An Autonomous Missile Defense System

Daine Richard Lesniak
Computer Science and Software Engineering Department
University of Wisconsin – Platteville
Platteville, WI 53818
lesniakd@uwplatt.edu

Douglas J. Hickok
Computer Science and Software Engineering Department
University of Wisconsin – Platteville
Platteville, WI 53818
hickokd@uwplatt.edu

Kristopher Whisler
Computer Science and Software Engineering Department
University of Wisconsin – Platteville
Platteville, WI 53818
whislerk@uwplatt.edu

Michael C. Rowe, Ph.D.
Computer Science and Software Engineering Department
University of Wisconsin – Platteville
Platteville, WI 53818
rowemi@uwplatt.edu

Abstract

This paper describes an independent study project at the University of Wisconsin – Platteville to design and implement an autonomous paper airplane/missile defense system. The paper airplane defense system is organized into three subsystems that detect and locate, track and target, and control a turret and shoot a compressed air gun.

This paper describes the details of design, development, integration, and calibration of the software components as well as the specification and construction of the hardware components of this system.

Introduction

Paper airplanes have become a public nuisance in some regions of the Midwest. This paper describes the design and development of an autonomous paper airplane/missile defense system using an off-the-shelf USB web cam, an air compressor, some hobby store electronic components, and a top-secret interceptor blowgun dart.

The system is composed of three subsystems that include:

- Airplane detection and location subsystem,
- Airplane tracking and targeting subsystem, and
- Turret control and shooting subsystem.

The next sections describe each of subsystem. Following the description of subsystems is a discussion of the integration and calibration of the system as a whole.

Project Goals

The inception of the automated missile defense project was inspired by several goals. The authors of this paper form a group of individuals at UW-Platteville who has worked on projects together and who are interested in machine vision and automated control. It was the intention to gain experience with a system that combined aspects of machine vision and automated control in an interesting and challenging manner. Automated missile defense was the natural choice given these goals.

Another goal for the project was to demonstrate the feasibility of a machine vision and automated control system for a class project. Too often, it seems the only options for class projects are scheduling programs and tracking systems, and GUIs tied to databases. While these are still challenging and important projects, variety is the spice of life. It is important to have project options with different facets, this is key to maintaining student interest. Showing the feasibility of projects such as this was the reason for making this project an independent study, thereby putting the project in an official curriculum framework.

Demonstration of the feasibility for projects of this nature is twofold. First, it must be shown that there are few entry barriers to these systems, especially those that would require large budgets (over $200) and timescales not permitted by a university program. The use of *Commercial Off The Shelf* components, or COTS, for both hardware and software is a major tool in keeping entrance barriers low, as COTS use lowers both cost and time to completion of a project. The second aspect to demonstrating the feasibility of such projects is showing that they can indeed be successful on an introductory basis. We found that this can be accomplished by applying the proper constraints to the problem, such as a white target on a black background, and keeping the targets movement fairly two dimensional.

The University of Wisconsin Platteville has a yearly Engineering Expo, where middle and high school students from around Wisconsin and neighboring states can come and see projects that have been developed by various technology and engineering majors. Engineering Expo is a method by which interested students can be exposed to technology and engineering, and help them with the problem of choosing an area for future study. The various majors use this event for shameless self-promotion, and field the flashiest and most interesting projects they have to offer. The automated missile defense project is one such project, and promises to be a great recruitment tool for the University of Wisconsin Platteville's computer science and software engineering programs.

Airplane Detection and Location Subsystem

The airplane detection and location subsystem is composed of a standard USB web cam that captures image frames and a software process that detects the presence of a paper airplane and locates its leading most pixel. To simplify this task we used a black background and white paper airplane. Processing speed is critical to make this process fast enough to intercept a paper airplane in flight. To that end, we capture frames in an easily processed form using operating system APIs and then process the resulting objects using built-in C# classes. The leading pixel of an airplane and a time stamp is forwarded to the tracking and targeting subsystem.

Image and Time Capture

The image capture device used for the automated missile defense project is a standard off the shelf USB web cam. Using a USB web cam furthered our goals of demonstrating the feasibility of low entrance barrier machine vision projects, and provided many technical advantages as well. USB web cams are designed specifically to be integrated with personal computers, and provide not only a convenient physical connection, but also driver support allowing the PC's operating system, and thereby our program, to utilize the web cam. Capture rates of off the shelf USB web cams are also quite adequate for shooting down paper airplanes; as quick as they are they are not so quick that they can escape the 30 frames per second provided.

The software side of image capture involves the avicap32 Window's API [1] and the built in classes in Visual Studio .Net C# [2]. The use of a standard API and built in classes allowed for rapid and successful low cost (students get a great discount on Visual Studio .Net) development of the image capture software. To capture an image, the automated missile defense program first makes an API call to capture a frame from the web cam into the Window's clipboard, thereby storing the data in RAM. Next the program creates a bitmap object from the data in the clipboard, this bitmap object is built into C# .Net and is very easy to process; making it the ideal end object of the image capture.

For proper target prediction, a time must be associated with every frame capture. A naïve approach would be to assume the time between each capture to be the same and predict the target a certain number of frame captures into the future. This assumption is incorrect however; due to the fact that the missile defense software runs on a standard PC with a multitasking OS. The strategy employed by the automated missile defense program is to capture a time stamp associated with each image, and use each capture's time stamp to maintain a running time line through the programs operation.   If we had chosen a real-time operating system the timestamps may not have been as important.

The automated missile defense system views and stores time as a continuous stream of milliseconds, with the start of the program at millisecond zero. To increment the number of milliseconds that have progressed since the program has started the automated missile defense program makes use of the System.DateTime class available through C# .Net and more importantly the System.DateTime.Now property which returns a System.DateTime representing the current date and time on the computer. By maintaining the System.DateTime of capture ($N - 1$) and comparing it with System.DateTime.Now of N we are able to tell how many milliseconds have passed between captures and update the programs time line accordingly.

Plane Detection

Plane detection is accomplished by maintaining a *trip grid,* a set of points that are to act as triggers to determine whether a captured image contains a plane or not. With each captured image, the trip grid points are checked one by one to determine if the RGB value at that point in the captured image is close enough to white to be considered part of a plane. If we go through the trip grid without finding a point belonging to a plane, the skies are considered clear and the capture process continues. Otherwise, if a single trip grid point is found to be white, the trip grid scanning process stops and the point found is used as the starting point for plane tip location process.

The points in the trip grid were selected such that they would be dense enough to detect a standard sized paper airplane and yet few enough to greatly increase the process of scanning. This balance had to be struck due to the fact that we certainly did not want to have a plane evade our detection, but could not afford scanning all 76800 points contained within a captured image. Striking this balance was accomplished visually. We started capturing images and displayed each image captured on the screen with the trip grid points we were currently evaluating colored bright red. We then adjusted the trip grid until we had a minimum number of points that would leave no gaps for a plane to pass through. By doing this, we were able to go from checking all 76800 points to checking, at most, only 3072 points.

In order to make this project a successful introduction to machine vision, certain appropriate constraints were placed on the project. As mentioned earlier, the system detects white planes against a black backdrop. This simplifies but does not completely solve the issue of plane detection, there is the question of what threshold an RGB value must have to be considered white. This problem was especially prevalent at edge points of a plane, where slightly grayer RGB values are observed. To address this, the automated missile defense system employs a modifiable RGB threshold and an additional constraint that there shall be direct lighting on the background. Direct lighting greatly increased the crispness of the edges, and made edge and tip detection much more consistent.

Tip location

In order to predict and target an object a characteristic position for the object must be determined and consistently located and utilized. While this is often done through a method such as clustering our choice of paper airplanes as targets gave us an easily located and natural choice for such a characteristic point: the tip. Each time a plane is detected, the image processing task needs to find the tip of the paper airplane, which along with the time of the capture is the only information to persist from capture to capture.

Locating the tip starts at the position in the trip grid determined to belong to a paper airplane. From this start point, points progressively farther to the left are evaluated until an edge is located, or in some cases the tip itself. If an edge is located rather than the tip from the initial leftward scanning, as is usually the case, the edge is followed left until the tip is located. These simple criteria to determine edges and the tip of the paper airplane were made possible largely by constraining the flight of the paper airplane in one direction perpendicular to the image capture device.

Edge and tip detection were both performed using simple comparisons of nearby points. Edges are detected by comparing points adjacent in the Y-axis, and should one point be white and one non-white, and edge is considered to be detected at the white pixel. The tip of the paper airplane is detected at the first point found in the tip search that matches the criteria of having non-white points three pixels above, three pixels below and three pixels to the left.

The system view of airplane detection and location is summarized in Figure 1.

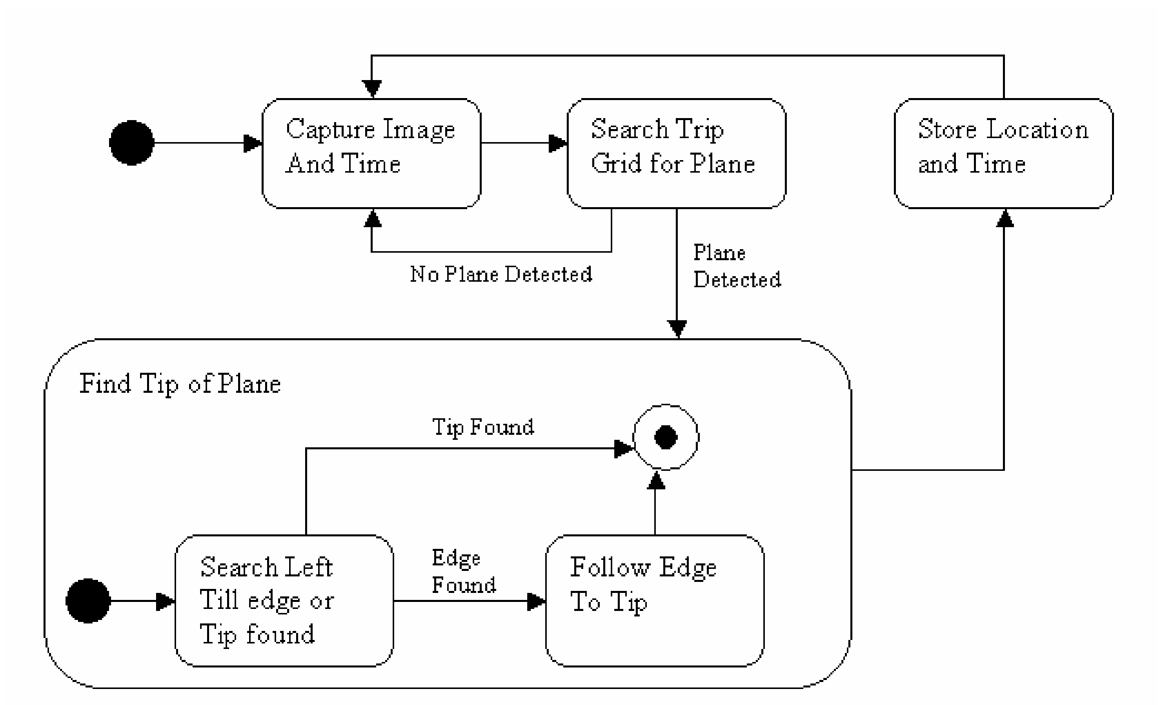

Figure 1: State Machine View of Airplane Detection and Location

Airplane Tracking and Targeting Subsystem

The tracking and targeting subsystem of the automated missile defense project uses three plane locations with associated times to estimate the likely path of the paper airplane. This likely path is targeted using a coordinate to shoot at and a time at which to shoot, both of which are provided by the tracking and targeting subsystem for use in the turret control and shooting subsystem. Along with the target location and time to shoot, the tracking and targeting subsystem provides a "best guess" estimated target point when it has collected two plane locations. This allows the turret control and shooting subsystem to starting moving to a likely targeting location early and get the jump on the paper airplane. It is also the responsibility of the tracking and targeting subsystem to take into account various calibration factors and to translate from the image captures coordinate system. The image capture subsystem uses a 320 x 240 coordinate grid, whereas the turret control and shooting subsystem uses a 256 x 256 coordinate grid.

There are three main stages to the target prediction process, each of which requires the latest XY coordinate of the plane and the time associated with the image frame. Which stage the target prediction is in depends on whether it is the first, second, or third coordinate and time given to the target prediction subsystem. The subsystem's inputs are coordinates and times, and its outputs are the estimated target coordinate, the actual target coordinate, which stage it has most recently completed, and the time at which the air gun should fire at the target coordinate.

The first stage of the target prediction process starts with initializing the data to starting values, clearing information used to predict the target, and setting the time at which to fire to infinity. The first stage stores the coordinate and associated time passed to it by plane location subsystem.

When the target prediction subsystem is passed a point in the second stage, it saves the point and uses the two known locations and times to estimate a targeting coordinate. This estimated targeting coordinate is passed to the turret control subsystem, which allows it to start positioning itself in roughly the right shooting position. While a "best guess" target location is estimated, it is important to note that the second phase of the target prediction process does not calculate a fire time, as the system is not meant to fire at this point. The estimated target is found using the calculated velocity of the paper airplane given the first two points and times. While this is a good estimate, acceleration must be taken into account, requiring a third coordinate and time, and therefore, one more stage in the target prediction process.

The third and final stage of the target prediction process stores the third coordinate and time, and uses this data in conjunction with the first two to calculate the target point and the time at which to fire. This target point is more accurate than the estimated target point as it is calculated using the start point, the velocity, and the acceleration. Also, rather than calculating the target based on a straight-line path, the targeting subsystem plots the planes path with a curve. This target point is the point at which the plane will be at time (now + lead-time). This time is also the fire time that the target prediction subsystem sets.

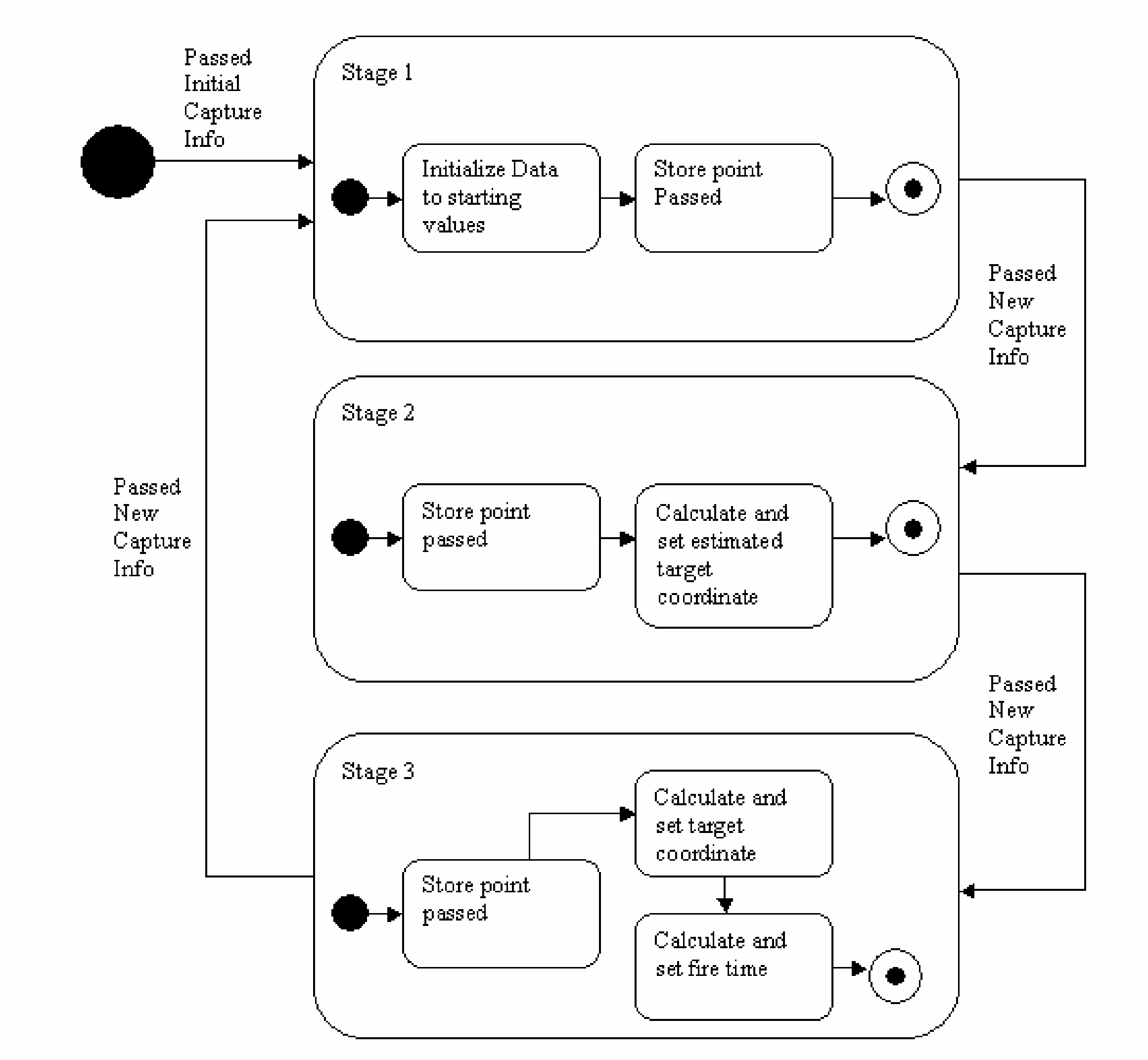
Passed Initial Capture Info
Stage 1
Initialize Data to starting values
Store point Passed
Passed New Capture Info
Stage 2
Store point passed
Calculate and set estimated target coordinate
Passed New Capture Info
Passed New Capture Info
Stage 3
Calculate and set target coordinate
Store point passed
Calculate and set fire time

Figure 2: State Machine view of the Tracking and Targeting subsystem.

Turret Control and Shooting Subsystem

The idea behind the design of the turret hardware and hardware control subsystems was to have the hardware perform like any other PC peripheral. That is to say, the turret hardware would receive commands from the host computer and perform whatever actions the received commands dictate; aiming and firing the cannon.

The turret subsystem is comprised mainly of two common hobby servos, a small array of electrically actuated pneumatic valves, a controller board, and an ordinary ATX power supply. With an unloaded movement rate of sixty degrees in fifteen hundredths of a second, the servos are the slowest part of the entire system and their movement rates needed to be taken into account in the target prediction portion of the software.

The micro controller that was chosen for this project was the Acroname BrainStem GP 1.0 [3]. This particular controller board was chosen not only for it's servo controlling capabilities for aiming the turret, but also for its digital output ports for opening the air valves to fire the cannon. Furthermore, with its robust set of built in commands, writing the code to interact with the BrainStem was quite easy. The only roadblock that was encountered was properly interpreting the packets received from the BrainStem for indication of a healthy connection between the host computer and the controller. This was solved with some minor digging through the documentation [3] and example programs that were available to Acroname customers.

Since all the programming for this project done using C# and Microsoft .Net 2003, some special measures had to be taken so that the turret control software on the host computer could communicate with the BrainStem controller. The issue with the .Net architecture is that it does not readily support serial communications [4]. To solve this, the MSCOMM OCX from Visual Studio 6 [5] was added to the project.

For the actual communications between the host computer and the BrainStem module, messages are sent through a serial connection. The message packets are simple byte arrays of up to eight bytes. The contents of the packets, starting at the lowest byte, is the IIC address of the module, the number of bytes in the packet, the byte value assigned to the command to be executed, and each additional byte is a parameter for the command [3]. Before any other commands can be successfully sent to the BrainStem module, the host machine must maintain what Acroname calls a heart beat. This is basically a handshaking loop between the host computer and the BrainStem to ensure that there is a good connection between the two. The loop starts with the BrainStem sending a heart beat up packet, which triggers a response from the host computer. Once the BrainStem receives the reply, it sends out a heartbeat down packet, which is similarly replied to by the host computer. This loop is repeated for as long as the system is running. For as long as this loop is maintained, other command packets can be sent to the BrainStem. As far as the command for moving the servos, two parameters are passed, one for the servo port and one for the position for the servo to move to. Since the normal range of a servo is broken up into 256 positions, the command requires a byte value between 0 and 255, inclusive, and the BrainStem then uses that value to output pulse-width modulated signal to the corresponding servo port.

As stated earlier, the turret hardware is being powered by a standard ATX power supply. This actually is perfect for this particular combination of hardware, as the air valves require twelve volts, the BrainStem needs between six and twelve volts, and the servos require five to six volts, all of which can be supplied by any PC power supply. An ATX power supply was used due to its availability (junk is cheap) however, to safely enable the power supply, a *debounce* circuit had to be constructed and attached to the appropriate wires [6]. *Debounce* refers to the smoothing of a series of spikes and sages that occur when a circuit is closed.

System Integration

The three subsystems were originally developed independently, with proper attention paid to what interfaces were required. While this allowed rapid development of the subsystems, it added an integration phase to the project. The order of integration is as follows: first, the image-capture and plane detection subsystem was integrated with the target tracking and detection subsystem. Next, the combination of these two subsystems was integrated with the turret control and shooting subsystem to make the fully automated missile defense system.

Integration in two steps rather than all at once reduced the risk involved and allowed for testing of the integration of the target tracking and prediction subsystem with the image capture and plane detection subsystem. This was accomplished by having an image displayed that showed not only the point that was being considered the tip of the airplane, but also the point that was being considered the target. Invariably the plane would indeed go through the target point, signaling that the system was ready to have the turret control and shooting subsystem integrated.

Calibration

In order for this system to be successful, it was important to realize there were many factors that could vary the timing and flight characteristics of both the target paper airplane and our interceptor blowgun darts. To address these, the automated missile defense project has many variable attributes that allow for on the fly calibration of the System. Some of these variables are the lead-time for the target prediction, the XY adjust for the target prediction, the trip grid for plane detection, and the threshold RGB value to determine if a pixel is white.

The plane detection relies on the modifiable variables for the density of the trip grid and the threshold value for white. Though these both have default values that have been determined experimentally, they are set up in such a way that they are easy to modify should conditions change. The threshold white value is actually specified by its red, green, and blue component, making it very flexible. Instances in which the threshold white would need to be changed would be based on paper used and rooms with unusual lighting or even by time of the day or night for which the systems own direct lighting could not quite compensate. Altering the trip grid density would become necessary should the paper size and therefore airplane size change, or if the distance were to change dramatically between the automated missile defense system and the backdrop.

Target tracking and prediction incorporates two modifiable variables: lead-time and XY adjust. Lead-time is to accommodate the speed of the servos, speed to the top-secret missile, and the speed of paper airplanes. Like the trip grid density, lead-time is set to a default value, but is easily calibrated to changing conditions. The XY adjust is a catch-all modifiable variable that provides flexible calibration for anything from mechanical changes due to wear and tear to unique wind conditions in the demo room. XY adjust is used to alter the target coordinate from what was originally calculated.

Summary

To summarize, this paper describes how a standard web cam, readily available software, some hobby store electro-mechanical devices, highly advanced top-secrete interceptor blowgun darts, and a roll or two of duct tape can be used to produce an autonomous real-time paper airplane defense system. This system should prove to be an excellent recruitment tool for UW-Platteville's computer science and software engineering program for many Engineering Expos to come, and will hopefully serve as a blueprint for an interesting addition to the usual roster of class projects.

References

[1] Writing Video Capture Drivers and Applications for Windows 95 and Windows NT, http://www.nomadelectronics.com/VidCap/capture%20using%20vfw/CAPTURE.DOC, 1995.

[2] Product Information for Visual Studio .NET 2003, http://msdn.microsoft.com/vstudio/productinfo/

[3] Acroname BrainStem GP 1.0, http://www.acroname.com/brainstem/ref/ref.html

[4] Coad, N., "Serial COM Simply in C# " http://www.devhood.com/tutorials/tutorial_details.aspx?tutorial_id=320

[5] Mackenzie, D., "Coding in the Blue Glow", Visual Studio 6 MSCOMM OCX, Microsoft Developer Network, 2003, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncodefun/html/code4fun12102003.asp.

[6] *FryGuy*, "Using an ATX computer power supply to make a scalable multipurpose power supply", http://forum.ecoustics.com/bbs/messages/1/Using_an_ATX_computer_power_supply_to_make_a_scalable_multipurpose_power_supply-75902.doc