PDV has level flight at the altitude of around 350 meters and the horizontal location of around 240 meters from the target.

Step 2. [-2 sec] The ground station operator sends a signal notifying the PDV program to start the state estimation. The signal is sent through the wireless 802.11b IPX network. The PDV starts state estimation.

Step 3. [0 sec] The PDV is launched by RC signal. The PDV initiates axial control.

Step 4. [3 sec] The ground station operator initiates tracking the target image and the PDV start vision-based estimation.

Step 5. [3.5 sec] The PDV initiates directional controls.

Step 6. [9.5 sec] The parachute is released at the altitude of around 30 meters.

Step 7. [15 sec] The ground station operator sends a signal notifying the PDV program to complete the mission.

Real precision airdrop tests will be performed in the near future, and a 10 meter $\times$ 10 meter square of white cloth will be used as a target point on the ground for testing the vision system and algorithms.

A more realistic human operator's tracking model can be made and the Monte-Carlo simulation can be applied to analyze the software simulation results and provide the accuracy average.

Regarding the vision-based target tracking models, an image processing module could be used to replace a human operator's target tracking. In order to develop the target tracking module, a Fast Fourier Transform algorithm can be used for extracting all the edges or features from the camera image. In addition, the temporal edges of our target on the camera image can be anticipated by making use of both the state estimate vector and the target shape during the state estimation. Finally, pattern recognition algorithms to compare the edge-extracted image with the anticipated-target edges can be applied for finding the center of the target on the camera image [29].

This research showed that vision sensors were a cheap means that allowed high precision deployment of sensor networks. Vision sensors may be a cheap alternative to high performance GPS with the comparable or better performance. More researches on vision systems are required to find the better vision systems.

Appendix A

The Extended Kalman Filter Setup

A.1 System Dynamics Matrix

First, with the navigation equation

$$\mathbf{f_1}(\mathbf{x}) = C\mathbf{v}_B = \begin{bmatrix} C_{11}u + C_{12}v + C_{13}w \\ C_{21}u + C_{22}v + C_{23}w \\ C_{31}u + C_{32}v + C_{33}w \end{bmatrix},$$

the followings are obtained:

$$F_{11} = \frac{\partial \mathbf{f_1}(\mathbf{x})}{\partial \mathbf{p}_L} = 0_{3\times3}, \quad F_{12} = \frac{\partial \mathbf{f_1}(\mathbf{x})}{\partial \mathbf{v}_B} = C$$

$$\begin{aligned} F_{13} &= \frac{\partial \mathbf{f_1}(\mathbf{x})}{\partial \mathbf{q}} \\ &= 2\begin{bmatrix} q_0u - q_3v + q_2w & q_1u + q_2v + q_3w & -q_2u + q_1v + q_0w & -q_3u - q_0v + q_1w \\ q_3u + q_0v - q_1w & q_2u - q_1v - q_0w & q_1u + q_2v + q_3w & q_0u - q_3v + q_2w \\ -q_2u + q_1v + q_0w & q_3u + q_0v - q_1w & -q_0u + q_3v - q_2w & q_1u + q_2v + q_3w \end{bmatrix} \end{aligned}$$

$$F_{14} = \frac{\partial \mathbf{f_1}(\mathbf{x})}{\partial \gamma_{P,Q,R}} = 0_{3\times3}, \quad F_{15} = \frac{\partial \mathbf{f_1}(\mathbf{x})}{\partial \gamma_{a_x,a_y,a_z}} = 0_{3\times3}$$

Second, with the force equation

$$\mathbf{f_2}(\mathbf{x}) = (\mathbf{a}_m + \gamma_\mathbf{a}) + C^{-1}\mathbf{g} - (\Omega_3 + \gamma_{\Omega_3})\mathbf{v}_B,$$

we get the followings:

$$F_{21} = \frac{\partial \mathbf{f_2}(\mathbf{x})}{\partial \mathbf{p}_L} = 0_{3\times3}, \quad F_{22} = \frac{\partial \mathbf{f_2}(\mathbf{x})}{\partial \mathbf{v}_B} = -(\Omega_3 + \gamma_{\Omega_3})$$

$$F_{23} = \frac{\partial \mathbf{f_2}(\mathbf{x})}{\partial \mathbf{q}} = \frac{\partial}{\partial \mathbf{q}} C^{-1}\mathbf{g} = \frac{\partial}{\partial \mathbf{q}} \begin{bmatrix} 2(q_1q_3 - q_0q_2)g \\ 2(q_2q_3 + q_0q_1)g \\ (q_0^2 - q_1^2 - q_2^2 + q_3^2)g \end{bmatrix} = 2g \begin{bmatrix} -q_2 & q_3 & -q_0 & q_1 \\ q_1 & q_0 & q_3 & q_2 \\ q_0 & -q_1 & -q_2 & q_3 \end{bmatrix}$$

$$F_{24} = \frac{\partial \mathbf{f_2}(\mathbf{x})}{\partial \gamma_{P,Q,R}} = -\frac{\partial}{\partial \gamma_{P,Q,R}} \gamma_{\Omega_3}\mathbf{v}_B = -\frac{\partial}{\partial \gamma_{P,Q,R}} \begin{bmatrix} -\gamma_R v + \gamma_Q w \\ \gamma_R u - \gamma_P w \\ -\gamma_Q u + \gamma_P v \end{bmatrix} = \begin{bmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{bmatrix}$$

$$F_{25} = \frac{\partial \mathbf{f_2}(\mathbf{x})}{\partial \gamma_{a_x,a_y,a_z}} = I_{3\times 3}$$

Third, with the attitude equation

$$\mathbf{f_3}(\mathbf{x}) = -\frac{1}{2}(\Omega_4 + \gamma_{\Omega_4})\mathbf{q},$$

the followings are obtained:

$$F_{31} = \frac{\partial \mathbf{f_3}(\mathbf{x})}{\partial \mathbf{p}_L} = 0_{3\times 3}, \quad F_{32} = \frac{\partial \mathbf{f_3}(\mathbf{x})}{\partial \mathbf{v}_B} = 0_{4\times 3}$$

$$F_{33} = \frac{\partial \mathbf{f_3}(\mathbf{x})}{\partial \mathbf{q}} = -\frac{1}{2}(\Omega_4 + \gamma_{\Omega_4})$$

$$\begin{aligned} F_{34} &= \frac{\partial \mathbf{f_3}(\mathbf{x})}{\partial \gamma_{P,Q,R}} = -\frac{1}{2}\frac{\partial}{\partial \gamma_{P,Q,R}} \gamma_{\Omega_4}\mathbf{q} \\ &= -\frac{1}{2}\frac{\partial}{\partial \gamma_{P,Q,R}} \begin{bmatrix} \gamma_P q_1 + \gamma_Q q_2 + \gamma_R q_3 \\ -\gamma_P q_0 - \gamma_R q_2 + \gamma_Q q_3 \\ -\gamma_Q q_0 + \gamma_R q_1 - \gamma_P q_3 \\ -\gamma_R q_0 - \gamma_Q q_1 + \gamma_P q_2 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} q_1 & q_2 & q_3 \\ -q_0 & q_3 & -q_2 \\ -q_3 & -q_0 & q_1 \\ q_2 & -q_1 & -q_0 \end{bmatrix} \end{aligned}$$

$$F_{35} = \frac{\partial \mathbf{f_3}(\mathbf{x})}{\partial \gamma_{a_x,a_y,a_z}} = 0_{4\times 3}$$

A.2 Process Noise Matrix

$E\left[\mathbf{w}_I\mathbf{w}_I^T\right]$:

$$\begin{aligned}\mathbf{w}_I &= -\nu_{\mathbf{a}} + \nu_{\Omega_3}\mathbf{v}_B \\ &= -\begin{bmatrix}\nu_{a_x}\\ \nu_{a_y}\\ \nu_{a_z}\end{bmatrix} + \begin{bmatrix}0 & -\nu_R & \nu_Q\\ \nu_R & 0 & -\nu_P\\ -\nu_Q & \nu_P & 0\end{bmatrix}\begin{bmatrix}U\\ V\\ W\end{bmatrix} \\ &= -\begin{bmatrix}\nu_{a_x} + \nu_R V - \nu_Q W\\ \nu_{a_y} - \nu_R U + \nu_P W\\ \nu_{a_z} + \nu_Q U - \nu_P V\end{bmatrix}\end{aligned} \tag{A.1}$$

Assuming no correlations between ν_{a_x}, ν_{a_y}, ν_{a_z}, ν_P, ν_Q, ν_R and let

$$\begin{array}{lll}\sigma_{a_x}^2 = E(\nu_{a_x}^2) & \sigma_{a_y}^2 = E(\nu_{a_y}^2) & \sigma_{a_z}^2 = E(\nu_{a_z}^2)\\ \sigma_P^2 = E(\nu_P^2) & \sigma_Q^2 = E(\nu_Q^2) & \sigma_R^2 = E(\nu_R^2)\end{array} \tag{A.2}$$

then we have

$$E\left[\mathbf{w}_I\mathbf{w}_I^T\right] = \begin{bmatrix}\sigma_{a_x}^2 + \sigma_R^2V^2 + \sigma_Q^2W^2 & -\sigma_R^2VU & -\sigma_Q^2WU\\ -\sigma_R^2VU & \sigma_{a_y}^2 + \sigma_R^2U^2 + \sigma_P^2W^2 & -\sigma_P^2WV\\ -\sigma_Q^2WU & -\sigma_P^2WV & \sigma_{a_z}^2 + \sigma_Q^2U^2 + \sigma_P^2V^2\end{bmatrix} \tag{A.3}$$

$E\left[\mathbf{w}_{II}\mathbf{w}_{II}^T\right]$:

$$\begin{aligned}\mathbf{w}_{II} &= \tfrac{1}{2}\nu_{\Omega_4}\mathbf{q} \\ &= \frac{1}{2}\begin{bmatrix}\nu_P q_1 + \nu_Q q_2 + \nu_R q_3\\ -\nu_P q_0 - \nu_R q_2 + \nu_Q q_3\\ -\nu_Q q_0 + \nu_R q_1 - \nu_P q_3\\ -\nu_P q_0 - \nu_Q q_1 + \nu_P q_2\end{bmatrix}\end{aligned} \tag{A.4}$$

Thus,

$$E\left[\mathbf{w}_{II}\mathbf{w}_{II}^T\right] = \frac{1}{4}\begin{bmatrix}d_{11} & d_{12} & d_{13} & d_{14}\\ d_{21} & d_{22} & d_{23} & d_{24}\\ d_{31} & d_{32} & d_{33} & d_{34}\\ d_{41} & d_{42} & d_{43} & d_{44}\end{bmatrix} \tag{A.5}$$

where

$$\begin{array}{ll}
d_{11} = \sigma_P^2 q_1^2 + \sigma_Q^2 q_2^2 + \sigma_R^2 q_3^2 & d_{22} = \sigma_P^2 q_0^2 + \sigma_R^2 q_2^2 + \sigma_Q^2 q_3^2 \\
d_{33} = \sigma_Q^2 q_0^2 + \sigma_R^2 q_1^2 + \sigma_P^2 q_3^2 & d_{44} = \sigma_R^2 q_0^2 + \sigma_Q^2 q_1^2 + \sigma_P^2 q_2^2 \\
d_{12} = d_{21} = -\sigma_P^2 q_1 q_0 + \sigma_Q^2 q_2 q_3 - \sigma_R^2 q_3 q_2 & d_{13} = d_{31} = -\sigma_P^2 q_1 q_3 - \sigma_Q^2 q_0 q_2 + \sigma_R^2 q_1 q_3 \\
d_{14} = d_{41} = \sigma_P^2 q_1 q_2 - \sigma_Q^2 q_2 q_1 - \sigma_R^2 q_3 q_0 & d_{23} = d_{32} = \sigma_P^2 q_0 q_3 - \sigma_Q^2 q_3 q_0 - \sigma_R^2 q_2 q_1 \\
d_{24} = d_{42} = -\sigma_P^2 q_0 q_2 - \sigma_Q^2 q_3 q_1 + \sigma_R^2 q_2 q_0 & d_{34} = d_{43} = -\sigma_P^2 q_3 q_2 + \sigma_Q^2 q_0 q_1 - \sigma_R^2 q_1 q_0
\end{array} \quad (A.6)$$

$E\left[\mathbf{w}_{II}\mathbf{w}_I^T\right]$:

$$\begin{aligned}
&E\left[\mathbf{w}_{II}\mathbf{w}_I^T\right] \\
&= -\tfrac{1}{2}E\left\{\begin{bmatrix} \nu_P q_1 + \nu_Q q_2 + \nu_R q_3 \\ -\nu_P q_0 - \nu_R q_2 + \nu_Q q_3 \\ -\nu_Q q_0 + \nu_R q_1 - \nu_P q_3 \\ -\nu_P q_0 - \nu_Q q_1 + \nu_P q_2 \end{bmatrix} \begin{bmatrix} \nu_{a_x} + \nu_R V - \nu_Q W \\ \nu_{a_y} - \nu_R U + \nu_P W \\ \nu_{a_z} + \nu_Q U - \nu_P V \end{bmatrix}^T \right\} \\
&= -\tfrac{1}{2}\begin{bmatrix} -\sigma_Q^2 q_2 W + \sigma_R^2 q_3 V & \sigma_P^2 q_1 W - \sigma_R^2 q_3 U & -\sigma_P^2 q_1 V + \sigma_Q^2 q_2 U \\ -\sigma_Q^2 q_3 W - \sigma_R^2 q_2 V & -\sigma_P^2 q_0 W + \sigma_R^2 q_2 U & \sigma_P^2 q_0 V + \sigma_Q^2 q_3 U \\ \sigma_Q^2 q_0 W + \sigma_R^2 q_1 V & -\sigma_P^2 q_3 W - \sigma_R^2 q_1 U & \sigma_P^2 q_3 V - \sigma_Q^2 q_0 U \\ \sigma_Q^2 q_1 W - \sigma_R^2 q_0 V & \sigma_P^2 q_2 W + \sigma_R^2 q_0 U & -\sigma_P^2 q_2 V - \sigma_Q^2 q_1 U \end{bmatrix}
\end{aligned} \quad (A.7)$$

$E\left[\mathbf{w}_{III}\mathbf{w}_{III}^T\right]$:

$$E\left[\mathbf{w}_{III}\mathbf{w}_{III}^T\right] = \begin{bmatrix} 0.001^2 & 0 & 0 \\ 0 & 0.001^2 & 0 \\ 0 & 0 & 0.001^2 \end{bmatrix} \quad (A.8)$$

$E\left[\mathbf{w}_{IV}\mathbf{w}_{IV}^T\right]$:

$$E\left[\mathbf{w}_{IV}\mathbf{w}_{IV}^T\right] = \begin{bmatrix} 0.001^2 & 0 & 0 \\ 0 & 0.001^2 & 0 \\ 0 & 0 & 0.001^2 \end{bmatrix} \quad (A.9)$$

Constructing the process noise matrix, the following sensor noises are applied:

$$\sigma_{a_x} = \sqrt{E(\nu_{a_x}^2)} = 0.1\ m/sec^2 \quad \sigma_{a_y} = \sqrt{E(\nu_{a_y}^2)} = 0.1\ m/sec^2 \quad \sigma_{a_z} = \sqrt{E(\nu_{a_z}^2)} = 0.1\ m/sec^2$$
$$\sigma_P = \sqrt{E(\nu_P^2)} = 0.01\ rad/sec \quad \sigma_Q = \sqrt{E(\nu_Q^2)} = 0.01\ rad/sec \quad \sigma_R = \sqrt{E(\nu_R^2)} = 0.01\ rad/sec$$

A.3 Measurement Matrices

Introducing

$$\vec{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{bmatrix} \equiv \mathbf{x}_{cam\to tar_C} = C^{-1}(\vec{x}_{target_L} - \vec{p}_L) - \vec{r}_{cam_B}$$

yields

$$\begin{aligned} h_1(\mathbf{x}) &= \frac{\{\mathbf{x}_{cam\to tar_C}\}_y}{\{\mathbf{x}_{cam\to tar_C}\}_x} \cdot \frac{1}{tan\frac{\alpha_x}{2}} \times \frac{N_x'}{2} = \frac{\mu_y}{\mu_x} \cdot \frac{1}{tan\frac{\alpha_x}{2}} \times \frac{N_x'}{2} \\ h_2(\mathbf{x}) &= -\frac{\{\mathbf{x}_{cam\to tar_C}\}_z}{\{\mathbf{x}_{cam\to tar_C}\}_x} \cdot \frac{1}{tan\frac{\alpha_y}{2}} \times \frac{N_y'}{2} = -\frac{\mu_z}{\mu_x} \cdot \frac{1}{tan\frac{\alpha_y}{2}} \times \frac{N_y'}{2} \end{aligned}$$

Thus,

$$H_{11} = \frac{\partial h_1(\mathbf{x})}{\partial \mathbf{p}_L} = \frac{\frac{\partial \mu_y}{\partial \mathbf{p}_L}\mu_x - \mu_y \frac{\partial \mu_x}{\partial \mathbf{p}_L}}{\mu_x^2} \cdot l_x$$

$$H_{21} = -\frac{\partial h_2(\mathbf{x})}{\partial \mathbf{p}_L} = -\frac{\frac{\partial \mu_z}{\partial \mathbf{p}_L}\mu_x - \mu_z \frac{\partial \mu_x}{\partial \mathbf{p}_L}}{\mu_x^2} \cdot l_y$$

where

$$\begin{aligned} \frac{\partial \mu_x}{\partial \mathbf{p}_L} &= \frac{\partial}{\partial \mathbf{p}_L}\left(-C^{-1}\mathbf{p}_L\right)_x = \text{1st row of } -C^{-1} \\ &= -\begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \frac{\partial \mu_y}{\partial \mathbf{p}_L} &= \frac{\partial}{\partial \mathbf{p}_L}\left(-C^{-1}\mathbf{p}_L\right)_y = \text{2nd row of } -C^{-1} \\ &= -\begin{bmatrix} 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \frac{\partial \mu_z}{\partial \mathbf{p}_L} &= \frac{\partial}{\partial \mathbf{p}_L}\left(-C^{-1}\mathbf{p}_L\right)_z = \text{3rd row of } -C^{-1} \\ &= -\begin{bmatrix} 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \end{aligned}$$

and

$$H_{13} = \frac{\partial h_1(\mathbf{x})}{\partial \mathbf{q}} = \frac{\frac{\partial \mu_y}{\partial \mathbf{q}}\mu_x - \mu_y\frac{\partial \mu_x}{\partial \mathbf{q}}}{\mu_x^2} \cdot \frac{1}{tan\frac{\alpha_x}{2}} \times \frac{N_x'}{2}$$

$$H_{23} = -\frac{\partial h_2(\mathbf{x})}{\partial \mathbf{q}} = -\frac{\frac{\partial \mu_z}{\partial \mathbf{q}}\mu_x - \mu_z\frac{\partial \mu_x}{\partial \mathbf{q}}}{\mu_x^2} \cdot \frac{1}{tan\frac{\alpha_y}{2}} \times \frac{N_y'}{2}$$

where

$$\frac{\partial \mu_x}{\partial \mathbf{q}} = -\frac{\partial}{\partial \mathbf{q}}\left(C^{-1}\mathbf{p}_L\right)_x = -\frac{\partial}{\partial \mathbf{q}}\left\{(q_0^2+q_1^2-q_2^2-q_3^2)x_L + 2(q_1q_2+q_0q_3)y_L + 2(q_1q_3-q_0q_2)z_L\right\}$$
$$= -2\begin{bmatrix} q_0x_L+q_3y_L-q_2z_L & q_1x_L+q_2y_L+q_3z_L & -q_2x_L+q_1y_L-q_0z_L & -q_3x_L+q_0y_L+q_1z_L \end{bmatrix}$$

$$\frac{\partial \mu_y}{\partial \mathbf{q}} = -\frac{\partial}{\partial \mathbf{q}}\left(C^{-1}\mathbf{p}_L\right)_y = -\frac{\partial}{\partial \mathbf{q}}\left\{2(q_1q_2-q_0q_3)x_L + (q_0^2-q_1^2+q_2^2-q_3^2)y_L + 2(q_2q_3+q_0q_1)z_L\right\}$$
$$= -2\begin{bmatrix} -q_3x_L+q_0y_L+q_1z_L & q_2x_L-q_1y_L+q_0z_L & q_1x_L+q_2y_L+q_3z_L & -q_0x_L-q_3y_L+q_2z_L \end{bmatrix}$$

$$\frac{\partial \mu_z}{\partial \mathbf{q}} = -\frac{\partial}{\partial \mathbf{q}}\left(C^{-1}\mathbf{p}_L\right)_z = -\frac{\partial}{\partial \mathbf{q}}\left\{2(q_1q_3+q_0q_2)x_L + 2(q_2q_3-q_0q_1)y_L + (q_0^2-q_1^2-q_2^2+q_3^2)z_L\right\}$$
$$= -2\begin{bmatrix} q_2x_L-q_1y_L+q_0z_L & q_3x_L-q_0y_L-q_1z_L & q_0x_L+q_3y_L-q_2z_L & q_1x_L+q_2y_L+q_3z_L \end{bmatrix}$$

A.4 Magnetic Sensor Measurement Matrix

$$H_3 = \tfrac{\partial \mathbf{h_3}(\mathbf{x})}{\partial \mathbf{q}}$$
$$= 2\begin{bmatrix} q_0b_N+q_3b_E-q_2b_D & q_1b_N+q_2b_E+q_3b_D & -q_2b_N+q_1b_E-q_0b_D & -q_3b_N+q_0b_E+q_1b_D \\ -q_3b_N+q_0b_E+q_1b_D & q_2b_N-q_1b_E+q_0b_D & q_1b_N+q_2b_E+q_3b_D & -q_0b_N-q_3b_E+q_2b_D \\ q_2b_N-q_1b_E+q_0b_D & q_3b_N-q_0b_E-q_1b_D & q_0b_N+q_3b_E-q_2b_D & q_1b_N+q_2b_E+q_3b_D \end{bmatrix}$$

Appendix B

Software Simulation in the MATLAB Simulink

B.1 Software Simulation Framework

B.2 Navigation Diagram

B.2.1 IMU

B.2.2 Magnetic Sensor

B.2.3 Target Tracking

B.3 Controller Diagram

B.3.1 Pixel-Y (Vertical Directional) Controller

B.3.2 Pixel-X (Horizontal Directional) Controller

B.3.3 Axial Controller

B.4 PDV Dynamics Diagram

B.5 Servo Dynamics Diagram

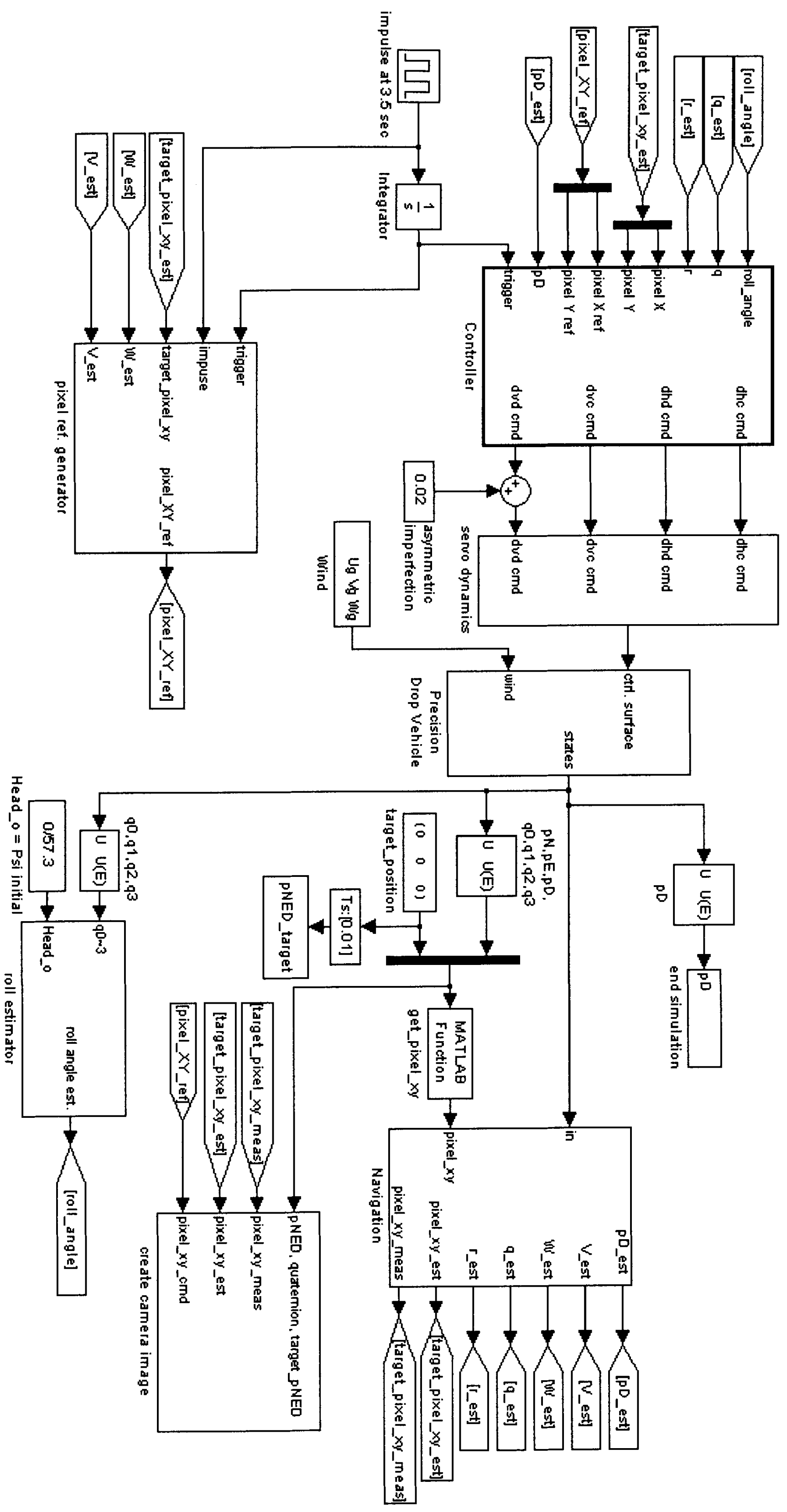


Figure B-1: Software Simulation Framework

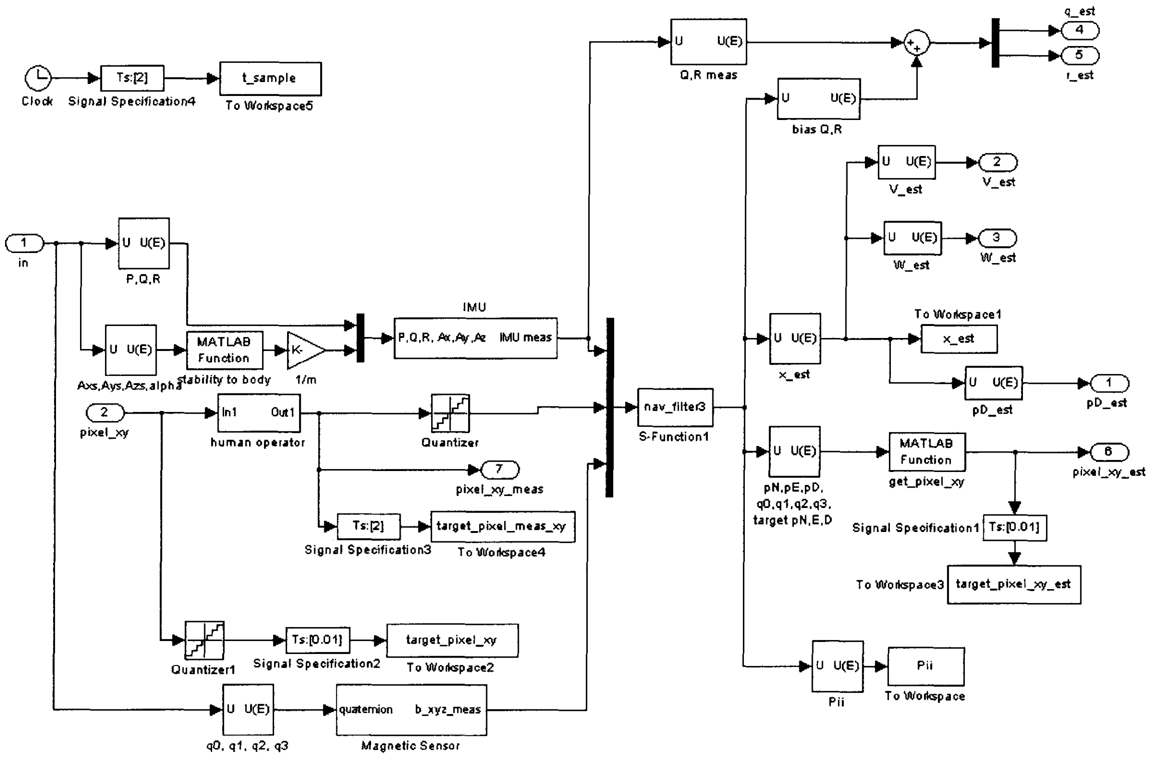


Figure B-2: Navigation Diagram

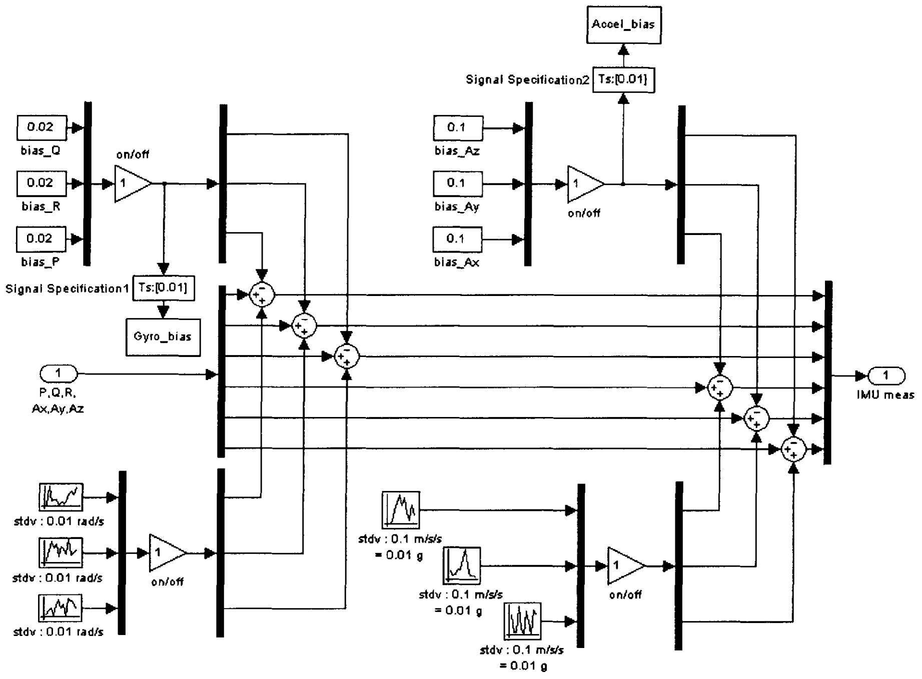


Figure B-3: IMU Diagram

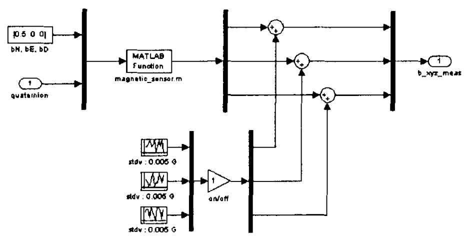


Figure B-4: Magnetic Sensor Diagram

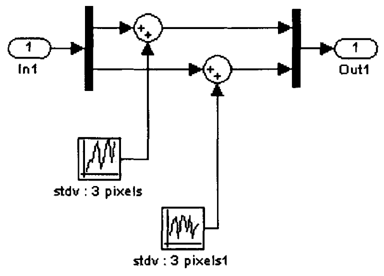


Figure B-5: Target Tracking Diagram

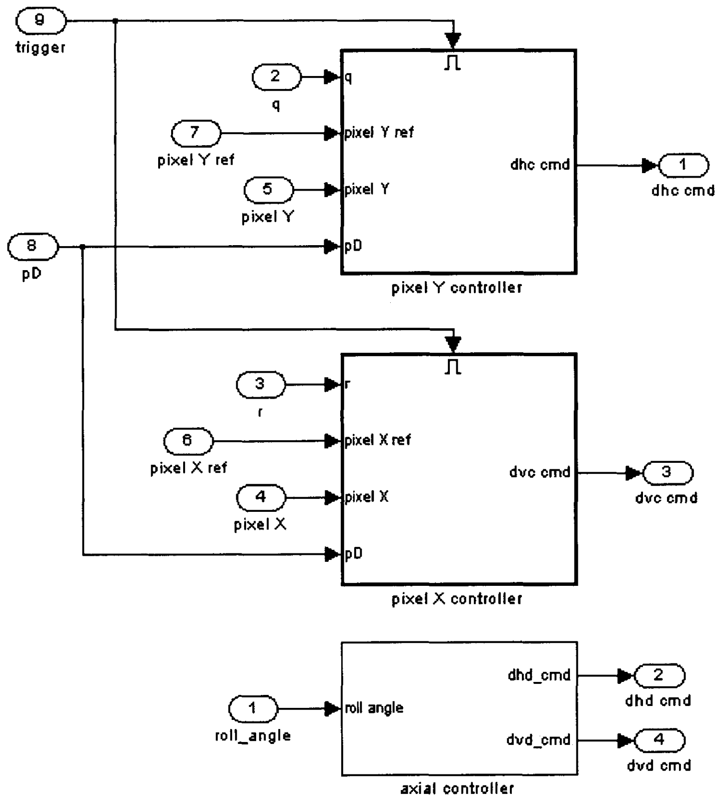


Figure B-6: Controller Diagram

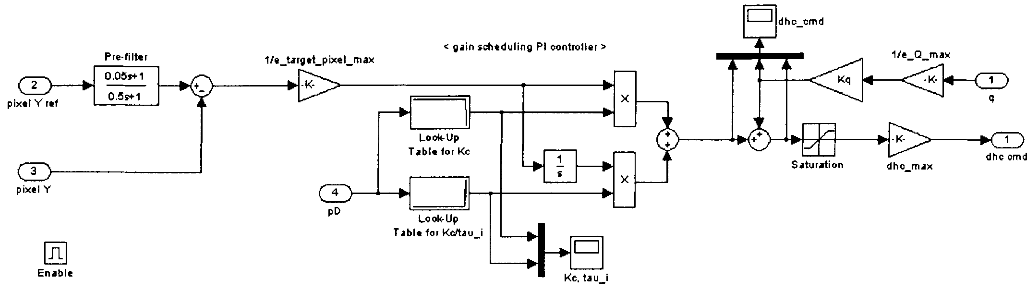


Figure B-7: Pixel-Y (Vertical Directional) Controller Diagram

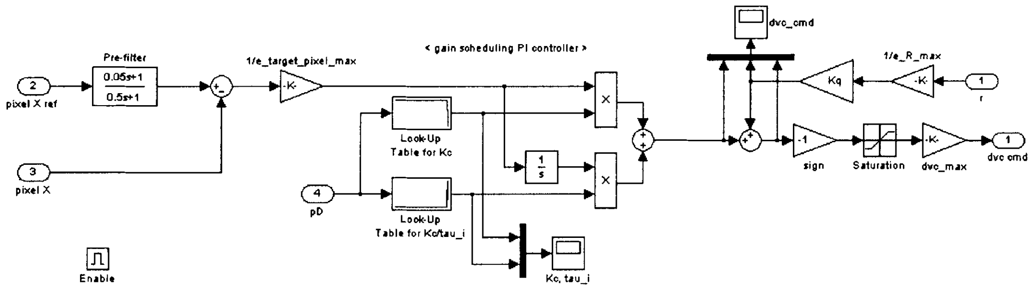


Figure B-8: Pixel-X (Horizontal Directional) Controller Diagram

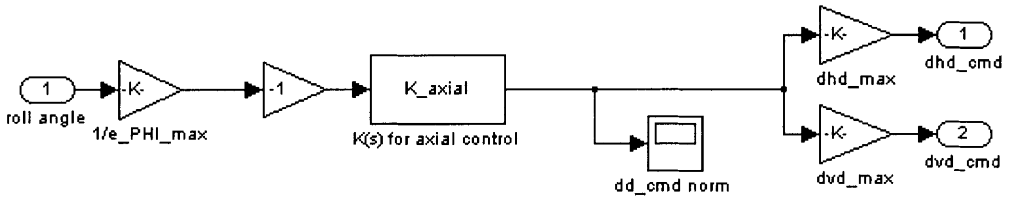


Figure B-9: Axial Controller Diagram

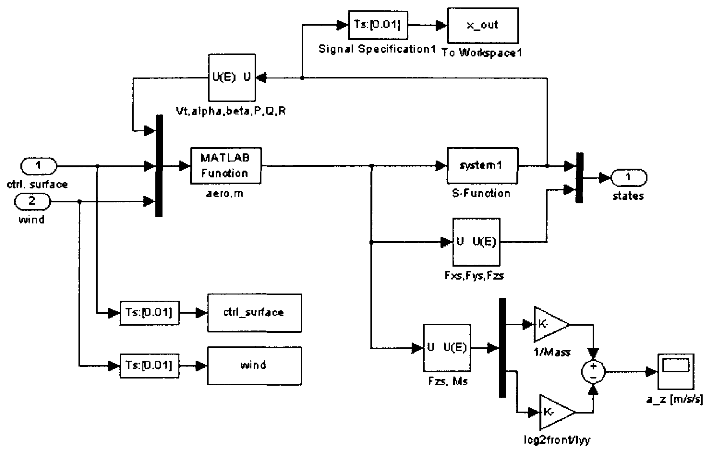


Figure B-10: PDV Dynamics Diagram

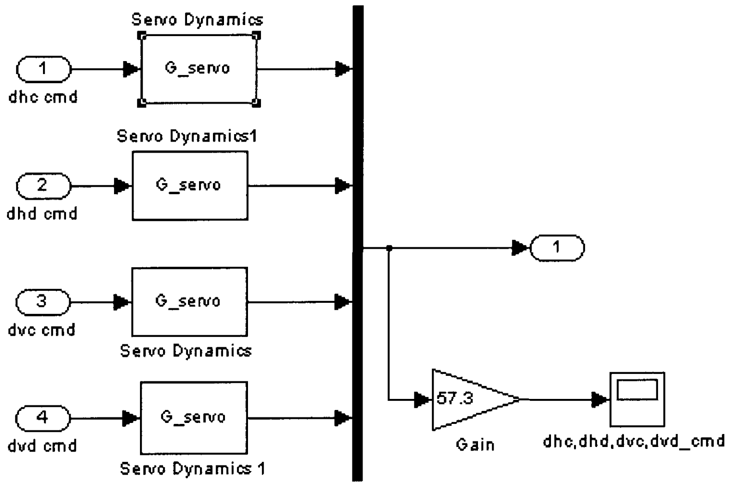


Figure B-11: Servo Dynamics Diagram

Bibliography

[1] CompuLab, Inc. *http://www.compulab.co.il/*. WWW

[2] Analog Devices, Inc. *http://www.analog.com/*.

[3] Crossbow Technology, Inc. *http://www.xbow.com/*. WWW

[4] Maxim Integrated Products, Inc. *http://www.maxim-ic.com/*. WWW

[5] Sensoray Company, Inc. *http://www.sensoray.com/*. WWW

[6] Honeywell International, Inc. *http://www.magneticsensors.com/*. WWW

[7] Pontech, Inc. *http://www.pontech.com/*. WWW

[8] Futaba, Inc. *http://www.futaba.com/*. WWW

[9] Supercircuits, Inc. *http://www.supercircuits.com/*. WWW

[10] Polaris Industries, Inc. *http://www.polarisusa.com/*. WWW

[11] Sterling Computer, Inc. *http://www.theportablepc.com/*. WWW

[12] D-link Systems, Inc. *http://www.dlink.com/*. WWW

[13] Linksys, A Division of Cisco Systems, Inc. *http://www.linksys.com/*. WWW

[14] Agere Systems Inc., formerly the Microelectronics Group of Lucent Technologies. *http://www.agere.com/* WWW

[15] Hitec RCD USA Inc. *http://www.hitecrcd.com/*. WWW

[16] Neal A. Carlson. *Fast Triangular Formulation of the Square Root Filter*. Vol. 11, NO. 9, AIAA, September 1973.

[17] Brian L. Stevens and Frank L. Lewis. *Aircraft Control and Simulation.* John Wiley and Sons, Inc., 1992.

[18] Arthur Gelb. *Applied Optimal Estimation.* MIT Press, 1974.

[19] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control - Analysis and Design.* John Wiley and Sons, Ltd., 1996.

[20] Paul Zarchan and Howard Musoff. *Fundamentals of Kalman Filtering - A Practical Approach, volumn 190, Progress in Astronautics and Aeronautics.* AIAA, 2000.

[21] Sanghyuk Park. *Avionics and Control System Development for Mid-Air Rendezvous of Two Unmanned Aerial Vehicles.* Ph.D dissertation, MIT, February 2004.

[22] Damian Toohey. *Development of a Small Parafoil Vehicle for Precision Delivery.* Master dissertation, MIT, September 2003.

[23] Jan Roskam. *Airplane Flight Dynamics and Automatic Flight Controls.* Roskam Aviation and Engineering Corporation, Kansas, 1979.

[24] Jr. John D. Anderson. *Fundamentals of Aerodynamics, 2nd ed.* McGraw-Hill, Maryland, 1991.

[25] Vladislav Gavrilets. *Avionics Systems Development for Small Unmanned Aircraft.* Master dissertation, MIT, June 1998.

[26] Vladislav Gavrilets. *Autonomous Aerobatic Maneuvering of Miniature Helicopters* Ph.D dissertation, MIT, May 2003.

[27] Katsuhiko Ogata. *Modern Control Engineering, 2nd ed.* Prentice-Hall, Inc. 1990.

[28] Gene F. Franklin, J. David Powell, Michael Workman. *Digital Control of Dynamic Systems, 3rd ed.* Addison Wesley Longman, Inc. 1998.

[29] Nikola K. Kasabov. *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering.* MIT Press, 1998.