VISION-ONLY AIRCRAFT FLIGHT CONTROL

Christophe De Wagter[1]
Delft University of Technology, Delft, The Netherlands
Alison A. Proctor[2] *and Eric N. Johnson*[3]
Georgia Institute of Technology, Atlanta, GA, 30332

Abstract

Building aircraft with navigation and control systems that can complete flight tasks is complex, and often involves integrating information from multiple sensors to estimate the state of the vehicle. This paper describes a method, in which a glider can fly from a starting point to a predetermined end location (target) precisely using vision only. Using vision to control an aircraft represents a unique challenge, partially due to the high rate of images required in order to maintain tracking and to keep the glider on target in a moving air mass. Second, absolute distance and angle measurements to the target are not readily available when the glider does not have independent measurements of its own position. The method presented here uses an integral image representation of the video input for the analysis. The integral image, which is obtained by integrating the pixel intensities across the image, is reduced to a probable target location by performing a cascade of feature matching functions. The cascade is designed to eliminate the majority of the potential targets in a first pruning using computationally inexpensive process. Then, the more exact and computationally expensive processes are used on the few remaining candidates; thereby, dramatically decreasing the processing required per image. The navigation algorithms presented in this paper use a Kalman filter to estimate attitude and glideslope required based on measurements of the target in the image. The effectiveness of the algorithms is demonstrated through simulation of a small glider instrumented with only a simulated camera.

Nomenclature

Δ	angle between the camera centerline and the line through the center of the window
δ	actuator deflection
γ	track angle
θ	pitch angle
φ	roll angle
ψ	heading angle with respect to the desired flight path
$C_{l\delta a}$	aileron effectiveness
$C_{m\delta e}$	elevator effectiveness
$C_{n\delta r}$	rudder effectiveness
C_{INT}	model error integration coefficient
d	distance to the window
dh	vertical off-track position
dt	discrete time step
dy	lateral off-track position
EKF	Extended Kalman Filter
FOV	Field of View
GN&C	Guidance Navigation and Control
S	width of window
I	Grayscale Image Intensity
II	Integral Image of Pixel Intensities
k	iteration number

[1] Graduate Research Assistant
Email: C.deWagter@student.tudelft.nl
[2] Graduate Research Assistant
Email: alison_proctor@ae.gatech.edu
[3] Lockheed Martin Assistant Professor of Avionics Integration
Email: Eric.Johnson@aerospace.gatech.edu

px	horizontal position in pixels
py	vertical position in pixels
RI	Region of Interest
u	forward velocity
UAV	Unmanned Aerial Vehicle

Subscripts

W	window
mid	midline of the image
des	desired
a	aileron
r	rudder
p	in pixels
trim	trim position
unroll	unrolled
H	horizontal component
V	vertical component

Introduction

Autonomous mechanical flight has historically utilized a wide array of sensors to estimate the precise state of the vehicle. This is a contrast to what is seen in nature where a bird uses senses similar to our own to navigate and maintain a predetermined course. Although, other senses are utilized vision appears to be dominant. These ideas have prompted this research, which in contrast to most previous autonomous vehicles, uses camera images only to estimate the vehicle's state.

The task chosen to demonstrate the algorithms is to find an open window on the side of a building using vision, and fly autonomously through the window. This straight forward objective provides a framework for developing and demonstrating the possibilities of GN&C using only vision.

There are two main bodies of previous work in this area. The first research area is in the stability aspects of vision based control. Until recently this research has dealt principally in the development of vision based control laws for manufacturing robots; however, the increasing desire to launch fleets of UAV's which operate as a team has ignited new research in vision-based feedback control for aerial vehicles, as described in [1]. The second, and most prevalent, area is research for developing autonomous vehicles which utilize vision in conjunction with other sensors. In this case the vision is used for trajectory planning as described in [2], or for precise determination of relative position as described in [3].

This paper focuses on utilizing vision alone to determine the state of the vehicle, the flight plan, and generate control commands. This differs from the more common target seeking missile, which also uses traditional sensors. A completely vision based flight control system has the advantage of having a very simple hardware configuration, making it a low cost solution for an autopilot system. This also makes it attractive as a completely isolated backup system.

Flight Configuration

There are four main components needed to fly: an aircraft, an Image Processing and Flight Control Computer (FCC), a Ground Control Station (GCS), and a video camera. The final configuration of these systems was chosen in order to maximize the reusability of the system for this and other projects.

Aircraft

In order to achieve the objective, the aircraft needs to be small enough to fit through a realistically sized window; this limits the payload capacity of the aircraft. Based on this constraint it was determined that the most suitable aircraft would be a small glider. By eliminating the engine, the payload capacity of the aircraft was maximized, and the logistics were minimized.

Image Processing and Flight Control Computer (FCC)

In order to eliminate the need for customized hardware, the image processing and the flight control calculations are performed on remote computer located on the ground or on another aerial vehicle near the glider. This eliminates the size constraint for this component. The FCC uses a framegrabber to obtain still images from the video camera. By utilizing a framegrabber, the images

are sent directly to the FCC's memory at a high rate. The framegrabber chosen for this is the Sensoray Model 311, which can provide images to the FCC at 30 frames/sec. Once the images are processed servo control signals are generated, encoded and sent back to the glider using a standard R/C transmitter.

Ground Control Station (GCS)

The GCS is a standard laptop that runs either Windows or Linux. The GCS software generates an OpenGL based display of the vehicle's position estimate as well as the location of the window, and the results of the image processor. The GCS communicates with the FCC via wireless Ethernet; this maximizes the flexibility of the system, allowing the GCS operator to monitor the progress from a remote location. Through this data link, the GCS operator can send commands to the aircraft, and the FCC reports state estimates and the results of the image processor back to the ground.

Camera

The camera chosen was a small 350 line CCD video camera, and the video is sent to the FCC via video transmitter.

Figure 1 shows one support configuration for the glider, where the FCC and video receiver are carried onboard a second autonomous vehicle. Although, this configuration is the most flexible, determining a position and velocity estimate for the glider is more difficult, and the glider control must be sufficiently accurate such that the window remains in the field of view of the camera.

Figure 1 Support Configuration for the Glider

Machine Vision

The problem of tracking a moving window at a high rate in real-time is not as easy for a machine as it is for a living creature. The image analysis routines need to be robust, fast and precise. A strategy for tracking the window was developed based on [4], which uses Rapid Object Detection applied on face recognition.

High speed and high precision are opposing requirements, precision usually requires more processing time, whereas speed requires less. An approach to this dilemma was found in [4]. The idea is to consider every static image coming from the frame grabber as a separate picture, which is thought to contain the target. In order to find the position of the target window as fast as possible, every possible location of the window in pixels is considered, and assigned a score. If the score of the most likely location exceeds a certain threshold, the image is considered to be good and the location is recorded.

Integral Image

The image analysis uses pattern matching to detect the square window. High speed pattern matching is achieved by using an Integral Image of Pixel Intensities. Finding the average pixel intensity of a square region is then reduced to four array references, or a multiple of this for more complex patterns. The integral image is found as:

$$II(x, y) = \sum_{x' \le x,\, y' \le y} I(x', y') \qquad \text{Eq 1}$$

where I(x, y) is the grayscale image intensity and II(x, y) is the integral image. Each array location in the integral image represents the sum of the intensities of all of the pixels above and to the left of it. Calculating the integral image only requires one pass only over the original image with the value at each pixel location being:

$$II(x, y) = II(x-1, y) + II(x, y-1) - II(x-1, y-1) + I(x, y) \qquad \text{Eq 2}$$

Then the sum the intensities of any rectangle can be computed with four array references, using the sum

$$\begin{aligned} rect(x, y, w, h) &= II_4 + II_1 - II_2 - II_3 \\ &= II(x + w, y + h) + II(x, y) \\ &\quad - II(x + w, y) - II(x, y + h) \end{aligned} \qquad \text{Eq 3}$$

as demonstrated in Figure 2.

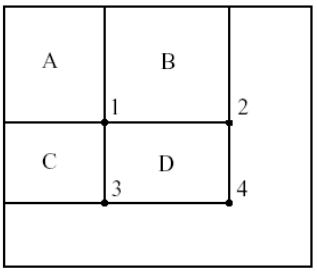

Figure 2 Feature Definition in an Integral Image

Cascade of Classifiers

Assigning a probability to every possible pixel location of a window can be a time consuming problem. By utilizing a Rejective Cascade of Classifiers, as shown in Figure 3 this process was made considerably faster.

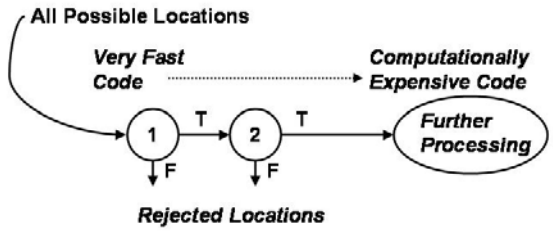

Figure 3 Rejective Cascade of Classifiers

The classifier begins by scoring each pixel against a simple criterion using an efficient algorithm. The location is then either rejected or accepted based on the score. If the location is accepted for further evaluation, a second classifier is used which has a more specific criteria and a more computationally expensive algorithm. This process is then repeated until the location is rejected or all of the levels of the cascade have been completed. This approach allows highly probable candidates to be examined in detail while highly improbable candidates are eliminated immediately.

A well designed cascade can tremendously speed up the processing time of an image, while achieving a high positive detection ratio. However, care must be taken when designing each stage of the classifier to ensure that the correct location is not eliminated early.

The first stage of the process is to determine a region of interest (RI) in the image. The RI is a subset of the real image where the target is thought to reside. The size and location of this region are determined from location of the target in the previous image and the estimate of the vehicle's current state. During certain stages of flight this can dramatically decreases the processing time as it can rule out a large portion of the image before the integral image is even created.

The second stage of the process is to build an integral image of the RI and apply the classification scheme shown in Figure 4. Since, during daylight hours, an open window is generally darker than the wall on which it is contained, the criteria were chosen to find features which have a dark rectangular inner region with a lighter border.

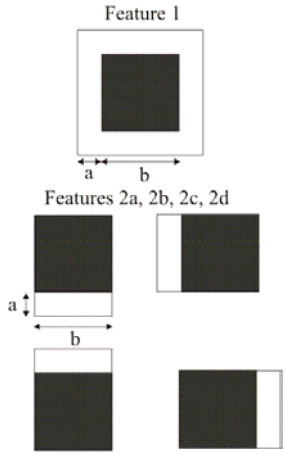

Figure 4 Cascade of Features

The first classifier has only one comparison. It checks if the average intensity of the inside of the feature is darker than the average intensity of the border. This simple and robust classification usually eliminates 70 percent of possible window locations with minimal chance for rejecting a true window location. If the location passes, it's score is the ratio of intensity of the inner region and the border, with the smallest score being the best.

If the location passes the first test, the second classifier is applied. The second classifier utilizes the same window characteristic, but checks it more

vigorously. It has four comparisons and checks whether all four sides of sides of the border are all individually darker than the center of the window. This removes a lot of false positives, like trees and other objects, which pass the first classifier. The features previous score is then multiplied by the ratio of intensities again to get the final score for that feature. The feature with the lowest score is chosen, and the position of its centroid in pixels, px and py, are stored as the final window location.

Additional filtering, which looks for the correct window orientation based on the state estimate could have been performed on the few features which passed the second level of the cascade. However, during development, it was determined that by utilizing these two filters the quality of recognition was satisfactory.

Size and Rotation

Once the window location is known, the size and rotation is measured, as shown in Figure 5. The horizontal size of the window in pixels, S_{Hp}, is measured by simple thresholding using the intensity levels found in the cascade. A region around the features centroid known to be larger than the window is selected and all of the pixels whose intensity is less than the average intensity of the border are counted. This method is easy to implement and it proved to work well for homogenous windows.

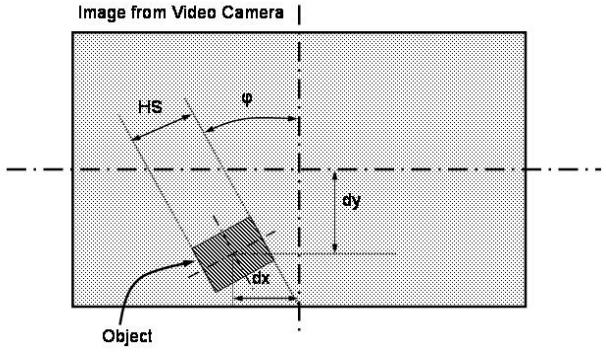

Figure 5 Image Measurements

Since the window may take up as few as 15 pixels, it was necessary to utilize a very rough measurement of the windows rotation. During the size measurement procedure, the farthest point from the center in each quadrant was designated the corner. Since each corner should be at a 45 degree angle from the x axis of the window, the average deviation from this gives a measure of the windows rotation angle, φ_W, as long as the aircraft roll remains less than 45°.

Navigation and Control Strategies

Once the image analysis has determined the position of the window in the video frame, the control commands can be derived. Several strategies for controlling the aircraft were investigated. The simplest being to simply keep the window centered in the camera frame. This turned out to have a number of problems, which are discussed in the following section. These investigations led to the final solution which decouples the lateral and vertical movements and utilizes an Extended Kalman Filter (EKF) to estimate the vehicle state. This allowed a more sophisticated controller to be used.

Keeping the Target Centered

The simplest idea was to have a controller keep the window centered on the camera image at all times. The elevator would control the vertical deviations and the ailerons would control the horizontal. However, this setup has shown stability problems in the lateral navigation.

The issue is that the position of the window, when not centered, is highly affected by the roll angle of the aircraft, φ. Suppose, for example, there is a lateral disturbance, and the window ends up right of the center. The lateral controller would then turn right to compensate. This implies a right roll angle, which will have the window move upwards. The vertical controller would then compensate, and will start aiming higher than the actual window.

Uncoupled roll-position interaction

A better method is to uncouple the lateral and vertical roll interaction by removing the roll movement of the camera. This is easily achieved, once a reliable value of φ is available. The unrolled target location becomes:

$$\begin{aligned} px_{unroll} &= (px - px_{mid})\sin(\phi) \\ &\quad + (py - py_{mid})\cos(\phi) \end{aligned} \qquad \text{Eq 4}$$

$$py_{unroll} = -(px - px_{mid})\sin(\phi) + (py - py_{mid})\cos(\phi) \quad \text{Eq 5}$$

Since all of the measurements are made in pixels and all of the state estimates are in a local coordinate system, it is necessary to convert between the two systems. In doing so it is helpful to define a scale factor which relates the pixel width of the image to the field of view.

$$SF = \frac{\text{width of image in pixels}}{\tan\left(\frac{FOV}{2}\right)} \quad \text{Eq 6}$$

Since images usually have a different number of pixels and a different FOV in the horizontal and vertical directions, this scale factor can be different for the two directions.

Simply using the uncoupled vertical and horizontal displacement angles to determine the controller commands can give good results, but is sensitive for camera angle errors or crosswind. The problem can be seen in Figure 6, where the unrolled angle from the centerline of the camera to the line through the window is Δ_H, which can be approximated by:

$$\Delta_H = \frac{(px_{unroll} - px_{mid})}{SF_X} \quad \text{Eq 7}$$

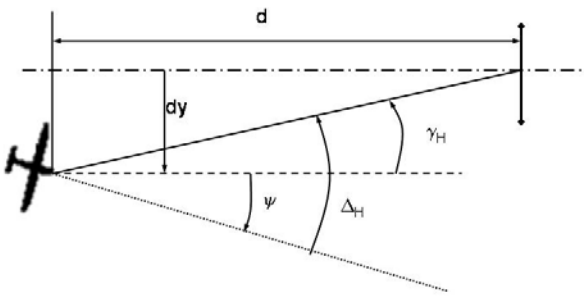

Figure 6 Off-track and Angle Measurements

Only one measurement for lateral navigation, Δ_H, is feasible from the images. However, this measurement really incorporates ψ and the track angle, γ_H, where γ_H arises from dy. In addition to the fact it is not possible to distinguish the contributions of these two parameters, their gains can vary a lot during the flight. When the vehicle is far from the target, Δ_H is sensitive to changes in ψ and less sensitive to changes dy. On the other hand, when it is close, dy can cause a large error in Δ_H compared to ψ. This creates a difficult gain scheduling problem, especially, when precise distance measurements are not available.

Therefore, it seems that it is important to estimate dy and ψ as well as φ. However, the crosswind also needs to be estimated with this measurement.

Navigation and Controller Design

This section describes the design of the Extended Kalman Filter (EKF) which is described in [5]. There are two main parts to an EKF, a nonlinear process model and a nonlinear measurement model. By comparing the real measurements to measurements which are predicted by the model, it is possible to generate a gain matrix, called the Kalman gain, which combines the measurements and the information from the dynamic model in such a way as to minimize the error covariance of the final solution.

An EKF requires an initial guess for each of the state variables and covariance information about the process model and about each of the individual measurements. In order for the EKF to converge quickly, it is important to have accurate initial conditions and to wisely choose covariance information.

The following measurements are retrieved from each image during the flight:

- Horizontal Window Size
- Window Rotation
- Horizontal position of the Window
- Vertical position of the Window

The filter then uses this data to estimate following information about the state of the vehicle relative to a window of known size and location:

- Roll Angle
- Distance to the Window and Vehicle Speed
- Off-track Position and Yaw

Dynamic Model of Glider

The model of the glider is a very important part of the filter, as it gives the filter its required

precision. On the other hand, obtaining values for a model can be a difficult job, and if different gliders are used, the model will have to be updated over and over. Therefore, the choice was made to keep the model as simple as possible. The glider model in the navigation filter is propagated forward in time using Euler integration and is completely described using only the following five parameters:

- $C_{l\delta a}$ Aileron effectiveness
- $C_{m\delta e}$ Elevator effectiveness
- $C_{n\delta r}$ Rudder effectiveness
- δ_{trim} Elevator trim setting
- C_{INT} Model error integrator coefficient

Each portion of the model is described in the sections that follow.

Roll:

Rolling motion is described with equation

$$\phi_{k+1} = \phi_k + C_{m_{\delta a}} \delta_{a_k} dt \qquad \text{Eq 8}$$

where dt is the discrete time step for the update. Choosing the model noise covariance wisely, this simple model gives a satisfactory propagation of roll.

Distance and Speed:

The next step is the distance measurement. Image analysis provides a value for the approximate size of the window, which can be turned into a distance. The vehicle distance model used extremely simple:

$$d_{k+1} = d_k - u_k dt \qquad \text{Eq 9}$$

where d is the distance to the window and u is the forward speed. In the filter u is modeled as a constant speed, but the estimation of u can change from measurement information.

Off-track position and Yaw:

The lateral camera angle, Δ_H, is the sum of the off-track angle, γ_H, and the heading error, ψ, as illustrated in Figure 6. It is difficult to accurately measure either one of these separately from a single image. One could go back to image analysis to make additional measurements; angles of the window border could be measured to attempt to calculate the off-track position, since it causes an angle to occur between the top and bottom line of the window. These angles, however, are too small to be useful for this application. Instead, the EKF will be used to estimate off-track distances during simulator flight tests, using only the four measurements described above. The coupled two-state model is:

$$dy_{k+1} = dy_k + \sin(\psi_k) u_k dt \qquad \text{Eq 10}$$

$$\psi_{k+1} = \psi_k + g \frac{\tan(\varphi_k)}{u_k} dt \qquad \text{Eq 11}$$

By using the EKF, it is possible to ignore the fact that ψ and dy are coupled and let the filter sort things out.

Vertical Flight Path Angle:

For the vertical flight path, there is a similar ambiguity to that seen in the lateral direction. The vertical measurement obtained encompasses both the pitch and an altitude error above the initial glide path. This issue is compounded when there is insufficient trim data for the elevator. However, it was determined that, by allowing the flight path to move with the aircraft and recalculating the desired pitch angle so that the glider intercepts the window, acceptable controller performance is achieved. Therefore, it is sufficient to model the pitch angle as:

$$\theta_{k+1} = \theta_k + C_{m_{\delta e}} (\delta_e - \delta_{trim}) dt \qquad \text{Eq 12}$$

Measurement Model

The measurement model in an EKF predicts what the measurement should be based on the state estimation predicted by the dynamic model.

Window Size:

The predicted window size in pixels is directly related to the distance the vehicle is from the window. The relation between window size measurement in pixels and distance state in feet is derived from:

$$\hat{S}_{H p_{k+1}} = \frac{S_{HW}}{2 d_{k+1}} \cdot SF_X \qquad \text{Eq 13}$$

where S_{HW} is the real dimension of the window in feet, and SF_X is the scale factor from Eq 6 for the horizontal direction.

Rotation Angle:

The predicted roll angle of the window is simply:

$$\phi_{W\,k+1} = -\phi_{k+1} \qquad \text{Eq 14}$$

Vertical and Horizontal Position:

In order to predict the vertical and horizontal position of the window on the image, the distance to the window dy, ψ, and θ need to be estimated.

Using small angle approximations, the lateral position of the window can be estimated as:

$$\begin{aligned} \hat{px}_{k+1} &= \frac{dy_{k+1}\cos(\psi_{k+1}) + d_{k+1}\sin(\psi_{k+1})}{d_{k+1}} \cdot SF_X \\ &\approx \left(\psi_{k+1} + \frac{dy_{k+1}}{d_{k+1}}\right) \cdot SF_X \end{aligned} \qquad \text{Eq 15}$$

Similarly, by making a small angle approximation and assuming that the altitude error is zero, the vertical position of the window can be calculated as:

$$\begin{aligned} \hat{py}_{k+1} &= \frac{dh_{k+1}\cos(\theta_{k+1}) + d_{k+1}\sin(\theta_{k+1})}{d_{k+1}} \cdot SF_Y \\ &\approx \theta_{k+1} \cdot SF_Y \end{aligned} \qquad \text{Eq 16}$$

Enhancing the Off-Track Estimation

Although this simple model gives good results when simulating flights, one thing that this model can not deal with is crosswind, or an error in camera alignment. Both of them can be compensated for by flying with a single crab angle.

However, the crab angle is an additional state to be estimated. Instead of adding the crosswind to the EKF state, it was estimated separately using the error in the model of ψ for input. Errors in the ψ prediction are mainly due to lateral wind, the camera not being aligned with the direction of flight or the roll angle having a bias. The approach developed in this paper is to integrate ψ, in order to estimate the necessary crab angle. The idea being that, since the heading error itself is being driven to zero by the controller, the remaining error would be the result of cross wind or misalignment. Then, by driving the heading to the crab angle instead, the errors in the heading will be minimized.

The Controller

The controller was designed around a standard proportional feedback controller. Using an integrator was investigated; however, it turned out to be too slow for the short high precision flight. Therefore, the integrator was replaced by a proportional term on the flight path angles. Since, the angle error is very small at large distances it does not influence the controller until the glider gets close to the window.

The lateral controller uses two loops. The first determines the desired roll angle φ_{des}, which is derived from dy and γ_H, and limited to ±30° to prevent 360° rolls.

$$\phi_{des} = K_P dy - K_d \phi + K_\alpha(\Delta_H - \gamma_H) \qquad \text{Eq 17}$$

Then it has a high gain controller for the desired roll angle:

$$\delta_a = K_R(\phi - \phi_{des}) \qquad \text{Eq 18}$$

The vertical controller sets the pitch angle to the vertical angle between the window and the camera centerline, Δ_V, minus the trim position.

$$\delta_e = K_P(\Delta_V - \theta_{trim}) \qquad \text{Eq 19}$$

This causes the glider to fly a curved path instead of a straight line.

Simulation Results

The system presented in this paper was tested in two ways. Firstly, the effectiveness of the image analysis was tested using multiple video fragments of different kinds of windows and square openings. The results were promising, with the image processor tracking the windows in real time for all the scenarios, as shown in Figure 7.

Figure 7 Image Processing Results

Once the image processor was tested, the whole system was implemented in a real time simulation environment described in [6]. This environment contains a dynamic model of the glider, simulated video transmission and datalinks, GCS software, and the navigation and image processing code discussed in this paper. For Closed Loop simulations, an scene generated representation of the camera view was used as an image. This tested the navigation filter and real time image tracking. The structure of the simulation is shown in Figure 8.

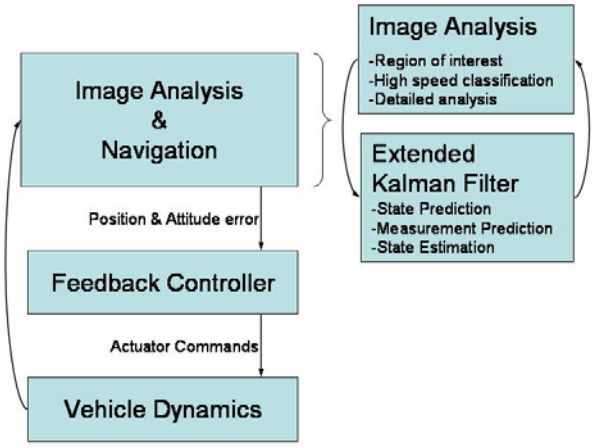

Figure 8 Simulation Architecture

Figure 9 shows the simulation scene generator. The blue aircraft represents the output of the navigation filter and the red aircraft is the position of the simulation model.

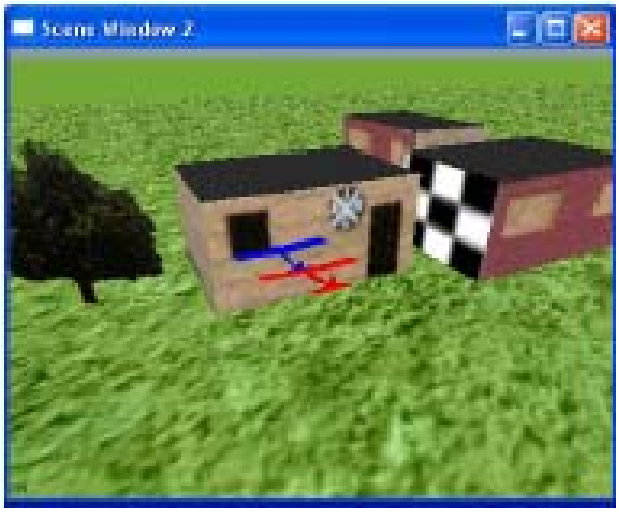

Figure 9 Screenshot of the Simulator Graphics

With the simulator it was possible to test all aspects of the EKF design. Figure 10 shows the distance and speed estimates obtained by the filter during a simulated flight and Figure 11 shows the results of the off-track distance and yaw angle estimations. In these cases, the glider must overcome an initial perturbation of 15° in φ; however, the crosswind is zero.

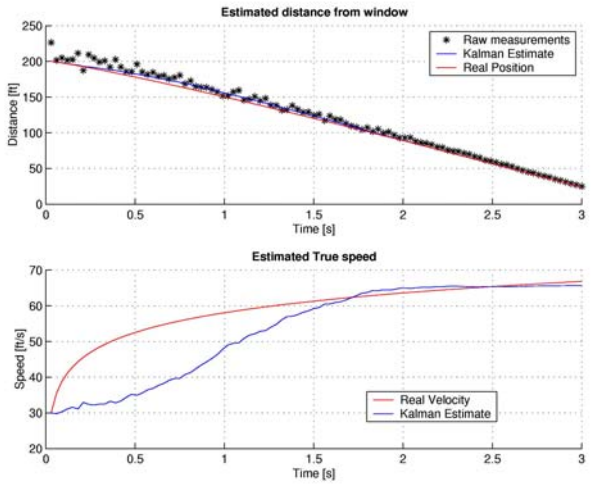

Figure 10 Distance Estimation to the Window without Crosswind

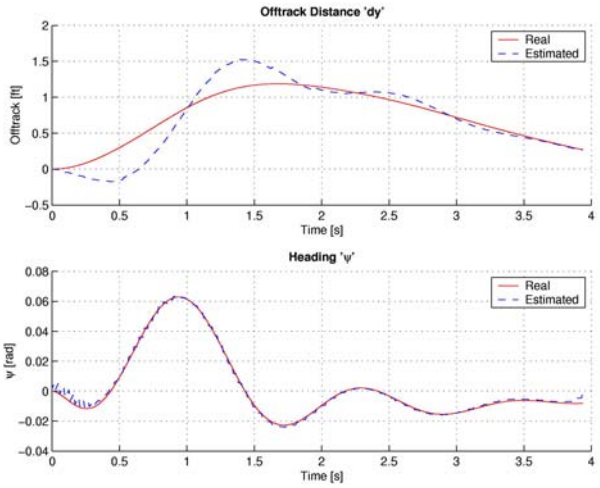

Figure 11 Off-Track Estimation without Crosswind

Figure 12 shows results from simulator run with heavy left crosswind and the same initial perturbation of 15° to the right in φ. In subplot b) one can see the difference between φ and the estimated $\hat{\varphi}$. In plot c) the glider finds to the necessary crab angle to compensate the crosswind. Plot d) has the off-track error in time. Notice the accurate trend with a scale error, since absolute measurements are not available.

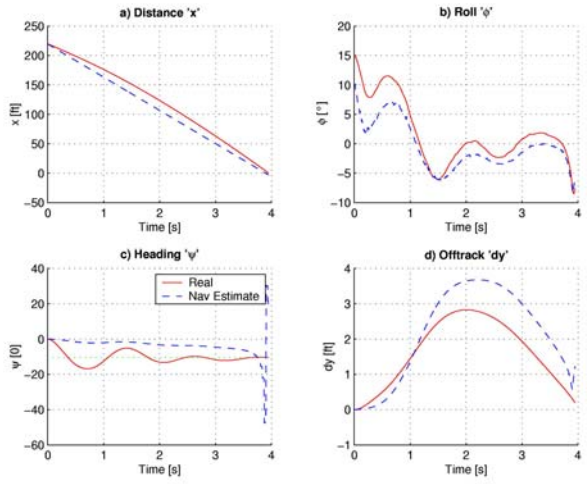

Figure 12 State Estimations with 9ft/sec Crosswinds

Figure 13 shows the results when the initial perturbation is in the same direction as the crosswind would generate. The filter is then forced to believe the glider behavior is caused mainly by the initial perturbation and needs a lot more time to estimate crosswind. Nevertheless the error-integrator flies the glider to the center of the target.

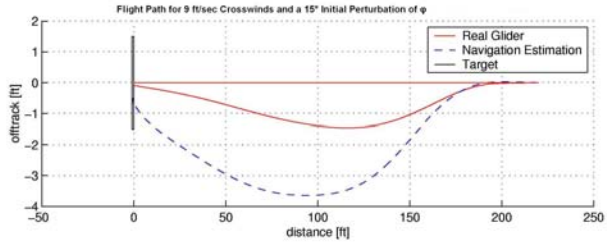

Figure 13 Off-Track Estimation with Crosswind and Perturbation in the Same Direction

Figure 14 shows the vertical results from simulator. The glide slope is not constant since it just controls the vertical camera angle Δ_V to a certain predetermined value related to the glide slope.

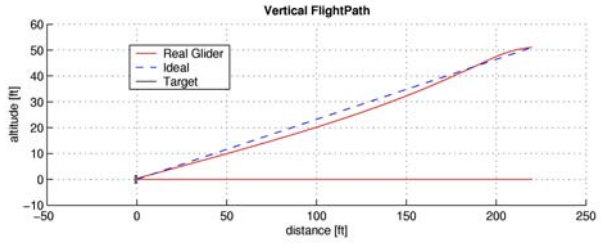

Figure 14 Vertical Flight Path

Conclusions and Further Research

Simulation tests showed that vision-only aircraft flight control is effective, even with very few image analysis measurements and a simple glider model.

Simulations showed that a 1/3 crosswind versus launch speed velocity ratio is the maximum the glider can handle, without losing track of the window. Should 10 feet/second winds be expected, the glider would have to fly at least 25 to 35 feet per second forward speed.

There are two main research areas that extend from this preliminary examination of visually guided aerial vehicles. The first step will be to take the methods outlined in this paper from simulation and apply them in actual flight testing. The second is to explore other possible applications, including guidance to a landing on a runway.

Acknowledgements

The authors would like to acknowledge the contributions of Tobias Breithaupt, Henrick Christophersen, Suresh Kannan, and Wayne Pickell, who made this research possible.

References

[1] Vidal, Rene, Omid Shakernia, Shankar Sastry,2003, "*Formation Control of Nonholonomic Mobile Robots with Omnidirectional Visual Servoing and Motion Segmentation"* To appear: 2003 IEEE Conference on Robotics and Automation .

[2] Sinopoli, Bruno, Mario Micheli, Gianluca Donato and T. John Koo, 2001, "*Vision Based Navigation for an Unmanned Air Vehicle*", Proceedings of the IEEE International Conference on Robotics and Automation, May 2001,pp 1757-1765.

[3] Saripalli , Srikanth, James F. Montgomery, and Gaurav S. Sukhatme, 2002, "*Vision-based autonomous landing of an unmanned aerial vehicle*," IEEE International Conference on Robotics and Automation, Volume 3, 2002 pp. 2799-2804

[4] Viola, P., M. Jones, 2001, *Rapid Object Detection using Boosted Cascade of Simple Features*, Computer Vision and Pattern Recognition.

[5] Weltch, Greg, Gary Bishop,, 2002, *An Introduction to the Kalman Filter*, University of North Carolina at Chapel Hill, TR 95-041.

[6] Johnson, E. and S. Mishra.,2002, *Flight Simulation for the Development of an Experimental UAV*, Proceedings of the AIAA Modeling and Simulation Technologies Conference.