AUTOMATIC TARGETING SYSTEM FOR UNMANNED AIR VEHICLE OPERATIONS

Submitted by
LOW CHIN MIN MIKE
U024997B

Department of Mechanical Engineering

In partial fulfillment of the requirements for the Degree of Bachelor of Engineering
National University of Singapore

Session 2005/2006
Unmanned Air Vehicles (UAV) have been widely used in the battlefield. They are usually employed for surveillance and intelligence gathering. An Automatic Targeting System (ATS) would greatly enhance its capability for such reconnaissance. Therefore this project aims to design and integrate such a system onto UAVs. The ATS includes the camera system, the gun system and the wireless communication system. The camera system provides the ‘eyes’ for the UAV, the gun system aims and ‘shoots’ the target, and the wireless communication system reports the actual GPS position of the target. All these systems required are carefully selected to cater for the UAVs flying performance and each category of system is tested on ground before integrating it onto the UAV. Eventually, the whole ATS is tested in flight to ensure the full function and operability in the practical environment.
ACKNOWLEDGEMENT

The author would like to extend sincere gratitude to his project supervisor, Associate Professor Gerard Leng Siew Bing for his guidance and support throughout the entire duration of this project.

The author would also like to thank the Masters students, Oi Tze Liang, Teoh Wei Lit and Tan Han Yong, for their help in conducting flight tests and making this project a success.

The author would also like to extend his gratitude to the staff of Dynamics Laboratory for their assistance for the duration of the project.

The author would also like to thank his wife, Christine Lim, and daughter, Anne Low, for their understanding and support for this project.
**TABLE OF CONTENTS**

ABSTRACT .......................................................................................................................... i

ACKNOWLEDGEMENT ........................................................................................................ ii

TABLE OF CONTENTS ..................................................................................................... iii

LIST OF FIGURES ........................................................................................................ viii

LIST OF TABLES ............................................................................................................... x

**CHAPTER 1 INTRODUCTION**

1.1 Thesis Background ........................................................................................................ 1

1.2 Objectives .................................................................................................................... 1

1.3 Progressive Goals ...................................................................................................... 2

1.4 Structure of Dissertation .......................................................................................... 2

**CHAPTER 2 TARGET IDENTIFICATION SYSTEM**

2.1 Overview of Target Identification System .................................................................. 3

2.2 Camera System .......................................................................................................... 4

2.2.1 Selection of Camera ............................................................................................. 4

2.2.2 CMUCam2+ ......................................................................................................... 5

2.2.3 Field of View Experiment ................................................................................... 7

2.2.4 Modeling of Camera Constraints ........................................................................ 8

2.3 Communications ....................................................................................................... 10

2.3.1 RS232 Serial Communication ............................................................................. 11

2.4 In-Flight Computer (PC104) .................................................................................... 11

2.5 Program Functions Development ............................................................................ 12
Table of Contents

2.5.1 Initialization of Serial Transmit and Receive............................12
2.5.2 Initialization of CMUCam2+ ....................................................13
2.5.3 Target Searching .......................................................................13
2.5.4 Calculation of Target Coordinates ............................................14
2.5.5 Reporting...................................................................................15
2.6 Ground Test of Target Identification System ............................................15

CHAPTER 3 ARCHITECTURE OF AUTOMATIC TARGETING SYSTEM
3.1 Architecture of ATS...................................................................................17
3.2 Gun System................................................................................................18
  3.2.1 Infra-Red Transmitter................................................................18
  3.2.2 Servo Motor ..............................................................................19
  3.2.3 Servo Motor Range Experiment................................................19
  3.2.4 Servo Motor Angular Change per Increment Experiment ........21
  3.2.5 Resolution of Gun System ........................................................21
3.3 Communications ........................................................................................23
  3.3.1 Wireless Communication..........................................................23
3.4 Program Functions Development ..............................................................24
  3.4.1 Calculation of Target Centroid with respect to Gun System ....24
  3.4.2 Servo Motor Control .................................................................25
  3.4.3 Infra-Red Transmitter Control ..................................................25
  3.4.4 Wireless Transmission ..............................................................26
  3.4.5 Ground Control Base Station Reporting ...................................26
3.5 Ground Test of Automatic Targeting System............................................26
## CHAPTER 4 INTEGRATION OF AUTOMATIC TARGETING SYSTEM WITH UNMANNED AIR VEHICLE

1. Algorithm of Integrated System with UAV .................................................. 28
2. Micropilot (MP2028g) ................................................................................. 29
3. Communications .......................................................................................... 30
   
   3.1 RS232 Serial Communication .................................................................. 30
4. Interconnection of Subsystems Communications ......................................... 31
5. Integration of Automatic Targeting System ............................................... 32
6. Program Functions Development ............................................................... 33
   
   6.1 Initialization of Serial Transmit and Receive ............................................ 33
   6.2 Initialization of Micropilot ....................................................................... 34
   6.3 Height and Heading Update from Micropilot .......................................... 34
   6.4 GPS Update from Micropilot ................................................................... 34
   6.5 GPS Reporting ....................................................................................... 35
7. Flight Test of Automatic Targeting System ............................................... 36

## CHAPTER 5 DESIGN OF PROGRAMMING ARCHITECTURE

1. Overview of Algorithm of Source Code .................................................... 38
2. Main Functions of Source Code ................................................................. 39
   
   2.1 Initialization ......................................................................................... 39
   2.2 Real-time Information Update ............................................................... 39
   2.3 Target Searching .................................................................................. 40
   2.4 Actuation .............................................................................................. 40
   2.5 Reporting ............................................................................................. 40
## Table of Contents

5.3 Algorithm Considerations .................................................................................. 41  
5.3.1 Redundancy in Height Information ......................................................... 41  
5.3.2 Height Estimation ..................................................................................... 41  
5.3.3 Target Identification Criteria .................................................................. 42  
5.3.4 Flexibility of Using Different Camera System ...................................... 42  
5.3.5 Flexibility of Using Different Servo Motors ......................................... 42  
5.3.6 Flexibility of Gun System Mounting ...................................................... 43  
5.3.7 Background Update of Flight Data .......................................................... 43  
5.3.8 Feedback on Wireless Transmission ...................................................... 44  

### CHAPTER 6 CONCLUSION  
6.1 Conclusion ........................................................................................................ 45  
6.2 Future Development .......................................................................................... 45  
   6.2.1 Target Lasing for Guided Weapons ..................................................... 45  
   6.2.2 Real-Time Pictures Download ............................................................. 46  
   6.2.3 Flexibility of Altering Target Identification Criteria ....................... 46  
   6.2.4 Tracking Multiple Targets ................................................................. 46  
   6.2.5 Tracking Airborne Targets ................................................................. 46  

### REFERENCES  
.................................................................................................................. 47  

### APPENDICES  
APPENDIX A: Performance Indicator For Vision Systems ............................ A-1  
APPENDIX B: CMUCam2+ ................................................................................ B-1
APPENDIX C: Experiment to find Field of View of Camera.............................C-1
APPENDIX D: PC104 Specifications .................................................................. D-1
APPENDIX E: Futaba 3001 Servo Motor Specifications ......................................E-1
APPENDIX F: Experiment to Find Maximum Servo Angle.................................F-1
APPENDIX G: WUSB54G Specifications............................................................... G-1
APPENDIX H: MP2028g Specifications................................................................. H-1
APPENDIX I: Source Codes For ATS .................................................................I-1
LIST OF FIGURES

Figure 1: Overview of Target Identification System .................................................3
Figure 2: CMUCam2+ ..............................................................................................5
Figure 3: RGB Values from CMUCam2 GUI ..........................................................6
Figure 4: Tracking of Red Ball ..............................................................................6
Figure 5: Graph of Horizontal Field of View of CMUCam2+ ..................................7
Figure 6: Graph of Vertical Field of View of CMUCam2+ ......................................7
Figure 7: Scenario where the target is captured only in 2 frames .........................8
Figure 8: Diagram of Successful Target Acquisition .............................................9
Figure 9: Graph of Maximum Height wrt Target Length .......................................10
Figure 10: Graph of Maximum Speed wrt Height ..................................................10
Figure 11: Serial Interface Connector ....................................................................10
Figure 12: PC/104 .................................................................................................12
Figure 13: Pixel Representation of CMUCam2+ ......................................................14
Figure 14: Diagram for Ground Distance ..............................................................14
Figure 15: Setup for Ground Test of Target Identification System .......................16
Figure 16: Architecture of ATS ............................................................................17
Figure 17: Gun System .........................................................................................18
Figure 18: IR Field of Transmission .....................................................................19
Figure 19: Setup for Experiment on Servo Motor ..................................................20
Figure 20: Picture of markings noted for 2m ..........................................................22
Figure 21: Graph of Experiment Variance ..............................................................22
Figure 22: Linksys Wireless-G USB Network Adaptor WUSB54G .......................23
Figure 23: Grid System of the Camera and the Gun ..............................................24
Figure 24: Setup for Ground Test of ATS ..........................................................27
Figure 25: Tracking of Red Target .....................................................................27
Figure 26: Algorithm of Integrated System with UAV ......................................28
Figure 27: Micropilot MP2028g .......................................................................29
Figure 28: Interconnection of Systems ...............................................................31
Figure 29: Systems Integration ..........................................................................32
Figure 30: JR Helicopter with Integrated Systems ...........................................33
Figure 31: Contents of Box ................................................................................33
Figure 32: Zoomed in view of Gun System .......................................................33
Figure 33: Camera and GPS Coordinate System ..............................................35
Figure 34: Target LED Array .............................................................................36
Figure 35: Flight Test Setup ...............................................................................36
Figure 36: Search for Blue Target ......................................................................37
Figure 37: Search for Red Target ......................................................................37
Figure 38: Overview of Source Code ...............................................................38
Figure 39: Different Views and Dimensions of CMUCam2+ .........................B-1
Figure 40: Graphs of Camera Field of View .....................................................C-1
Figure 41: Sample Snapshot of CMUCam2+ ....................................................C-2
Figure 42: Futaba S3001 Specifications .............................................................E-1
Figure 43: Futaba S3001 Dimensions ...............................................................E-1
Figure 44: Graph of Maximum Servo Angle ....................................................F-1
Figure 45: WUSB54G .....................................................................................G-1
LIST OF TABLES

Table 1 : Specifications of the COTS Vision Systems ..........................................4
Table 2 : Port Settings for CMUCam2+ .................................................................11
Table 3 : Port Settings for MP2028g .................................................................30
Table 4 : Specifications of WUSB54G4G ...........................................................G-1
CHAPTER 1 INTRODUCTION

1.1 Thesis Background

This project is an industrial collaborative project with DSO National Laboratories. The objective of the DSO National Laboratories is to eventually have multiple Unmanned Air Vehicles (UAV henceforth) conducting a planned search pattern in a given search area to find known target. When the target is located, the UAV will mark the target location and relay the information back to the Ground Control Station (GCS henceforth).

This project focuses only on a single UAV and specifically the Automatic Targeting System (ATS henceforth). It simulates a single UAV flying a given search pattern to find a known target. The ATS onboard will search and detect the target against background clutter. Once the target is located, an onboard gun is rotated to point at the target and shoots. At the same time, the position of the target is relayed back to the GCS.

1.2 Objectives

This project aims to design, integrate and test an ATS for UAV operations. The design will involve the development of algorithms for a camera system to detect a target signature against background clutter and relaying the target information back to the GCS. The system will then be integrated on existing platforms and thereafter tested under different field conditions. In order to accomplish this project, three progressive milestones have been set.
1.3 Progressive Goals

For the first milestone, the aim is to have the camera system track a stationary target and thereafter transfer data regarding the target to the flight computer which must then display the coordinates in an external display. The purpose is to be able to establish communications between the camera and the flight computer.

The second milestone’s objective is to be able to track the stationary target and also send the required actuator commands to correct the position of an aerial robotic platform. The target coordinates are then transmitted to the GCS.

The final milestone is to integrate the whole system and flight test the aerial robotic platform in open field condition. The aim is to succeed in identifying target placed on the ground, and also to transmit the target coordinates to GCS.

1.4 Structure of Dissertation

This thesis comprises of 6 chapters. Chapter 1 introduces and defines the objectives of this project. Chapter 2 shows the target identification system. Chapter 3 describes the architecture of the automatic targeting system. Chapter 4 highlights the integration of the ATS into the UAV and the flight tests that follow. Chapter 5 gives an overview of the design of the programming architecture. Chapter 6 concludes this thesis.
CHAPTER 2 TARGET IDENTIFICATION SYSTEM

2.1 Overview of Target Identification System

![Diagram of Target Identification System]

Figure 1: Overview of Target Identification System

Figure 1 shows the overview of the target identification system. Serial communication will firstly be initialized to enable communication between the computer and the camera. Thereafter, the camera system will be initialized. It will be ready to accept target identification criteria from the computer and start its search for the target. If the target is not in sight, the system will continue its search algorithm. Once the target is found, the target information will be gathered and processed to calculate the centroid of the target. The centroid is then used to calculate and find the coordinates of the target with respect to the camera. This information is subsequently displayed on the monitor. Thereafter, the system continues to search for the target and the whole search process is repeated.
Chapter 2 Target Identification System

2.2 Camera System

One of the main components of the target identification system is the camera. It is the ‘eyes’ of the entire system. Therefore much emphasis is placed in the selection of the camera system.

2.2.1 Selection of Camera

The ATS is implemented using the vision sensors. It is selected with the consideration that it is able to work for a large range of flight altitude as well as being able to accommodate complex target identities. With that in mind, the low cost Commercial Off-The-Shelf (COTS) vision systems with high processing speed and light weight are sought after. Table 1 shows the specification of the 2 COTS vision system that are found to satisfy the requirements.

Table 1: Specifications of the COTS Vision Systems

<table>
<thead>
<tr>
<th>S/N</th>
<th>Model</th>
<th>Max Resolution</th>
<th>Frame Rate (fps)</th>
<th>Weight (g)</th>
<th>Cost (S$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>CMUCam2+</td>
<td>176 x 255 pixels</td>
<td>26 (@88 x 143 pixels)</td>
<td>40</td>
<td>279</td>
</tr>
<tr>
<td>2.</td>
<td>EyeCam Camera C2 with EyeCon EyeBot-Controller M4</td>
<td>640 x 480 pixels</td>
<td>10 – 15 (@160 x 120 pixels)</td>
<td>216</td>
<td>1744</td>
</tr>
</tbody>
</table>

A performance indicator (P.I.) with the parameters shown in Equation 1 is used to assess the decision for the vision systems. The vision system with the highest P.I. is the most suitable vision systems for the ATS.

Equation 1: Performance Indicator for Vision Systems

\[ P.I. = \frac{\text{Maximum Resolution}_{\text{pixels}} \times \text{Frame Rate}_{\text{fps}}}{\text{Weight}_{\text{g}} \times \text{Cost}_{\text{SS}}} \]
The simple analysis in Appendix A shows that the \((P.I.)_{CMUCam2+}\) is greater than the \((P.I.)_{EyeCam}\) and therefore the CMUCam2+ is chosen as the vision system for the ATS.

### 2.2.2 CMUCam2+

![CMUCam2+](image)

The CMUCam2+ (Figure 2) is a camera system produced by Carnegie Mellon University. It requires a power supply which produces anywhere from 6 to 15 volts of DC power and capable of supplying at least 200mA of current. This is provided by a battery supply comprising of 8 AA sized rechargeable batteries. More details on the camera are listed in Appendix B.

For the ATS, the CMUCam2+ is set to high resolution and can capture images at a resolution of 176 by 255 pixels at a frame rate of approximately 10 frames per second. It uses a silicon chip that contains a grid of boxes, each of which is sensitive to different colours of light. After passing through the lens, light generate different voltage proportional to the amount of light and are converted into numerical value (in the range of 16 to 240) for each of the three channels namely, the red, green and blue (RGB) channel. Therefore in order to
track a target, the known RGB values have to be known. Figure 3 shows how the RGB values are obtained using the CMUCam2+ Graphics User Interface. Since light is not perfectly uniform and thereby affecting the colour of the target, the tolerances need to be set for each of the three channels to accommodate for these variations. However, relaxing these boundaries too much will accept many unwanted colours.

![Figure 3: RGB Values from CMUCam2 GUI](image)

Given the RGB values and the tolerances of the target, the camera processes each new image frame from the camera and examines each pixel row by row. If the pixel is within the range specified, it will be marked as tracked. Noise filtering is also used to reduce fluctuations by grouping sequential tracked pixels. With all the tracked pixels, it will create a bounding box (Figure 4) with the top most, bottom most, left most and right most values stored. These values will then be transmitted to the computer through the serial communication.

![Figure 4: Tracking of Red Ball](image)
2.2.3 Field of View Experiment

The field of view of the camera is vital for calculations in the target identification system. It has to be accurate and therefore the field of view of the camera for both the horizontal and vertical axis has to be obtained experimentally. The experimental setup and results are shown in Appendix C. From the experiment, 2 graphs are obtained for the horizontal and vertical field of view. The horizontal field of view (Figure 5) obtained is 44.91° and the vertical field of view (Figure 6) is 29.76°.

![Horizontal Field of View Graph](image1)

**Figure 5 : Graph of Horizontal Field of View of CMUCam2+**

![Vertical Field of View Graph](image2)

**Figure 6 : Graph of Vertical Field of View of CMUCam2+**
2.2.4 Modeling of Camera Constraints

The camera system, like any other camera, has some constrains in terms of the height, speed and target size. The worst case scenario in Figure 7 is used to determine the maximum flight speed of the camera system for target acquisition.

![Figure 7: Scenario where the target is captured only in 2 frames](image)

It assumes that the 2 successive field of views does not overlap nor does it leave any gap in the search area. Thus the maximum flight speed, \( V_{\text{max}} \) is given by Equation 2.

**Equation 2: Maximum Velocity of Camera wrt Height**

\[
V_{\text{max}} = \frac{2 \times \text{height} \times \tan \left( \frac{\text{FOV}}{2} \right)}{\text{t}_{\text{refresh}}}
\]

where
- \( V_{\text{max}} \) = maximum velocity
- height = height of camera
- FOV = Field of View of camera in degrees
- \( \text{t}_{\text{refresh}} \) = time for each refresh
In order for successful target acquisition, it is assumed that each of the successive field of views capture only half of the target (Figure 8). And the number of pixels captured must be at least more than the minimum required for machine vision to acquire. Similar ratio is used to derive the inequality where the proportion of the target pixel over the ground pixel is equal to the proportion of the target length over the actual ground distance. Therefore for successful acquisition of the target, the height is governed by the inequality in Equation 3.

**Equation 3 : Minimum Height wrt Target Size**

\[
\text{height} \leq \left( \frac{\text{grdpix}}{\text{tgtpix}} \times \frac{\text{Length}}{2} \times \frac{1}{2 \times \tan \left( \frac{\text{FOV}}{2} \right)} \right)
\]

where
- height = height of camera
- grdpix = camera coverage in pixels
- tgtpix = target length in pixel
- Length = actual length of target
- FOV = Field of View of camera in degrees

The refresh time is limited by the micropilot MP2028\textsuperscript{g} with refresh rate of 5Hz (10Hz for CMUCam2+). Therefore \( t_{\text{refresh}} \) is 0.2 seconds. The ground pixel is limited by the lower resolution of the horizontal axis as compared to the vertical axis and thus grdpix is 176 pixels. And the corresponding field of view (FOV) is taken to be 44.73°. Minimum target pixel required for machine vision to acquire is
assumed to be at least 5 pixels. Therefore $tgtpix$ is 5 pixels. With all these parameters, the graphs of camera height against minimum target size as well as the maximum velocity is plotted and shown in (Figure 9 & Figure 10). The target size will determine the maximum height required and the corresponding maximum speed that the system can fly in order to have a successful acquisition of the target.

![Graph of Height against Target Length](image1)

![Graph of Height against Vmax](image2)

**Figure 9 : Graph of Maximum Height wrt Target Length**

**Figure 10 : Graph of Maximum Speed wrt Height**

## 2.3 Communications

The camera system needs to establish communications with the computer. The CMUCam2+ uses the transistor-transistor logic (TTL) as its means of data output. Therefore a Serial Interface Connector (Figure 11) is used to convert TTL to RS232 serial port.

![Serial Interface Connector](image3)
2.3.1 RS232 Serial Communication

The serial port (RS232) provides the means of communication between the camera and the flight computer. Data is transmitted in binary (1’s and 0’s) and it is therefore crucial to establish handshaking, both software and hardware, between the two nodes, namely the camera and the computer. It is also important to synchronize the rate of data transfer (baud rate), each byte size and parity. This allows the data transmitted to be coded back to readable data for processing. The Windows operating system, unlike the DOS, does not allow direct access to the serial port. It uses the WIN32 APIs (Application Programming Interfaces) to handle the serial communications via the I/O ports. Programming is done in Visual C++ to establish communications through the serial port. The settings for the serial communication with the camera are listed in Table 2.

<table>
<thead>
<tr>
<th>PORT SETTINGS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate (Bits per second)</td>
<td>115200</td>
</tr>
<tr>
<td>Data Bits</td>
<td>8</td>
</tr>
<tr>
<td>Parity</td>
<td>None</td>
</tr>
<tr>
<td>Stop Bits</td>
<td>1</td>
</tr>
<tr>
<td>Flow Control</td>
<td>None</td>
</tr>
</tbody>
</table>

2.4 In-Flight Computer (PC104)

The main processing brain of the ATS is the in-flight computer, PC104 (Figure 12). It is actually a very small computer system of the size of the palm. PC104 is essentially a personal computer (PC) with a different form factor and most of the program development tools used for PC can be used for a PC104 system. It is powered by lithium polymer batteries supplying 16.8V and 1500mAh. It is chosen mainly due to its size and weight which is critical for the
UAV platform to carry. It is also chosen due to the computing power that is needed to run the programs. Appendix D lists the specifications of the PC104.

![Image of PC/104](image)

**Figure 12 : PC/104**

The PC104 executes the commands programmed in Visual C++. It initializes the serial communication and camera system. It then sets the system to search for the target. Once the target is detected, the data is transferred to the PC104. With the data, it works out the centroid of the target by using the average of the vertical and horizontal values of the tracked box to estimate the centroid.

### 2.5 Program Functions Development

The whole target identification system comprises of many processes and is controlled mainly by the program written. It has to initialize the serial communication as well as the camera system. Thereafter it has to code the search algorithm and do some calculations before reporting the target coordinates. As such, several functions are written to address each process.

#### 2.5.1 Initialization of Serial Transmit and Receive

This function establish handshaking, both software and hardware, between the camera and computer. It synchronizes the rate of data transfer (baud rate), the
byte size of each transmission as well as the parity. This enables data to be sent
and read correctly.

The data received from the camera system are in ASCII strings and
comprises of many different information. This has to be broken up into each
individual data sections and sorted out before converting the required data to
numbers for useful calculation.

Likewise, the data sent to the camera system are in ASCII strings also.
Commands to the camera using the calculated values have to be converted to
strings and packaged into the format understandable by the camera.

2.5.2 Initialization of CMUCam2+

The CMUCam2+ needed to be initialized to set the required configuration
for target identification. This code is written to set the camera register to auto
exposure and auto gain off. It also sets the camera to high resolution mode.

2.5.3 Target Searching

The target search criteria in terms of RGB values are processed with the
given tolerance. All the minimum and maximum values of each individual RGB
value are calculated and registered onto the camera enabling the camera to search
for that target parameter.
2.5.4 Calculation of Target Coordinates

With the calculated centroid of the target, the system needs to calculate the actual ground coordinates of the target. The centroid is calculated in terms of pixels and therefore there is a need to convert it to actual ground distance based on the height of camera from the ground.

Figure 13 shows the pixel representation of the CMUCam2+. The target centroid \((x)\) is calculated and the position is related in terms of pixels. In order to calculate the actual ground distance from the center, Figure 14 is used to formulate the equations for each horizontal and vertical axis. Equation 4 first calculates the actual ground distance of the centroid from the edge of the camera coverage. This value is then used to calculate the actual distance from the camera’s centre using Equation 5.

**Equation 4 : Actual Ground Distance from Edge**

\[
\text{edge} \_ \text{dist} = 2 \times \text{height} \times \tan \left( \frac{\text{FOV}}{2} \right) \times \frac{\text{Centroidpix}}{\text{grdpix}}
\]
Chapter 2 Target Identification System

Equation 5 : Actual Ground Distance from Center

\[
\text{centre\_dist} = \left[ \text{height} \times \tan \left( \frac{\text{FOV}}{2} \right) \right] - \text{edge\_dist}
\]

where
- height = height of camera
- grdpix = camera coverage in pixels
- edge\_pix = actual ground distance of point from edge of camera coverage
- centre\_dist = actual ground distance of point from centre of camera
- Centroidpix = position of centroid in pixel
- FOV = Field of View of camera in degrees

Computing for both the horizontal and vertical axis will result in the coordinates of the target centroid with respect to the camera’s centre.

It can be observed that the height information is critical. It is variable and depends on the flying altitude of the UAV. The height information is obtained by the micropilot discussed in Section 4.2.

2.5.5 Reporting

This function processes the coordinates of the target and displays it on the monitor. It gives the ground location of the target centroid with respect to the camera in terms of distance to the left/right and up/down.

2.6 Ground Test of Target Identification System

In order to test the target identification system, a ground test is conducted. A big grid chart with each small grid measuring 10cm by 10cm is used. A red target of size 10cm by 10cm is placed against the grid with white background. From the constraint of the camera addressed in Section 2.2.4, the maximum
distance of the camera is supposed to be 2.14m away for successful target acquisition. Therefore the camera, which is mounted on a tripod, is placed near the maximum range at 2m away from the centre of the chart and programmed to track the red target (Figure 15). As the target moves from point to point, the corresponding position readout is calculated and shown on the monitor. When a blue target is placed against the grid chart, there was no position readout and the system was in continuous search mode. This concluded that serial communication is correctly established between the computer and the camera system. It also displayed the ability of the target identification system to identify the correct target and also track it.

*Figure 15: Setup for Ground Test of Target Identification System*
CHAPTER 3 ARCHITECTURE OF AUTOMATIC TARGETING SYSTEM

3.1 Architecture of ATS

The architecture of the ATS is shown in Figure 16. The target identification system forms the foundation for ATS. Once the target is identified, it will calculate the target coordinates with respect to the camera. Since the gun system is offset away from the camera, the system will transpose the target coordinates to that of the gun system. With the coordinates of the target aligned to the gun system, it then actuates the gun to point and ‘shoot’ at the target. The location and data of the target will then be sent to the GCS. Thereafter, the search routine is repeated.
3.2 Gun System

With the tracking system established, the subsequent aim is to actuate and correct the position of an aerial robotic platform namely the gun system. In this case, an infra-red (IR) transmitter is chosen to simulate the onboard gun system (Figure 17). The IR transmitter is mounted onto 2 Futaba 3001 servo motors (See Appendix E for specifications) which are used to rotate the IR pointer in both the pan and tilt direction. The 2 servos are carefully and properly aligned and fabricated to attach together. Thereafter, the servo motors are connected to the camera and pulse can be sent through the camera to set the position of each servo motor. The actual ‘shooting’ is achieved by switching on and off the IR pointer via the digital I/O ports.

3.2.1 Infra-Red Transmitter

The IR transmitter is controlled by the digital I/O port of the PC104. Therefore the digital I/O port has to be initialized in order to send commands to activate or deactivate the IR transmitter. Whenever ‘shooting’ is required, signals will be sent to switch the IR on. Otherwise, the IR is maintained in the off state.
The IR transmitter is sheathed to reduce the scope of its transmission. An experiment is conducted to determine the field of transmission of the sheathed IR transmitter by placing the IR transmitter at various distances from the wall and moving the IR receiver along it. Measurements are taken when the IR receiver is activated. The result in Figure 18 shows that the field of transmission is 36.27°.

### 3.2.2 Servo Motor

Control of the servo motors is done through the servo ports on the CMUCam2+. The CMUCam2+ has 5 servo ports and is able to support up to 5 servo motors and therefore a separate servo controller is not required. The required settings of the servo motors are sent to the camera system and it sends out the correct pulse to the servo motors.

### 3.2.3 Servo Motor Range Experiment

Since the servo motors are controlled through the camera and the camera could only send pulse range of 400μs to 1820μs, an experiment is conducted to
calculate the maximum range of the operation of the servo motor. A laser pointer is mounted onto a servo motor and the whole mechanism is placed at various distances from the long chart which is against the wall (Figure 19). Maximum and minimum pulses are sent to the servo motor and a mark is noted onto the chart wherever the laser pointer points. The distances between the marks are measured and plotted onto a graph against the various displacement of the servo motor and the maximum angle is obtained (See Appendix F). The experiment concluded that each servo motor covers a range of 138.65° which was much more than the camera field of view. This enables coverage over every position captured by the camera.

![figure 19](image.png)

**Figure 19**: Setup for Experiment on Servo Motor

It is critical to set the servo setting to centre and align the pointer perpendicular to the wall before starting the experiment. The misalignment can be noted by the difference between the maximum servo pulse position and the minimum servo pulse position from the centre.
3.2.4 Servo Motor Angular Change per Increment Experiment

The camera controls the servo motor by sending the required pulses. However, the commands to control the servo motors define it to the active region of 46 to 210 with 128 being the centre position. Therefore the characteristic of the servo motor angular change per active region increment has to be determined experimentally.

The setup is the same as in the servo motor range experiment. Various regular active region setting increment are sent and marked. The angular change with respect to each increment is then calculated and the average is determined to be 0.84°/increment.

3.2.5 Resolution of Gun System

An experiment is conducted to determine the resolution of the gun system. The gun system with the laser pointer is set to point at the camera’s centre before pointing at the target which is placed at the corner of the field of view of the camera. The variance of the point from the target centroid is then noted. A sample of 50 readings each is taken for various distances of the gun system to the target. Figure 20 shows a sample of the markings taken for the setup where the target is 2m from the gun system.
The maximum variance from the centroid of the target for each setup is plotted onto a graph (Figure 21) and the result shows the variance from the target centroid to be approximately 0.4%.
3.3 Communications

An important part of the ATS is the communications with the GCS. Due to the distance between the UAV and the GCS, the best form of communication is by wireless means.

3.3.1 Wireless Communication

The wireless communications uses the Linksys Wireless-G USB Network Adaptor WUSB54G (Figure 22). It weighs 80g and measures 9.1cm by 2.3cm by 7.1cm. It is connected to the PC104 via the USB port. The adapter communicates over the 802.11g wireless standard, which is one of the newest wireless standards, to communicate with the GCS. 802.11g is an IEEE wireless networking standard that specifies a maximum data transfer rate of 54Mbps, an operating frequency of 2.4GHz, and backwards compatibility with 802.11b devices. Detailed specification is listed in Appendix G.
3.4 Program Functions Development

The ATS is built upon the foundation of the target identification system and has added more processes. The alignment of the gun system and the control of it have to be precisely calculated to ensure accuracy. Also, the new wireless communication with the GCS needs to be established. All these new processes are coded as follows.

3.4.1 Calculation of Target Centroid with respect to Gun System

The centroid of the target is calculated with respect to the camera. It needs to be transposed onto the grid system of the gun (Figure 23) in order to control the gun to point at the target correctly. This code deals with that transposition. It transposes the target coordinates with the measured offset of the gun from the camera.

![Figure 23 : Grid System of the Camera and the Gun](image)

Figure 23 : Grid System of the Camera and the Gun
3.4.2 Servo Motor Control

This function controls the exact positioning of the 2 servo motors. It makes use of the angular change per increment of the servo motors to determine the pulse required to send to each of the servo motors.

The angle of the target with respect to the camera is given by Equation 6. The pulse required can be obtained by dividing the angle with the angular change per increment (Equation 7) where the angular change per increment is given by the maximum servo angle divided by the maximum servo setting range.

**Equation 6 : Angle of Target from Camera**

\[
\text{angle} = \tan^{-1}\left(\frac{\text{pos} + \text{offset}}{\text{height}}\right)
\]

where
- height = height of camera
- angle = angle of target from camera in degrees
- pos = position of centroid wrt camera in cm
- offset = offset of gun system wrt camera in cm

**Equation 7 : Pulse Required for Servo Setting**

\[
\text{pulse} = \frac{\text{angle}}{\text{servo\_angle}} \times \text{servo\_range}
\]

where
- pulse = pulse required
- angle = angle of target from camera in degrees
- servo\_angle = maximum angle of servo motor
- servo\_range = maximum range of setting of servo motor

3.4.3 Infra-Red Transmitter Control

This code initializes the digital I/O port of the PC104 and sets the IR transmitter to the off state. When ‘shooting’ is required, the IR transmitter would be set to the on state. Otherwise, it is maintained in the off state.
3.4.4 Wireless Transmission

The target coordinates are transmitted to the GCS via wireless communications. A code is written to establish the gateway and IP address for communications. The Win32 APIs is used to handle an interface to the telnet portion of the TCP/IP package and uses the Client for Microsoft Network Internet Protocol TCP/IP settings to connect to the receiving station. When the target data is obtained and calculated, the target coordinates are sent through this gateway.

3.4.5 Ground Control Base Station Reporting

The function for the ground control base station uses the same logic as that used for wireless transmission. It establishes the gateway between the ATS and the GCS and waits continuously to receive information. Once information is received, the base station will display it on the monitor.

3.5 Ground Test of Automatic Targeting System

A final ground test for the whole system is carried out. The same large grid chart was used. The camera is placed on a table which is 2m away from the chart. A laser pointer is mounted onto the two servo motors and are placed 15cm offset from the camera (Figure 24). The ATS is programmed to track the red target. As the red target moves, the laser pointer moves correspondingly to point onto the target (Figure 25). This displayed the accurate actuation and tracking of the gun system onto the target. The coordinates of the target are also displayed on the monitor of the flight computer as well as the monitor of the GCS to show correct wireless transmission of data.
Chapter 3 Architecture of Automatic Targeting System

Figure 24: Setup for Ground Test of ATS

Figure 25: Tracking of Red Target
CHAPTER 4 INTEGRATION OF AUTOMATIC TARGETING SYSTEM WITH UNMANNED AIR VEHICLE

4.1 Algorithm of Integrated System with UAV

The algorithm of the integrated system with the UAV is as shown in Figure 26. After initializing the serial communications, the system concurrently initializes the camera system as well as the micropilot system. The camera system will perform its task of searching for the target while the micropilot system continually updates the UAV Global Positioning System (GPS) position, height

![Figure 26: Algorithm of Integrated System with UAV](image-url)
and heading. When the target is identified, the actual real-time height, heading and GPS position are obtained for calculations and to pinpoint the GPS position of the target. That information is then reported to the GCS before it continues searching.

4.2 Micropilot (MP2028\textsuperscript{g})

![Micropilot MP2028\textsuperscript{g}](image)

Figure 27: Micropilot MP2028\textsuperscript{g}

The control of the autonomous UAV is carried out by the micropilot MP2028\textsuperscript{g} (Figure 27). It is the world’s smallest UAV autopilot weighing 28g and measuring 4cm by 10cm. It has a power of 140mA at 6.5V and the supply voltage can range from 4.2V to 26V. The detailed specifications of the micropilot are listed in Appendix H.

The MP2028\textsuperscript{g} consists of an on-board Global Positioning System (GPS), gyro unit as well as an air data unit. From these units, the UAV flight data i.e. position, heading, turn rate, pitch and roll angles, airspeed, altitude, etc, are determined. Some of these data are used by the ATS, specifically the altitude, heading and GPS position data.
4.3 Communications

Unlike the camera system, the micropilot does not need a serial converter. The communication link between the micropilot and the computer is via the RS232 serial port on both the MP2028® and the PC104.

4.3.1 RS232 Serial Communication

As in all serial port initialization, it is crucial to establish handshaking between the two nodes, namely the micropilot and the computer. To enable correct reading and sending of data, it is necessary to synchronize the baud rate, each byte size and parity. The settings for the serial communication with the micropilot are listed in Table 3.

<table>
<thead>
<tr>
<th>PORT SETTINGS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate (Bits per second)</td>
<td>9600</td>
</tr>
<tr>
<td>Data Bits</td>
<td>8</td>
</tr>
<tr>
<td>Parity</td>
<td>None</td>
</tr>
<tr>
<td>Stop Bits</td>
<td>1</td>
</tr>
<tr>
<td>Flow Control</td>
<td>Hardware</td>
</tr>
</tbody>
</table>
4.4 Interconnection of Subsystems Communications

Figure 28: Interconnection of Systems

Figure 28 shows how the systems are interconnected. The PC104 receives data from the MP2028 through the serial communication. The exchange of data between the PC104 and the CMUCam2+ is also via serial communications as well. The gun system however is controlled by the servo ports of the CMUCam2+ and the digital I/O ports of PC104. Data link from the PC104 to the GCS is achieved through wireless communication.
4.5 Integration of Automatic Targeting System

Figure 29 illustrates the systems that are to be integrated onto the JR helicopter which is used in this project as the UAV platform. All the individual systems are fitted onto the JR helicopter as shown in Figure 30. The CMUCam2+ and PC104 are enclosed in the box (Figure 31) below the helicopter. A round hole is cut on the underside of the box to expose the camera lens. The WUSB54G is connected and exposed at the tail boom of the helicopter for unobstructed wireless transmission. The MP2028g is placed in the nose area of the helicopter while the gun system is attached to the leading edge of the box (Figure 32) with an offset of 34cm directly at the top of the vertical axis of the camera.
Chapter 4 Integration of ATS with Unmanned Air Vehicle

4.6 Program Functions Development

With the addition of the micropilot, more functions need to be written to initialize the system as well as to extract data from it.

4.6.1 Initialization of Serial Transmit and Receive

This function is similar to the initialization of serial transmit and receive for the camera system. It establishes handshaking, both software and hardware, between the micropilot and the computer and synchronizes the rate of data transfer (baud rate), the byte size of each transmission as well as the parity. This enables data to be sent and read correctly.
The data received from the micropilot is also similar to that of the camera system. The same processing has to be done to extract the required data as well as the sending of commands to request for different types and classes of raw data.

4.6.2 Initialization of Micropilot

The micropilot has to be initialized for it to run through all its internal checking and reporting of all its sensors. A code is written to enable this initialization so that the micropilot will be ready to give correct information from its sensors.

4.6.3 Height and Heading Update from Micropilot

The altitude and heading information is obtained by the gyro and air data unit of the micropilot. Since there are a lot of other data related to these sensory units of the micropilot, there is a need to sieve out the required information. A function is written to sieve out the altitude and heading information to update the ATS.

4.6.4 GPS Update from Micropilot

GPS position update of the UAV is obtained from the micropilot GPS system. The GPS data received comprises of a lot of other data related to the GPS system. Therefore a function has to be written to selectively obtain the required data. This information is then updated on the ATS.
4.6.5 GPS Reporting

With the target coordinates calculated with respect to the camera, it is required to calculate the GPS position of the target. A code is written to transpose the target coordinates to GPS positioning (Figure 33).

![Figure 33: Camera and GPS Coordinate System](image)

This function makes use of the current heading and the present GPS position of the UAV to obtain the GPS position of the target. The target GPS position can be calculated using Equation 8 and Equation 9.

**Equation 8 : Target GPS Northing**

\[
Tgt\_GPS\_N = GPS\_N + [(\text{grd\_dist}) \cos (\theta + \alpha)]
\]

**Equation 9 : Target GPS Easting**

\[
Tgt\_GPS\_E = GPS\_E + [(\text{grd\_dist}) \sin (\theta + \alpha)]
\]

where

- \(\theta\) = heading of UAV in degrees
- \(\alpha\) = angle of target centroid wrt camera in degrees
- \(Tgt\_GPS\_N\) = target GPS Northing
- \(Tgt\_GPS\_E\) = target GPS Easting
- \(GPS\_N\) = current GPS Northing
- \(GPS\_E\) = current GPS Easting
- \(\text{grd\_dist}\) = maximum range of setting of servo motor
4.7 Flight Test of Automatic Targeting System

A flight test is conducted to show the functionality and operability of the integrated system in the actual practical world. Uniformly coloured LED panel (Figure 34) is used as target with IR receiver attached to switch the LED off when ‘shot’ by the airborne IR transmitter which is part of the ATS.

![Figure 34: Target LED Array](image)

The ATS is mounted onto the box right below the radio controlled helicopter and initialized before it is flown to a height between 1 to 2 meters. The helicopter is then flown across 2 lit LED panels with different colours simulating 2 different targets (Figure 35).

![Figure 35: Flight Test Setup](image)
Chapter 4 Integration of ATS with Unmanned Air Vehicle

With that blue LED set as its target criteria, the ATS switches off the blue LED panel when the helicopter flies over the blue target (Figure 36). The GCS also displayed the GPS coordinates of the blue target centroid. However, when it flew past the red LED panel, the LED remains on.

![Figure 36: Search for Blue Target](image)

When the ATS is set to track for red LED, the blue LED remains on when the helicopter flew past the blue target (Figure 37). Only the red LED is switched off when the helicopter flies over the red target. The GPS coordinates of the red target centroid is also displayed on the GCS.

![Figure 37: Search for Red Target](image)

This concluded the success of the ATS in identifying and ‘shooting’ the correct target placed on the ground and also transmitting the target coordinates to GCS.
5.1 Overview of Algorithm of Source Code

Figure 38: Overview of Source Code

Figure 38 shows the overview of the algorithm of the source code. The whole system begins by initializing all the various systems required for the ATS. After initialization, the micropilot system runs in the background independently of the camera system. The micropilot continually updates the ATS on the UAV’s parameters. The camera system at the same time carries out its search for the
target. When the target is identified, the ATS gathers the real-time parameters of the UAV and computes the target coordinates. It then actuates the gun system to point and ‘shoot’ the target before reporting the target GPS position to the GCS. Thereafter, it continues its search pattern. The detailed source code is listed in Appendix I.

5.2 Main Functions of Source Code

The overall algorithm of the source code can be broadly grouped into a few categories, namely the initialization, update, search, actuation and reporting. The algorithm begins with the initialization phase where it initializes all the required systems. Once that is done, it starts its update phase in the background while performing the search phase. When the target is identified, it moves on to the actuation phase and concludes it with the reporting phase before resuming the search phase again.

5.2.1 Initialization

The first part of the source code deals with the initialization of all the required systems. It initializes the serial communications which is required to initialize the CMUCam2+ camera system as well as the MP2028® micropilot. It also initializes the digital I/O ports for use by the IR transmitter.

5.2.2 Real-time Information Update

The updating of information to the ATS is done continuously by the micropilot. It updates the real-time information such as the altitude, heading and
GPS position. This function needs to run in a continuous close loop and is therefore being threaded in order for the PC104 to multitask it to run in parallel to target searching.

5.2.3 Target Searching

This is the main algorithm of the ATS. With the given target identification criteria, it sets the CMUCam2+ to search continually in a close loop for the target. When the target is identified, it processes the data to be used by the gun system and also for reporting purposes.

5.2.4 Actuation

For the actuation algorithm, it makes use of the target data processed from the searching algorithm as well as the height information updated by the micropilot to actuate the gun system to point at the target. It actuates and aligns the 2 servo motors to point the IR transmitter at the target. It then actuates the IR transmitter to ‘shoot’ at the target. If the target remains in sight, the ATS will actuate the gun system to continually track and point at the target.

5.2.5 Reporting

The reporting algorithm makes use of the target information obtained from target searching as well as the heading and GPS position update from the micropilot. With those information, it works out the GPS position of the target and sends it down to the GCS via the wireless link.
5.3 Algorithm Considerations

In the process of programming for the ATS, several considerations are taken into account to make the system robust, reliable and efficient. Listed below are the considerations.

5.3.1 Redundancy in Height Information

As highlighted from Section 2.5.4, the height is a critical value needed to compute the target coordinates. Height information is a global variable and is being updated by the micropilot. To cater for spike or no readings from the micropilot, another global variable is catered to capture and store the last readable height information. Since the update of the micropilot is 5Hz that means the last height information is only 0.2sec apart and is therefore quite reliable. Furthermore, there is an additional backup to estimate the height with the given target parameters. Therefore there are 3 levels of redundancy in obtaining the height information.

5.3.2 Height Estimation

In the event that the height updating system by the micropilot is not available, there is still this backup function to estimate the height of the UAV using Equation 10. It makes use of the actual target size and the pixel size of the target to extrapolate the height. However, this assumes that the UAV is flying is an upright position with no banking.
Equation 10: Height Estimation

\[
\text{height} = \sqrt{\frac{\text{tgt\_area}}{\frac{\text{grd.pix\_area}}{\text{tgtpix\_area}}}} \left( \frac{1}{4\left(\tan\frac{\text{FOV}_H}{2}\right)\left(\tan\frac{\text{FOV}_V}{2}\right)} \right)
\]

where
- \( \text{tgt\_area} \) = actual target area
- \( \text{grd.pix\_area} \) = pixel area of ground (176 x 255)
- \( \text{tgtpix\_area} \) = pixel area of target captured
- \( \text{FOV}_H \) = Horizontal Field of View of camera
- \( \text{FOV}_V \) = Vertical Field of View of camera

5.3.3 Target Identification Criteria

The target criteria parameters are stored as global variables in the beginning of the source code. This allows ease in changing the parameters to cater for different mission which may require targeting of different target.

5.3.4 Flexibility of Using Different Camera System

The critical parameters of the camera system are defined clearly in the beginning of the source code. Camera specific parameters like the horizontal and vertical field of view as well as the horizontal and vertical pixel resolution are used by the ATS for target identification. These values change with each different camera used. Therefore, clearly defining these parameters in the beginning of the code will enable flexibility in changing the camera for a better one in the future if the need arises.

5.3.5 Flexibility of Using Different Servo Motors

For the gun system, the control of the gun is dependent on the servo motors used. If there is a change in the gun system, it may result in a need to
change the servo motors used to cater for the weight required or due to space limitation. Therefore the parameters of the servo motors are also clearly defined in the beginning to allow flexibility in using different servo motors for the ATS.

5.3.6 Flexibility of Gun System Mounting

The gun system is connected to the camera system but it can be placed anywhere on the UAV. This is due to the offset already catered to align the gun system to the camera system. Once the gun system is fixed onto a certain location, the offset from the camera is measured and the parameter changed in the beginning of the source code. This does not restrict the placement of the gun system but allows it to be mounted anywhere on the UAV.

5.3.7 Background Update of Flight Data

Information regarding the UAV is updated by the micropilot. It collects and updates the height, heading and GPS position of the UAV. If information is retrieved only when the target is found, it will take up some time and cause a lag in the system. Since the ATS also requires to carry out the search algorithm simultaneously, the updating is threaded to enable multitasking of both the micropilot updating system as well as the camera searching system. Threading allows it to run in the background while the ATS carries out its main task of searching for the target.
5.3.8 Feedback on Wireless Transmission

Target information is sent to the GCS via the wireless link. However, this information is only obtained and sent when the target is identified by the system. During its normal search mode, no target information is obtained and thus nothing is sent to the GCS. This may cause uncertainty in the function of the wireless communication system. Therefore a blank string is sent whenever the system is in search mode to enable the GCS to acknowledge and display that the ATS is searching for target.
CHAPTER 6 CONCLUSION

6.1 Conclusion

This report has highlighted the design of the ATS. It begins with designing the algorithm for the target identification system and builds on to develop the ATS. It shows the systems used and the development of algorithms to detect target against background clutter and relaying the target information back to the GCS. The whole system has been designed and tested carefully and progressively before integrating it onto the UAV. The ATS onboard the UAV is then tested in the open field condition to assess its operability in the real world. All these have been tested and integrated with successful.

This has proven the concept of using an ATS for the UAV. If we can replace the CMUCam2+ with a military specified camera, better and clearer target identification can definitely be achieved. Furthermore, this ATS can provide the basic platform to build on more sophisticated systems.

6.2 Future Development

This project establishes the basic foundation for ATS. More sophisticated and detailed designs could be implemented on this foundation. Listed below are some of the recommended future developments.

6.2.1 Target Lasing for Guided Weapons

Future projects can study on improving the gun system. A more robust and actual shooting mechanism can be designed to actually shoot and destroy the
target. Another alternative would be to change the gun system and use this system as a platform to perform target lasing for guided weapons.

6.2.2 Real-Time Pictures Download

Currently, the flight computer only sends target information to the GCS. Since the communication link between the flight computer and GCS has been established, there is a possibility that pictures can be transmitted. Research can be done on sending real time pictures down to the GCS. This will give commanders a view onto the real-time target area as well.

6.2.3 Flexibility of Altering Target Identification Criteria

The target identification criteria is currently preloaded before the mission. It will be better if the criteria can be changed by the GCS. This will enable flexibility of changing target when the UAV is already airborne.

6.2.4 Tracking Multiple Targets

This ATS system is designed to search for a single target. Further research can be done to design a system that can track multiple targets. The study can even be extended to allow the system to track multiple targets with different target criteria.

6.2.5 Tracking Airborne Targets

Study can also be done on using this ATS for airborne targets. The study will focus more on targeting very fast moving targets like the airplanes.
REFERENCES


APPENDIX A: Performance Indicator For Vision Systems

\[ P.I. = \frac{(\text{Maximum Resolution})_{\text{pixels}} \times (\text{Maximum Frame Rate})_{\text{fps}}}{(\text{Weight})_e \times (\text{Cost})_{ss}} \]

The maximum resolution of the vision system is related to the maximum altitude that the platform can fly. And the maximum frame rate is related to the maximum speed that the platform can fly. Therefore the higher the values of these 2 factors, the better will be the performance of the vision system. As such, they are directly proportional to the P.I.

The weight relates to the amount of thrust that the platform needs as well as the other payloads that the platform can carry. The lighter the weight, the lower the thrust required and also the more of other payloads the platform can carry. Therefore the lower the weight and cost of the vision system, the more desirable is the system. As such, these 2 factors are inversely proportional to the P.I.

\[ (P.I.)_{\text{CMUCam2+}} = \frac{(176 \times 255) \times (26)}{(40) \times (279)} = 104.6 \]

\[ (P.I.)_{\text{EyeCam}} = \frac{(640 \times 480) \times (15)}{(216) \times (1744)} = 12.2 \]

Since the P.I. of CMUCam2+ is higher than that of the EyeCam, CMUCam2+ will be chosen as the vision system for the ATS. Even though the resolution is lower as compared to the EyeCam, the performance is still well within the limit required for the ATS.
APPENDIX B: CMUCam2+

Figure 39: Different Views and Dimensions of CMUCam2+

Figure 39 shows the top and side views of the CMUCam2+ as well as its dimensions. The CMUcam2+ consists of a SX52 microcontroller interfaced with a 0V6620 Omnivision CMOS camera that allows high-level data to be extracted from the camera's streaming video. Its primary function is to track and monitor highly contrasting color regions. It can also detect motion, provide color statistics, and transmit image information to a computer for additional processing.

Since the board is designed to work with microcontrollers, it has a serial interface using TTL voltage levels (0-5V). When using the CMUcam2+ with an
RS-232 serial port, such as on a PC or USB-to-serial adapter, a level-shifter is needed to convert TTL signals to RS-232 signals.

When compared with our other CMUcam2 products, the CMUcam2+ has these important differences:

- Thinner profile
- Lighter weight
- Uses only the more capable OV6620 camera
- No level-shifter on board for more efficient connection to +5V controllers
- Minimal packaging: does not include printed manual, AC adapter, or CD-ROM
- Different connector arrangement with horizontal pins for easier stacking

The CMUcam2+ has the following functionality:

- Track user-defined colors at up to 50 Frames Per Second (FPS)
- Track motion using frame differencing at 26 FPS
- Find the centroid of any tracked data
- Gather mean color and variance information
- Gather a 28 bin histogram of each color channel
- Manipulate horizontally pixel-differenced images
- Transfer a real-time binary bitmap of the tracked pixels in an image
- Adjust the camera’s image properties
- Dump a raw image (single or multiple channels)
- Up to 176 X 255 resolution
- Supports multiple baudrates
- Control 5 servo outputs
APPENDIX C: Experiment to find Field of View of Camera

A large grid chart of 10cm by 10cm squares is drawn and put up on the wall. The camera which is mounted on a tripod is carefully placed at various distances from the chart and snapshots are taken. The field of views are obtained by using graphs (Figure 40) of distance of camera from chart against the actual distance captured on the snapshot. Figure 41 shows a sample of a snapshot taken from the CMUCam2+.

![Graphs of Camera Field of View](image)

**Horizontal Field of View**

\[ y = 0.4133x \]

\[ y = -0.4133x \]

**Vertical Field of View**

\[ y = 0.2657x \]

\[ y = -0.2657x \]

Figure 40: Graphs of Camera Field of View
Figure 41: Sample Snapshot of CMUCam2+
## APPENDIX D: PC104 Specifications

### Specifications

<table>
<thead>
<tr>
<th>&gt;&gt; Systems</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Applicable Models</td>
<td>Support all boards with PC/104 Bus</td>
</tr>
<tr>
<td>Chipset</td>
<td>16C550 UART with 16-byte FIFO</td>
</tr>
<tr>
<td>Function (For 16550-Compatible type of UART) adjusted by jumper setting.</td>
<td>The 9th pin of both ports can be set by jumper for either 5V or 12V output. LPT1 for SPP/EPP/ECP</td>
</tr>
<tr>
<td>Interrupt Levels(IRQ)</td>
<td>COM1 Default: IRQ10</td>
</tr>
<tr>
<td></td>
<td>COM2 Default: IRQ11</td>
</tr>
<tr>
<td></td>
<td>LPT1 Default: IRQ5</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>Typical: +5V @400 mA</td>
</tr>
<tr>
<td></td>
<td>Maximum: +5V @950 mA</td>
</tr>
<tr>
<td>MTBF</td>
<td>175,200 hrs (20 years)</td>
</tr>
<tr>
<td>Dimension</td>
<td>90mm x 96mm, 3.6&quot; x 3.8&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&gt;&gt; I/O Ports</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Connector</td>
<td>PC/104 interface</td>
</tr>
<tr>
<td>Output Connector</td>
<td>COM1: 10-pin Box-head connector</td>
</tr>
<tr>
<td></td>
<td>COM2: 10-pin Box-head connector</td>
</tr>
<tr>
<td></td>
<td>LPT1: 26-pin Box-head connector</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&gt;&gt; Environment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation Temperature</td>
<td>0°C<del>65°C(32°F</del>140°F)</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>-40°C<del>85°C(-40°F</del>185°F)</td>
</tr>
<tr>
<td>Humidity</td>
<td>Operation Humidity 20<del>90%, Storage Humidity 20</del>95%</td>
</tr>
</tbody>
</table>
APPENDIX E: Futaba 3001 Servo Motor Specifications

Figure 42 shows the specifications and Figure 43 shows the dimensions of the Futaba S3001 Servo Motor.

<table>
<thead>
<tr>
<th>S3001 Standard</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FUTM0022</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Volts</th>
<th>Torque</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8V</td>
<td>44.4 oz-in.</td>
<td>0.22 sec/60°</td>
</tr>
<tr>
<td>6.0V</td>
<td>56.9 oz-in.</td>
<td>0.19 sec/60°</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6 x 0.6 x 1.4 in.</td>
<td>1.6 oz.</td>
</tr>
</tbody>
</table>

Figure 42: Futaba S3001 Specifications

Figure 43: Futaba S3001 Dimensions
APPENDIX F: Experiment to Find Maximum Servo Angle

Figure 44 shows the graph for maximum servo angle.

Maximum servo angle = \( \tan^{-1}(2.6079) + \tan^{-1}(2.6937) \)

= 69.02° + 69.63°

= 138.65°
APPENDIX G: WUSB54G Specifications

Listed in Table 4 below are the specifications of the Linksys Wireless-G USB Network Adapter WUSB54G (Figure 45).

Table 4: Specifications of WUSB54G

<table>
<thead>
<tr>
<th>SPECIFICATIONS OF WUSB54G</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
</tr>
<tr>
<td><strong>Standards</strong></td>
</tr>
<tr>
<td><strong>Ports</strong></td>
</tr>
<tr>
<td><strong>Channels</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>LEDs</strong></td>
</tr>
<tr>
<td><strong>Transmitted Power</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Receive Sensitivity</strong></td>
</tr>
<tr>
<td><strong>Security features</strong></td>
</tr>
<tr>
<td><strong>WEP key bits</strong></td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
</tr>
<tr>
<td><strong>Unit Weight</strong></td>
</tr>
<tr>
<td><strong>Certifications</strong></td>
</tr>
<tr>
<td><strong>Operating Temp.</strong></td>
</tr>
<tr>
<td><strong>Storage Temp.</strong></td>
</tr>
<tr>
<td><strong>Operating Humidity</strong></td>
</tr>
<tr>
<td><strong>Storage Humidity</strong></td>
</tr>
</tbody>
</table>
## APPENDIX H: MP2028<sup>g</sup> Specifications

### Specifications - MP2028<sup>g</sup>

<table>
<thead>
<tr>
<th>Servos &amp; Mixing</th>
<th>MP2028&lt;sup&gt;g&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>elevon</td>
<td>yes</td>
</tr>
<tr>
<td>flaperons</td>
<td>yes</td>
</tr>
<tr>
<td>4 servo flap/aileron</td>
<td>yes</td>
</tr>
<tr>
<td>separate flaps</td>
<td>yes</td>
</tr>
<tr>
<td>v-tail</td>
<td>yes</td>
</tr>
<tr>
<td>split rudders</td>
<td>yes</td>
</tr>
<tr>
<td>number of servos</td>
<td>8/16/24</td>
</tr>
<tr>
<td>servo update rate</td>
<td>50 hz</td>
</tr>
<tr>
<td>separate servo and main battery power supply</td>
<td>yes</td>
</tr>
<tr>
<td>separate voltage monitor for main and servo battery</td>
<td>yes</td>
</tr>
<tr>
<td>supplies</td>
<td></td>
</tr>
<tr>
<td>integrated RC override</td>
<td>yes</td>
</tr>
<tr>
<td>servo resolution</td>
<td>11 bit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control System</th>
<th>MP2028&lt;sup&gt;g&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>inner loop update rate</td>
<td>30 hz</td>
</tr>
<tr>
<td>gain scheduling for optimum performance</td>
<td>yes</td>
</tr>
<tr>
<td>rudder aileron feed forward for improved turn</td>
<td>yes</td>
</tr>
<tr>
<td>performance</td>
<td></td>
</tr>
<tr>
<td>aileron elevator feed forward for improved altitude</td>
<td>yes</td>
</tr>
<tr>
<td>hold during turns</td>
<td></td>
</tr>
<tr>
<td>autonomous takeoff and landing</td>
<td>yes</td>
</tr>
<tr>
<td>user definable PID feedback loops (for camera</td>
<td>16</td>
</tr>
<tr>
<td>stabilization etc)</td>
<td></td>
</tr>
<tr>
<td>user definable table lookup functions</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sensors</th>
<th>MP2028&lt;sup&gt;g&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>airspeed sensor range (kph)</td>
<td>500</td>
</tr>
<tr>
<td>altimeter range (meters above launch point)</td>
<td>12000</td>
</tr>
<tr>
<td>pitch, Roll, and Yaw Gyros</td>
<td>yes</td>
</tr>
<tr>
<td>y-acellerometer to co-ordinate turns</td>
<td>yes</td>
</tr>
</tbody>
</table>
### Navigation

<table>
<thead>
<tr>
<th>Feature</th>
<th>MP2028®</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Hz GPS update rate</td>
<td>yes</td>
</tr>
<tr>
<td>move servo at waypoint</td>
<td>yes</td>
</tr>
<tr>
<td>change altitude at waypoint</td>
<td>yes</td>
</tr>
<tr>
<td>change airspeed at waypoint</td>
<td>yes</td>
</tr>
<tr>
<td>user definable holding patterns</td>
<td>yes</td>
</tr>
<tr>
<td>user definable error handlers (loss of GPS, low battery etc.)</td>
<td>yes</td>
</tr>
<tr>
<td>RPV and UAV modes</td>
<td>yes</td>
</tr>
<tr>
<td>supports DGPS accuracy</td>
<td>yes</td>
</tr>
<tr>
<td>waypoints</td>
<td>1,000</td>
</tr>
</tbody>
</table>

### Telemetry, Datalog & Video

<table>
<thead>
<tr>
<th>Feature</th>
<th>MP2028®</th>
</tr>
</thead>
<tbody>
<tr>
<td>telemetry (user defined fields transmitted each second)</td>
<td>100</td>
</tr>
<tr>
<td>telemetry update rate</td>
<td>5hz</td>
</tr>
<tr>
<td>size of onboard datalog</td>
<td>1.5 MB</td>
</tr>
<tr>
<td>datalog update rate</td>
<td>5hz</td>
</tr>
<tr>
<td>video overlay (number of user definable fields)</td>
<td>16</td>
</tr>
<tr>
<td>video overlay uses low cost industry standard video overlay boards</td>
<td>yes</td>
</tr>
<tr>
<td>pressure altitude and pressure airspeed available on video overlay</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Ground control station

<table>
<thead>
<tr>
<th>Feature</th>
<th>MP2028®</th>
</tr>
</thead>
<tbody>
<tr>
<td>HORIZON® ground control software included with system</td>
<td>yes</td>
</tr>
<tr>
<td>MP2028® autopilot simulator for operator training</td>
<td>yes</td>
</tr>
<tr>
<td>ground control station developer's kit</td>
<td>yes</td>
</tr>
<tr>
<td>gains can be adjusted in-flight</td>
<td>yes</td>
</tr>
<tr>
<td>change waypoints in-flight</td>
<td>yes</td>
</tr>
<tr>
<td>payload servos controlled from ground station</td>
<td>yes</td>
</tr>
<tr>
<td>fly in RC mode via datalink (both stabilized and normal)</td>
<td>yes</td>
</tr>
<tr>
<td>point and click waypoint editor</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Physical Characteristics

<table>
<thead>
<tr>
<th>Feature</th>
<th>MP2028®</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight (including GPS receiver, gyros, and sensors)</td>
<td>28 grams (not including GPS antenna)</td>
</tr>
<tr>
<td>power (including GPS receiver, gyros, all sensors and GPS antenna)</td>
<td>140 mA @ 6.5V</td>
</tr>
<tr>
<td>supply Voltage</td>
<td>4.2 - 26V</td>
</tr>
<tr>
<td>size - length (cm)</td>
<td>10.0 cm</td>
</tr>
<tr>
<td>size - width (cm)</td>
<td>4.0 cm</td>
</tr>
<tr>
<td>size - height (cm)</td>
<td>1.5 cm</td>
</tr>
<tr>
<td>software upgradable in the field</td>
<td>yes</td>
</tr>
</tbody>
</table>
Listed below are some of the characteristics of the micropilot MP2028®.

- World’s smallest UAV autopilot; 28 grams, 4 cm by 10 cm
- GPS waypoint navigation with altitude and airspeed hold
- Completely independent operation including autonomous takeoff, bungee launch, hand launch and landing
- Powerful script language command set
- Open architecture – all state fields fully accessible
- Fully integrated with 3-axis gyros & accelerometers, GPS, pressure altimeter, pressure airspeed sensors, all on a single circuit board
- Extensive data logging and telemetry capabilities
- UAV configuration wizard and installation video simplify installation
- Includes HORIZON® ground control software
APPENDIX I: Source Codes For ATS

Automated Targeting System
using CMUCAM2+

Created by: Mike Low
Updated on: 17 Mar 2006

#include <winsock.h>
#include <cstdlib>
#include <windows.h>
#include <stdio.h>
#include <iostream>
#include <math.h>
#include <string>
#include <time.h>
#include "PracticalSocket.h"
#include "dscud.h"

using namespace std;

// Fixed parameters (in cm and degrees)

// Camera System
#define H_FOV 44.91
#define V_FOV 29.76
#define H_pix 176
#define V_pix 255
#define tolerance 25
#define pi 3.14159

// Gun System
#define servo_angle 138.65
#define servo_range 165  // 46 to 210
#define x_offset 0  // offset of the servos wrt CMUCAM2+
#define y_offset -34     // 34cm above camera

// ******************************************
//                FUNCTIONS
// ******************************************

// Camera System
void CheckSerial(HANDLE);
void ErrorHandler(char *, DWORD);
void serialDCB_setup(HANDLE);
void serialTimer_setup(HANDLE);
void init_camera(HANDLE);
void set_trackcolour(HANDLE);
int minvalue(int, int);
int maxvalue(int, int);
void trackcolour(HANDLE);
double estimate_height(int);
double displacement(double, double, double, double);
void report_position(double, double);

void send_blank();
void send_string(double, double);

// GPS System

// ******************************************
//                FUNCTIONS
// ******************************************

// Camera System
void CheckSerial(HANDLE);
void ErrorHandler(char *, DWORD);
void serialDCB_setup(HANDLE);
void serialTimer_setup(HANDLE);
void init_camera(HANDLE);
void set_trackcolour(HANDLE);
int minvalue(int, int);
int maxvalue(int, int);
void trackcolour(HANDLE);
double estimate_height(int);
double displacement(double, double, double, double);
void report_position(double, double);

// GPS System
void send_blank();
void send_string(double, double);
// Gun System
void aim_target(HANDLE, double, double, double);
void set_servo(HANDLE, int, int);

// Micropilot System
void serialPilotDCB_setup(HANDLE);
void init_micropilot(HANDLE);
int micropilot_sensor(HANDLE, int);
int micropilot_gps(HANDLE, int);

// ******************************************
//       GLOBAL VARIABLES DECLARATION
// ******************************************

DWORD WINAPI Micropilot(LPVOID);
HANDLE hPilot;
double height, lastheight;
double heading, lastheading;
double GPS_E1, GPS_E2, GPS_N1, GPS_N2;

// Target Criteria
int tgt_R = 120;    // RED target
int tgt_G = 16;
int tgt_B = 16;
int tgt_length = 10;  // in cm
int tgt_breadth = 10;  // in cm

// ******************************************
//               MAIN PROGRAM
// ******************************************

int main()
{
    DWORD dwThreadId, dwThrdParam = 1;
    HANDLE hThread;

    HANDLE hCamera;
    BOOL success;
    DWORD numWrite, numRead;
    char str[40] = {0};
    double xpos, ypos;
    char *s;
    char *tempstr;
    int Tpacket[8] = {0};
    int j;
    int start = 0;
    int found = 0;
    int tgtMX, tgtMY;
    int tgtx1, tgtx2, tgty1, tgty2;
    time_t startTime, endTime;

    cout << "Fights On!!" << endl << endl;
    time(&startTime);     // Log down the start time

    //========================================================================
    // I. INITIALIZATION
    //
    //    a) Serial Port Communications
    //       1) CMUcam2+ Comms   (COM1)
    //       2) Micropilot Comms (COM2)
    //    b) CMUcam2+
    //    c) MP2028 Micropilot
    //    d) Digital IO
    //}
Appendix I

Initialising the serial ports

~~~~~~~~~~~~~~~~~~~~

cout << "Initialising Serial Port Communications......." << endl;

hCamera = CreateFile ("COM1", GENERIC_READ | GENERIC_WRITE, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
CheckSerial(hCamera);
serialDCB_setup(hCamera);          // Setting parameters for the serial port.
serialTimer_setup(hCamera);         // Setting timeouts for the serial port
EscapeCommFunction(hCamera, SETDTR); // Sends the DTR (data-terminal-ready) signal to hCamera

hPilot = CreateFile ("COM2", GENERIC_READ | GENERIC_WRITE, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
CheckSerial(hPilot);
serialPilotDCB_setup(hPilot);       // Setting parameters for the serial port.
serialTimer_setup(hPilot);          // Setting timeouts for the serial port
EscapeCommFunction(hPilot, SETDTR); // Sends the DTR (data-terminal-ready) signal to hCamera

Initialising camera and micropilot.

~~~~~~~~~~~~~~~~~~~~~~~~~

init_camera(hCamera);       // Initialising Camera
set_trackcolour(hCamera);  // Set Tracking Parameters
init_micropilot(hPilot);    // Initialising Micropilot

Initialising digital IO port.

~~~~~~~~~~~~~~~~~~~~

DSCB dscb;    // handle used to refer to the board
DSCCB dsccb;    // structure containing board settings
BYTE config_bytes[2];  // holds configuration for DIO ports
ERRPARAMS errorParams; // structure for returning error code and error string

A. DRIVER INITIALIZATION

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if( dscInit( DSC_VERSION ) != DE_NONE )
{
    dscGetLastError(&errorParams);
    fprintf( stderr, "dscInit error: %s %s
", dscGetErrorString(errorParams.ErrCode), errorParams.errstring );
}

B. BOARD INITIALIZATION

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

dsccb.base_address = 0x300; // default address to automate program

if(dscInitBoard(DSC_OMMDIO, &dsccb, &dscb)!= DE_NONE)
{
    dscGetLastError(&errorParams);
    fprintf( stderr, "dscInitBoard error: %s %s
", dscGetErrorString(errorParams.ErrCode), errorParams.errstring );
}

C. IR Working Codes

~ Setup the following parameters for the user interrupt:
// config_bytes[ ] for dscDIOSetConfig
// IR: init to port 4, bit 4

// config_bytes[0] = 0x89;       // Port 1A(0)-Output, Port 1B(1)-input, Port 1C(2)-output
config_bytes[1] = 0x89;       // Port 2A(4)-Input, Port 2B(5)-Output, Port 2C(6)-input

dscDIOSetConfig(dscb, config_bytes);
Appendix I

// Set IR to standby
// ~~~~~~~~~~~~~
dscDIOOutputBit (dscb, 4, 4, 0); // Output port 4, bit 4, set to low

//=========================================================================
// II. REAL-TIME INFORMATION UPDATE
//
// a) MP2028 Micropilot updates in the background
// 1) Height
// 2) Heading
// 3) GPS position
//=========================================================================

hThread = CreateThread(NULL,   // no security attributes
    0,    // use default stack size
    Micropilot,  // thread function
    &dwThrdParam, // argument to thread function
    0,    // use default creation flags, possible CREATE_SUSPENDED
    &dwThreadId); // returns the thread identifier

SetThreadPriority(hThread, 15);
if (hThread == NULL)
cerr << "CreateThread failed.;"

//=========================================================================
// III. TARGET SEARCHING
//
// a) Search for target
//=========================================================================

do
{
    trackcolour(hCamera); // Start Tracking
    Sleep(60);
    if (start != 1)
    {
        cout << endl << "Camera’s Tracking!!" << endl;
        start = 1;
    }

    // Reading data from the Camera serial port.
    success = ReadFile(hCamera, str, 40, &numRead, 0);
    if (!success)
        ErrorHandler("In ReadFile", GetLastError());

    // Processing data received.
    // ~~~~~~~~~~~~~
    j = 0;
    tempstr = strstr(str, "T");
    s = strtok(tempstr, ", ");
    while ((s!=NULL) && (j<7))
    {
        switch(j)
        {
            case 1:
                tgtMX = atoi(s);
                break;
            case 2:
                tgtMY = atoi(s);
                break;
            case 3:
                tgtx1 = atoi(s);
                break;
            case 4:
                tgty1 = atoi(s);
                break;
        }
    }
}
case 5:
    tgtx2 = atoi(s);
    break;

case 6:
    tgty2 = atoi(s);
    break;
}

j++;
    s = strtok(NULL, " ");
}

DONE

// Searching for target
// ~~~~~~~~~~~~~~~
if ((tgtMX < 1) && (tgtMY < 1))
{
    if (found == 1)
    {
        cout << endl << "Target not in sight" << endl;
        // Off IR
        // ~~~~~
        dscDIOOutputBit (dscb, 4, 4, 0); // Output port 4, bit 4, set to low
    }
    cout << "Searching for target......\r";
    send_blank();
    found = 0;
}

// Target Found!
// ~~~~~~~~~~~
else
{

// IV. ACTUATION & REPORTING

// a) Shoot target
// b) Report target position to GCS

// Locating target position
// ~~~~~~~~~~~~~~~
if (found == 0)
{
    cout << endl << endl << "Target found!!!" << endl;
    found = 1;
}

// Estimating height if both height and lastheight are not known
if (height == 0) height = estimate_height((tgtx2 - tgtx1 + 1) * (tgty2 - tgty1 + 1));

xpos = displacement(height, H_pix, tgtMX, H_FOV); // right is -ve
ypos = displacement(height, V_pix, tgtMY, V_FOV); // down is -ve

aim_target(hCamera, (xpos*(-1)), ypos, height);

// Shoot IR
// ~~~~~~~~
dscDIOOutputBit (dscb, 4, 4, 1);  // Output port 4, bit 4, set to high
report_position(xpos, ypos);  // Display position on ATS
send_string((xpos*(-1)), ypos);  // Send position to GCS
}
FlushFileBuffers(hCamera);  // Clears the buffers for hCamera
EscapeCommFunction (hCamera, CLRDTR);  // Clears the DTR (data-terminal-ready) signal
time(&endTime);  // end after 10800 seconds (3hrs)

} while ((endTime - startTime) < 10800); // end after 10800 seconds (3hrs)
Appendix I

//=========================================================================
// V. SHUTTING DOWN
//
//      a) CMUcam2+
//      b) Serial Port
//      c) Digital IO
//=========================================================================

// Shutting down the CMUcam2+.
strcpy(str, ":");
WriteFile(hCamera, str, strlen(str), &numWrite, 0);
CloseHandle (hCamera);
CloseHandle (hPilot);
dscFree();  // freeing digital IO resources

return 0;
}

.isTrue

I-6
void serialDCB_setup(HANDLE hFile)
{
    // Setting parameters for the serial port.
    DCB dcbSerialParams = {0};
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);

    if(!GetCommState(hFile, &dcbSerialParams))
    {
        cout << "Error getting serial port state!" << endl;
    }
    dcbSerialParams.BaudRate = CBR_115200;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;

    if(!SetCommState(hFile, &dcbSerialParams))
    {
        ErrorHandler("In SetCommState", GetLastError());
        cout << "Error setting serial port state!" << endl;
    }
}

void serialPilotDCB_setup(HANDLE hFile)
{
    // Setting parameters for the serial port.
    DCB dcbSerialParams = {0};
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);

    if(!GetCommState(hFile, &dcbSerialParams))
    {
        cout << "Error getting serial port state!" << endl;
    }
    dcbSerialParams.BaudRate = CBR_9600;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;

    if(!SetCommState(hFile, &dcbSerialParams))
    {
        ErrorHandler("In SetCommState", GetLastError());
        cout << "Error setting serial port state!" << endl;
    }
}

void serialTimer_setup(HANDLE hFile)
{
    // Setting timeouts for the serial port.
    COMMTIMEOUTS timeouts={0};
    timeouts.ReadIntervalTimeout=50;
    timeouts.ReadTotalTimeoutConstant=50;
    timeouts.ReadTotalTimeoutMultiplier=10;
    timeouts.WriteTotalTimeoutConstant=50;
    timeouts.WriteTotalTimeoutMultiplier=10;

    if(!SetCommTimeouts(hFile, &timeouts))
    {
        ErrorHandler("In SetCommTimeouts", GetLastError());
        cout << "Timeout error has occurred!!!" << endl;
    }
}
void init_camera(HANDLE hFile)
{
    char cmdstr[30] = {0};
    DWORD numWrite;
    BOOL success;

    cout << "Initialising CMUcam2+.....\n" << endl;

    strcpy(cmdstr, "CR 19 32\r");
    success = WriteFile(hFile, cmdstr, strlen(cmdstr), &numWrite, 0);
    if (!success)
        ErrorHandler("In WriteFile", GetLastError());
    Sleep(500);

    strcpy(cmdstr, "HR 1\r");
    success = WriteFile(hFile, cmdstr, strlen(cmdstr), &numWrite, 0);
    if (!success)
        ErrorHandler("In WriteFile", GetLastError());
    Sleep(500);

    cout << "Camera's initialised!!!\n" << endl;
}

void set_trackcolour(HANDLE hFile)
{
    int rmin, rmax, gmin, gmax, bmin, bmax;
    char tempstr[5] = {0};
    char sendST[50] = {0};
    DWORD numWrite;
    BOOL success;

    // Calculating the min and max range of RGB
    rmin = minvalue(tgt_R, tolerance);
    rmax = maxvalue(tgt_R, tolerance);
    gmin = minvalue(tgt_G, tolerance);
    gmax = maxvalue(tgt_G, tolerance);
    bmin = minvalue(tgt_B, tolerance);
    bmax = maxvalue(tgt_B, tolerance);

    strcpy(sendST, "ST ");
    itoa(rmin, tempstr, 10);
    strcat(sendST, tempstr);
    strcat(sendST, " ");
    itoa(rmax, tempstr, 10);
    strcat(sendST, tempstr);
    strcat(sendST, " ");
    itoa(gmin, tempstr, 10);
    strcat(sendST, tempstr);
    strcat(sendST, " ");
    itoa(gmax, tempstr, 10);
    strcat(sendST, tempstr);
    strcat(sendST, " ");
    itoa(bmin, tempstr, 10);
    strcat(sendST, tempstr);
    strcat(sendST, " ");
    itoa(bmax, tempstr, 10);
    strcat(sendST, tempstr);
    strcat(sendST, "\r");

    // example "ST 240 255 240 255 240 255\r"
    success = WriteFile(hFile, sendST, strlen(sendST), &numWrite, 0);
    if (!success)
        ErrorHandler("In WriteFile", GetLastError());
}

int minvalue(int rgb, int tol)
{
    int min;
    min = rgb - tol;
    if (min > 255) min = 255;
    if (min < 0) min = 0;
    return min;
}

int maxvalue(int rgb, int tol)
{
    int max;
    max = rgb + tol;
    if (max > 255) max = 255;
    if (max < 0) max = 0;
    return max;
}

void trackcolour(HANDLE hFile)
{
    char sendTC[10] = {0};
    DWORD numWrite;
    BOOL success;
    strcpy(sendTC, "TC\r");
    success = WriteFile(hFile, sendTC, strlen(sendTC), &numWrite, 0);
    if (!success)
        ErrorHandler("In WriteFile", GetLastError());
}

double estimate_height(int T_area_pix)
{
    double ht, tgt_area, grd_area;
    tgt_area = tgt_length * tgt_breadth;
    grd_area = 2 * tan(H_FOV * (pi/180)/2) * 2 * tan(V_FOV * (pi/180)/2);
    ht = sqrt((176 * 255 * tgt_area) / (T_area_pix * grd_area));
    return ht;
}

double displacement(double ht, double grdpix, double centroidpix, double fov)
{
    double edge_dist, centre_dist;
    edge_dist = (2 * ht * tan(fov * (pi/180)/2)) * (centroidpix/grdpix);
    centre_dist = ((ht * tan(fov * (pi/180)/2)) - edge_dist);
    return centre_dist; // +ve if point is left/up
}

void report_position(double xpos, double ypos)
{
    if (xpos > 0) strcpy(xloc, "left");
    else
    {
        xpos = xpos * (-1);
        strcpy(xloc, "right");
    }
    if (ypos > 0) strcpy(yloc, "up");
    else
    {
        ypos = ypos * (-1);
        strcpy(yloc, "down");
    }
}
cout << "The target is ";
    cout.width(2);
    cout << (int)xpos << " cm ";
    cout << xloc << " and ";
    cout.width(2);
    cout << (int)ypos << " cm ";
    cout << yloc << " from the centre!"
        cout << " Tgt aimed! Info sent! r";
}

void send_blank()
{
    char sendstr[255];

    strcpy(sendstr, "x");
    string servAddress = "212.212.0.33";  //Fixed at 212.212.0.33 (BaseServer IP)
    unsigned short commsServPort = 6112;
    int sendstrLen = strlen(sendstr);

    // Send the string to the server
    try
    {
        UDPSocket sock;

        // send the string (sendstr) to the server at (servAddress) thru Port (commsServPort)
        sock.sendTo(sendstr, sendstrLen, servAddress, commsServPort);
    }
    catch (SocketException &e)
    {
        cout << "help!!!" << endl;
        cerr << e.what() << endl;
        exit(1);
    }
}

void send_string(double xpos, double ypos)
{
    char tempstr[3] = {0};
    char GPS_tempstr[8] = {0};
    char sendstr1[255];
    char sendstr2[255];
    char sendstr3[255];
    double pix_angle;
    double dist;
    double N2_GPS, E2_GPS;
    double N_dist, E_dist;

    // Transposing from camera to GPS grid
    pix_angle = atan(xpos/ypos);
    dist = sqrt(pow(xpos,2) * pow(ypos,2));
    N_dist = dist * (cos(pix_angle + heading));
    E_dist = dist * (sin(pix_angle + heading));

    N2_GPS = GPS_N2 + ((N_dist/11100000) * 1000000);  // 1 = 111km = 11100000cm
    E2_GPS = GPS_E2 + ((E_dist/11100000) * 1000000);  // 1 = 111km = 11100000cm

    strcpy(sendstr3, "The target GPS is N");
    itoa((int)GPS_N1, GPS_tempstr, 10);
    strcat(sendstr3, GPS_tempstr);
    strcat(sendstr3, ".");
    itoa((int)N2_GPS, GPS_tempstr, 10);
    strcat(sendstr3, GPS_tempstr);
    strcat(sendstr3, " and E");
    itoa((int)GPS_E1, GPS_tempstr, 10);
    strcat(sendstr3, GPS_tempstr);
    strcat(sendstr3, ", ");
    itoa((int)E2_GPS, GPS_tempstr, 10);
    strcat(sendstr3, GPS_tempstr);
    strcat(sendstr3, ");
}
// Determining target position in terms of left/right/up/down
if (xpos > 0) strcpy(xloc, "right");
else
{
    xpos = xpos * (-1);
    strcpy(xloc, "left");
}

if (ypos > 0) strcpy(yloc, "up");
else
{
    ypos = ypos * (-1);
    strcpy(yloc, "down");
}

strcpy(sendstr1, "The target is ");
itoa((int)xpos, tempstr, 10);
strcat(sendstr1, tempstr);
strcat(sendstr1, "cm ");
strcat(sendstr1, xloc);
strcat(sendstr1, " and ");
itoa((int)ypos, tempstr, 10);
strcat(sendstr1, tempstr);
strcat(sendstr1, "cm ");
strcat(sendstr1, yloc);
strcat(sendstr1, " from the centre!");

// Present GPS of UAV
strcpy(sendstr2, "The current GPS for the UAV is N");
itoa((int)GPS_N1, GPS_tempstr, 10);
strcat(sendstr2, GPS_tempstr);
strcat(sendstr2, ".");
itoa((int)GPS_N2, GPS_tempstr, 10);
strcat(sendstr2, GPS_tempstr);
strcat(sendstr2, " and E");
itoa((int)GPS_E1, GPS_tempstr, 10);
strcat(sendstr2, GPS_tempstr);
strcat(sendstr2, ".");
itoa((int)GPS_E2, GPS_tempstr, 10);
strcat(sendstr2, GPS_tempstr);

string servAddress = "212.212.0.33";  // Fixed at 212.212.0.33 (BaseServer IP)
unsigned short commsServPort = 6112;
int sendstrLen1 = strlen(sendstr1);
int sendstrLen2 = strlen(sendstr2);
int sendstrLen3 = strlen(sendstr3);

// Send the string to the server
try
{
    UDPsocket sock;

    // send the string (sendstr1) to the server at (servAddress) thru Port (commsServPort)
    sock.sendTo(sendstr1, sendstrLen1, servAddress, commsServPort);
    sock.sendTo(sendstr2, sendstrLen2, servAddress, commsServPort);
    sock.sendTo(sendstr3, sendstrLen3, servAddress, commsServPort);
}
catch (SocketException &e)
{
    cout << "help!!!!!" << endl;
    cerr << e.what() << endl;
    exit(1);
}
void aim_target(HANDLE hFile, double xpos, double ypos, double height)
{
    double x, y;
    double pan_angle, tilt_angle;
    int pulse, servo_pos;
    x = xpos + x_offset;
    y = ypos + y_offset;

    // Setting Pan Servo (x-axis if required)
    if (x != 0)
    {
        pan_angle = (atan2(x, height))*(180/pi);
        pulse = (pan_angle/servo_angle)*servo_range;
        servo_pos = 128 + pulse;
        set_servo(hFile, servo_pos, 1);  // Set Pan Servo 1
    }

    // Setting Tilt Servo (y-axis if required)
    if (y != 0)
    {
        tilt_angle = (atan2(y, height))*(180/pi);
        pulse = (tilt_angle/servo_angle)*servo_range;
        servo_pos = 128 + pulse;
        set_servo(hFile, servo_pos, 2);  // Set Tilt Servo 2
    }
}

void set_servo(HANDLE hFile, int servo_pos, int servo_no)
{
    char cmdstr[30] = {0};
    char tempstr[2] = {0};
    DWORD numWrite;
    BOOL success;
    strcpy(cmdstr, "SV ");
    itoa(servo_no, tempstr, 10);
    strcat(cmdstr, tempstr);
    strcat(cmdstr, " ");
    itoa(servo_pos, tempstr, 10);
    strcat(cmdstr, tempstr);
    strcat(cmdstr, "\r");
    // example "SV 1 46\r"
    success = WriteFile(hFile, cmdstr, strlen(cmdstr), &numWrite, 0);
    if (!success)
        ErrorHandler("In WriteFile", GetLastError());
}

void init_micropilot(HANDLE hFile)
{
    BOOL success;
    DWORD numWrite, numRead;
    char str[10] = {0};
    char rec_str[120] = {0};
    int i;
    strcpy(str, "SSSS"); // Sensor Report
    WriteFile(hFile, str, strlen(str), &numWrite, 0);
    for (i=0; i<4; i++)  // Sieving out unwanted info
    {
        success = ReadFile(hFile, rec_str, 120, &numRead, 0);
        if (!success)
            ErrorHandler("In ReadFile", GetLastError());
    }
    FlushFileBuffers(hFile);  // Clears the buffers for hPilot
    EscapeCommFunction (hFile, CLRDTR);  // Clears the DTR (data-terminal-ready) signal
int micropilot_sensor(HANDLE hFile, int data_req) {
    BOOL success;
    DWORD numWrite, numRead;
    char str[10] = {0};
    char rec_str[120] = {0};
    char *s;
    char *tempstr;
    int i, j, data;

    strcpy(str, "SSSS"); // Sensor Report
    WriteFile(hFile, str, strlen(str), &numWrite, 0);

    for (i=0; i<4; i++) // Sieving out unwanted info
    {
        success = ReadFile(hFile, rec_str, 120, &numRead, 0);
        if (!success)
            ErrorHandler("In ReadFile", GetLastError());
    }

    // Reading data from the serial port.
    success = ReadFile(hFile, rec_str, 120, &numRead, 0);
    if (!success)
        ErrorHandler("In ReadFile", GetLastError());

    j = 1;
    tempstr = rec_str;
    s = strtok(tempstr, " .-");
    while (j<10) // since max data required is 9
    {
        if (data_req == j) data = atoi(s);
        j++;
        s = strtok(NULL, " .-");
    }

    /*
    Complete list of Sensors data
    1  hdg
    2  pitch
    3  roll
    4  yaw
    5  xd
    6  yd
    7  spd
    8  palt
    9  agl
    10  alt
    11  el
    12  ai
    13  rdr
    14  th
    15  rollc
    16  pitchc
    17  yawc
    18  temp
    */
    FlushFileBuffers(hFile); // Clears the buffers for hPilot
    EscapeCommFunction(hFile, CLRDTR); // Clears the DTR (data-terminal-ready) signal
    return data;
}
int micropilot_gps(HANDLE hFile, int data_req)
{
    BOOL success;
    DWORD numWrite, numRead;
    char str[10] = {0};
    char rec_str[256] = {0};
    char *s;
    char *tempstr;
    int i, j;
    int data;

    strcpy(str, "GGGG"); // GPS Report
    WriteFile(hFile, str, strlen(str), &numWrite, 0);

    for (i=0; i<4; i++) // Sieving out unwanted info
    {
        success = ReadFile(hFile, rec_str, 255, &numRead, 0);
        if (!success)
            ErrorHandler("In ReadFile", GetLastError());
    }

    // Reading data from the serial port.
    success = ReadFile(hFile, rec_str, 255, &numRead, 0);
    if (!success)
        ErrorHandler("In ReadFile", GetLastError());

    j = 1;
    tempstr = rec_str;
    s = strtok(tempstr, " .-");
    while (j<11) // since max data required is 10
    {
        if (data_req == j) data = atoi(s);
        j++;
        s = strtok(NULL, " .-");
    }

    /*
    Complete list of GPS data
    1  VE
    2  VN
    3  VV
    4  hdgT
    5  hdgM
    6  Vel
    7  Pos E (degrees)
    8  Pos E (decimal)
    9  Pos N (degrees)
    10 Pos N (decimal)
    11 Time
    12 GPS Status
    13 DGPS Status
    */
    FlushFileBuffers(hFile);                     // Clears the buffers for hPilot
    EscapeCommFunction(hFile, CLRDTR);          // Clears the DTR (data-terminal-ready) signal

    return data;
}