JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 4, April 2007

Real-Time Vision-Based Relative Aircraft Navigation

Eric N. Johnson,* Anthony J. Calise,† Yoko Watanabe,‡ Jincheol Ha,‡ and James C. Neidhoefer§
Georgia Institute of Technology, Atlanta, GA 30332-0150

This paper describes two vision-based techniques for the navigation of an aircraft relative to an airborne target using only information from a single camera fixed to the aircraft. These techniques are motivated by problems such as "see and avoid", pursuit, formation flying, and in-air refueling. By applying an Extended Kalman Filter for relative state estimation, both the velocity and position of the aircraft relative to the target can be estimated. While relative states such as bearing can be estimated fairly easily, estimating the range to the target is more difficult because it requires achieving valid depth perception with a single camera. The two techniques presented here offer distinct solutions to this problem. The first technique, Center Only Relative State Estimation, uses optimal control to generate an optimal (sinusoidal) trajectory to a desired location relative to the target that results in accurate range-to-target estimates while making minimal demands on the image processing system. The second technique, Subtended Angle Relative State Estimation, uses more rigorous image processing to arrive at a valid range estimate without requiring the aircraft to follow a prescribed path. Simulation results indicate that both methods yield range estimates of comparable accuracy while placing different demands on the aircraft and its systems.

Nomenclature

Section II. Image Processing

$\Psi(x, y, t)$	active contour curve
$\phi(x, y)$	conformal mapping or speed function
$I(x, y, t)$	normalized gray-scale image intensity function
G_σ	Gaussian smoothing filter
v	constant inflation term
L_ϕ	length functional

Section III. Dynamics of the Camera Motion

ω	angular velocity of camera relative to target
V	translational velocity of camera relative to target
P_{camera}	position of target in camera frame
P_{image}	position of target in image frame
d	parameter defined as $1/X$
η_x, η_y	process noise

Received 22 February 2006; revision received 11 September 2006; accepted for publication 11 September 2006.

*Lockheed Martin Assistant Professor of Avionics Integration, Aerospace Engineering Deportment, Georgia Institute of Technology, Atlanta, GA 30332-0150
†Professor, Aerospace Engineering Department, Georgia Institute of Technology, Atlanta, GA 30332-0150
‡Graduate Research Assistant, Aerospace Engineering Department, Georgia Institute of Technology, Atlanta, GA 30332-0150
§Research Engineer, Aerospace Engineering Department, Georgia Institute of Technology, Atlanta, GA 30332-0150

$\sigma_{\eta x}$, $\sigma_{\eta y}$ standard deviation of the process noise
$\sigma_{\xi x}$, $\sigma_{\xi y}$ standard deviation of the measurement noise

Section IV. Center Only Relative State Estimation (CORSE)

t_k time at which the k^{th} measurement is taken
z_k measurement at time t_k
X_m camera position in aircraft body axis
$X - X_m$ target position relative to camera
ξ_k measurement noise at time t_k
a_s aircraft acceleration
x_k vector of estimator states for process model
v_k discretized zero mean process noise
Φ_k state transition matrix at time t_k
I identity matrix
$\hat{x}_k$ predicted state estimate
P_k covariance matrix of error for updated state estimate
P_k^- covariance matrix of error for predicted state estimate
K_k Kalman gain matrix
u_Y guidance commands in the y direction
t_f time at which aircraft reaches destination
V_Y optimal velocity solution in the y direction
Y optimal position solution in the y direction
K weight for guidance command u_Y
d inverse of range

Section V. Subtended Angle Relative State Estimation (SARSE)

x_k vector of estimator states at time t_k
t_k time at which the k^{th} measurement is taken
z_k measurement at time t_k
u unit vector pointing from camera to target
α angle subtended by wingspan of target
a_s aircraft acceleration
K_k Kalman gain matrix
$\hat{x}_k^-$ predicted state estimate
P_k covariance matrix of error for updated state estimate
P_k^- covariance matrix of error for predicted state estimate
R_k measurement noise covariance matrix
Q_k process noise covariance matrix

I. Introduction

MOST present-day airborne navigation systems use multi-sensor fusion to accurately estimate vehicle states. However, the use of multiple sensors, which typically include GPS and Inertial Measurement Units (IMUs), can result in complex and expensive navigation systems not suitable for small, disposable UAVs. This paper focuses on the development of a simpler and less expensive navigation system that uses only data from a single camera to generate navigation information (relative to another vehicle) suitable for autonomous UAV guidance and control.

Birds and insects are two examples in nature that demonstrate the feasibility of using vision information to generate navigation solutions suitable for in-flight guidance and control. This paper introduces two methods that similarly rely on visual input for airborne navigation. The specific scenario considered involves an aircraft with a single camera following and navigating relative to an airborne target, such as another aircraft. The methods developed are applicable to operations such as leader-follower setups or formation flying and can be adapted to other types of UAV operations

as well. In addition, these methods could be adapted for navigation relative to known landmarks, thus providing a viable alternative to GPS for navigation in an earth fixed reference frame.

In each of these methods, the single camera provides noisy measurements of the target's horizontal and vertical position in its image plane, and an Extended Kalman Filter (EKF) is then used to estimate the position and velocity of the aircraft relative to the target. Any changes in the relative state in the axes of the camera's image plane will be readily apparent to the following aircraft. However, changes in the range from the camera to the target will be harder to determine due to the difficulty of depth perception with monocular vision. For example, when the aircraft flies straight towards a non-accelerating target, the difference between subsequent image frames will be small, resulting in poor estimation of the distance between the aircraft and the target. If these poor range estimates are used to control a vehicle performing station-keeping with another aircraft, dangerous deviations from the desired location could occur.

In many vision-based systems, this problem is solved by having multiple distinctly located cameras (or eyes). In such systems, depth perception arises from differences between images taken simultaneously from each of the cameras. In monocular systems, however, only differences between successive camera images can be used. Both methods presented in this paper use monocular image sequences differently to produce valid relative state estimates.

The first method presented in this paper, Center Only Relative State Estimation (CORSE), guides the motion of the camera to maximize the usable information available in the image sequence to derive accurate range estimates. In this way, the performance of the state estimator can be significantly improved.[1] The best camera motion for range estimation is parallel to the camera's image plane;[2] if the vehicle flies to the target with a meandering path, more accurate range estimates can be obtained and potentially dangerous path deviations avoided. In the CORSE algorithm, the EKF provides the variance of the estimation error for each state. These variances are then minimized in order to maximize the estimation accuracy. The CORSE algorithm performs this optimization by setting a performance index to minimize the variance of the error and then solving the resulting optimization problem subject to the known camera motions, in order to obtain an optimal flight path to some desired position relative to the target. It is shown here that this problem can be solved analytically. The resulting optimal guidance policy is then compared to a simple linear guidance policy using a flight simulator. For a similar example where tracking is performed with "bearing only" the reader is referred to.[4]

The second algorithm presented in this paper, Subtended Angle Relative State Estimation (SARSE), uses more information derived from camera images without resorting to controlling camera motions. In particular, the image processing system is used to estimate the locations of the wingtips of the target in addition to just the location of the target center. Then, the angle subtended by the wingspan of the target, relative to the camera position, is used as an input to the EKF. The resulting performance of the SARSE algorithm is demonstrated in this paper using a flight simulator.

This paper starts by describing the image processing techniques used by the CORSE and SARSE algorithms. Next, the derivation of the equations describing the camera motion is presented. This is followed by detailed descriptions of the CORSE and SARSE algorithms, including simulation-based results for each. Finally, conclusions on the work are presented.

II. Image Processing

The image processing in this work uses an active contour to calculate the location of the center of the target aircraft (which is used by the CORSE algorithm) as well as the locations of its wingtips and the angle subtended by the line between the wingtips (which are used by the SARSE algorithm). This work is based on work done in[3] and extended to use a Fast Marching Level Set method[13] for computational speed. For similar image processing applications, the reader is referred to.[5]

A. Background on Active Contours

As described in,[3] active contours are processes that use image coherence to track features of interest over time. They fit naturally into control frameworks and have often been employed in conjunction with Kalman filtering.[6] Active contours have the ability to conform to various object shapes and have been used for segmentation, edge detection, shape modeling, and visual tracking.

In the classical theory of deformable contours, energy minimization methods are used to move splines under the influence of image dependent forces and constraints set by the user. There are a number of potential problems associated with these approaches, such as contour initialization, existence of multiple minima, and the selection of the elasticity parameters. Moreover, natural criteria for the splitting and merging of contours (or for the treatment of multiple contours to track multiple objects) are not readily available.

In[7,8] an active contour model is proposed based on the level set formulation of the Euclidean curve shortening equation. Specifically, the model is

$$\frac{\partial \Psi}{\partial t} = \phi(x, y)\|\nabla\Psi\| \left(\operatorname{div}\left(\frac{\nabla\Psi}{\|\nabla\Psi\|} \right) + \upsilon \right) \tag{1}$$

in which the function $\phi(x, y)$ depends on the given image and is used as a "stopping criterion." Typically the term $\phi(x, y)$ is chosen to be small near an intensity-based edge and acts to stop evolution when the contour gets close to an edge. One may define

$$\phi(x, y) = \frac{1}{1 + \|\nabla G_\sigma * I\|^2} \tag{2}$$

where I is the (grey-scale) intensity of pixel x, y and G_σ is a Gaussian smoothing filter.[7,8] The function $\Psi(x, y, t)$ evolves in equation (1) according to the associated level set flow for planar curve evolution in the normal direction with speed as a function of curvature. This concept was introduced in.[9,10]

The curve shortening part of the active contour evolution, namely

$$\frac{\partial \Psi}{\partial t} = \|\nabla\Psi\| \operatorname{div}\left(\frac{\nabla\Psi}{\|\nabla\Psi\|} \right) \tag{3}$$

is derived as a gradient flow for shrinking the perimeter. As explained in,[7] the constant inflation term v is added in equation (1) in order to keep the evolution moving in the proper direction.[11]

Thus the model represented in equation (1) will be modified in a manner consonant with length/area minimizing ideas. The ordinary arc-length function along a curve $C = (x(p), y(p))^{\mathrm{T}}$ with parameter p is changed as,

$$\begin{aligned} \mathrm{d}s &= (x_p^2 + y_p^2)^{1/2}\,\mathrm{d}p \\ &\downarrow \\ \mathrm{d}s_\phi &= (x_p^2 + y_p^2)^{1/2}\phi\,\mathrm{d}p \end{aligned}$$

where $\phi(x, y)$ is a positive differentiable function. Then the corresponding gradient flow for shortening length is computed relative to the new metric $\mathrm{d}s_\phi$.

The length function L_ϕ is defined as

$$L_\phi = \int_0^1 \left\| \frac{\partial C}{\partial p} \right\| \phi\,\mathrm{d}p$$

Then, by taking the first variation of the modified length function L_ϕ and using integration by parts as in,[12] the result is

$$L'_\phi(t) = -\int_0^{L_\phi(t)} \left\langle \frac{\partial C}{\partial t}\phi\kappa\vec{N} - (\nabla\phi\cdot\vec{N})\vec{N} \right\rangle$$

which means that the direction in which the L_ϕ perimeter is shrinking the fastest is given by

$$\frac{\partial C}{\partial t} = (\phi\kappa - (\nabla\phi\cdot\vec{N}))\vec{N} \tag{4}$$

This is the gradient flow corresponding to the minimization of the length functional L_ϕ. The level set version of this is

$$\frac{\partial \Psi}{\partial t} = \phi\|\nabla\Psi\| \operatorname{div}\left(\frac{\nabla\Psi}{\|\nabla\Psi\|} \right) + \nabla\phi\cdot\nabla\Psi \tag{5}$$

This evolution should attract the contour very quickly to the feature that lies at the bottom of the potential well described by the gradient flow in equation (5). We may also add a constant inflation term (as is done in[7,8]), to derive

a modified version of (1) given by

$$\frac{\partial \Psi}{\partial t} = \phi \|\nabla \Psi\| \left(\operatorname{div} \left(\frac{\nabla \Psi}{\|\nabla \Psi\|} \right) + \upsilon \right) + \nabla \phi \cdot \nabla \Psi \tag{6}$$

B. The Fast-Marching Level Set Method

The specific active contour model used in this application is based on the fast marching level set method.[13] Level set methods use a narrowband approach which performs calculations on image data only in a limited region near the active contour (also called the interface) to calculate how the contour evolves. This greatly reduces computational requirements and enhances the method's viability for real-time applications.

The Fast Marching method is the extreme narrowband case as it uses a band width of only a single pixel (the calculation of the "crossing time" for each pixel uses only the direct neighbors of that pixel). In the fast marching method the interface moves in only one direction and passes over each pixel only once. Thus a unique value of "crossing time" is associated with each pixel. The resulting crossing time function $T(x, y)$ is a time-invariant level set function that satisfies the equation

$$\phi(x, y)\|T(x, y)\| = 1 \tag{7}$$

which is a form of the well-known Eikonal equation.[13] In equation (7), ϕ is the called the "speed function". It can be seen in equation (7) that for large speed function values, the corresponding crossing time will be small (which means that pixel is processed relatively early), and for small speed function values, the corresponding crossing time will be large.

1. The Fast Marching Formulation

Once $\phi(x, y)$ has been calculated at each pixel in the image, the goal is to numerically solve equation (7) for $T(x, y)$ over a subset of the image data located around the expected target location. The numerical solution of equation (7) will result in active contours propagating across the image and "wrapping" themselves around objects in the image. A two dimensional numerical approximation of equation (7) is:[13]

$$[\max(D_{ij}^{-x}T(x, y), 0)^2 + \min(D_{ij}^{+x}T(x, y), 0)^2 + \max(D_{ij}^{-y}T(x, y), 0)^2 + \min(D_{ij}^{+y}T(x, y), 0)^2] = \frac{1}{\phi(x, y)^2} \tag{8}$$

where D in equation (8) is a difference operator notation such that, $D_{ij}^{+x}T = (T_{i+1,j} - T_{i,j}/\Delta x)$.

In the Fast Marching method, all pixels in an image belong to one of three sets. The first set, called the "Known" set, contains all pixels over which the interface has already passed. The second set, called the "Trial" set, contains the pixels in the interface. And the third set, called the "Far" set, contains those pixels which have not yet been processed. Details of the standard Fast Marching algorithm are available in[13,22] and the algorithm is summarized here as follows:

FAST MARCHING ALGORITHM

S1. Begin loop: Find the pixel in the "Trial" set with the smallest value of $T(x, y)$ and label this pixel "A".
S2. Add "A" to the "Known" set and remove it from the "Trial" set. Check if a stopping criteria has been reached, if not, proceed to step S3.
S3. Add all pixels neighboring "A" that are not in the "Known" set to the "Trial" set. If any of these pixels were in the "Far" set, remove them.
S4. Recompute the values of $T(x, y)$ for all pixels neighboring "A" by solving equation (8). Return to step S1.

Figure 1 illustrates how the Fast Marching procedure is used to evolve an active contour on a 9×10 image grid (90 total pixels) at 9 distinct steps in the solution process.

In Fig. 1 the blue pixels are part of the "Known" set, the red pixels are part of the "Trial" set, the yellow pixels are part of the "Far" set, the green curve represents the active contour, and the pixel in the "Trial" set with the lowest value of $T(x, y)$ is labeled "A". A description of the process (more detailed than the summary above) is as follows:

1. Start by solving for the speed function, $\phi(x, y)$ over the entire image.

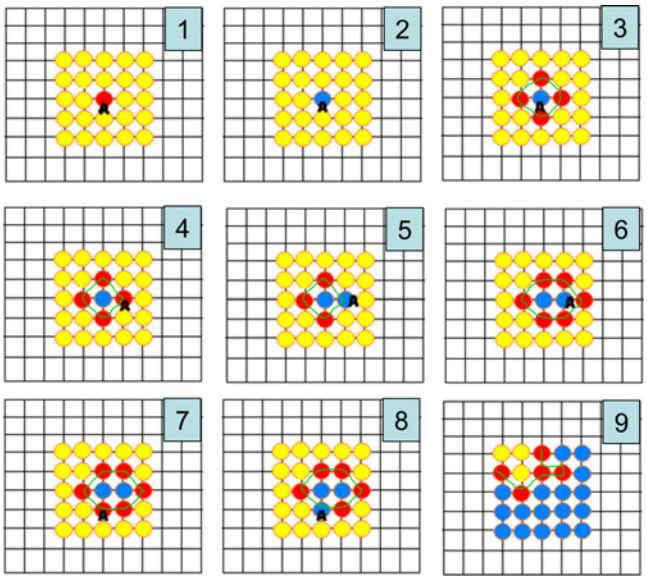

Fig. 1 Active contour evolution with the fast marching level set method.

2. On the initial iteration, a pixel is chosen and added to the "Trial" set (the red pixel in Fig. 1.1). $T(x, y)$ is set equal to zero for this pixel.
3. A heap sorting algorithm[14] is then used to find the pixel in the "Trial" set with the smallest value of $T(x, y)$ and this pixel is labeled "A". This corresponds with step S1.
4. Pixel "A" is then removed from the "Trial" set and added to the "Known" set. This corresponds with step S2 as well as Fig. 1.2.
5. The neighbors of "A" that are not in the "Known" set are then added to the "Trial" set. If any of these pixels were in the "Far" set, they are removed. This corresponds to step S3 as well as Fig. 1.3.
6. Equation (8) is then solved for $T(x, y)$ at all neighbors of "A" in the "Trial" set. This corresponds to step S4.
7. The process then loops back to 3 and iterates until a stopping criteria is satisfied. Thus Fig. 1.4 corresponds to step S1, Fig. 1.5 corresponds to step S2, Figure 1.6 corresponds to steps S3 and S4 however only 3 neighboring pixels are added to the 'Trial' set since one neighboring pixel is already in the "Known" set. Fig. 1.7 corresponds to step S1, etc. Finally, Fig. 1.9 shows the result of this procedure after several iterations.

Thus, the contour propagates across the image, and comes to rest near areas of high contrast, and in particular, near dark objects (as explained in Section B.2). Video of an actual contour propagation using real camera imagery can be seen here: http://pdf.aiaa.org/JournalsOnline/PDFFiles/15429423_v4n4/aiaa/15429423/v4n4/Multimedia/23410Movie1.avi

2. Construction of the Speed Function

In this work, the speed function is calculated as follows:

$$\phi(x, y) = \alpha\left(\frac{1}{1 + \|\nabla I(x, y)\|^2}\right) + \gamma I(x, y) \tag{9}$$

where $I(x, y)$ is the grey-scale intensity of pixel x, y and $\nabla I(x, y)$ is the gradient of the intensity (contrast) at x, y calculated with respect to its neighboring pixels.

Since in real image data, the illumination of various parts of the image can drastically change from frame to frame, the intensity values are normalized as follows,

$$I(x, y) = \frac{I_o(x, y) - \min(I)}{\max(I) - \min(I)} \tag{10}$$

Fig. 2 A. Original image, B. Processed image with noise from clouds, C. Processed image with optimized values of α and γ.

so in each frame the maximum value of the intensity is 1 and minimum value is 0. In equation (10), $I_o(x, y)$ is the original image intensity data from a frame grabber. For dark pixels (small values of $I(x, y)$) with high contrast, the speed will be small and $T(x, y)$ will be large (as seen in Equation (7)) which means that dark pixels will be processed later than brighter pixels. For either bright pixels (large values of $I(x, y)$) or pixels with low contrast, the speed function will be large and $T(x, y)$ will be small meaning these pixels will be processed earlier than darker pixels. Thus the first term in equation (9) will drive the contour to areas of high intensity gradient (edges of objects), and the second term will help to drive the contour to dark objects. The second term helps to distinguish between objects such as aircraft and noise such as clouds since aircraft normally appear darker than their surroundings in aerial images, and clouds normally appear lighter than their surroundings.

In equation (9), α and γ are scaling factors for setting the relative weight of contrast and intensity. For different weather conditions or different times of day, the optimal values of α and γ can change. To date, the values of α and γ have been tuned "by hand" both off-line (in lab conditions) and, if necessary, at the flying site. Future work is intended to include the development of numerical schemes to automatically tune α and γ based on weather and lighting conditions. Figure 2 part A shows an original image and part B shows the result of processing the image with the Fast Marching Level Set Method. Notice that in part B, the intensity gradients of the edges of the cloud have been detected and classified as "final" contours. Part C shows the same image processed with the same method with values of α and γ optimized to filter out the "noise" caused by the clouds. Video footage showing a situation similar to that shown in Fig. 2 part B can be seen here: http://pdf.aiaa.org/JournalsOnline/PDFFiles/15429423_v4n4/aiaa/15429423/v4n4/Multimedia/23410Movie2.avi and a video showing a situation similar to that shown in Fig. 2 part C can be seen here: http://pdf.aiaa.org/JournalsOnline/PDFFiles/15429423_v4n4/aiaa/15429423/v4n4/Multimedia/23410Movie3.avi

3. Implementation for Flight Testing

When applying this method to real flight test data, the assumption is made that the target aircraft will always be above the horizon. Future work will include enhancing the robustness of the image processing techniques to include cases where the target is both above or below the horizon. Thus, the initial "Trial" pixels used to start the contour propagation are located at the top of the image and the contour propagation is stopped at the horizon as shown in Figs. 3 and 4.

In this work, in order to process each image faster, the heap sorting algorithm is only called at every iteration until the total number of points in the heap exceeds some number (in this case 500 points). Then when the heap size exceeds the threshold, the heap sorting algorithm is called only once for every several iterations (in this case 20 iterations).

To terminate the contour propagation for each individual image frame, two stopping criteria are examined each time the process reaches step S2 (as described in Section B.1). The first criterion uses a minimum threshold value for the speed function, $\phi(x, y)$ "Trial" pixel. Thus the propagation is stopped when the speed function value associated with the pixel on the active contour with the minimum value of crossing time (pixel "A" in Fig. 1 found by heap sorting algorithm) is less than a given threshold value.

The second criterion is used only after the target is registered in the previous frame. This criterion uses an expected value for the number of pixels the target will occupy in each image frame. The expected value is calculated using the

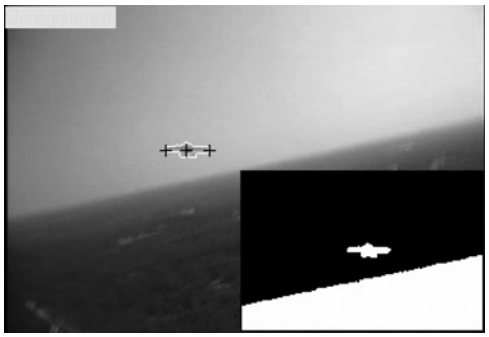

Fig. 3 Original image and the image after processing with the fast marching method (inset).

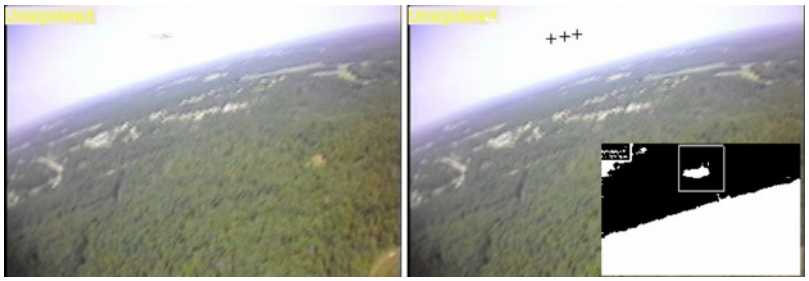

Fig. 4 Intensity saturation caused by the camera pointing to near the sun can cause noisy results.

assumption that the target will be nearly the same size in the current image as it was in the previous image (which assumes the trailing aircraft is neither gaining nor losing distance relative to the target aircraft, and also that the target aircraft is not maneuvering). This assumption, may limit the current overall implementation in some applications, such as mid-air docking (where the follower must close on the leader), however, there are many other applications for which it is well suited, such as formation flying, and tracking of ground based targets. Also, this assumption is part of the image processing algorithm which is only a subset of the overall system. The filtering aspects of the system should need very little modification for use in docking-type applications.

Thus, the propagation is stopped if the total number of pixels in a limited region around the previous target position minus the total number of processed pixels (pixels in the same region that are also in the "Known" set) reaches the expected value. This criterion is helpful in cases when the intensity in an image saturates, such as when the camera points directly into the sun and some part of the target can not be distinguished. In such cases (see the left image in Fig. 4) the intensity gradients may not be discernable by the algorithm, and thus contours can propagate past the target.

4. Post Processing

After the contour propagation is stopped, pixels in the "Known" set are set to 0, and all other pixels are set to 1. From this "binary image" several contours are extracted using a threshold on the recalculated intensity gradient.

The "center of mass" for each closed contour is then calculated. The method of moments is used to find the major and minor axes of each closed contour and the outermost points of the major axis are labeled as candidates for the target's wingtips. These are then used along with an expected span value and other calculated geometric characteristics (such as the aspect ratio and the angle between the line segments formed by the center-of-mass and the wingtips) to determine which closed contour corresponds to the target aircraft. The expected value of the span is calculated based on knowledge of the actual target aircraft span, and the distance from the target aircraft (thus the assumption is again used that the target is not maneuvering with respect to the trailing aircraft). These "feature points" are shown with black crosses in Figs. 3 and 4. A video showing a situation similar to that shown in Fig. 3 can be seen here: http://pdf.aiaa.org/JournalsOnline/PDFFiles/15429423_v4n4/aiaa/15429423/v4n4/Multimedia/23410Movie4.avi

III. Dynamics of the Camera Motion

In this section, the dynamics of the camera motion are derived. Consider a camera frame and an image frame as shown in Fig. 5. Let ω be the angular velocity and V the relative velocity of the camera with respect to the target. Also, define the positions of the target in the camera frame and in the image frame as follows:

$$P_{camera} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad P_{image} = \begin{bmatrix} y \\ z \end{bmatrix}$$

$$y = \frac{Y}{X}, \quad z = \frac{Z}{X}$$

The dynamics of P_{camera} can be described as

$$\dot{P}_{camera} = -V - \omega \times P_{camera} \tag{11}$$

To facilitate the analytic solution of the optimal control problem, an assumption is made that the camera does not rotate either through gimbal dynamics or correction. The dynamics of Equation (11) then become:

$$\begin{aligned} \dot{X} &= -V_X \\ \dot{Y} &= -V_Y \\ \dot{Z} &= -V_Z \end{aligned} \tag{12}$$

where V_X, V_Y and V_Z are the relative translating velocity components along the X, Y and Z-axes, respectively. The translation-only assumption is idealistic only if one assumes the camera rotates as a result of aircraft motion. Aircraft mounted non-rotating cameras have been in wide use for many years in the film industry and as stable observation platforms[23] and the simulation based results shown for the CORSE algorithm were generated by simulating such a system; an aircraft with a mounted camera which translated, but did not rotate with the aircraft. However, the translation-only assumption was used in the CORSE algorithm primarily to facilitate an analytic solution of the optimal control problem. And if the CORSE algorithm was to be applied with small expendable UAVs (which probably wouldn't have a gyro-stabilized gimbaled camera mount), it would be desirable to re-derive the equations

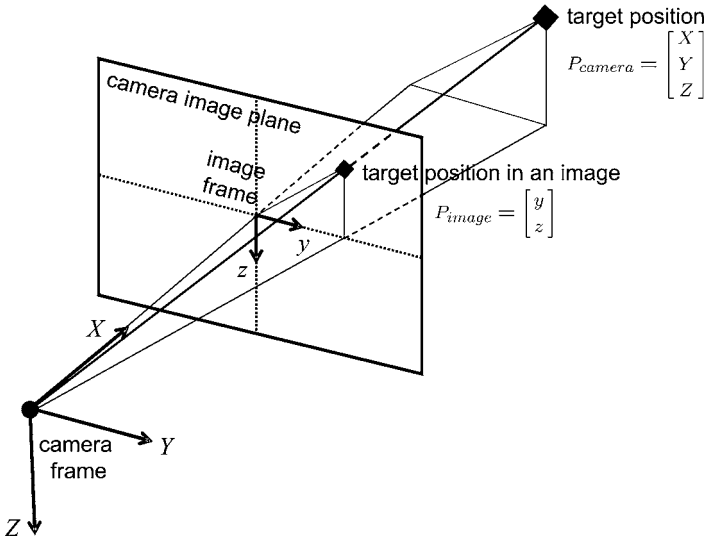

Fig. 5 Camera frame and image frame.

without using the translation-only assumption. The SARSE algorithm presented below is more general and does not use a translation-only assumption.

The CORSE results presented below were generated using a simulated camera which translated but did not rotate with the aircraft

For convenience, a new parameter $d = 1/X$ is introduced in place of X.[2] Then, Equation (12) yields:

$$\dot{d} = V_X d^2 \tag{13}$$

The dynamics are considered as a second order system from the guidance inputs

$$\begin{aligned} \dot{V}_X &= u_X + \eta_X \\ \dot{V}_Y &= u_Y + \eta_Y \\ \dot{V}_Z &= u_Z + \eta_Z \end{aligned} \tag{14}$$

where the target's acceleration is assumed to be zero.

In the equations above, η_X, η_Y and η_Z denote the process noise. The average of the process noise $\eta = [\eta_X \ \eta_Y \ \eta_Z]^{\mathrm{T}}$ is zero, and its covariance matrix is given by

$$E[\eta(t)\eta(\tau)^{\mathrm{T}}] = Q\delta(t-\tau)$$

$$Q = \begin{bmatrix} \sigma_{\eta X}^2 & 0 & 0 \\ 0 & \sigma_{\eta Y}^2 & 0 \\ 0 & 0 & \sigma_{\eta Z}^2 \end{bmatrix}$$

The values of the standard deviation of the process noise ($\sigma_{\eta x}$, $\sigma_{\eta y}$, $\sigma_{\eta z}$) were determined by tuning them to match empirical data collected during flight tests. These values are held constant during algorithm execution, a common practice in similar EKF applications. While, constant process noise standard deviations could require retuning in different conditions, the authors have found the values used to result in a robust implementation.

Thus, the dynamics of the camera motion can be represented by equations (12) through (14).

IV. Center Only Relative State Estimation (CORSE)

A. The Measurement Model

The camera gives a series of images and the target center position is detected in each image using a simplified image processing procedure. The target center position is specified by its horizontal and vertical distance from the center of the image, for each image at time t_k in the series. The measurement is, in this case, the distance from the center of each image to the center position of the target. Measurement noise is then added to the detected target position, and the measurements are thus modeled as follows.

$$z_k = \begin{bmatrix} y_k \\ z_k \end{bmatrix} + \begin{bmatrix} \xi_{yk} \\ \xi_{zk} \end{bmatrix} = \begin{bmatrix} Y_k/X_k \\ Z_k/X_k \end{bmatrix} + \begin{bmatrix} \xi_{yk} \\ \xi_{zk} \end{bmatrix} = \begin{bmatrix} Y_k d_k \\ Z_k d_k \end{bmatrix} + \begin{bmatrix} \xi_{yk} \\ \xi_{zk} \end{bmatrix} \tag{15}$$

$\boldsymbol{\xi}_k = [\xi_{xk} \ \xi_{y_k}]^{\mathrm{T}}$ represents measurement noise with zero mean and covariance so that

$$E\begin{bmatrix} \boldsymbol{\xi}_k & \boldsymbol{\xi}_l^{\mathrm{T}} \end{bmatrix} = \begin{cases} R_k & l = k \\ 0 & l \neq k \end{cases}$$

$$R_k = \begin{bmatrix} \sigma_{\xi y}^2 0 \\ 0 \sigma_{\xi z}^2 \end{bmatrix}$$

In a manner similar to that used in the process noise model development, the values of the standard deviation of the measurement noise ($\sigma_{\xi x}$, $\sigma_{\xi y}$, $\sigma_{\xi z}$) were determined by tuning them to match empirical data collected during flight

tests. These values are held constant during algorithm execution, a common practice in similar EKF applications. While, constant measurement noise standard deviations could require retuning in different conditions, the authors have found the values used to result in a robust implementation.

This nonlinear measurement model can be written as

$$z_k = h(x_k) + \xi_k \tag{16}$$

B. The Process Model

The estimator states are defined as

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 1/X \\ Y \\ Z \\ V_X/X \\ V_Y \\ V_Z \end{bmatrix} = \begin{bmatrix} d \\ Y \\ Z \\ V_X d \\ V_Y \\ V_Z \end{bmatrix}$$

The time derivatives of X and V are given by (12–14) so the process model is represented as

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_1 x_4 \\ -x_5 \\ -x_6 \\ x_4^2 + x_1 u_X \\ u_Y \\ u_Z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ x_1 \eta_X \\ \eta_Y \\ \eta_Z \end{bmatrix} = \begin{bmatrix} x_1 x_4 \\ -x_5 \\ -x_6 \\ x_4^2 + x_1 u_X \\ u_Y \\ u_Z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ x_1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \eta$$

This nonlinear model can be written as

$$\dot{x} = f(x, u) + G(x)\eta$$

This model is then discretized as follows:

$$\begin{aligned} x_{k+1} &= \Phi_k x_k + \upsilon_k \\ \Phi_k &\cong I + \left.\frac{\partial f}{\partial x}\right|_{x_k = \hat{x}_k} \Delta t \end{aligned} \tag{17}$$

where υ_k is the discretized zero mean process noise whose covariance matrix is approximated as

$$E\begin{bmatrix} \upsilon_k & \upsilon_l^{\mathrm{T}} \end{bmatrix} = \begin{cases} Q_k & l = k \\ 0 & l \neq k \end{cases}$$

$$Q_k \cong GQG^{\mathrm{T}}\Delta t$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1^2\sigma_{\eta \mathrm{X}}^2 \Delta \mathrm{t} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\eta \mathrm{Y}}^2 \Delta \mathrm{t} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\eta \mathrm{Z}}^2 \Delta \mathrm{t} \end{bmatrix}$$

C. The Extended Kalman Filter

Applying an Extended Kalman Filter[15–17] to the discrete-time system in (17), the state estimation update and prediction equations are as follows.

Update

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + K_k(z_k - \hat{z}_k^-) \\ P_k^{-1} &= (P_k^-)^{-1} + H_k^{\mathrm{T}} R_k^{-1} H_k \\ K_k &= P_k\, H_k^{\mathrm{T}} R_k^{-1} \\ H_k &= \left.\frac{\partial h}{\partial x_k}\right|_{x_k=\hat{x}_k^-} \end{aligned} \tag{18}$$

Prediction

$$\begin{aligned} \hat{x}_{k+1}^- &= \hat{x}_k + f(t, \hat{x}_k, u_k)\Delta t \\ P_{k+1}^- &= \Phi_k P_k \Phi_k^{\mathrm{T}} + Q_k \end{aligned} \tag{19}$$

The variables $\hat{x}_k$ and $\hat{x}_k^-$ stand for updated and predicted estimates of states at time t_k, and P_k and P_k^- are covariance matrices of their errors, respectively. K_k represents the Kalman gain matrix.

D. The Trajectory Optimization Formulation

The trajectory optimizer outputs a path from the aircraft's current position to a desired location relative to the target. If the aircraft then follows this path, the accuracy of the estimated states, particularly of the estimated range, is maximized.

As the vehicle approaches the target, the EKF (described in the previous section) gives estimates of states that describe the position and velocity of the target center relative to the camera. However, depending on the motion of the camera relative to the target, the range estimation might not have sufficient accuracy, because there might not be enough lateral motion to make the range X observable.

To improve the accuracy of the range estimation, the lateral motion of the camera is controlled to minimize the variance of the range estimation error. This is formulated as an optimization problem in which the inverse of P_k is maximized. Since P_k^- is the covariance matrix of the estimation error at time t_k calculated using measurements up to time t_{k-1}, our approach to maximize P_k^{-1}, given in Equation (18), is to maximize the second term $H_k^{\mathrm{T}} R_K^{-1} H_k$. Increasing the first diagonal element of this matrix helps to reduce the range estimation error. It is expanded as

$$[H_k^{\mathrm{T}} R_k^{-1} H_k]_{11} = \left(\frac{\hat{Y}_k^-}{\sigma_{\xi y}}\right)^2 + \left(\frac{\hat{Z}_k^-}{\sigma_{\xi z}}\right)^2$$

At the same time, it is also desirable to limit the magnitude of the control u_Y and u_Z to minimize the extra maneuvering required for range estimation. Therefore, the optimization problem is formulated as

$$\min_{u_Y, u_Z} \frac{1}{2} \int_0^{t_f} \left\{ -\left(\frac{\hat{Y}_k^-}{\sigma_{\xi y}}\right)^2 - \left(\frac{\hat{Z}_k^-}{\sigma_{\xi z}}\right)^2 + K_Y^2 u_Y^2 + K_Z^2 u_Z^2 \right\} \mathrm{d}t \tag{20}$$

which is subject to the camera motion dynamics given by Equations (12) through (14), where t_f represents the terminal time when the vehicle reaches the destination. This problem can then be solved with appropriate boundary conditions.

To simplify the optimization problem, we limit the camera motion to the X-Y plane, with the Z position held constant. Then $\hat{Z}_k^-$ is assumed to be constant and $u_Z = 0$, the second and third terms of the performance index in Equation (20) can be eliminated. Furthermore, to simplify the problem, constant V_X and $u_X = 0$ are assumed. This

reduces the problem to having only two states, Y and V_Y . The boundary conditions are given by specifying the initial and terminal states.

$$\begin{cases} Y(0) = Y_0 \\ V_Y(0) = V_{Y0} \end{cases} \qquad \begin{cases} Y(t_f) = Y_f \\ V_Y(t_f) = V_{Yf} \end{cases} \tag{21}$$

E. Analytical Solution

The analytical solution for this problem can be obtained by solving the Euler-Lagrange equations[18] The Hamiltonian is defined as

$$H = -\frac{1}{2}\frac{Y^2}{\sigma_{\xi y}^2} + \frac{1}{2}K_Y^2 u_Y^2 - \lambda_1 V_Y + \lambda_2 u_Y$$

The corresponding Euler-Lagrange equations are

$$\begin{aligned} \dot{\lambda}_1 &= -\frac{\partial H}{\partial Y} = \frac{Y}{\sigma_{\xi y}^2} \\ \dot{\lambda}_2 &= -\frac{\partial H}{\partial V_Y} = \lambda_1 \\ \frac{\partial H}{\partial u_Y} &= K_Y^2 u_Y + \lambda_2 = 0 \end{aligned} \tag{22}$$

where λ_1 and λ_2 are Lagrange multipliers. From these equations and the camera motion dynamics, a differential equation for u_Y can be derived as follows:

$$\frac{\mathrm{d}^4 u_Y}{\mathrm{d}t^4} = \frac{1}{(\sigma_{\xi y} K_Y)^2} u_Y$$

The explicit solutions for Y, V_Y, and u_Y are:

$$\begin{aligned} Y(t) &= -\frac{\sigma_{\xi y}}{K_Y}(A \sin t' + B \cos t' - Ce^{t'} - De^{-t'}) \\ V_Y(t) &= \frac{\sqrt{\sigma_{\xi y} K_Y}}{K^2}(A \cos t' - B \sin t' - Ce^{t'} + De^{-t'}) \\ u_Y(t) &= -\frac{1}{K_Y^2}(A \sin t' + B \cos t' + Ce^{t'} + De^{-t'}) \end{aligned} \tag{23}$$

where $t' = t/\sqrt{\sigma_{\xi y} K_Y}$. A, B, C, and D are chosen to satisfy the boundary conditions given by Equation (21).

The derived optimal path corresponding to Equation (23) is sinusoidal. Its frequency is $1/\sqrt{\sigma_{\xi y} K_Y}$, which is determined by the control input weight K_Y. Since V_X is assumed to be constant, small values of K_Y give more lateral motion before the vehicle reaches the destination and large values of K_Y give less. Because the lateral motion is what enables an estimation of range from a single camera, greater lateral motion corresponds to a more accurate range estimation. Therefore, small values of K_Y are desirable for increasing the range estimation accuracy. However, in practical applications, it is also important to limit the magnitude of guidance input u_Y to avoid excessive motions, which calls for the use of a large value for K_Y. Thus, there is a trade-off between the accuracy of the range estimation and the complexity of the flight path.

One approach to deal with this trade-off is to change the value of K_Y by using the variance of the range estimation error. As shown in Figs. 10 and 13, the variance of the range estimation error can be an index of the range estimation accuracy. A smaller variance corresponds to a more accurate estimate. Thus, we can increase K_Y as the variance becomes small and decrease K_Y as the variance becomes large to guide the vehicle to the specified relative position with sufficient range estimation accuracy and without having to undergo excessive lateral motion.

F. Simulation-Based CORSE Results

1. *Camera Motion Control*

The optimal guidance policy is used for the lateral camera motion. Suppose the desired target position relative to the vehicle is at $[X_f \; Y_f]$. Since V_X is a constant, considering the current time as zero, the terminal time t_f can be estimated by using the latest estimates of d and V_X as

$$\hat{t}_{fk} = \frac{(X_f - \hat{X}_k)}{\hat{V}_{Xk}} = \frac{X_f}{\hat{V}_{Xk}} - \frac{1}{\hat{V}_{Xk}\hat{d}_k}$$

Using this $\hat{t}_{fk}$ as the terminal time, the flight path is re-optimized at the each time t_k. In other words, the coefficient B is solved with $\hat{t}_{fk}$ at each time and the control input u_{Yk} is obtained by

$$\begin{aligned} u_{Yk} &= -\frac{1}{K_Y^2}(B_k + C_k + D_k) \\ &= -\frac{1}{K_Y^2}\left(2B_k + \frac{K_Y}{\sigma_{\xi y}}\hat{Y}_k\right) \end{aligned}$$

To signify that B is computed at each time, B_k is used instead of B in the above equations.

For the comparison of the range estimation results, the simulation is also performed using a linear guidance law.

$$u_{Yk} = -\hat{V}_{Yk} - 0.2(Y_f - \hat{Y}_k)$$

2. *The 6-DOF Multi-Aircraft Simulation*

Figure 6 is a screen capture of the display of the flight simulator used for this work. It includes two 14 ft wingspan fixed wing aircraft, configured as a leader and a follower. A flight controller and guidance system is already implemented in the simulator. The controller is an adaptive neural network flight controller,[19–21] and it determines actuator commands based on the navigation system output and position, velocity, and attitude commands. The follower aircraft has a camera and its image is also simulated. The synthetic images are processed and provide the same type of output that would be expected in an actual flight. The image processor uses active contours for fast visual tracking,

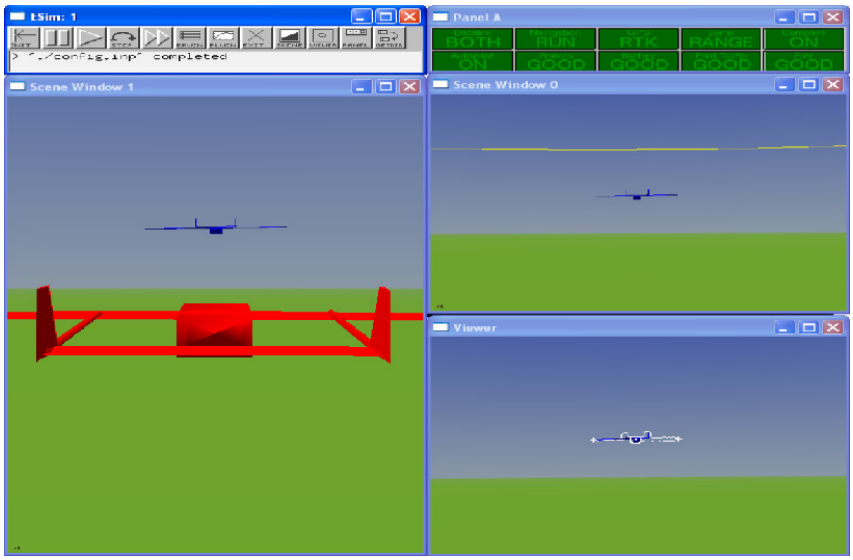

Fig. 6 Multi-aircraft simulation.

as discussed in Section II. The right top window in Fig. 6 shows a synthetic image in the simulator, and the right bottom window displays the image processing results.

The CORSE algorithm is then applied to a two aircraft formation flight, with the camera on the follower aircraft and the leader aircraft being the target. From the output of the image processor, the EKF estimates the leader's position and velocity relative to the follower. Then, using the optimal guidance policy, a relative position command is generated from the estimate.

Values of the parameters used in the simulation are shown below.

- Sampling time

$$\Delta t = 0.01$$

- Standard deviations of measurement noise

$$\sigma_{\xi y} = \sigma_{\xi z} = 1.0 \times 10^{-4}$$

- Standard deviations of process noise

$$\sigma_{\eta x} = \sigma_{\eta y} = 1.0 \times 10^{-4}$$

- Weight for the input u_Y

$$K_Y = \frac{16}{\sigma_{\xi y}} \left(\frac{1}{\sqrt{\sigma_{\xi y} K_Y}} = \frac{1}{4} \right)$$

- Initial condition

$$\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = \begin{bmatrix} 100 \\ 5 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} \mathrm{V_{X0}} \\ \mathrm{V_{Y0}} \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

- Terminal condition

$$\begin{bmatrix} X_f \\ Y_f \\ Z_f \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} V_{Xf} \\ V_{Yf} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Prior knowledge of all states is assumed to be available to initialize the EKF. The initial estimates are chosen by taking the true values and adding a random error of magnitude 1.0×10^{-4}.

3. Results

An example of the resulting trajectory and the guidance input u_Y are shown in Figs. 7 and 8. There is an obvious difference between the linear and optimal trajectories depicted in these figures. With the linear guidance law, the vehicle travels almost straight to the target and there is very little lateral motion. On the other hand, the optimal guidance law gives a meandering flight path to the target, always maintaining enough lateral velocity to estimate the range. While such a trajectory may limit the usefulness of the technique in some applications, such as docking or mid-air refueling, there are many other applications such as loose formation flying, or ground-based or airborne target tracking for which it is well suited.

Figure 9 displays examples of the range estimation errors for both the linear and optimal guidance cases. Under linear guidance, the lateral position Y and velocity V_Y both become small as the vehicle approaches the destination, resulting in a loss of range observability and a relatively large range estimation error. However, with the CORSE optimal guidance law, the range estimation error decreases to zero and the EKF provides an accurate range estimate. The estimation accuracy is also illustrated by the computed variance of the estimation errors. Figure 10 shows a comparison between time histories of the variance of the range estimation error, averaged over 100 trials, between the two cases.

The estimation errors of states other than the range are relatively small in both the linear and optimal guidance cases. Figures 11 and 12 show a comparison between the estimation and guidance results with and without the optimal guidance policy.

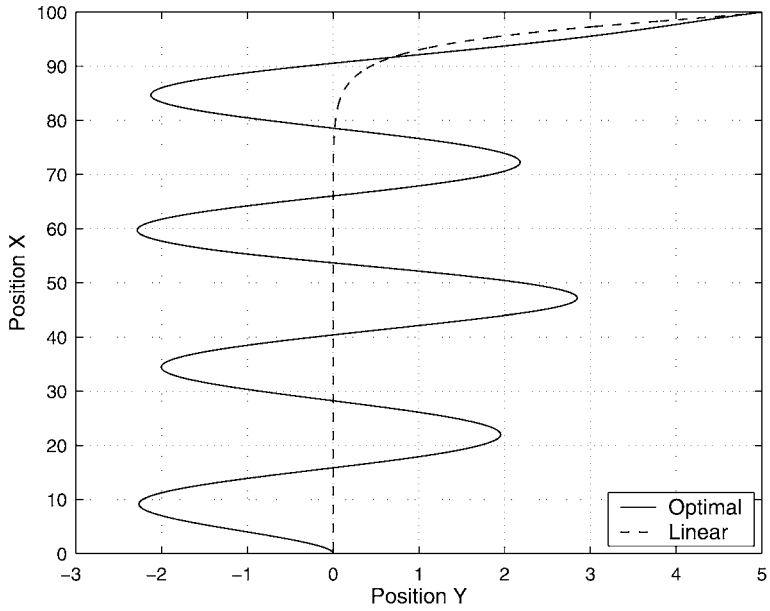

Fig. 7 Vehicle trajectory.

In this simulation, the follower aircraft changes its relative position from [$\Delta x = 100$ ft, $\Delta y = 10$ ft, $\Delta z = 0$ ft] to [$\Delta x = 50$ ft, $\Delta y = 0$ ft, $\Delta z = 0$ ft]. In one case (Fig. 11), the relative position command is given as a step command at a time of 20 seconds. In the other case (Fig. 12), the optimal path given by Equation (23) is utilized as the command from 20 seconds to a fixed terminal time 60 seconds. Without the optimal guidance policy, there remains a steady

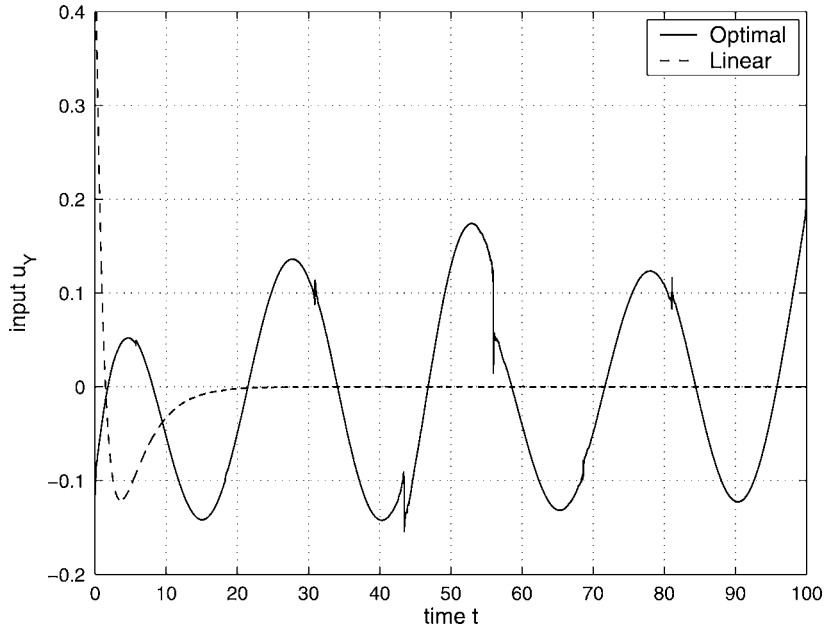

Fig. 8 Control input u_Y.

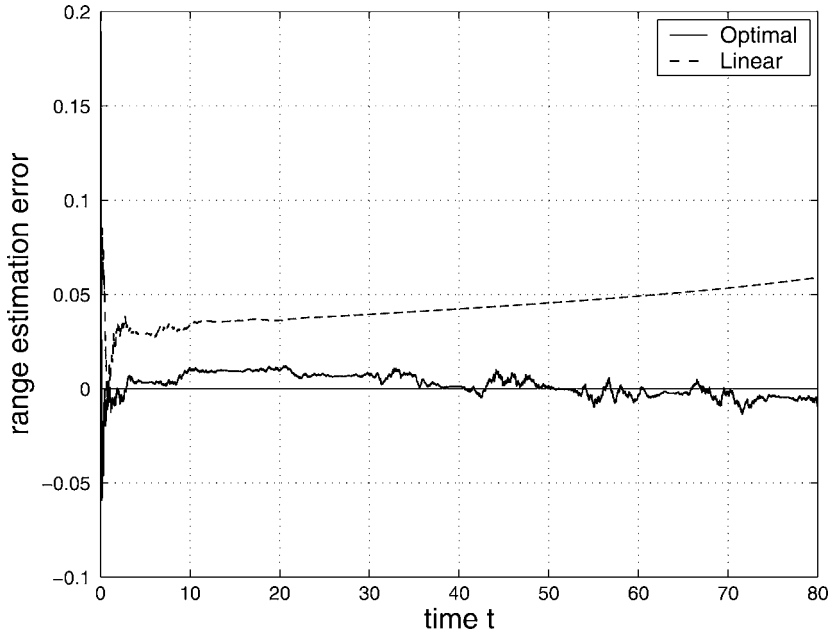

Fig. 9 Range estimation error.

state error in range, and the follower aircraft is guided to the position about 20 feet farther than its command. With the optimal guidance policy, the distance between two aircraft is accurately estimated and the follower is guided to the desired position. Figure 13 shows a profile of the variance of the range estimation error. Thus, it is observed that the optimal guidance policy accurately estimates range.

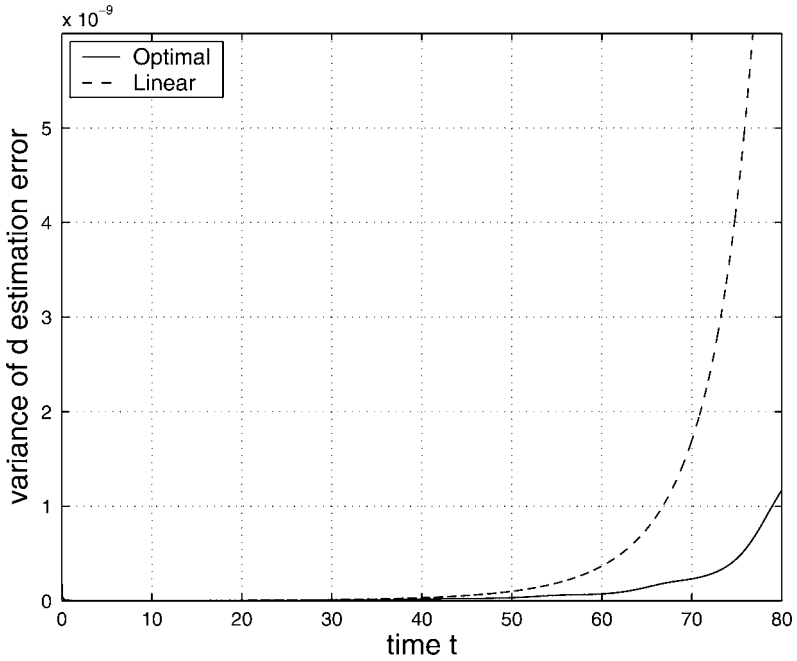

Fig. 10 Variance of range estimation error.

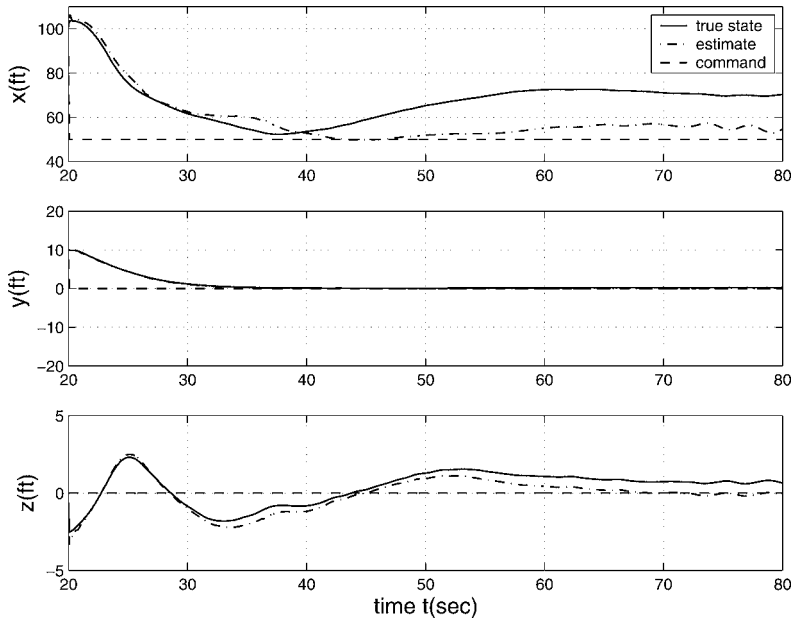

Fig. 11 Step command for position estimation without optimal guidance.

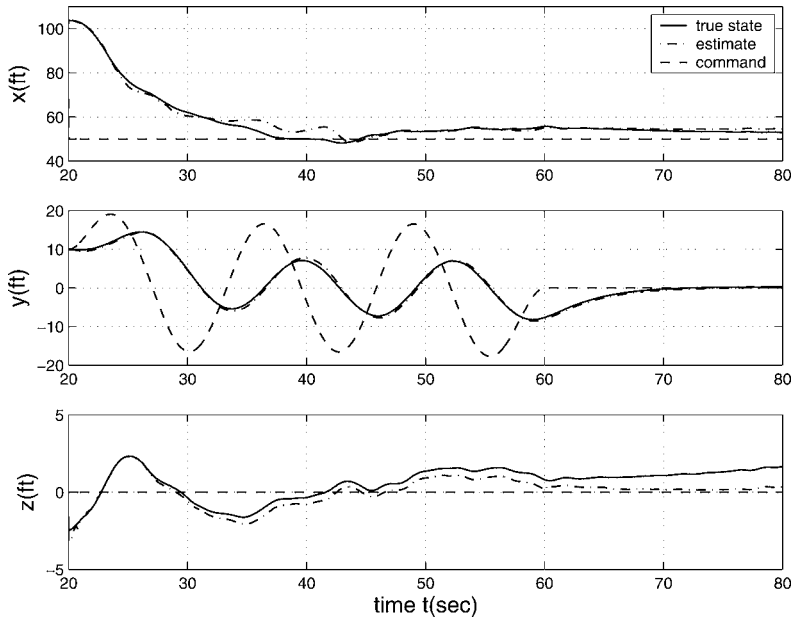

Fig. 12 Optimal path command for position estimation with optimal guidance.

V. Subtended Angle Relative State Estimation (SARSE)

A. The Measurement Model

The SARSE algorithm differs from the CORSE algorithm in several ways. One major difference lies in the level of image processing that is required. Unlike the CORSE algorithm's image processing, which determines

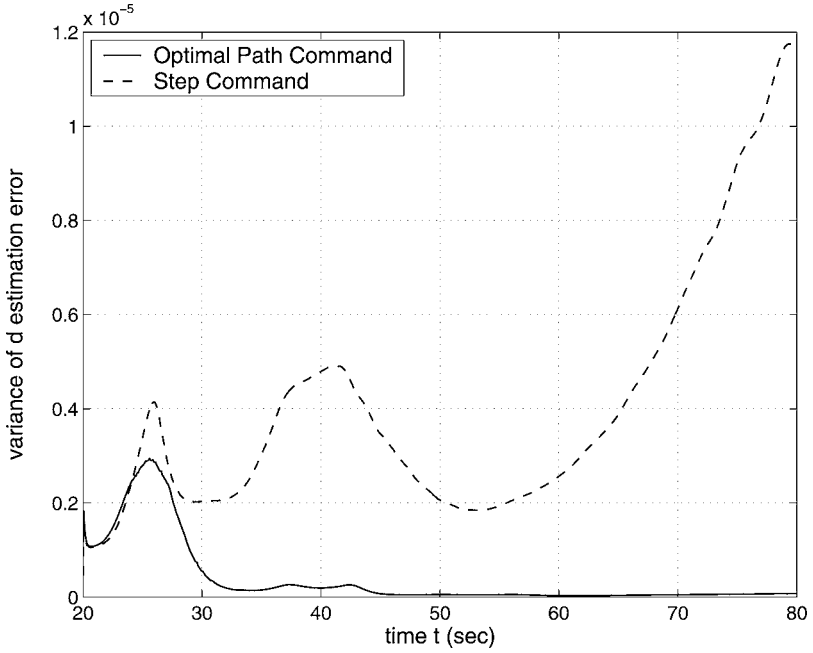

Fig. 13 Variance of range estimation error.

only the target center for a given image, the SARSE algorithm uses a more advanced technique to estimate the locations of the wing tips in the image as well. These locations are then used to calculate the angle whose vertex lies at the follower aircraft's center and which is subtended by the line between the target's wingtips (see Fig. 14).

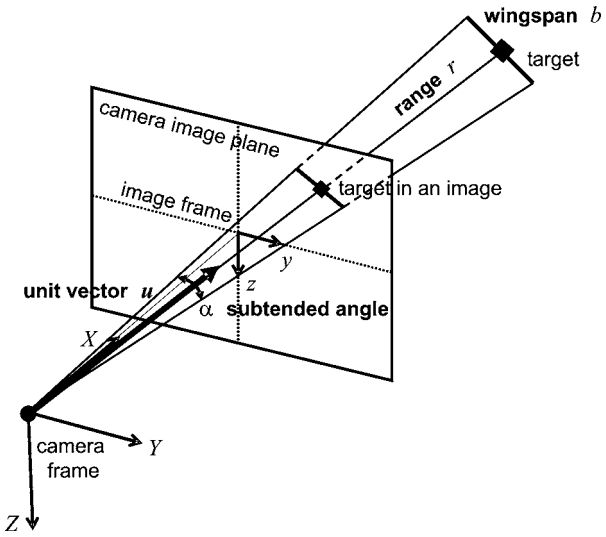

Fig. 14 Subtended angle relative to camera aircraft and target.

In the SARSE algorithm, the estimator states are

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} u_X \\ u_Y \\ u_Z \\ \dot{u}_X \\ \dot{u}_Y \\ \dot{u}_Z \\ 1/r \\ \dot{r}/r \\ b \end{bmatrix}$$

And the measurement derived from each image is

$$z_k = \begin{bmatrix} z_{1k} \\ z_{2k} \\ z_{3k} \\ z_{4k} \end{bmatrix} = \begin{bmatrix} u_{Xk} \\ u_{Yk} \\ u_{Zk} \\ \alpha_k \end{bmatrix} + \begin{bmatrix} \xi_{Xk} \\ \xi_{Yk} \\ \xi_{Zk} \\ \xi_{\alpha k} \end{bmatrix} = \begin{bmatrix} x_{1k} \\ x_{2k} \\ x_{3k} \\ 2\tan^{-1}(x_{7k}x_{9k}/2) \end{bmatrix} + \begin{bmatrix} \xi_{Xk} \\ \xi_{Yk} \\ \xi_{Zk} \\ \xi_{\alpha k} \end{bmatrix}$$

where r is a range to the target, $u = [u_X, u_Y, u_Z]$ is a unit vector pointing to the target and α is the angle subtended by the target's wingspan whose actual length is b. Measurement noise $\xi_k = [\xi_{Xk}, \xi_{Yk}, \xi_{Zk}, \xi_{\alpha k}]$ is modeled in the same manner as described in Section IV.A for the CORSE algorithm. Then, the measurement model is written as

$$z_k = h(x_k) + \xi_k \tag{24}$$

B. The Process Model

The process model of x is written as follows:

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ (-a_X(1-x_1^2) + a_Y x_1 x_2 + a_Z x_1 x_3)x_7 - 2x_4x_8 - x_1(x_4^2 + x_5^2 + x_6^2) \\ (a_X x_1 x_2 - a_Y(1-x_2^2) + a_Z x_2 x_3)x_7 - 2x_5x_8 - x_2(x_4^2 + x_5^2 + x_6^2) \\ (a_X x_1 x_3 + a_Y x_2 x_3 - a_Z(1-x_3^2))x_7 - 2x_6x_8 - x_3(x_4^2 + x_5^2 + x_6^2) \\ -x_7x_8 \\ -(a_X x_1 + a_Y x_2 + a_Z x_3)x_7 + x_4^2 + x_5^2 + x_6^2 - x_8^2 \\ 0 \end{bmatrix}$$

where $a = [a_X, a_Y, a_Z]$ is the vehicle acceleration and the target acceleration is assumed to be zero.

The acceleration a is modeled by

$$a = \begin{bmatrix} a_X \\ a_Y \\ a_Z \end{bmatrix} = \begin{bmatrix} \tilde{a}_X \\ \tilde{a}_Y \\ \tilde{a}_Z \end{bmatrix} + \begin{bmatrix} \eta_X \\ \eta_Y \\ \eta_Z \end{bmatrix}$$

where $\tilde{a}$ is control input.

The standard deviations for the process and measurement noise models in the SARSE algorithm were determined in the same manner as the CORSE algorithm.

Thus, the nonlinear process model can be written as

$$\dot{x} = f(x, \tilde{a}) + G(x)\eta \tag{25}$$

C. The Extended Kalman Filter

For the measurement and process models described in Equations (24) and (25), the extended Kalman filter is designed as follows:

Update

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + K_k(z_k - \hat{z}_k^-) \\ K_k &= P_k^- H_k^{\mathrm{T}}(H_k P_k^- H_k^{\mathrm{T}} + R_k)^{-1} \\ P_k &= P_k^- - K_k H_k P_k^- \end{aligned} \tag{26}$$

Prediction

$$\begin{aligned} \hat{x}_{k+1}^- &= \hat{x}_k + f(\hat{x}_k, a_{sk})\Delta t \\ P_{k+1}^- &= \Phi_k P_k \Phi_k^{\mathrm{T}} + Q_k \end{aligned} \tag{27}$$

where $\hat{x}_k$ is an updated estimate and $\hat{x}_k^-$ is a predicted estimate, R_k and Q_k are measurement and process noise covariance matrices, and P_k and P_k^- are their error covariance matrices. Φ_k and H_k are obtained from the process and measurement model as follows:

$$\Phi_k = I + \frac{\partial f(\hat{x}_k, a_k)}{\partial \hat{x}_k}\Delta t$$

$$H_k = \frac{\partial h(\hat{x}_k^-)}{\partial \hat{x}_k^-}$$

D. Simulation-Based SARSE Results

The image processor outputs new data whenever a camera image is updated. So the sampling time is not constant and is calculated at each iteration. It is generally about $\Delta t \approx 0.02$ seconds.

The process noise is assumed to be on the vehicle acceleration in each (body) axis. The values for the noise variance are set as

$$\sigma_X^2 = 0.03, \quad \sigma_Y^2 = 0.00015, \quad \sigma_Z^2 = 0.003$$

The measurement noise covariance matrix R_k is given by

$$R_k = \begin{bmatrix} 0.0001 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix}$$

The initial values for the EKF prediction and for its error covariance are set as

$$\hat{x}_0^- = \begin{bmatrix} \hat{x}_{01}^- \\ \hat{x}_{02}^- \\ \hat{x}_{03}^- \\ \hat{x}_{04}^- \\ \hat{x}_{05}^- \\ \hat{x}_{06}^- \\ \hat{x}_{07}^- \\ \hat{x}_{08}^- \\ \hat{x}_{09}^- \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.01 \\ 0 \\ 14.16 \end{bmatrix}, \quad P_0^- = I$$

The initial prediction implies that the target is 100 feet ahead of the camera without any relative motion, which is almost the same as the actual distance, and the target wingspan is assigned its actual value as well.

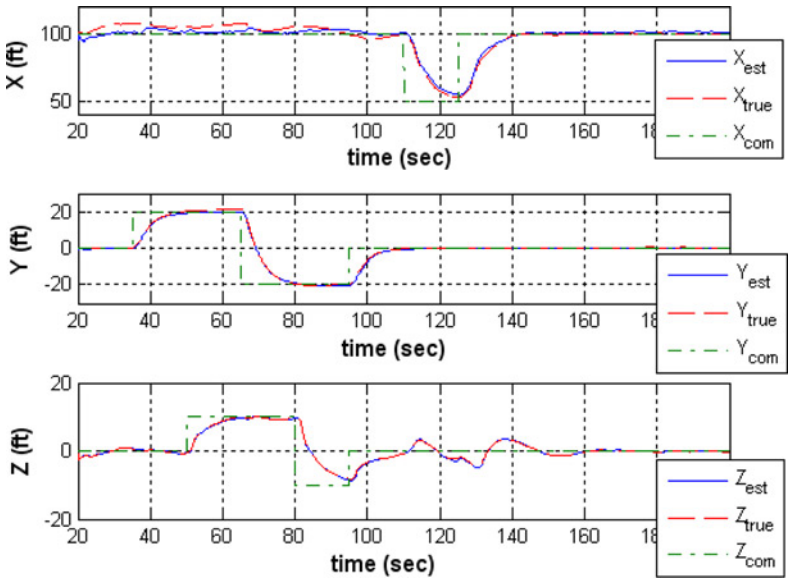

Fig. 15 Relative position estimation results.

The vehicle is guided by its relative position command. In this simulation, the commands are given as follows, with the time specified in seconds:

$$
\begin{aligned}
t < 35 \qquad &: p_c = [-100, 0, 0] \\
35 \le t < 50 &: p_c = [-100, -20, 0] \\
50 \le t < 65 &: p_c = [-100, -20, -10] \\
65 \le t < 80 &: p_c = [-100, 20, -10] \\
80 \le t < 95 &: p_c = [-100, 20, 10]
\end{aligned}
$$

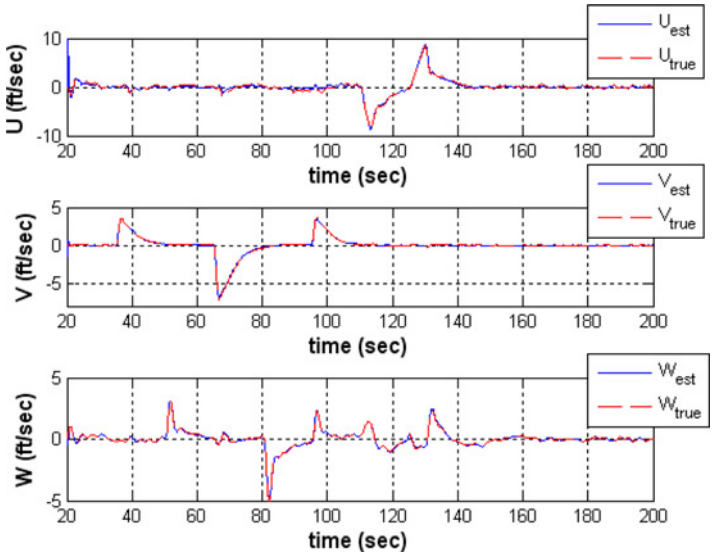

Fig. 16 Relative velocity estimation results.

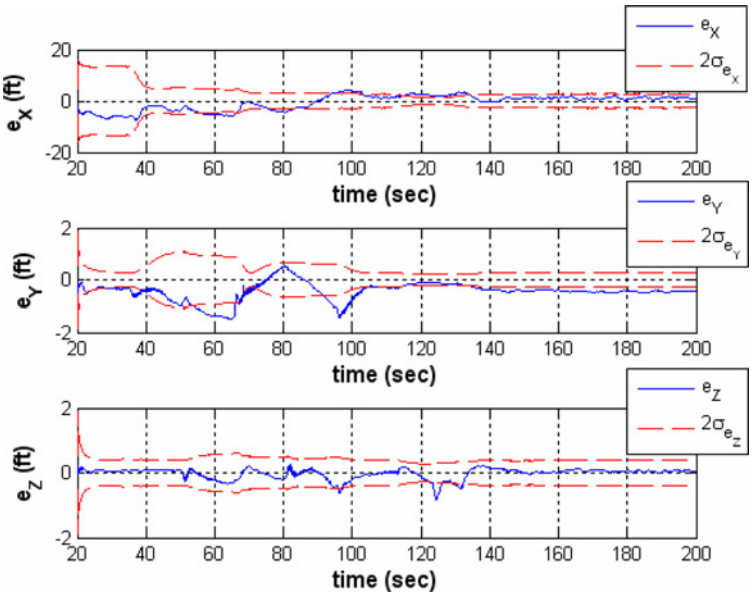

Fig. 17 Error in estimation of relative position.

$$95 \leq t < 110 : p_c = [-100, 0, 0]$$
$$110 \leq t < 125 : p_c = [-50, 0, 0]$$
$$t \geq 125 \quad : p_c = [-100, 0, 0]$$

The estimation and guidance results are shown in Figs. 15 and 16. Using the estimated value, the vehicle is guided to follow the relative position commands. (The steady state error in the x position is due to the flight controller.) Figures 17 and 18 show estimation errors for each state. The range estimation has the largest error among all the

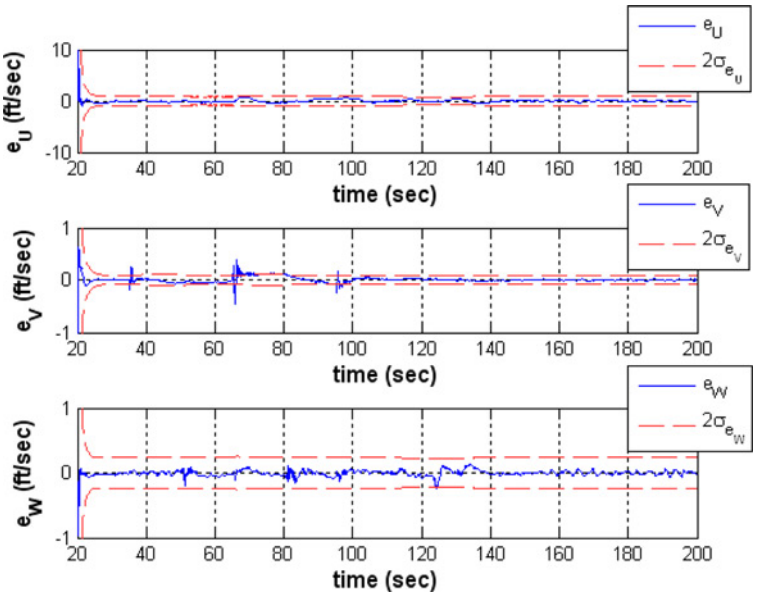

Fig. 18 Error in estimation of relative velocity.

states; however, its error remains less than 5% and is small enough that the estimated range value can still be used legitimately for vehicle guidance.

VI. Flight Test Results

A. Flight Test Hardware

The flight tests described here were image-based UAV formation flying tests where the GTMax rotorcraft UAV was flown in a non-homogenous formation with the GTEdge fixed-wing UAV. In these tests, the GTMax was the follower; it used the SARSE algorithm described above for navigation relative to the GTEdge which was the formation leader. Video of the actual flight test taken from a ground-based camera can be seen here: http://pdf.aiaa.org/JournalsOnline/PDFFiles/15429423_v4n4/aiaa/15429423/v4n4/Multimedia/23410Movie5.mpg and video of the same test taken from the camera on-board the follower aircraft can be seen here: http://pdf.aiaa.org/JournalsOnline/PDFFiles/15429423_v4n4/aiaa/15429423/v4n4>/Multimedia/23410Movie6.mpg

This section provides details on the flight hardware, including the the GTMax and GTEdge airframes, their associated avionics (and data describing how the SARSE algorithms performed on this hardware), and camera sensors used.

Georgia Tech's GTMax UAV (see Fig. 19) utilizes a Yamaha R-Max industrial helicopter airframe with the following characteristics:

- Rotor diameter: 10.2 feet
- Length: 11.9 feet (including rotor)
- Engine: gasoline, two-cylinder, water-cooled, 246 cc, 21 horsepower
- Max weight: 205 pounds
- Payload including avionics: >66 pounds
- Endurance of approximately 60 min (hover)
- Generator, battery, and electric starter
- Yamaha Attitude Control System (YACS).

The GTMax baseline flight avionics, including sensors and processing equipment, adds approximately 35 pounds to the basic airframe weight. The avionics configuration used for these tests includes:

- A 266 MHz Pentium II Embedded PC, 500 MB Flash Drive, and an 850 MHz Pentium III Embedded PC, 2 GB Flash Drive
- An Inertial Science ISIS-IMU Inertial Measurement Unit and a NovAtel OEM-4 differential GPS

Fig. 19 Georgia Tech's GTMax UAV.

Table 1 SARSE camera specifications.

Parameter	Value
Camera model	Sony FCB-EX780B
Weight	8.1 oz.
Size (L × W × H)	3.25 in. × 2 in. × 2.4 in.
Angle of View	45° (wide angle) to 2° (spot)
Effective pixels	680,000
Electrical power	1.6 W (motors inactive), 2.7 W (motors active)
Zoom	25 × optical, 12 × digital
Lighting conditions	2.5l × minimum, near IR capability

- A Honeywell HMR-2300, three-axis magnetometer, and an ultra-sonic sonar altimeter
- Vehicle telemetry (RPM, voltage, remote pilot inputs, low fuel warning) from YACS
- An actuator control interface to the YACS, 11 Mbps Ethernet data link and Ethernet switch, and a FreeWave 900 MHz spread-spectrum serial data link.

The power distribution system utilizes the onboard generator, which outputs 12V DC. It includes a hot-swappable connection to use external power and each component has a dedicated individual circuit breaker.

The flight software runs on the two onboard computers, referred to as "Onboard 1" (the 266 MHz Pentium II used as the primary flight computer) and "Onboard 2" (the 850 MHz Pentium III used as a secondary computer). The baseline guidance, navigation (GPS-based) and control algorithms are run on Onboard 1 while the SARSE algorithms are run on Onboard 2. The Ground Control Station (GCS) software normally runs on one or more laptop computers and allows system operators to interact with the onboard systems through the spread-spectrum data link.

Georgia Tech's GTEdge UAV uses a modified Aero-Works 33% Edge 540T airframe that has the following characteristics: Wing span: 105 inches; Length 94 inches; Wing planform area: 1870 sq. inches; Empty weight: 29 pounds; Desert Aircraft 9.8HP Engine; Engine weight: 5.8 pounds; Propeller size: 26 × 12, Electronic Ignition. The GTEdge baseline flight avionics configuration along with sensors and processing equipment includes:

- An FCS20 data acquisition and flight control computer featuring a 255 MHz TI-DSP with an integrated Altera EP1S40 FPGA, A/D converters, four RS232 ports, and 12 General Purpose 5V IO pins
- Three Axis IMU, Analog Devices +/−300 degrees Solid State Rate Gyros with a Temperature Sensor, and two 2-axis +/−10 g accelerometers (X, Y, Z, X2)

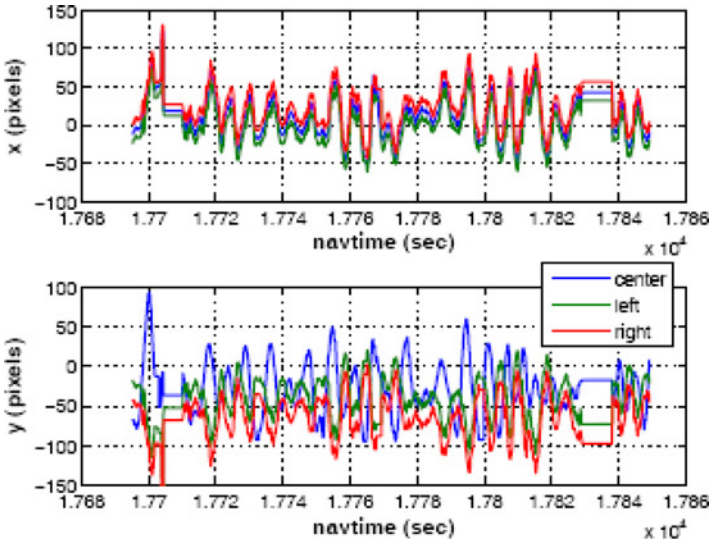

Fig. 20 Image processor outputs.

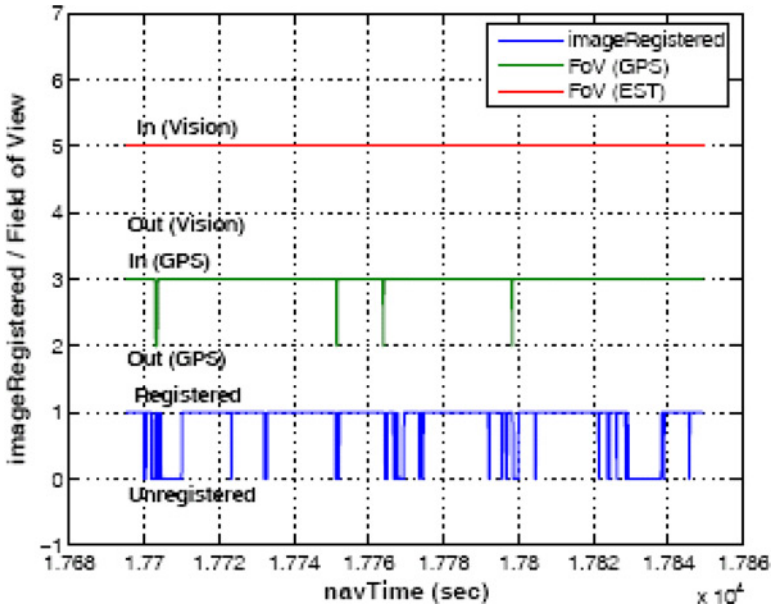

Fig. 21 Image registered in field of view.

- An extended Kalman filter based navigation system capable of estimating Euler Angles at 100 Hz
- μBlox TIM-LF GPS module, a FreeScale Airspeed Sensor, Ethernet data link and an Ethernet switch, and a FreeWave 900 MHz spread spectrum serial data link.

Both the GTMax and GTEdge systems are transported in an air-conditioned mobile Ground Control Station (GCS) equipped with a generator.

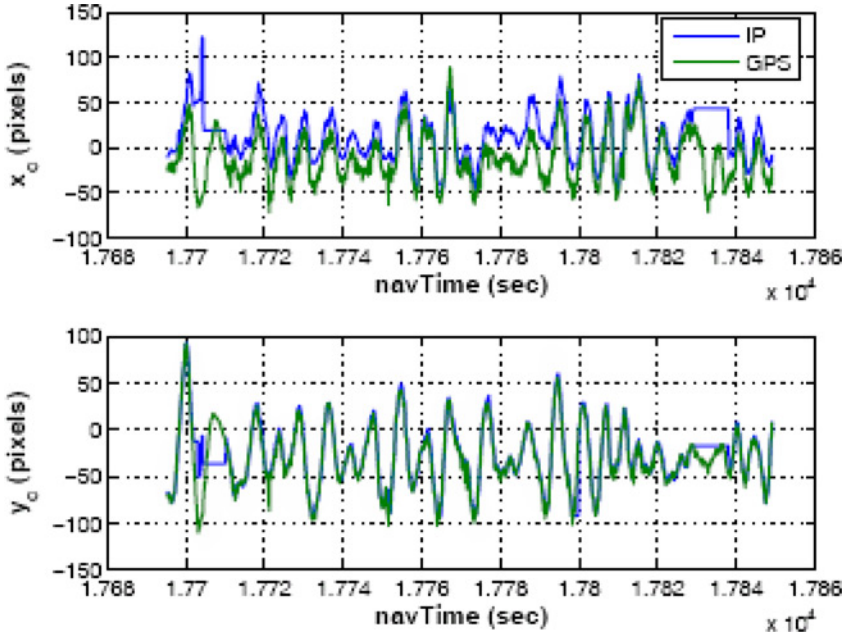

Fig. 22 Leader's center position (IP result vs. GPS estimate).

The camera sensor used to generate image data for the SARSE algorithms is a Sony FCB-EX780B color CCD block video camera. This camera features image stabilization, near IR capabilities, user configurable shutter control, and data transmission rates up to 38.4 Kb/sec. Some important specifications for the camera system are given in Table 1.

B. Image Processing

Figure 20 shows actual flight test data of the leader's center, and left and right wingtip positions which are detected by the follower's image processor in successive images. An "imageRegistered" flag is 1 when the image processor detects a target (i.e., the leader aircraft) and the flag is 0 when it does not. In Fig. 21, it is apparent that the image processor detects the target most of the time. Fig. 22 compares the leader's center position as detected by the image processor with values estimated using GPS data. Lateral position (x_c) has some bias, but vertical position (y_c) matches very well. The bias in x_c is probably due to the yaw angle difference of the leader and the follower.

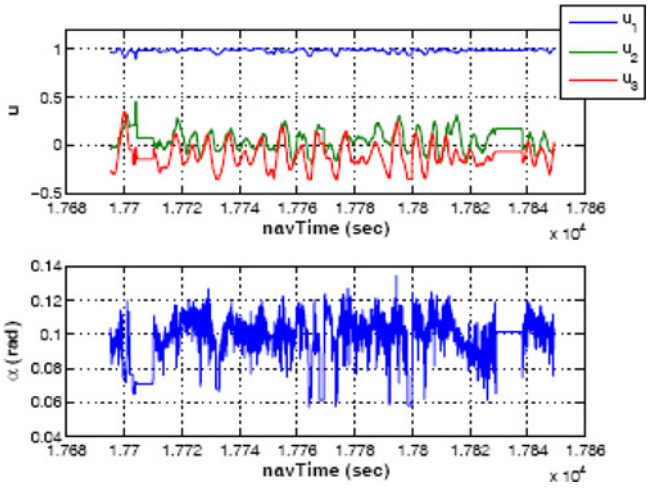

Fig. 23 Measurement vector for the SARSE EKF.

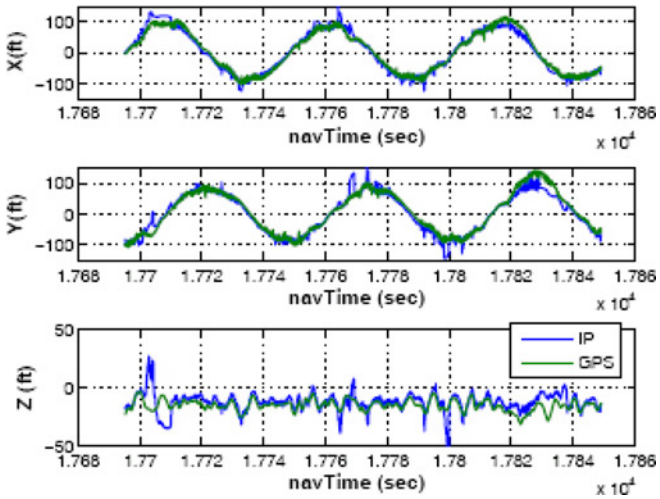

Fig. 24 Estimated relative position.

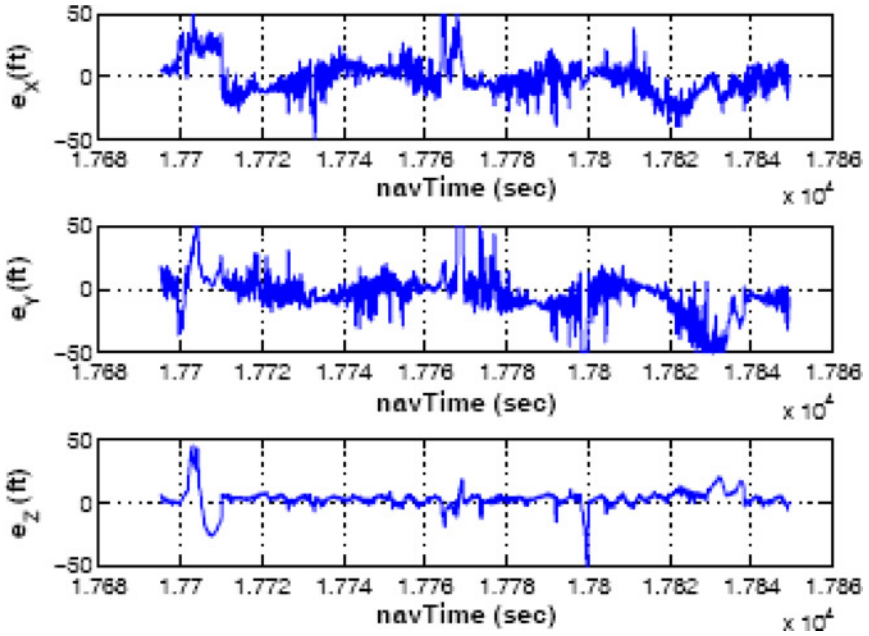

Fig. 25 Position estimation error.

C. EKF Measurements

Figure 23 is a measurement vector for the SARSE EKF, calculated by using the image processor outputs in Fig. 20. The measurements are a unit vector $\boldsymbol{u}$ from the camera to the leader's center in a camera frame and a subtended angle by the leader's wingspan as discussed in the description of the SARSE algorithm above.

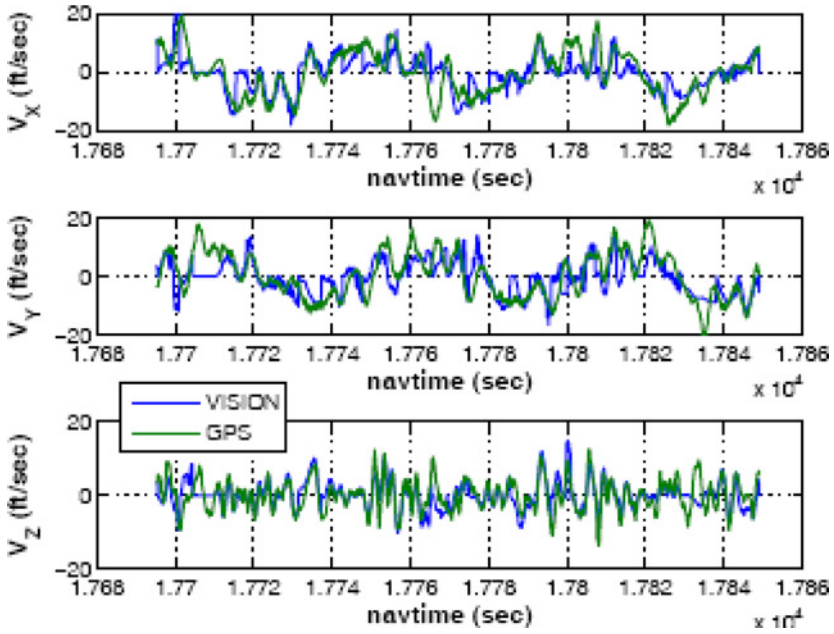

Fig. 26 Estimated relative velocity.

D. Relative State Estimates

Figures 24, 26, and 28 compare the Vision-estimated and GPS-estimated relative position, velocity and acceleration. Figures 25, 27, and 29 are the vision-based estimation errors assuming the GPS-based estimates as truth. Because of the range estimation error, the followers range tracking was somewhat inaccurate, however, the vertical position estimation error is relatively small, and thus was able to keep the leader in its field of view. This accurate Z estimate is because of the accurate y_c measurement from the image processor.

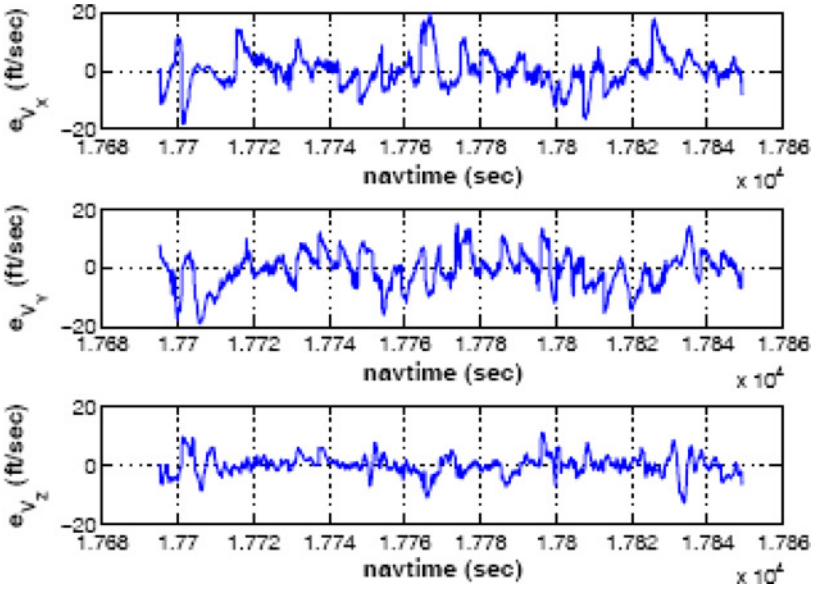

Fig. 27 Velocity estimation error.

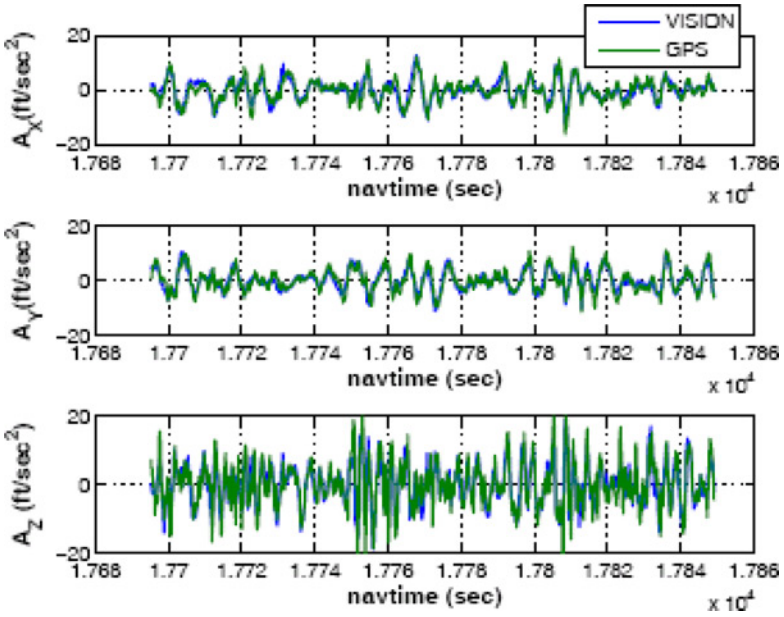

Fig. 28 Estimated relative acceleration.

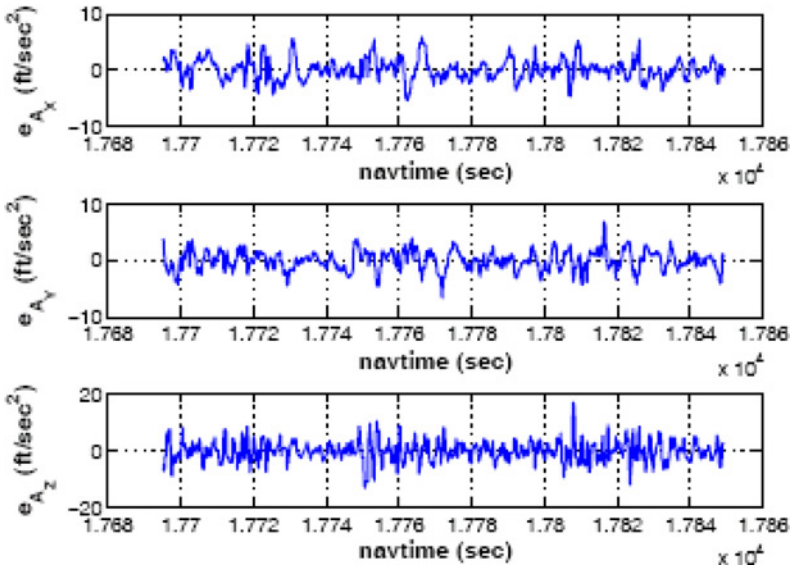

Fig. 29 Acceleration estimation error.

The actual algorithmic loop time of the SARSE algorithms running on the actual flight hardware (an 850 MHz Pentium III Processor) was approximately 0.1 seconds. In other words, the SARSE algorithm was run at approximately 10 Hz. Analysis of approximately 10 minutes worth of continuous flight data (∼6000 data points) showed that the actual mean value of the loop time was 0.1020 seconds with a standard deviation of 0.0159 seconds. A segment of this data is shown in Fig. 30. In Fig. 30 it can be seen that loop time was often less than 0.08 seconds and occasionally (less than 2% of the time) slightly higher than 0.24 seconds. The rare anomalous higher loop times are attributed to latency in the image processing algorithms.

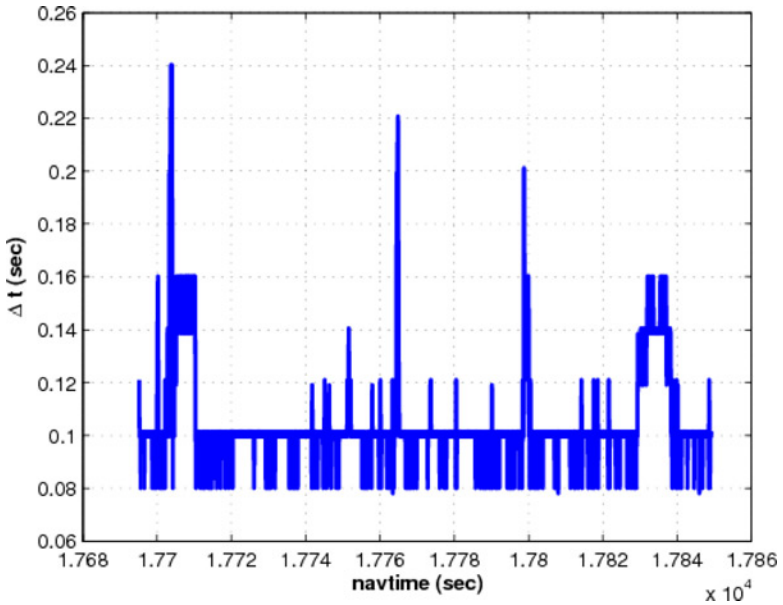

Fig. 30 SARSE algorithm loop time on a 850 MHz Pentium III Processor.

VII. Conclusions

In this paper, two successful vision-based techniques for the navigation of an aircraft relative to an airborne target using only information from a single camera were introduced. This represents an important contribution to the goal of reducing both cost and complexity of airborne navigation systems, most of which currently rely on sophisticated sensor and communication suites, such as GPS and/or Inertial Measurement Units (IMUs), to achieve similar results. Also, since the methods presented in this paper use only a single fixed camera, they will be of great interest to researchers developing autonomous micro-UAVs which can typically carry only very small, lightweight sensors and processing units with low power requirements.

The first method, Center Only Relative State Estimation (CORSE), guides the motion of the camera to maximize the usable information available in the image sequence to derive accurate range estimates. CORSE uses very simple image processing methods that output only the center of the target in each image.

The second algorithm, Subtended Angle Relative State Estimation (SARSE), uses a more complex image processing technique to derive more information from each camera image in order to successfully estimate the range. In particular, the image processing system is used to estimate the locations of the wingtips of the target in addition to just the location of the target center. Then, the angle subtended by the wingspan of the target, relative to the camera position, is used as an input to the EKF.

While both of these methods are successful at airborne vision-based relative state estimation using only a single vision sensor, they have distinct advantages and disadvantages. The CORSE algorithm uses very simple image processing whose only output is the center of the target aircraft in each image. This minimizes the amount of processing power necessary for the navigation system, which results in a system that has minimal sensor and processing requirements. This type of simple processing system is a very attractive feature for micro UAVs where minimal weight and volume are critical. The disadvantage of the CORSE algorithm is that it requires the aircraft to follow a flight path that allows observation of range. This would not be a valid option for systems in which many aircraft are flying in close proximity to one another, or if the target and following aircraft were flying through restrictive terrain. The SARSE algorithm uses a more complex image processing technique to locate both wingtips of the target in each image. Thus, it may require more powerful (and subsequently larger, heavier, and more expensive) onboard processing than the CORSE algorithm. However, using this enhanced image processing; SARSE can provide valid range estimates without requiring the aircraft to follow any particular path. Finally, flight test data results are presented. These results include EKF measurements, relative state estimates, estimation errors, and image processing lock data for experiments in which the SARSE algorithms were used to perform relative navigation and formation flying between two UAVs. Statistics show that the SARSE algorithms can be run consistently at 10 Hz on modern flight computer hardware.

Acknowledgments

This work was supported in part by AFOSR MURI, #F49620-03-1-0401 as well as grants from NSF, AFOSR, ARO, MRI-HEL, and an STTR through Georgia Tech. We also acknowledge Allen Tannenbaum who contributed to the development of the image processing algorithms.

References

[1]Frew, E., and Rock, S., "Trajectory Generation for Constant Velocity Target Motion Estimation Using Monocular Vision." *IEEE Int'l Conference on Robotics and Automation*, 2003.

[2]Metthies, L., and Kanade, T., "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences." *International Journal of Computer Vision*, Vol. 3, 1989, pp. 209–236.

[3]Ha, J., Alvino, C., Gallagher, P. Niethammer, M. Johnson, E., and Tannenbaum, A., "Active Contours and Optical Flow for Automatic Tracking of Flying Vehicles", American Control Conference, 2004.

[4]Aidala, V., and Hammel, S., "Utilization of Modified Polar Coordinates for Bearings-Only Tracking", *IEEE Transactions on Automatic Control*, AC-28(3):283.294, March 1983.

[5]Niethammer, M., and Tannenbaum, A., "Dynamic Level Sets for Visual Tracking", *Proceedings of the Conference on Decision and Control*, IEEE, Vol. 5, 2003, pp. 4883–4888.

[6]Blake, A., and Isard, M., *Active Contours*, Springer, New York, 1998.

[7]Caselles, V., Catte, F., Coll, T., and Dibos, F., "A Geometric Model for Active Contours in Image Processing." *Numerische Mathematik*, Vol. 66, 1993, pp. 1–31.

[8]Malladi, R., Sethian, J., and Vermuri, B., "Shape Modeling with Front Propagation: A Level Set Approach." *IEEE PAMI*, Vol. 17, 1995, pp. 158–175.

[9]Osher, S., and Sethian, J., "Fronts Propagation with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations." *Journal of Computational Physics*, Vol. 79, 1988, pp. 12–49.

[10]Sethian, J., "Curvature and the Evolution of Fronts." *Commun. Math. Phys.*, Vol. 101, 1985, pp. 487–499.

[11]Lauziere, Y., Siddiqi, K., Tannenbaum, A., and Zucker, S., "Area and Length Minimizing Flows for Segmentation." *IEEE Trans. Image Processing*, Vol. 7, 1998, pp. 433–444.

[12]Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A., and Yezzi, A., "Conformal Curvature Flows: from Phase Transitions to Active Vision." *Archive for Rational Mechanics and Analysis*, Vol. 134, 1996, pp. 275–301. A short version of this paper has appeared in the *Proceedings of ICCV*, June 1995.

[13]Sethian, J., "A Fast Marching Level Set Method for Monotonically Advancing Fronts", Proc. Nat. Acad. Sci., Vol. v93, No. 4, 1996, pp. 1591–1595.

[14]Press, W., Vetterling, W., Teukolsky, S., and Flannery, B., "Numerical Recipes in C, The Art of Scientific Computing", Cambridge University Press, 1988.

[15]Brown, R., Hwang, P., *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley & Sons, 1997.

[16]Bletzacker, et al., "Kalman Filter Design for Integration of Phase III GPS with an Inertial Navigation System, "Computing Applications Software Technology Technical Papers, Los Alamitos, CA, 1988.

[17]Gelb, A, et. al., *Applied Optimal Estimation*, The M.I.T. Press, 1974.

[18]Bryson, E. Jr., Ho, Y., *Applied Optimal Control—Optimization, Estimation, and Control*." Taylor & Francis, 1975.

[19]Johnson, E., and Kannan, S., "Adaptive Flight Control for an Autonomous Unmanned Helicopter." *AIAA Guidance, Navigation and Control Conference*, AIAA-2002-4439, 2002.

[20]Johnson, E., and Kannan, S., "Adaptive Trajectory-Based Control for Autonomous Helicopters", In the Proceedings of the AIAA Digital Avionics Systems Conference (DASC), 2002.

[21]Johnson, E., and Kannan, S., "Adaptive Trajectory Control for Autonomous Helicopters," AIAA Journal of Guidance, Control, and Dynamics, Vol. 28, No. 3, 2005.

[22]Sethian, J., "Level Set Methods and Fast Marching Methods.", Cambridge University Press, 1996.

[23]Comparison of Gyro-Stabilized Camera Mounts: http://www.camerasystems.com/gyrostabilization.htm.

Ella Atkins
Associate Editor