ELEC495 Senior Desgin Project

Computer Pilot

Group 7

Nick McIntosh[1] Takashi V. Sato[2] Jing Chyuan Soong[3]
Chee Boon Tan[4] Yutaka Tsutano[5]

April 27, 2007

[1]E-mail: mcintoshnick69@hotmail.com
[2]E-mail: takabo31@bigred.unl.edu
[3]E-mail: jingchyuan_soong@yahoo.com
[4]E-mail: zhiwen020404@yahoo.com
[5]E-mail: yutaka@tsutano.com

Abstract

The purpose of this project is to build a fully autonomous stabilization and navigation system for a radio controlled (RC) helicopter. This project was chosen because it is an ambitious engineering design scheme that will challenge us as well as provide a level of sophistication that would allow it to be applicable to a year long project.

Contents

Chapter 1

Introduction

Helicopters present a challenging control problem with complex, noisy dynamics that prove to be significantly more difficult than fixed wing aircraft[**?**]. For instance, consider the hovering helicopter. With the main rotor spinning clockwise, when viewed from above, it produces a torque on the main chassis that is counter clockwise. The counter clockwise torque of the main rotor is corrected by a tail rotor that produces a rightward thrust, propelling the helicopter to the left. Therefore, for the helicopter to hover, it must be tilted slightly to the right to counteract its natural tendency to move to the left.

From the discussion above, you can see that precise information of the helicopters orientation and location is necessary for stable control. This control problem is solved using a combination of image processing and an inertial navigation system (INS). Furthermore, a centralized processing unit is developed to handle the incoming information and create control decisions to send back to the helicopter control system.

The results of this project can be further developed for more complicated applications such as underwater exploration, automated farming, and guided missiles. This report features the progress to date of an unmanned air vehicle (UAV) called the Computer Pilot project. All members of the design team contributed to this report.

Chapter 2

Project Goals and Major Objectives

The goal for this project is to design a control system and implement it on an RC helicopter.

- Design image processor to detect helicopter and determine its location and orientation.
- Design inertial navigation system to complement image processor.
- Program a microcontroller to take, as inputs, the data from the image processor and INS to make appropriate control decisions.

Chapter 3

Project Specifications

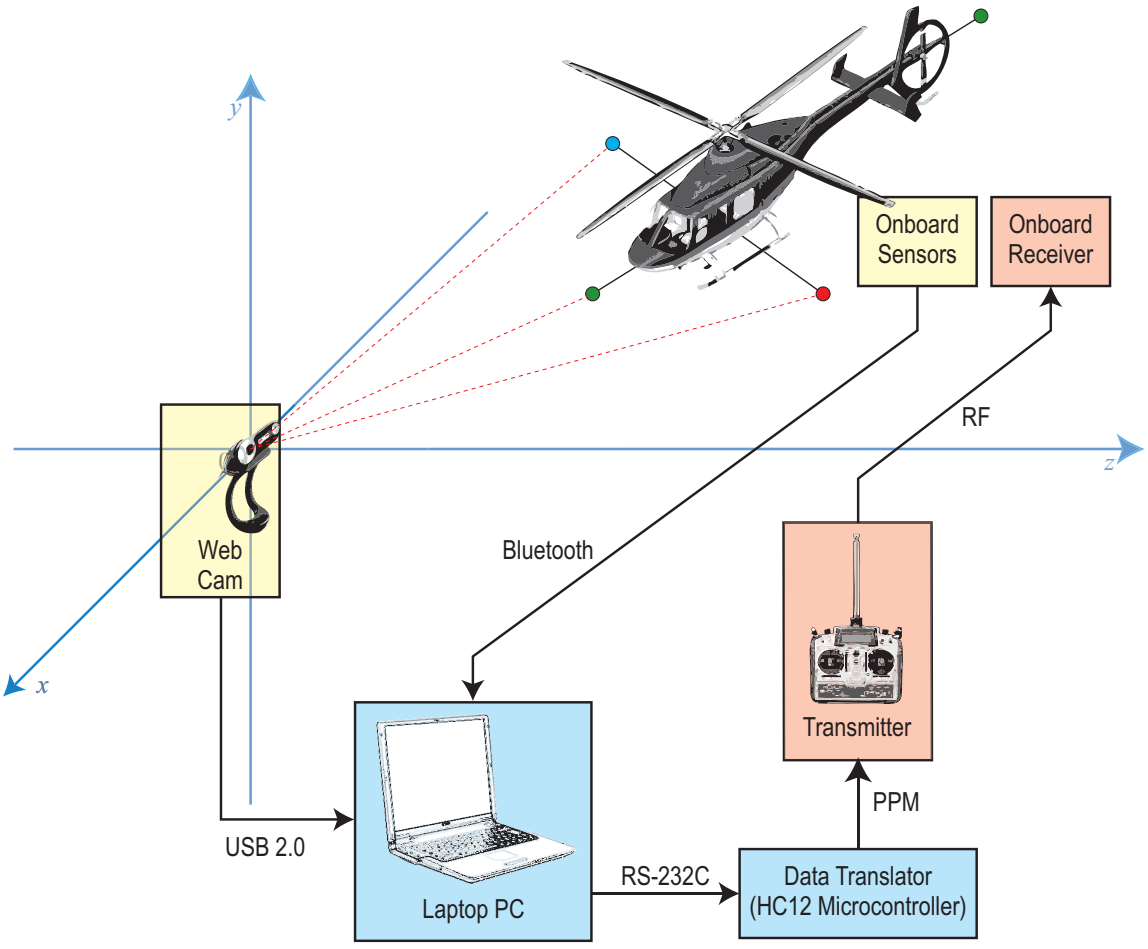


Figure 3.1: Overview of the system from the user's perspective.

To accomplish our goal, the location of the aircraft and the condition of its environment must be known with a high degree of accuracy. This is accomplished with the use of three main parts: image processor, inertial navigation system, data translator, and flight controller.

The helicopter is controlled by the flight controller software in the laptop PC. The flight controller software acquires the location of the helicopter from the image processor and the onboard sensors. Since each component uses different type of communication protocols, the data translator is needed to connect the components.

The control signal is transmitted by the transmitter. The transmitter can override the control signal, thus in case of emergency, a human pilot can take control of the helicopter immediately.

3.1 Image Processor

The image processor is a component which is responsible for converting camera images into the helicopter location in three dimensional coordinates. More specifically, it detects at least three bright points (LED beacons) on the helicopter, and applies nonlinear mathematical transform to get the helicopter position in 3-D coordinates.

This component is implemented as computer software, thus it runs on almost any Windows PCs equipped with camera devices. Also, the program is componentized so that the system runs on any computer such as Mac and Linux by adding some system dependent code.

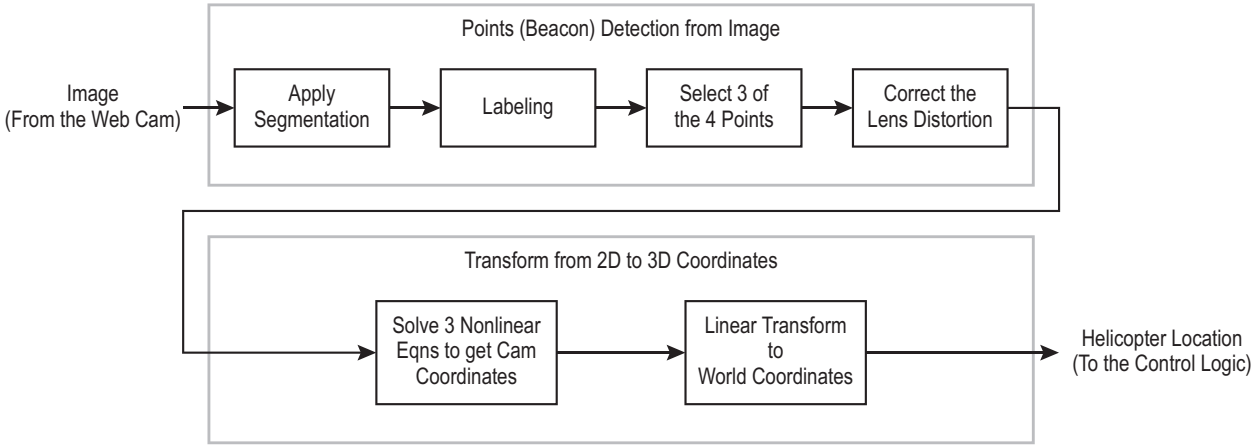


Figure 3.2: Overview of the Image Processor.

The overall process is shown in figure 3.2. As shown, the image processor consists of the following subcomponents:

- Point Detector—determines the position of the three beacons from the camera image.
- Coordinate Transformer—determines the position of the helicopter from the locations of the three beacons. In addition, this unit determines the position of the reference point of the helicopter and the attitude of the helicopter.

The implementation of this component is discussed in Section 4.1.

3.2 Inertial Navigation System

An Inertial Navigation System (INS) is responsible for providing measurements of pitch, roll and yaw of the system along with acceleration measurements in all directions. In addition, it is capable of measuring the height of the system as well as the angular velocity in each direction. These measurements can be used to determine the height and orientation of the vehicle it is mounted on. This inertial navigation system consists of a group of sensors such as an accelerometer, a magnetometer, a gyroscope, an ultrasonic rangefinder as well as a controller.

3.2.1 Requirements

- Obtain acceleration on three axis
- Obtain orientation on three axis
- Obtain rate of orientation change on three axis
- Obtain altitude of helicopter
- Transmit measured data to the data translator
- Receive control signal from the data translator

3.2.2 Sensors

Accelerometer

An accelerometer is an instrument used to measure the acceleration of an object. The accelerometer is installed in the system so that it measures the linear acceleration of the system in the inertial reference frame when the system rotates or moves. The triple-axis accelerometer (MMA 7261Q) from freescale semiconductor is capable of sensing movements in the X, Y and Z plane and measuring the system's acceleration in every direction.

Magnetometer

The strength, direction and fluctuation of magnetic fields can be measured using a magnetometer. The triple-axis magnetometer (Micromag3) used in the inertial navigation system measures the magnetic field in the X, Y and Z directions of the system. It is necessary for any orientation sensing or navigation system because it is used to determine the pitch, roll and direction (Yaw) of which the system is heading to.

Gyroscope

A dual-axis gyroscope (IDG-300) used in this inertial navigation system is very accurate in sensing the rate of rotation of the system about the X and Y axis. This precise measurement is necessary for accurately observing how fast the system is rotating.

Ultrasonic Rangefinder

The ultrasonic rangefinder uses sonar to measure the distance from the system to a reference frame. Therefore, by setting the ground as the reference frame, it can detect the height of the system with respect to the ground easily and accurately.

3.2.3 Sensor Controller

The controller is responsible for taking measurements from the sensors and communicating with the data translator. The unit controls each sensor sequentially to collect measurements. By performing multiple measurements and taking the average value, accuracy and dependability are improved. The unit communicates with the data translator using serial peripheral interface (SPI).

3.3 Data Translator

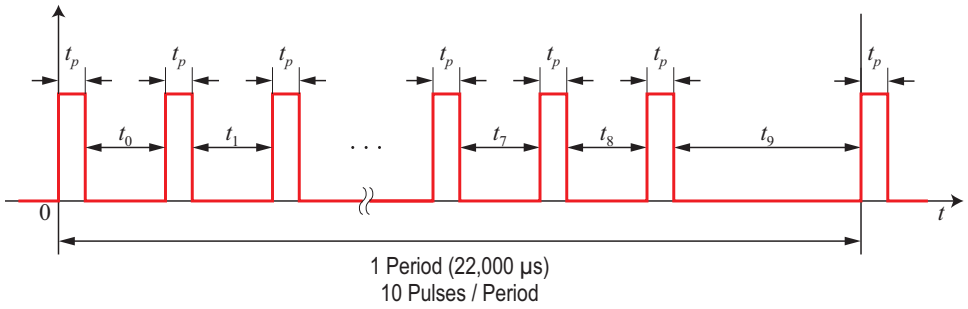


Figure 3.3: PPM waveform where $t_p = 400\,\mu$s.

The data translator serves as the communicator in the system. The primary task of the data translator is to control the helicopter based on the information from the flight controller. The other task is to communicate with the INS on the helicopter and translating the data for the flight controller.

3.3.1 Performance Specifications

- Performs data communication with PC using serial communication with RS-232
- Communication must occur more frequently than 30 times per second
- Communicate with the INS using SPI
- Meet specific data formats
- Generates Pulse Position Modulation (PPM) signal continuously
- PPM signal is updated every time a new data set is successfully received from PC

RS-232C is adopted for two reasons: it is fast enough for our application, and it is the easiest way to establish connection between the PC and the microcontroller. The communication must occur more frequently than 30 times per second because the image processing can optimally update its result at this rate. The communication with the INS is done serially using Serial Peripheral Interface (SPI). There are two data formats specified in the design at this point. First is the data format used in data reception from the PC; the other is the format of the PPM signal. To minimize the delay caused by the PPM generation, the old data must be replaced by the new data as soon as possible. However, the data should not be replaced until a successful reception of the new data set has occurred.

3.3.2 Hardware and Software

The data translator uses a Motorola 68HC12 microcontroller as its only hardware device. The microcontroller communicates with the PC and the INS. It also generates the PPM signal. The evaluation board, Adapt 912B32, has two serial interfaces (Serial Communication Interface and Serial Peripheral Interface) and timer subsystems, which are needed in the data translator.

The software is written using ANSI C language. The software is developed to perform both data communication and PPM generation. Since the PPM is used to control the helicopter, the program must be implemented such that the PPM generation should not experience any delay while the data translator is in operation.

3.3.3 Data Format

Serial Data Protocol from PC

Byte Number	**Value**	**Purpose**
0	00	Synchronization Bytes
1	00	
2	01	Data Type
3	$H(t_0)$	Data
4	$L(t_0)$	
5	$H(t_1)$	
6	$L(t_1)$	
...	...	
18	$H(t_8)$	
19	$L(t_8)$	
20	$H(t_9)$	
21	$L(t_9)$	
22	Parity	Parity (XOR of All Data)

Table 3.1: Data protocol.

The PC sends binary information to the data translator in specific protocol. The protocol is depicted in Table 3.1. The PC sends data in sequential bytes using RS-232C, with 9600 baud rate, 8-bit data, no-parity, and one start and one stop bit. The baud rate can be increased further to handle more information. The synchronization bytes and a data-type byte precede the data bytes. The synchronization bytes are used to detect the start of the data bytes and the data-type byte is used to determine the data type. Twenty two data bytes are sent immediately after the synchronization bytes. They represent ten 16-byte integers, and these numbers are used by the microcontroller to control the PPM signal. The entire set is repeatedly sent from the PC so that it can be received as soon as possible.

Seriad Data Protocol to PC

In order to send data received from the INS, a data format similar to the one discussed in the first part of Section 3.3.3 must be adopted. The detail is not determined at this moment.

PPM Signal

Pulse Position Modulation is used in the remote control system employed in the design. Since the system replaces human control of the helicopter with computer control, it is necessary to develop a device which generates control signal in PPM. The specification of the PPM signal is as follows. First, the period of the signal in which a whole set of information is transmitted is 22,000 micro seconds. Next, tp is the duration of each pulse in the signal. Each pulse is displaced from each other by t_0 thru t_9. These displacements between the pulses correspond to the control signal of each channel.

3.4 Flight Controller

The flight controller needs to control helicopters position and maintain stable hover. Since helicopter is free to move in three dimensional spaces, the controller has to work in these dimensions as well. The inputs to the helicopter are the 3D position of the helicopter, yaw, pitch and roll of helicopter. These inputs are used to calculate output signal levels from controller. The output is the data packet to the data translator which controls the servo movement on the helicopter. They are throttle, yaw, pitch and roll signals and allow full control over the helicopters movement.

Chapter 4

Implementation

4.1 Image Processor

4.1.1 Point Detector

Point Detector determines the position of the three beacons from the camera image. The detection consists of the following procedures:

1. Apply segmentation. In this process, the original image is converted to a black and white image. Any pixels brighter than a certain threshold are changed to white, and others are changed to black. Figure 4.1(b) and Figure 4.1(c) are the images after this process is applied to Figure 4.1(a) with threshold of 50% and 75%, respectively. In this case, threshold of 75% is more appropriate because the white part of the image more faithfully represents the positions of the LEDs.

2. Label each of the islands. After the segmentation process, the image is still a set of pixels. In this process, each set of continuous white pixels, what is called "island", is given a unique ID number. At the same time, the area and the center of gravity is calculated for each island.

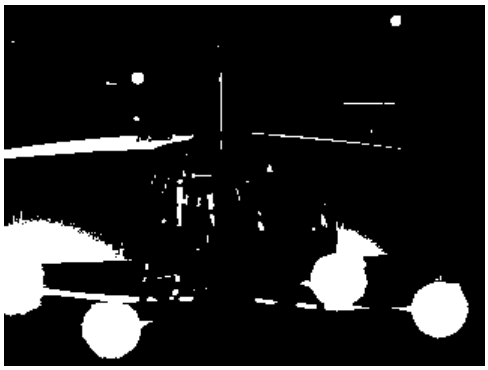
   To identify each island, depth first search (DFS) is used by considering each pixel as a vertex of a simple graph. Since the graph size is large, the search requires a large amount of computing power and careful implementation against stack overflow[1]. Figure 4.1(d) shows the labeled image.

3. Select 3 of the 4 beacons. In this process, 3 islands are selected so that each of them must be one of the 4 LEDs based on color and area information. Figure 4.1(d) shows the result of this process.

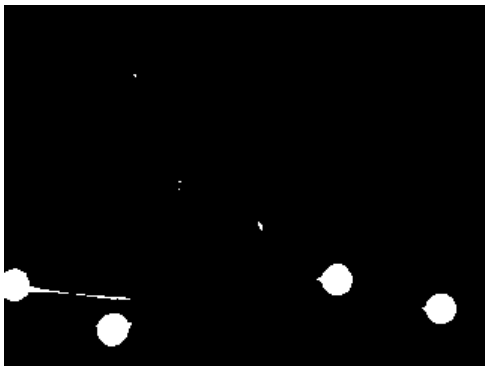
4. Correct the lens distortion. Since the center of gravity is already calculated in the labeling process, the positions of the selected islands are already known. However, not

[1]Typical DFS algorithm uses recursive function calls. When the graph size becomes too large, the function call level becomes too deep and causes stack overflow. Though it is possible to determine the maximum number of function calls quantitatively, the parameters such as the image size and the stack size are not fixed by the system, thus it is best to avoid recursive algorithm to ensure the generality of the program.

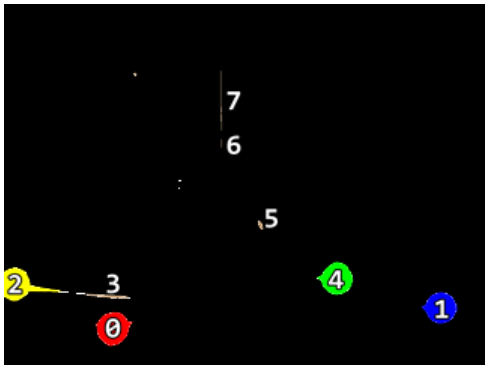
(a) Original image.



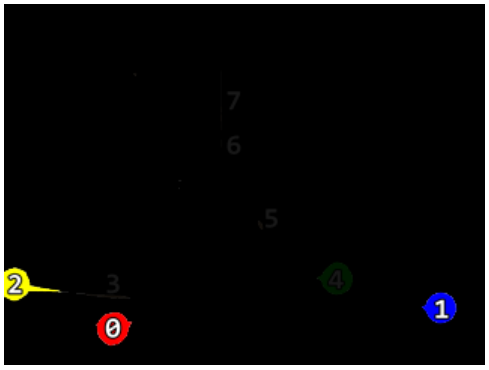
(b) Segmentation: 50%.



(c) Segmentation: 75%.



(d) Islands are labeled.



(e) 3 islands are selected.

Figure 4.1: Point detection.

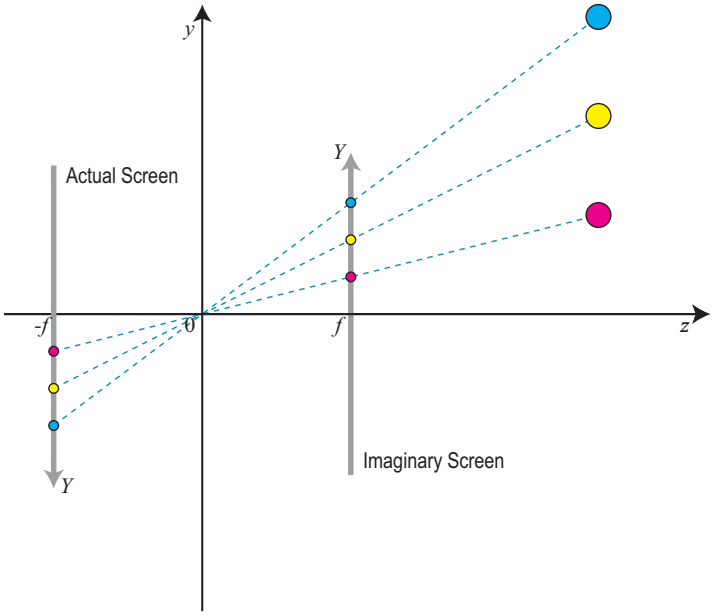


Figure 4.2: Imaginary screen used in the discussion.

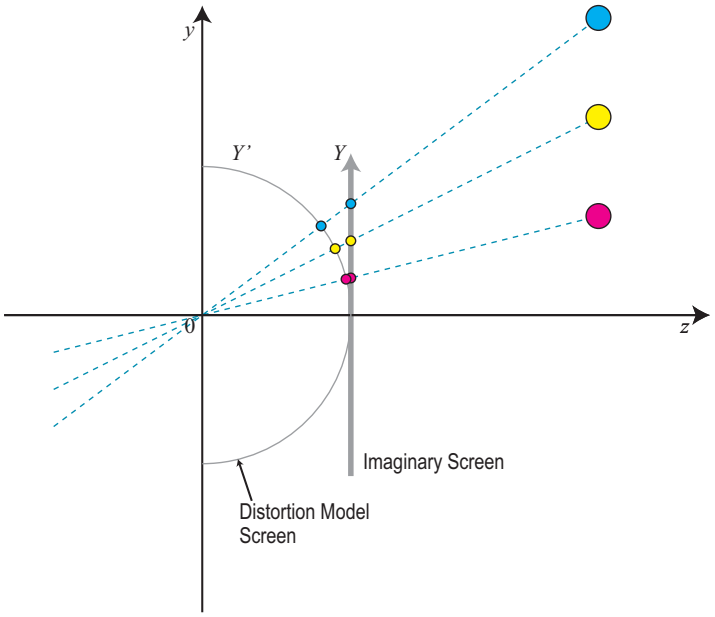


Figure 4.3: Lens distortion model. Using the geometry, the coordinates on the distorted screen can be mapped to the flat screen.

all lenses are ideal, and most of the lenses have distorted characteristics, thus correction is required. From Figure 4.3, correction can be done by using the following equations:

$$x = \frac{l \tan\left(\frac{\sqrt{x'^2+y'^2}}{l}\right)}{\sqrt{x'^2+y'^2}} x', \quad y = \frac{l \tan\left(\frac{\sqrt{x'^2+y'^2}}{l}\right)}{\sqrt{x'^2+y'^2}} y' \tag{4.1}$$

where x and y are the corrected positions, x' and y' are distorted positions, and l is a constant determined by the lens characteristics.

Point Detector is implemented as a DirectShow[2] filter to take full advantage of the platform's capabilities. As a result, the program achieves high generality in the platform and good performance in exchange of low portability to other operating systems. However, this subcomponent is kept small, so the low portability is insignificant. For example, this subcomponent can be implemented as a QuickTime component for Mac OS X to take advantage of the platform.

4.1.2 Coordinate Transformer

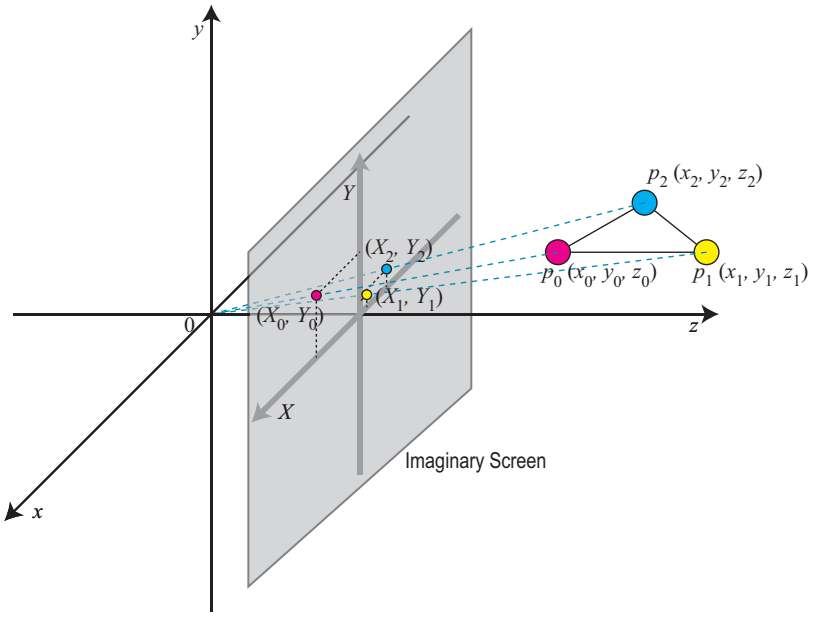


Figure 4.4: 2-D to 3-D conversion. Since the distance between each beacon's 3-D space is known, the three points on the imaginary screen can be uniquely converted into the 3-D coordinates with cross product verification.

The positions of the three LEDs given by point detector is in 2-D screen coordinates (Figure 4.4). They are converted into 3-D camera coordinates in this subcomponent. This component uses the following mathematical operations.

[2]See `http://windowssdk.msdn.microsoft.com/en-gb/library/ms783323(VS.80).aspx` for more details.

Let p_0, p_1 and p_2 be the points of the LEDs, and D_{ab} be the distance between the points p_a and p_b. Also the position of p_n be (X_n, Y_n) in 2-D screen coordinates and x_n, y_n, z_n in 3-D camera coordinates. Then

$$X_n = \frac{x_n}{z_n}, \quad Y_n = \frac{y_n}{z_n} \tag{4.2}$$

Now, the square of the distance can be shown as

$$(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2 = D_{ab}\,. \tag{4.3}$$

Using (4.2), the equation becomes

$$(X_b z_b - X_a z_a)^2 + (Y_b z_b - Y_a z_a)^2 + (z_b - z_a)^2 = D_{ab}\,, \tag{4.4}$$

or

$$(X_a^2 + Y_a^2 + 1)z_a^2 + (X_b^2 + Y_b^2 + 1)z_b^2 - 2(X_a X_b + Y_a Y_b + 1)z_a z_b - D_{ab} = 0\,. \tag{4.5}$$

Since there are three points, we have nonlinear equations

$$\begin{cases} f_0 = (X_0^2 + Y_0^2 + 1)z_0^2 + (X_1^2 + Y_1^2 + 1)z_1^2 - 2(X_0 X_1 + Y_0 Y_1 + 1)z_0 z_1 - D_{01} \\ f_1 = (X_0^2 + Y_0^2 + 1)z_0^2 + (X_2^2 + Y_2^2 + 1)z_2^2 - 2(X_0 X_2 + Y_0 Y_2 + 1)z_0 z_2 - D_{02} \\ f_2 = (X_1^2 + Y_1^2 + 1)z_1^2 + (X_2^2 + Y_2^2 + 1)z_2^2 - 2(X_1 X_2 + Y_1 Y_2 + 1)z_1 z_2 - D_{12} \end{cases}. \tag{4.6}$$

By solving this equation, the positions in 3-D camera coordinates can be obtained. However, there are two considerations:

- Equations (4.6) are nonlinear equations, and it is difficult to be solved symbolically. Therefore, it has to be solved numerically. In this case, Newton's method is used with the Jacobean

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_0}{\partial z_0} & \frac{\partial f_0}{\partial z_1} & \frac{\partial f_0}{\partial z_2} \\ \frac{\partial f_1}{\partial z_0} & \frac{\partial f_1}{\partial z_1} & \frac{\partial f_1}{\partial z_2} \\ \frac{\partial f_2}{\partial z_0} & \frac{\partial f_2}{\partial z_1} & \frac{\partial f_2}{\partial z_2} \end{bmatrix}, \tag{4.7}$$

  and the relation

$$\begin{bmatrix} \Delta z_0 \\ \Delta z_1 \\ \Delta z_2 \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} -f_0(z_0, z_1, z_2) \\ -f_1(z_0, z_1, z_2) \\ -f_2(z_0, z_1, z_2) \end{bmatrix}. \tag{4.8}$$

- Equations (4.6) have 2 solutions when $z > 0$. Therefore, the solution must be verified. In this case, verifying if the z-component of the cross product

$$\overrightarrow{p_0 p_1} \times \overrightarrow{p_0 p_2} \tag{4.9}$$

  is positive can be used since the helicopter will never be flipped over.

Unlike the Point Detector, this subcomponent is implemented as a C++ class. Therefore, this subcomponent can be used on any operating systems that support C++ just by recompiling the code.

Determining the Helicopter Position and Attitude

In 2-D to 3-D conversion, the position of the three LEDs are determined in 3-D space. Since the controller requires the position of the helicopter's reference point (possibly the center of gravity) and the attitude (pitch, roll, yaw) of the helicopter, the information should be converted into such format.

This conversion can be done by the following procedure:

1. Determine the inverse of an Affine transformation that represents a helicopter movement (rotation and translation) from the origin to the current position. This transformation can also be interpreted as a conversion of the position of the beacons from the local coordinates relative to the reference point to the world coordinates.

2. Using the Affine matrix determined in the previous step to determine the position and the attitude.

The problem of using this procedure is that the Affine transformation requires the position of the 4 beacons. Since only 3 of the positions are determined, it cannot be used as is. Careful observation reveals that this particular transformation includes rotation and translation, but not scaling. Using a cross product and applying the inverse of the rotation can solve this problem.

Let the positions of the beacons in the local coordinate as $\overrightarrow{l_0}$, $\overrightarrow{l_1}$ and $\overrightarrow{l_2}$ in the local coordinates and $\overrightarrow{w_0}$, $\overrightarrow{w_1}$ and $\overrightarrow{w_2}$ in the world coordinates. Also define $\mathbf{L}$ and $\mathbf{W}$ such that

$$\mathbf{L} = \left[\begin{array}{ccc} \overrightarrow{l_1} - \overrightarrow{l_0} & \overrightarrow{l_2} - \overrightarrow{l_0} & (\overrightarrow{l_1} - \overrightarrow{l_0}) \times (\overrightarrow{l_2} - \overrightarrow{l_0}) \end{array} \right] \tag{4.10}$$

$$\mathbf{W} = \left[\begin{array}{ccc} \overrightarrow{w_1} - \overrightarrow{w_0} & \overrightarrow{w_2} - \overrightarrow{w_0} & (\overrightarrow{w_1} - \overrightarrow{w_0}) \times (\overrightarrow{w_2} - \overrightarrow{w_0}) \end{array} \right] . \tag{4.11}$$

Then a rotation matrix $\mathbf{R}$ is

$$\mathbf{R} = \mathbf{W}\mathbf{L}^{-1} . \tag{4.12}$$

This rotation matrix can be used for calculating the attitudes as explained later. The position of the reference point (or the position of the helicopter) is

$$\overrightarrow{p} = \overrightarrow{w_k} - \mathbf{R}\overrightarrow{l_k} \tag{4.13}$$

where k is one of $\{0, 1, 2\}$.

Yaw is the rotation around y-axis and its conversion is represented by $\mathbf{R_y}$. Similarly, rhttp://en.wikipedia.org/wiki/Functionoll for $\mathbf{R_z}$, and pitch for $\mathbf{R_x}$. The direction of the rotations are defined such that they meets the requirements of the left-handed coordinate

system. Each of the rotation matrices are defined as

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \tag{4.14}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \tag{4.15}$$

$$\mathbf{R_z} = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.16}$$

The overall rotation is a result of the rotations of yaw, pitch then roll. Note that this order is extremely important for meaning rotation. Therefore,

$$\mathbf{R} = \mathbf{R_y}\mathbf{R_x}\mathbf{R_z} \tag{4.17}$$

$$= \begin{bmatrix} \cdots & \cdots & -\sin\theta_y\cos\theta_x \\ -\sin\theta_z\cos\theta_x & \cos\theta_z\cos\theta_x & \sin\theta_x \\ \cdots & \cdots & \cos\theta_y\cos\theta_x \end{bmatrix}. \tag{4.18}$$

Since $\mathbf{R}$ is already determined, the attitude $(\theta_x, \theta_y, \theta_z)$ can be determined by using (4.18) with the assumptions

$$-\frac{\pi}{2} < \theta_x < \frac{\pi}{2} \tag{4.19}$$

$$-\pi \leq \theta_y \leq \pi \tag{4.20}$$

$$-\frac{\pi}{2} < \theta_z < \frac{\pi}{2}. \tag{4.21}$$

4.2 Inertial Navigation System

The Inertial Navigation System is divided into two main sections. The first section includes the sensors: accelerometer, magnetometer, gyroscope and ultrasonic rangefinder. The other section contains the sensor controller. An overview of the system is shown Figure 4.5.

4.2.1 Hardware

Accelerometer

The triple-axis accelerometer (MMA7261) senses accelerations in the X, Y and Z plane. The accelerometer's readings can be obtained by using the analog-to-digital converter (ADC) of the microcontroller and from these values, the pitch and roll of the system can be evaluated. These readings will be used to keep track of how the system has turned, accelerated and decelerated.

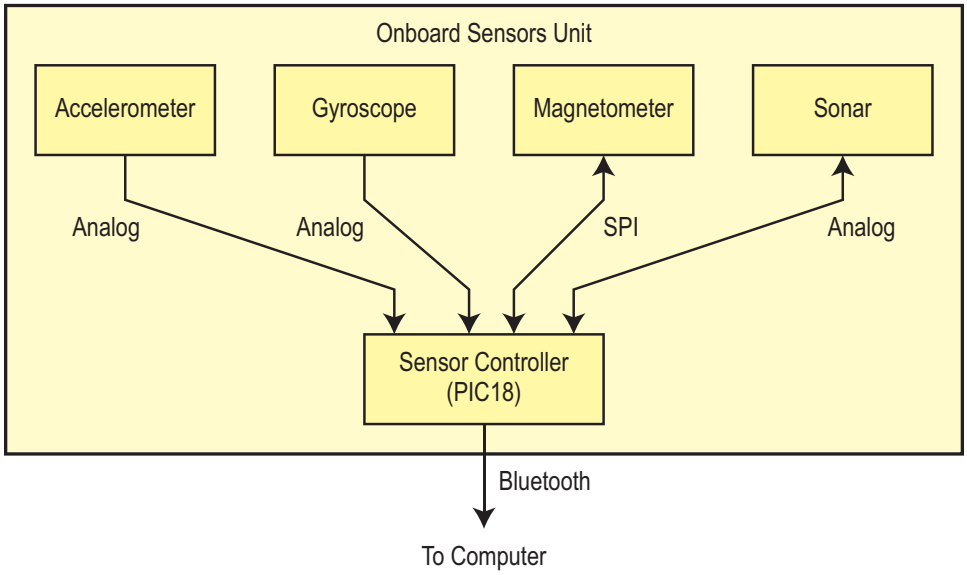


Figure 4.5: Overview of the Inertial Navigation System.

Magnetometer

The triple-axis magnetometer (Micromag3) measures the magnetic field of the earth in three directions to determine the pitch, roll and yaw of the system. The readings from the magnetometer are transferred to the microcontroller using Serial Peripheral Interface (SPI).

Gyroscope

The IDG-300 is an integrated dual-axis angular rate sensor to uses two sensors that sense the rate of rotation about the X and Y axis, respectively. The gyroscope defines a ground voltage (1.5V) to represent the plane where the X and Y axis lie. When there is rotation along the axis, the sensor's output voltage will increase according to the rate of rotation. The output voltage level reflects the rotation speed and therefore enables the determination of the angular acceleration. Since the microcontroller can only handle digital data, the information obtained from the gyroscope will need to be converted into digital form via ADC.

Ultrasonic Rangefinder

The MaxSonar EZ1 is able to provide sonar range information from 0.16 meter to 6.45 meters with a resolution of 0.025 meter. It is capable of detecting objects up to 6.45 meters which is sufficient for this project. The ultrasonic rangefinder operates by first transmitting signal to a reference frame and then measures the time needed to receive the reflected signal. This rangefinder has three different types of outputs including pulse width, analog voltage as well as serial digital.

Measurements

Using the output from both the X-axis and Y-axis of the magnetometer, yaw can be calculated using the following equation:

$$\text{Yaw} = \frac{180}{\pi} \tan^{-1}\left(\frac{Y}{X}\right) \tag{4.22}$$

Yaw has a range of 0 to 360 degrees where 0° is pointing to North, 90° is pointing to East, 180° is pointing to South and 270° is pointing to West.

The accelerometer has a selectable sensitivity of 480 mV/g, 360 mV/g, 160 mV/g and 120 mV/g. The sensitivity can be set by the user for a variety of applications. By measuring the output voltage from the Y-axis of the accelerometer, roll can be determined from:

$$\text{Roll} = \sin^{-1}\left(\frac{V_{\text{out},y} - 1.65\,\text{V}}{\text{selected sensitivity}}\right) \tag{4.23}$$

Roll also has a range of 0 to 360 degrees where 0° is horizontal, 90° is tilted to the right, 180° is inverted and 270° is tilted to the left.

In addition, by measuring the output voltage from the X-axis of the accelerometer, pitch can be calculated from:

$$\text{Pitch} = \sin^{-1}\left(\frac{V_{\text{out},x} - 1.65\,\text{V}}{\text{selected sensitivity}}\right) \tag{4.24}$$

Pitch has a range of -90 to 90 degrees where 0° is horizontal, -90° is tilted forward and 90° is tilted backward.

The angular velocity along the X and Y axis can be easily determined the readings from the gyroscope:

$$\text{Angular velocity}_x = \frac{V_{\text{out},x}}{2.0\,\text{mV}} \qquad (^\circ/\text{sec}) \tag{4.25}$$

$$\text{Angular velocity}_y = \frac{V_{\text{out},y}}{2.0\,\text{mV}} \qquad (^\circ/\text{sec}) \tag{4.26}$$

The full scale range for this dual-axis gyroscope is ±500 °/sec.

For the ultrasonic rangefinder, the distance between the system and the reference frame can be calculated from:

$$\text{Distance} = \frac{V_{\text{out},x}}{10\,\text{mV}} \qquad (^\circ/\text{sec}) \tag{4.27}$$

$$\tag{4.28}$$

This ultrasonic rangefinder provides very accurate readings of up to 255 inches with 1 inch increment.

Sensor Controller

The sensor controller is designed using Microchip PIC16F690 microcontroller. It is necessary to use a sensor controller in the INS because it is not feasible to interface each sensor to the

data translator directly. The number of wires we can use is limited by weight constraints. This microcontroller was chosen because the INS must be designed with minimal weight to reduce the load on a miniature helicopter. This microcontroller is in DIP with 20 pins is and very light compared to the 68HC12 evaluation board. The microcontroller fits our design needs perfectly because it has an A/D converter, Timer and has an SPI compatible serial interface.

4.2.2 Software

The software is being written in ANSI C language. Since the software is under progress, only the basic operation will be explained. The INS works repeating two steps. First, the sensor controller takes measurements of each sensor. Next, the data collected is transmitted to the data translator.

The measuring process is performed sequentially. The order and repetition of measurements are to be decided to optimize accuracy since each sensor has different accuracy and speed in measuring.

The data communication with the data translator is performed using SPI. The sensor controller sends data at a certain rate, say 50Hz. Then the data translator passes the data to PC. The communication is bi-directional so the PC can send control signals to initiate the calibration process and/or change measurement parameters of INS if desired.

4.3 Data Translator

4.3.1 Data Communication with PC

The data available from the PC is in the data format specified in Section 3.3.3. The data is sent serially via RS-232C to the microcontroller. The Serial Communication Interface (SCI) is employed to make the serial communication possible. Using SCI, the microcontroller can communicate the data in a series of bytes received. To receive the bytes from the PC, the program is dedicated for storing the received data. As you can see, the data format has synchronization bytes and a data type byte leading the actual data bytes. The logic tests every received byte in series to see if the synchronization bytes and the data-type byte have arrived in sequence. As soon as the synchronization is detected, the program starts to store the data bytes received after the synchronization bytes. Using this method, the PC can start sending the new data as soon as it is available, and the data translator can receive the new data accordingly.

The reverse process, sending data to PC is to be implemented. Basic operation of communication will be the same for the reception process because we utilize the same interface.

4.3.2 Data Communication with INS

The INS designed samples each sensor output multiple times and send average value to the data translator at rate of more than 30 times per second. Since the data is held in data translator momentary and sent to PC.

4.3.3 PPM Generation for the Transmitter

The PPM signal has the form of a square wave except that the duty cycle and frequency varies all the time. The microcontroller is used to toggle the output voltage between high and low to generate the square wave of our interest. To determine the timing for the toggling operation, a timer is used in the output compare configuration. The output compare is analogous to how an alarm clock works. You can choose anytime after the current time and set the alarm. As time passes by, time reaches the time you set and the alarm goes off.

Similarly, the timer uses a timer-counter register and output-compare register. The timer-counter register operates like a clock and advances at a certain rate, and the output-compare register holds the value when the alarm goes off. As soon as these two register values become equal, an interrupt is requested to the CPU. At each time the timer generates the interrupt request, a new value is stored in the output-compare register to set the next time to cause the interrupt. In summary, using the output compare function, a certain amount of time is measured for each high period and low period to attain desired PPM. The duration of high or low is specified by the data received from the PC. Since the PPM signal has 10 highs and 10 lows in every period, 20 numbers are used to control the timer. After every 20 output compare operations, either the new data set replaces the old data set or old data set is reused.

4.3.4 Software

The software is configured in a way to handle both of data communication and PPM generation at the same time. An interrupt driven program is used to achieve this. That is; the communicator part of the program is polling the arrival of new data either from PC or from INS at all times except when the interrupt request occurs from the timer. As soon as the interrupt is requested, the program takes a quick break from the data communication and updates the timer. Since the interrupt routine takes just tenths of clock cycles, the data communication is not interfered by interruption.

4.4 Flight Controller

4.4.1 Modeling the Helicopter and Assumptions

As everyone might know well, the dynamics of helicopter flight is so complex and hard to describe with mathematical relations we employed great deal of simplification based on observation and laying some assumptions.

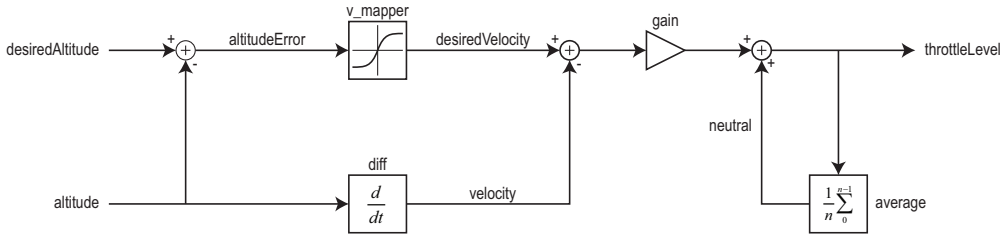
First assumption is that helicopter remains in hovering at low velocity in any direction as well as nearly no pitch and roll angles. Given this assumption, the helicopters four control axis; altitude, yaw, pitch, and roll can be treated as if they are independent from each other.

Second assumption is that helicopters thrust, vertical lift, is linearly related to main rotor deflection relative to horizon which is directly controlled by throttle signal. Also the pitching and rolling is limited to small angle so that thrust does not change at different pitch and roll angles, thus we assumed that we can control vertical force applied to helicopter freely.

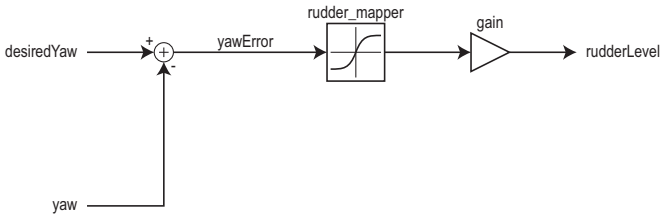
Third assumption is that the pitching and rolling movement of helicopter is directly controlled by servo movements of its axis and additionally, the speed of rotation has linear relation to servo movement. In other words, we can control how fast the helicopter pitches or rolls.

4.4.2 Design of Controller

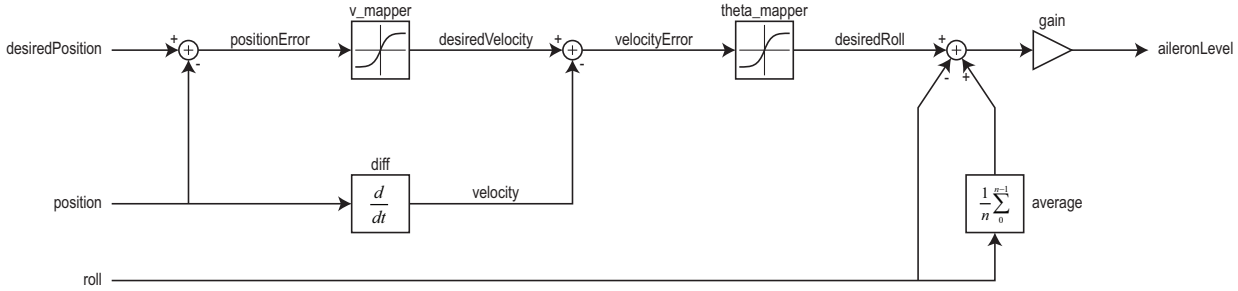
By observation, we learned that helicopters control has to base on position relative to destination and velocity relative to optimal velocity at moment. All of controllers are based on this approach.



(a) Altitude controller.



(b) Yaw controller.



(c) Roll/Pitch controller.

Figure 4.6: Overview of the controllers.

Altitude Controller

Altitude controller has current altitude and desired altitude as its inputs. Past two altitudes are used to determine current velocity by taking the time derivative. The output signal is computed as a function of current error in position and velocity as described in figure 4.6(a). Since we can control the force applied to the helicopter the output is proportional to error in velocity. After infinitesimal time duration, applied force yields change in velocity so that velocity error will decrease, and in return the position error will decrease until there is no error in position.

Yaw Controller

Yaw controller has current yaw and desired yaw as its input. Overview of yaw controller is in figure 4.6(b). Unlike altitude controller, yaw direction is ultimately unstable because the torque caused by main rotor has to be exactly cancelled by the tail rotor to maintain yaw stable. It is a must for our controller to rely on a yaw stabilizer which does cancel torque applied from main rotor by changing tail rotor deflection. This device reduces our problem significantly because the controller only needs to control creeping of heading. Thanks to the yaw stabilizer, the controller need to output signal proportional to heading error and the helicopter heading is smoothly controlled.

Pitch and Roll Controller

Pitch and Roll controller bases its decision on, current position, desired position, current velocity, desired velocity, current angle, desired angle. Due to the fact that pitch and roll posses almost identical characteristics, so we designed one kind of controller and applied for both axes. As shown in figure 4.6(c) the pitch and roll controller are different from other controller only in the rear end of the controller. This is because in these axes, we cannot control the force applied directly. We can only control its angle and the force applied to horizontal direction is approximated as

$$F \sin(\theta) \tag{4.29}$$

where F is the thrust and theta is angle of pitch or roll relative to vertical axis. Here, we took angle to be linearly proportional to acceleration for small angle of theta. So the controller controls the angle and controls velocity, in turn this controls position.

4.4.3 Testing and Result

These controllers are first tested individually with safety pilot. Using trainer function of transmitter one axis is passed to controller at a time and controller functionality was evaluated and calibrated. After attaining acceptable stability for each axis, two of controller is combined and being tested. Later three and finally all four axis of controller are connected and evaluated. This process was essential in success of our experiment because we could analyze each controller independently from other axis, at the same time we could avoid crashes.

The controller worked exceptionally reliable with reliable inputs from image processor. The controller was successful in keeping the helicopter in hovering, square path, turning 360 degrees. The stability was limited because our model for the helicopter was too simple to control the pitch and roll precisely.

The future possible improvement would be more detailed description of pitch and roll dynamics so that these axes can be controlled more efficiently so the movement will be more smooth and less of creep.

4.5 User Interface

Figure 4.7 shows the interface of the software on the observer's computer. The window shows the processed image, the position/attitude of the helicopter, controller outputs, and miscellaneous information useful for the observer. Most of the configurations required for the controllers and the image processor are read from a configuration file in a text format.

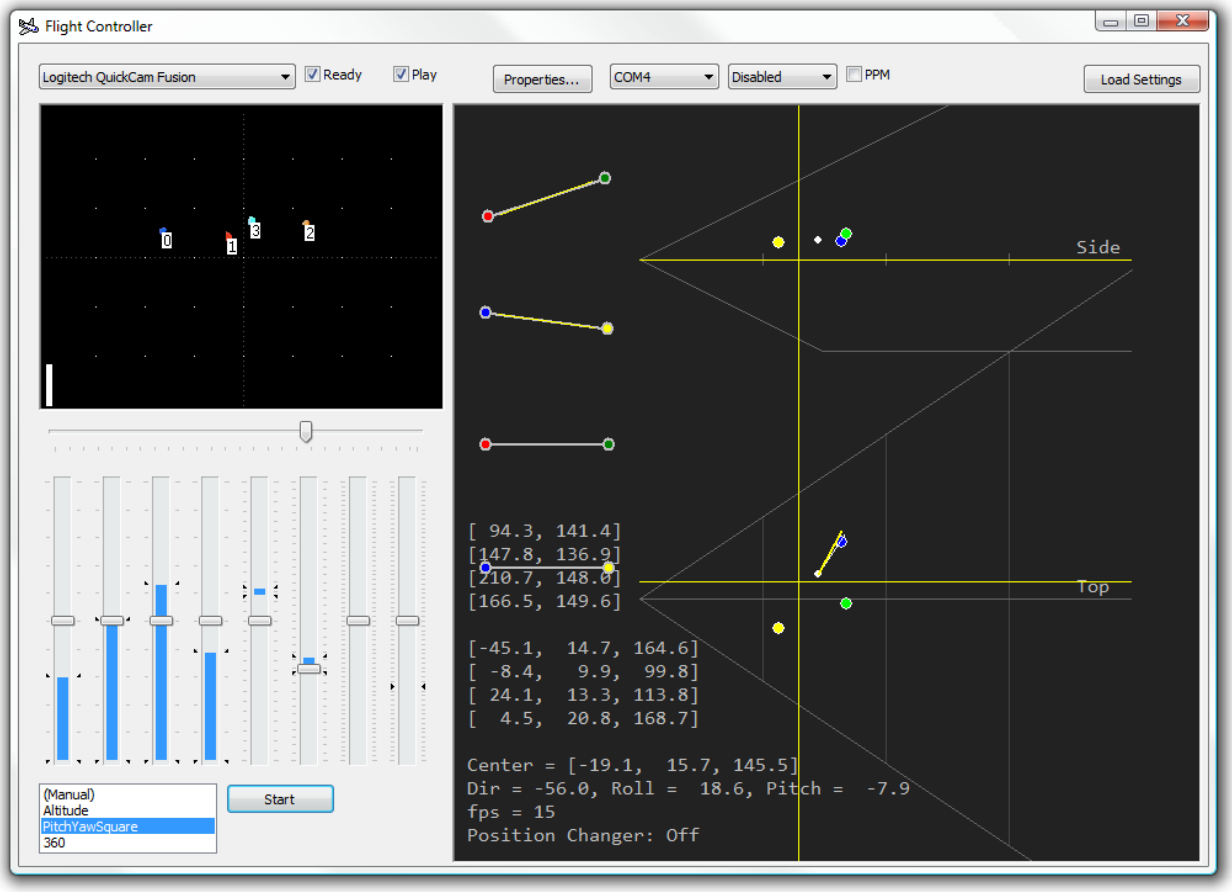


Figure 4.7: Screenshot of the flight controller with the controllers on.

Chapter 5

Project Milestones and Completion Dates

- Accelerometer had been tested and each axis output voltage had been measured to calculate the angle of pitch and roll (10/25/06).
- The PPM specification was obtained experimentally (9/25/06). The specification is necessary to develop the software.
- A test program for the PPM generation was written (10/7/06). This part of source code is most important in hardware interfacing.
- The Data Translator was tested using special set-up (10/18/06). This test verifies the completion of the Data Translator mini-project. Now, the team can fly the helicopter and make measurements for sensors and control theory development.
- The accelerometer is interfaced with 68HC12 by using voltage scalar and A/D converter (11/27/06).
- PIC microcontroller is programmed with a test program. This verifies the development environment is well functional (12/03/06).

Chapter 6

Mini Projects

6.1 Image Processor: Point Detection

In the Image Processor miniproject, the image processor component is implemented and tested.

6.1.1 Procedure

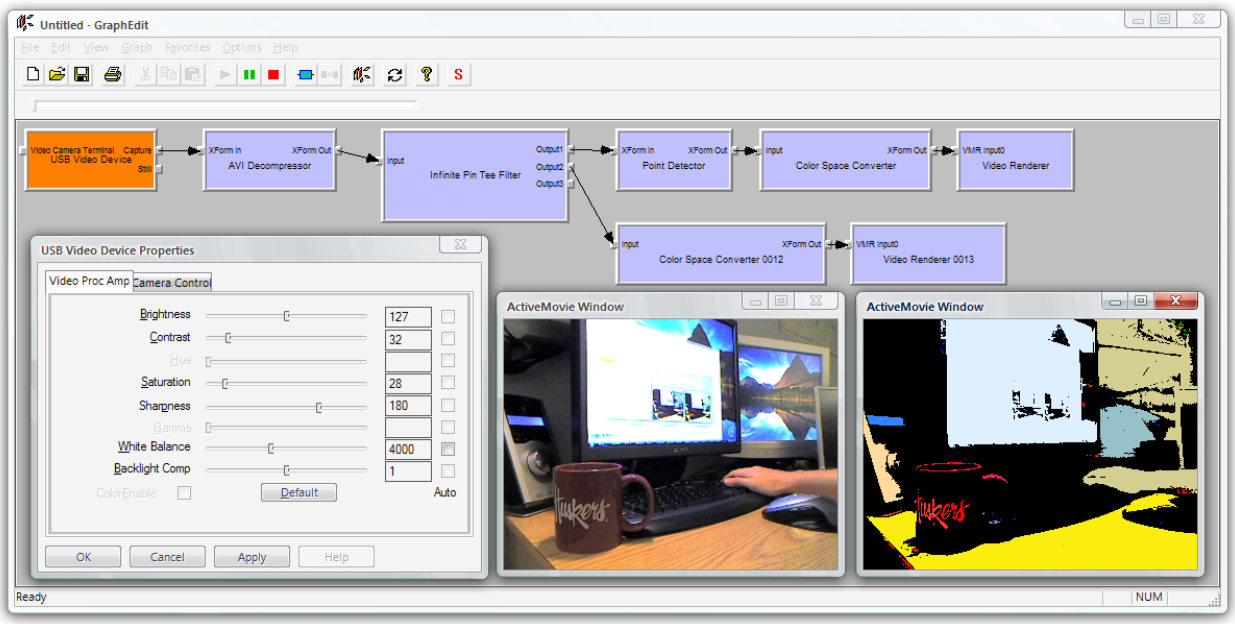


Figure 6.1: GraphEdit application.

The image processor component is implemented as described in Section 3.1. It is tested by using a program called GraphEdit which is shown in Figure 6.1.

Point detection unit is implemented as a DirectShow filter. Microsoft DirectShow is used for this project because it provides support for a wide range of camera devices. DirectShow's

component-based graph architecture allows us to extract any media streaming data just by writing a software component called a "filter".

It is important to note that the algorithm itself is portable between the platforms: only the implementation is specialized to one platform to maximize the performance of this highly demanding process.

Since our program, which is a user filter, is not ready at the point of demonstration, only processed images are shown on the screen. However, to demonstrate that the filter has enough information, we will show a processed image with

- detected island filled by average color,
- points indicating the islands' center of gravity.

6.1.2 Results

The image processor worked correctly. Using a laptop computer and a webcam with 320x240 resolution at 30 fps, the unit was able to process the image without unacceptable delay or dropped frames.

6.2 Image Processor: Coordinate Transformer

In this miniproject, Coordinate Transformer is implemented, and tested.

6.2.1 Procedure

This unit determines the helicopter position from three 2-D coordinates. The conversion is mathematically done using Newtons method. Therefore, this component should be platform independent. To make it platform independent, Coordinate Transformer is implemented as a C++ class.

C++ class itself cannot be tested without testers. Therefore, tester program is also created for this miniproject.

Using the tester program, the coordinate transformer is demonstrated. The program receives 3 points in 2-D coordinates, and determines the helicopter position from the location of the points.

6.2.2 Results

The system was tested with several test inputs. The results were verified by comparing to the calculations from Maple worksheet.

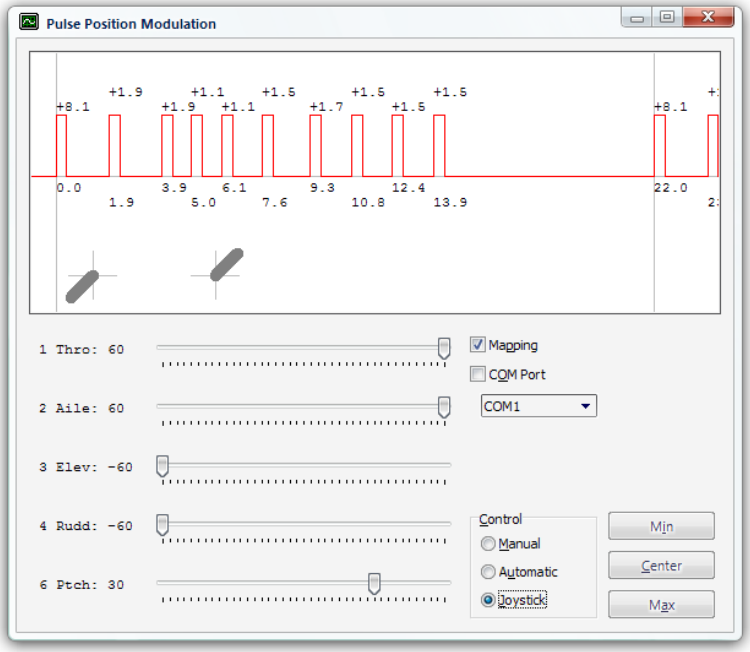


Figure 6.2: PPM signal generator.

6.3 Joystick-Simulator Connectivity Test

6.3.1 Procedure

The programmed data translator was tested using a special set-up as shown in Figure **??**. The system has inputs from the USB joystick controller and the input is processed in the PC program shown in Figure 6.2 and the data is sent to the microcontroller via RS-232C. Finally the microcontroller outputs PPM signal. The output was evaluated on the computer flight simulating software. The connectivity is verified by flying a helicopter in the simulator at different conditions; hovering, forward flight, and landing.

6.3.2 Results

The test system worked completely. There was no noise or time delay noticeable. This mini-project is finished.

6.4 Measuring and Displaying Pitch and Roll

In this mini project, the pitch and roll of the system is calculated and displayed in degrees on a PC screen.

6.4.1 Procedure

The Motorola MC68HC12 microcontroller will be used to receive data from the accelerometer and translate it to usable information that a PC can use to make control decisions. The triple-axis accelerometer has three analog outputs that swing from 1.13V to 2.17V. However, in order to fully utilize the analog to digital converter (ADC) in the microcontroller which accepts input from 0V to 5V, the output signal from the accelerometer is amplified and scaled down using a voltage scaler to get a 0V to 5V swing. This process is known as signal processing.

After conditioning the output signal from the accelerometer, it is then sent to the microcontroller. The programmed microcontroller will convert the analog input signal to digital signal and then transfer the useful data to the PC to perform the following calculations to determine the pitch and roll of the system:

$$\text{Roll} = \sin^{-1}\left(\frac{V_{\text{out},y} - 1.65\,\text{V}}{\text{Selected Sensitivity}}\right) \tag{6.1}$$

$$\text{Pitch} = \sin^{-1}\left(\frac{V_{\text{out},x} - 1.65\,\text{V}}{\text{Selected Sensitivity}}\right). \tag{6.2}$$

6.4.2 Results

The microcontroller successfully converted the conditioned analog signal from the accelerometer into a useful digital signal for the PC to calculate the pitch and roll. The calculated results were then displayed in the PC and the results were verified.

Chapter 7

Design Issues

7.1 Image Processor

7.1.1 Frame Rate

The image processor requires a large amount of computing power to process the images. Consequently, the CPU usage is always close to 100%. In the first phase of the development, we had experienced delay because of the high CPU usage. This was fixed by dropping frames as needed.

7.2 Data Translator

7.2.1 Obtaining the Specification of PPM

The documentation on PPM was not distributed from the manufacturer of the transmitter. To determine the specification of the PPM used for the transmitter, a set of measurements were taken. According to the measurements, working principles and specifications of the PPM were revealed.

7.2.2 Finding How SCI and RS-232C Works

Serial Communication Interface is an interface on the microcontroller that enables easy connection with the PC. Using SCI, RS-232C port of a PC can be used to send and receive data from the microcontroller. SCI makes it easy for the programmer to implement serial communication because it takes care of signal generation and detection automatically. To accomplish data communication, SCI functionality of the microcontroller is used. The SCI makes it possible to work with logic level in stead of hardware level, and makes the implementation of serial communication easy. Using this interface, we can simply call a function which returns a character (8 bit number) to capture one byte of data or call a function with a character to send one byte of data. By repeatedly using these functions, the data communication can be achieved with the master.

7.2.3 Developing Timer Function

A test program was written to test the timer functionality and interrupt driven programming. The test program was to generate a square wave slowly so that the output can be observed with LED. Once the program was loaded and running correctly development of the timer function was advanced to the actual PPM generation. The only difference is the duty cycle of the square wave is varied. The program was loaded and tested with an oscilloscope. The frequency and time period of the signal was measured and proved to be precise. The timer functionality was successfully developed.

7.2.4 Developing Serial Communication

We employed the integral approach in the development. The development was started by writing a test program that sends ASCII characters to the hyper terminal of the PC. This establishes that physical connection and the basic set-up of the SCI is correct. Following this, a test program is designed which echoes what was sent from the PC back to the hyper terminal. Completing this makes sure the program is capable of receiving, storing and sending information. Finally, the program was modified to receive and transmit any sequence of data.

7.3 Inertial Navigation System (INS)

7.3.1 Circuit Design

The circuit was redesigned to include a Bluetooth wireless module in order to eliminate the wired connection between the helicopter and the data translator. With this addition to the circuit, the PIC16F690 became inadequate due to its limited number of pins. To remedy this situation, the choice of microcontroller was changed to the PIC18F2520, which has 8 pins more than the PIC16F690. In addition to the extra pins, the PIC18F2520 provided a faster processing speed.

7.3.2 PCB Design

It is important to have a circuit design finalized before beginning the PCB design process so that the time spent is minimized. The circuit design was changing on nearly a daily basis. With every new change, the PCB has to be redesigned. As a result, much more time was spent designing the first PCB than was necessary. In order to minimize the area and weight of the PCB, DIP chips are switched to surface mount chips.

7.3.3 Testing and Results

All sensor output was too noisy to use due to helicopter vibration. IIR and digital filters were designed to remove the noise created by the vibration. Various capacitive filters were also added to data lines with. However the filtered data from the sensors were not as accurate

as the data from the image processor. Therefore, the INS was not used to supplement the image processor.

Chapter 8

Engineering Standards

8.1 Cost of Design

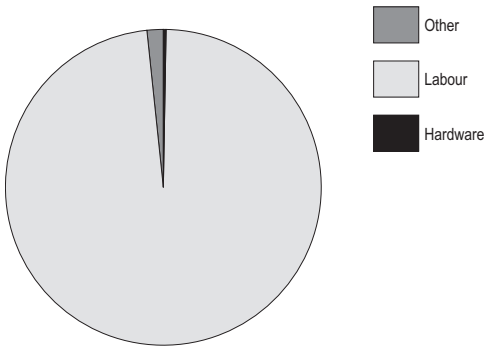


Figure 8.1: Cost of design.

Refer to Appendix A.1 for the details of the design costs. The analysis is based on the following assumptions:

- The cost of design is a fixed cost
- The design period is one year

It's seen in Figure 8.1 that the cost to design this system is almost entirely due to the engineer's salary. In reality, the design period can be much shorter when using advanced design software. An experience staff of engineers would certainly help. To return the teams standard of 20%, the prototype system would have to sell for $386,401.64.

8.2 Annual Cost of Production

Refer to Appendix A.2 for details of production costs. The analysis is based on the following assumption:

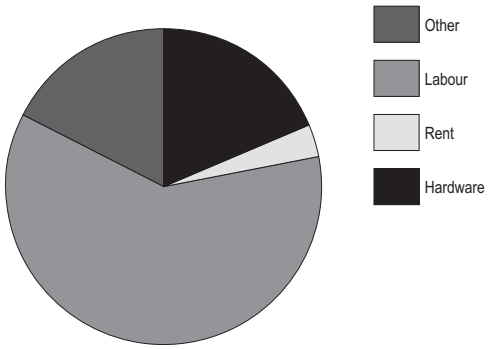


Figure 8.2: Cost of production.

- 1000 units are manufactured per year.
- Salary information is based on current average starting salary.
- All programming can be implemented on a single processor.

As seen in Figure 8.2, the cost of labor is more than half of the annual expenses. The reason for this expense is due to the unreasonable assumption that all five members of our design team will stay on for the duration. It is more likely that only one or two will move on to form the company, with the others receiving royalties.

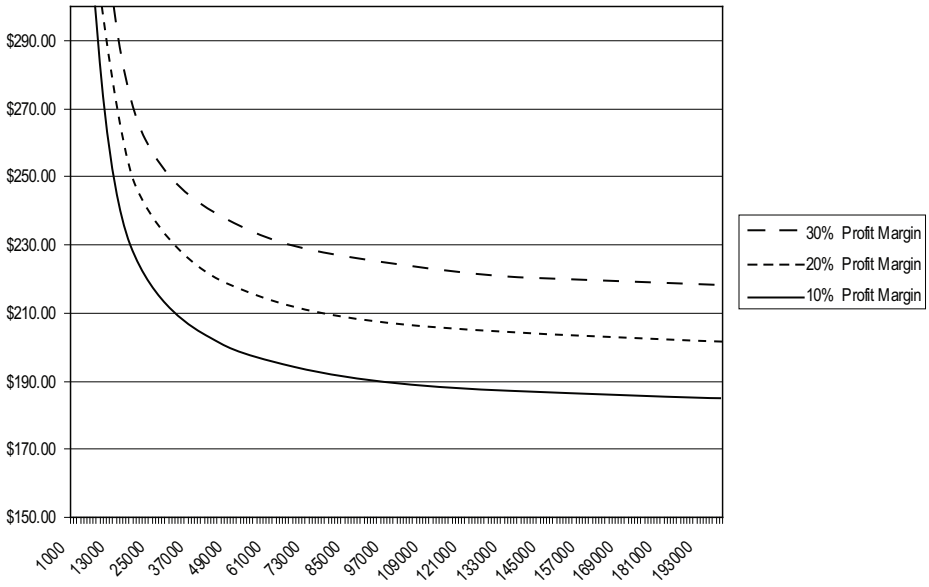


Figure 8.3: Per unit cost.

Figure 8.3 shows that the price per unit is not significantly reduced when the production reaches around 50,000. By choosing a 20% profit margin, which is typical in electronic

industry, and a production of 5000 per year, the product can be sold for $409.64. A more realistic production number, determined from a Japanese producer, is about only 100 units per year. At this rate of production, the cost per unit will rise to $10,857.23. If production is increased by just 50 units, the cost per unit will decrease to $7303.63. Increasing production significantly benefits the customer while keeping the profit margin steady.

The annual cost of production could be reduced if the responsibilities of the accountant and lawyer were placed on the already employed engineers.

8.3 Ethical Considerations

Autonomous navigation systems are widely used in military applications, where the intent is to bring harm to others. This system was designed in such a manner that it could not be used in a harmful manner. The design aspects that provide this protection include the requirement that the vehicle to be controlled must be in view and within range of the two video cameras as well as be tethered to the control system by the communication cables. Without long range capabilities, the probability that this system is used to harm others is very low.

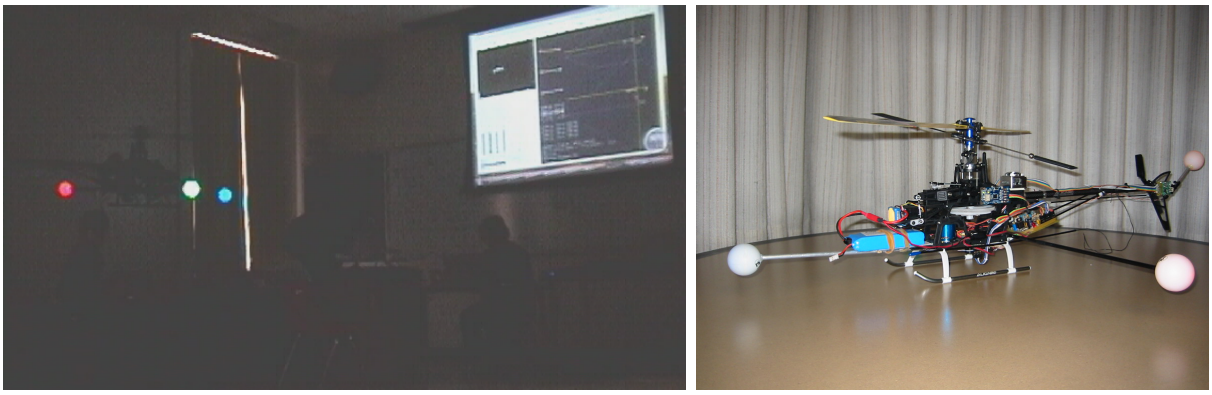
8.4 Safety Issues

The main rotor of the helicopter in this project presents a serious safety issue. In the unlikely event that the helicopter were to fly out of control, the spinning rotor could cause serious injury or damage to whomever or whatever was within it's path. In an attempt to prevent this mishap from occurring, the system is designed such that a radio control transmitter can be switched on instantaneously to manually override the automatic control system.

Chapter 9

Results

The computer pilot system satisfied all of the requirements. Furthermore, additional capabilities such as way-point flight are accomplished.



(a) Successful way-point flight.

(b) Helicopter with sensors and beacons.

Figure 9.1: Photographs taken in the testing phase of the project.

Appendix A

Cost Analysis

A.1 Cost of Design

			Cost	Subtotal	Total
Hardware	Microcontroller		$100.00	$1,370.00	$318,410.00
	INS		$150.00		
	Camera		$120.00		
	Miscellaneous		$1,000.00		
Rent			$0.00	$0.00	
Labor	Wages	5 Engineers	$275,000.00	$317,000.00	
		1 Maintenance	$0.00		
		10 Assembly	$0.00		
		1 Secretary	$0.00		
	Insurance		$42,000.00		
Other	Test Equipment		$5,000	$5,000	
	Phone/Internet		$0.00		
	Office Supplies		$0.00		
	Accountant/Lawyer		$0.00		

A.2 Cost of Production

			Cost	Subtotal	Total
Hardware	Microcontroller		$11,090.00	$203,690.00	$1,092,090.00
	INS		$12,600.00		
	Camera		$80,000.00		
	Miscellaneous		$100,000.00		
Rent			$36,000.00	$36,000.00	
Labor	Wages	5 Engineers	$275,000.00	$663,000.00	
		1 Maintenance	$55,000.00		
		10 Assembly	$204,000.00		
		1 Secretary	$20,000.00		
	Insurance		$109,000.00		
Other	Utilities		$55,000.00	$189,400.00	
	Phone/Internet		$48,000.00		
	Office Supplies		$2,400.00		
	Accountant/Lawyer		$84,000.00		

(Quantity = 1000)