Fusion of Unmanned Aerial Vehicle Range and Vision Sensors Using Fuzzy Logic and Particles

Gregory L. Sinsley[*], Lyle N. Long[†], Brian R. Geiger[‡],
Joseph F. Horn[§], and Albert F. Niessner[¶]

The Pennsylvania State University, University Park, PA 16802.

This paper presents a novel method for fusing data from a UAV's range and vision sensors. The range sensor is used to build an elevation map of the flying area. Fuzzy logic is used to detect red barrels in camera images. The world location of a target on the ground is found by fusing the terrain map with image data using both an extended Kalman filter and a particle filter. The target detection system has been tested using images collected onboard a UAV. The terrain mapping system and the fusion system were both tested in simulation.

Nomenclature

(o_x, o_y)	Pixel location of image center
(x_{im}, y_{im})	Point in pixel coordinates
$\mathbf{M_{int}}$	Camera intrinsic parameter matrix
$\mathbf{R}$	Rotation matrix
$f(\cdot)$	State equation
f	Camera focal length
$h(\cdot)$	Measurement equation
H	Jacobian matrix
$h_{terrain}$	Terrain elevation
h_{uav}	UAV altitude
K	Kalman gain
M	Jacobian matrix
N	Number of particles
$p(\cdot)$	Probability density
P	Covariance matrix
R	Covariance matrix
r	Range
s_x	Size of a pixel in the horizontal direction
s_y	Size of a pixel in the vertical direction
u	Process noise
v	Measurement noise
w	Weight
x	State
Y	Set of measurements
y	Measurement
z	Vertical component of position

[*]Graduate Research Assistant, Electrical Engineering, AIAA Student Member.
[†]Distinguished Professor, Aerospace Engineering, AIAA Fellow.
[‡]Graduate Research Assistant, Aerospace Engineering, AIAA Student Member.
[§]Associate Professor, Aerospace Engineering, AIAA Associate Fellow.
[¶]Senior Research Associate, Emeritus, Applied Research Laboratory.

American Institute of Aeronautics and Astronautics

Subscripts

ABC	Aircraft-body coordinates
k	Time index
NED	North-east-down coordinates

Symbols

θ	Zenith bearing
η	Normalizing constant
μ	Mean
ϕ	Azimuth bearing
ϕ_r	UAV roll
ψ_y	UAV yaw
Σ	Covariance matrix
σ^2	Variance
θ_p	UAV pitch

Superscripts

n	Particle index

I.   Introduction

In order to increase the autonomy level of unmanned aerial vehicles (UAVs) it is critical to effectively utilize sensor data to create an accurate model of the vehicle's environment. Without such a model, a UAV will not have the data it needs in order to make intelligent decisions. Therefore, this work will focus on utilizing data from a range sensor and visual camera to identify and locate objects on the ground. This data may then be used by a path planner,[1,2] or an intelligent system such as the Penn State University Applied Research Lab Intelligent Controller,[3,4] SOAR,[5] or many others.[6]

A range sensor is used in order to build a map of the terrain that the UAV flies over. There are many possible uses for such a map. It could be used for finding a suitable landing area, ground vehicle path planning, or for topography. In this paper we use the terrain map primarily to aid in the geolocation of ground targets. A single image from a camera is a bearings-only measurement, so it can not uniquely locate a target. To uniquely locate a target requires a stereo camera, multiple images from a moving camera, or fusion with an additional sensor. In this paper, the third approach is used, although multiple images are also used to improve upon the location.

In order to have a good environment map, it is vital to know the locations of important objects. In this paper a camera is used to locate a set of red barrels which line the edge of the flying field runway. The red barrels were chosen because they can very easily be identified from the air. They are also a useful thing to identify because knowing where the red barrels are located will help the UAV find the runway. The red barrels are found by using a technique based on fuzzy logic known as a Continuous Inference Network (CINet).[7] A CINet is a network of fuzzy "and" and "or" nodes that determine a confidence that a certain property exists.

Oftentimes, a target will be visible in more than one image frame. If this is the case, it is possible to improve upon the original position estimate by including the information from subsequent measurements. The most popular method for doing so is the Kalman filter, and its nonlinear variation, the extended Kalman filter (EKF).[8] The Kalman filter is optimal for linear systems with Gaussian noise. The extended Kalman filter is suboptimal for nonlinear systems with non-Gaussian noise, but in practice works well if the nonlinearities are minor, and the noise is approximately Gaussian. A more recent approach is the particle filter.[9] Particle filters make no assumptions about the structure of a system or its noise characteristics. Instead of maintaining a state estimate as a mean and covariance like the Kalman filter does, a particle filter approximates the state and its probability distribution as a collection of particles.

II.   Related Work

There have been several studies on terrain mapping that are related to Unmanned Vehicles. One application uses light detection and ranging (LIDAR) to detect a hazard-free landing area for a Mars lander.[10] In the reference, a scanning LIDAR sensor is used to generate a grid of elevation values. The grid is then

American Institute of Aeronautics and Astronautics

analyzed to find rough areas that constitute hazards that should be avoided. Another interesting approach is found in Leal *et al.*[11] In this approach a 3D surface is represented as a distribution of particles. This approach appears to be very accurate and flexible, the biggest problem seems to be the computational demands. They have tested it in simulation and on a ground vehicle, but it appears it should work on an air vehicle with sufficient computational resources.

The use of color filtering to identify large red balls is discussed in Ross *et al.*[2] In the reference an image is converted to the Hue-Saturation-Value (HSV) color space. The image is then filtered in order to find pixels that have a hue below 30 degrees or above 315 degrees (hue wraps around the color scale) and a saturation above 0.3 (this excludes pixels that are "washed out"). This creates a hard classification. In this paper fuzzy logic is used in a similar manner to create a fuzzy classification.

Because surveillance is a major application for UAVs, locating and tracking objects on the ground has been an important area of research. Ross *et al.*[2] describes an algorithm for finding an object on the ground using a single camera frame when a terrain map is known. In Pachter *et al.*[12] an algorithm is described that accurately locates an object on the ground as a batch linear regression process. Their algorithm also estimates biases in the UAV's attitude sensors. In Monda *et al.*[13] an algorithm is developed to track a moving target using a UAV equipped with a gimbaled camera. This paper deals with the range ambiguity in camera images by assuming that the target is always at a known constant altitude. The target was tracked using both an extended Kalman filter and an unscented Kalman filter (a variation of the Kalman filter that represents the mean and covariance as a deterministic group of samples called sigma points). Finally, Campbell and Wheeler[14] describe a system based on an unscented Kalman filter for locating both stationary and moving targets. For the stationary target, the target's elevation could be assumed to be known, or it could be estimated using bearings measurements from the camera. The moving target's altitude was assumed to be known.

Particle filters are a relatively new addition to the nonlinear estimation field, although they are based on older techniques from other fields.[15] Despite how new they are, particle filters have already gained a great deal of popularity because of their flexibility, especially in dealing with systems with severe nonlinearities and non-Gaussian noise. A recent book by Ristic *et al.*[9] provides a good tutorial on particle filters, and discusses their applications to many important tracking problems such as bearings-only tracking, range-only tracking, and radar tracking. Particle filters have also found a variety of applications in robotics. Thrun *et al.*[16] describe applications of particle filters for robot localization and for simultaneous localization and mapping (SLAM) in their book. Particle filters have also found applications in distributed data fusion, such as in Ong *et al.*[17]

III.   Terrain Mapping

There are several options for range sensors. The most elaborate are scanning laser range finders, such as the ones made by SICK.[18] Another solution that is suitable for UAVs is Cloud Cap Technology's AGL laser altimeter.[19] This sensor interfaces directly with Cloud Cap's Piccolo autopilots, and provides very accurate measurements of the vehicles height above ground level. The AGL laser altimeter useful range is limited to 370 meters with a maximum range of 1000 meters. A third alternative and by far the cheapest is the use of a camera and optical flow. In Marlow and Langelaan[20] it is shown that given a moving camera's velocity and angular rotation, the range to a stationary obstacle can be determined using optical flow. Optical flow measurements tend to be very noisy, but have the advantage of using an inexpensive sensor. This paper simulates a general sensor that provides measurements of the form

$$z = h_{uav} - h_{terrain} + v_z \tag{1}$$

where z is the measurement, h_{uav} is the height of the UAV, $h_{terrain}$ is the height of the terrain directly below the UAV, and v_z is zero mean Gaussian noise, with variance σ_z^2.

In order to build the terrain map, the flying area is first divided into a grid. The finer the grid is, the more accurate the map will be, but at the expense of additional memory and processing requirements. In this paper, the grid is fixed at 10 meters by 10 meters. Each grid cell is initialized with a mean height μ_h and a height standard deviation σ_h. If some information about the terrain is known in advance, it should be reflected in these values. Otherwise, the mean should be given a reasonable value (for the flying area in this paper all grid cells are set to 390 meters above sea level), and the standard deviation should be given a large value (in this case 25 m).

American Institute of Aeronautics and Astronautics

Because the system is linear, and has no process noise or dynamics, it can be updated by a simple recursive least squares algorithm.[8] The fact that each grid cell is one dimensional (i.e. terrain height is the only relevant variable) makes the estimation problem even more simple. Each time a new estimate comes in, the grid is updated using the following algorithm:

1. Use range sensor to obtain an altitude measurement $z = h_{uav} - h_{terrain} + v_z$

2. Determine which grid cell the UAV is directly over.

3. Update the mean and variance of the grid cell using the recursive least squares estimator:

$$K \quad = \quad \frac{\sigma_h^2}{\sigma_h^2 + \sigma_z^2} \tag{2}$$

$$\mu_h \quad = \quad \mu_h + K[(h_{uav} - z) - \mu_h] \tag{3}$$

$$\sigma_h^2 \quad = \quad (1 - K)\sigma_h^2 \tag{4}$$

IV. Target Identification

The red barrels which line the flying area are detected using a CINet. The CINet for determining if a colored blob in an image is a red barrel is shown in Figure 1. One advantage of fuzzy logic is that it utilizes natural language statements. For instance, the barrel CINet can be read from right to left as follows, "an object is a barrel if it is high or low hue, high saturation, and the size of a barrel".

The first layer of the CINet is a "fuzzification" step that translates pixel values in the image into fuzzy confidence levels. For instance, in order to translate the hue of a pixel into a red confidence level the hue is passed through the trapezoidal membership functions shown in Figures 2(a) and 2(b). The trapezoidal membership function classifies hue values below 25 degrees as "definitely low" (confidence of 1), hue values between 25 degrees and 45 degrees as "possibly low", and hue values above 45 degrees as "definitely not low" (confidence of 0). Figure 2(c) shows a similar membership function for saturation. For determining if a blob is an appropriate size, the number of pixels in a colored blob is counted and passed through the membership function shown in Figure 2(d).

The second and third layers of the CINet represent fuzzy logic functions. Stover *et al.*[7] describes elaborate fuzzy "and" and "or" functions with weights. In this paper, much simpler membership functions are used. A min is used for "and", and a max is used for "or". These simple functions were found to work fine for identifying the red barrels.
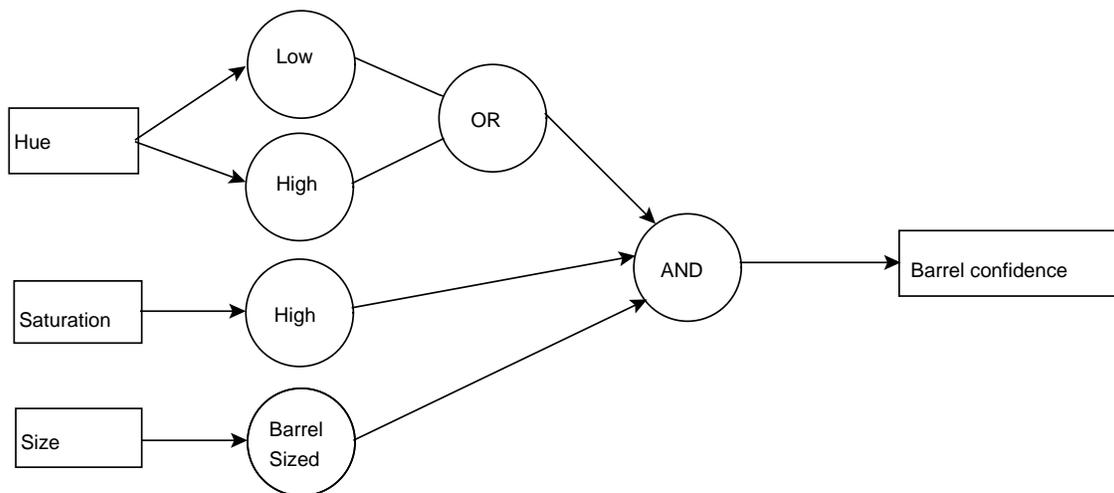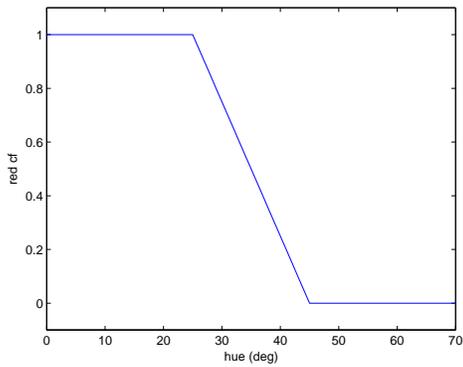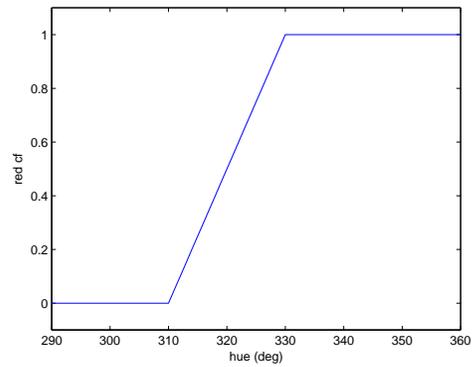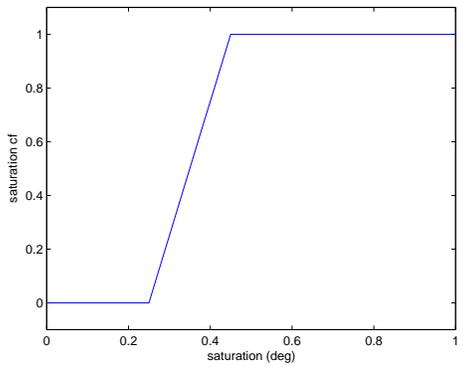

Figure 1. CINet for identifying a red barrel

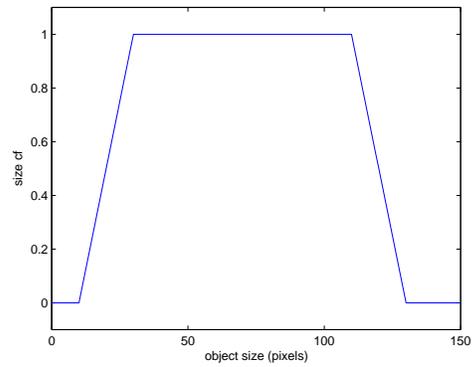American Institute of Aeronautics and Astronautics

(a) Low hue membership function

(b) High hue membership function

(c) High saturation membership function

(d) Barrel-sized membership function

Figure 2. Fuzzy membership functions

American Institute of Aeronautics and Astronautics

V.   Target State Estimation

State estimation can be performed using either an extended Kalman filter or a particle filter. Extended Kalman filters are relatively straightforward to design, and can be very computationally efficient, but suffer from the fact that they utilize a linear approximation to a nonlinear system. Particle filters are sometimes more accurate, but they are more difficult to design, and often requires significant memory space and processing time. These two techniques are discussed in more detail in the following sections.

V.A.   Extended Kalman Filter

The extended Kalman filter (EKF) is a nonlinear version of the popular Kalman filter. It works by linearizing the dynamics of a system about the current best guess at the target's state, then updating that estimate using the linear Kalman filter equations. Mathematical details of the EKF can be found in many textbooks including Simon.[8]

When a new target is detected, the filter must be initialized. This involves finding an estimate of the target's position, and the covariance of the uncertainty in the position. To estimate the position, first the pixel value of the target is converted to a set of bearings as follows

$$\phi \quad = \quad \arctan\left(\frac{(o_x - x_{im})s_x}{(o_y - y_{im})s_y}\right) \tag{5}$$

$$\theta \quad = \quad \arctan\left(\frac{\sqrt{(o_x - x_{im})^2 s_x^2 + (o_y - y_{im})^2 s_y^2}}{f}\right) \tag{6}$$

Where (x_{im}, y_{im}) is the target's location in pixel coordinates, (o_x, o_y) is the pixel location of the image center, s_x and s_y are the size of a single pixel (in millimeters) in the horizontal and vertical directions respectively, and f is the focal length of the camera (in millimeters). A discussion of these and other camera parameters can be found in an image processing book such as Trucco and Verri.[21]

Given a bearing to the target and a range to the target, the target's location can be uniquely determined. Unfortunately a range is not available, all that is available is an estimate of the UAVs height above ground level. Therefore, the target's location must be determined by the following iterative procedure:

1. Express the initial guess of the target's location in UAV aircraft-body coordinate (ABC) frame as follows

$$z \quad = \quad h_{uav} - h_{terrain} \tag{7}$$

$$x \quad = \quad z\tan(\theta)\cos(\phi) \tag{8}$$

$$y \quad = \quad z\tan(\theta)\sin(\phi) \tag{9}$$

   where $h_{terrain}$ is the terrain map's estimate of the height of the terrain directly below the UAV. Note that these equations assume that the camera's axis align with the UAV's. If this is not the case, an additional rotation will have to be performed.

2. Rotate the ABC location into the UAV's north-east-down (NED) frame using the following Equation[22]

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{NED} = \mathbf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{ABC} \tag{10}$$

   where

$$\mathbf{R} = \begin{bmatrix} \cos(\psi_y) & -\sin(\psi_y) & 0 \\ \sin(\psi_y) & \cos(\psi_y) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_p) & 0 & \sin(\theta_p) \\ 0 & 1 & 0 \\ -\sin(\theta_p) & 0 & \cos(\theta_p) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi_r) & -\sin(\phi_r) \\ 0 & \sin(\phi_r) & \cos(\phi_r) \end{bmatrix} \tag{11}$$

   and ϕ_r, θ_p, ψ_y are the UAV roll, pitch, and yaw, respectively.

American Institute of Aeronautics and Astronautics

3. Set $h_{terrain}$ to the terrain map's estimate of the height of the terrain at location (x, y)

4. Check for convergence (i.e. the new estimate of the target's location is less than a certain distance from the old estimate) and either exit or return to step 1.

Once the target's position in the UAV NED frame is found, it is translated into local coordinate frame that is fixed at a given point on the flying field, this avoids the complexity of the target moving as the UAV moves.

The initial uncertainty in the state estimate is somewhat difficult to estimate. There are many possible sources of error including the UAV's location, roll, pitch, and yaw, the uncertainty in the pixel location of the target, and the uncertainty in the terrain map. Even if all these errors are independent and Gaussian, they can create correlated, non-Gaussian errors in the target location due to the nonlinearity of the measurement. Fortunately, it has been found that setting the diagonal elements of the covariance matrix to some large value (on the order of 100 meters squared), and the off-diagonal elements to some small value (on the order of 10 meters squared) works just fine.

Every time a new measurement comes in, the target's state estimate is updated using the extended Kalman filter equations. Normally, the EKF is used on a system with the following state and measurement equations

$$x(k) = f(x(k-1), u(k-1)) \tag{12}$$
$$y(k) = h(x(k), v(k)) \tag{13}$$

Where $x(k)$ is the state at time k, $y(k)$ is the output at time k, and $w(k)$ and $v(k)$ are zero-mean Gaussian noise. Since we are only interested in a stationary target, however, the first equation can be omitted. This leaves us with the equation

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix} = h(x_{target}, x_{uav}(k), w(k)) \tag{14}$$

where dependence on the UAV's state has been made explicit. The function h can be determined from the camera model described in section VI.B. If the UAV is directly over the target, such that the range sensor is returning a range to target, a third measurement can be added so that

$$\begin{bmatrix} \phi \\ \theta \\ z \end{bmatrix} = h(x_{target}, x_{uav}(k), w(k)) \tag{15}$$

From one of these two equations, we can calculate two matrices of partial derivatives

$$H = \left. \frac{\partial h}{\partial x_{target}} \right|_{x_{target}(k)} \tag{16}$$

$$M = \left. \frac{\partial h}{\partial v} \right|_{x_{target}(k)} \tag{17}$$

The second matrix should account for all sources of uncertainty, but due to the difficulty of doing so, it is often assumed that the measurement noise is additive, in which case

$$y = h(x_{target}, x_{uav}(k)) + w(k) \tag{18}$$

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{19}$$

Also the covariance of the noise must be considered. It is assumed that v is zero mean and Gaussian and has covariance

$$R = E(vv^T) \tag{20}$$

American Institute of Aeronautics and Astronautics

Given the previous estimate of the state $\hat{x}_{k-1}$ and the covariance of the estimate, P_{k-1}, the EKF can be updated at each time step using the following equations

$$
\begin{align}
K_k &= P_{k-1}H^T(HP_{k-1}H^T + MRM^T)^{-1} \tag{21}\\
\hat{x}_k &= \hat{x}_{k-1} + K_k[y_k - h(\hat{x}_k)] \tag{22}\\
P_k &= (I - K_kH)P_{k-1} \tag{23}
\end{align}
$$

V.B. Particle Filter

The particle filter serves as a general approximation to a recursive Bayes filter.[8, 16] The Bayes filter seeks to calculate $p(x|Y_k)$, the probability density of a state at time k, x_k, conditioned on the measurements up to time k:

$$
Y_k = \{y_1, y_2, ..., y_k\} \tag{24}
$$

using Bayes rule. This probability is very difficult to calculate in practice. For a linear system with Gaussian noise, the Kalman filter calculates the probability. For other systems, some form of approximation must be used. The particle filter approximates this distribution as a set of particles (samples). Following the procedure in Thrun *et al.* the particle filter algorithm may be stated as follows:

1. Initialize the particle filter by drawing N samples from the known probability density $p(x_0)$.

2. At each time step k, draw a sample from the state transition probability density for each particle $x_{k-1}^{[n]}$

$$
x_k^{[n]} \sim p(x_k|u_k, x_{k-1}^{[n]}) \tag{25}
$$

3. Weight the new samples according to the latest measurement, z_k

$$
w_k^{[n]} = p(z_k|x_k^{[n]}) \tag{26}
$$

4. Generate N new samples by drawing a particle at random (with replacement) with probability proportionate to $w_k^{[n]}$

The last step in the procedure is known as the resampling or importance sampling step. The resampling step is important in that it ensures that the final particle distribution accurately approximates $p(x|Y_k)$. Unfortunately, the resampling step can reduce the information in the particle set. In the worst case, known as sample impoverishment, all but one particle may be eliminated. This is especially true in systems with low process noise, such as tracking a stationary target. In this case

$$
p(x_k|u_k, x_{k-1}^{[n]}) = p(x_{k-1}) \tag{27}
$$

so sampling $x_k^{[n]}$ from this distribution will not induce any diversity into the sample set. The easiest way to counteract this problem is to only resample occasionally instead of at every step. In this case, the weight of each particle will be updated by the following equation during steps where resampling does not take place

$$
w_k^{[n]} = p(y_k|x_k^{[n]})w_{k-1}^{[n]} \tag{28}
$$

In this paper, resampling only takes place when a target that has been in the camera's field of view for multiple image frames finally leaves the camera's field of view.

Another way to add more diversity to the particle set is to add a small number of the particles according to the measurement model instead of the motion model as follows

$$
x_k^{[n]} \sim p(y_k|x_k) \tag{29}
$$

American Institute of Aeronautics and Astronautics

It is difficult to determine what the importance weight of these new particles should be. In Thrun *et al.* it is suggested that the weights be determined by the integral

$$w_k^{[n]} = \int p(x_k^{[n]}|u_k, x_{k-1})bel(x_{k-1})dx_{k-1} \tag{30}$$

where $bel(x_{k-1})$ is the belief calculated at the previous step. This integral can be difficult to compute, but the authors have found that simply setting the weights of these new particles to the average of the weights of the previously generated particles works well.

Given these mathematical preliminaries, a particle filter can be applied to tracking a stationary target using a terrain map and a visual camera. The first time a target is seen, it is initialized as follows:

1. Convert the camera's pixel measurement to a bearing measurement using Equation 5.

2. Use the iterative procedure described in Section V.A to determine an initial guess of the target's position in the NED frame.

3. Generate N samples of the following random variables

$$\begin{align}
v_h &\sim N(0, \sigma_h^2) \tag{31}\\
v_\phi &\sim N(0, \sigma_\phi^2) \tag{32}\\
v_\theta &\sim N(0, \sigma_\theta^2) \tag{33}
\end{align}$$

where σ_h^2 is the uncertainty in the terrain map, as described in Section III and σ_ϕ^2 and σ_θ^2 are the noise in the bearing estimates, which is dependent on the camera and the image processing routine.

4. From the N samples of the noise variables, generate the ABC coordinates of the N particles using the following equations:

$$\begin{align}
\tilde{z} &= h_{uav} - h_{terrain} + v_h \tag{34}\\
\tilde{\phi} &= \phi + v_\phi \tag{35}\\
\tilde{\theta} &= \theta + v_\theta \tag{36}\\
\tilde{x} &= \tilde{z}\tan(\tilde{\theta})\cos(\tilde{\phi}) \tag{37}\\
\tilde{y} &= \tilde{z}\tan(\tilde{\theta})\sin(\tilde{\phi}) \tag{38}
\end{align}$$

5. Transform the particle coordinates from the ABC frame to the NED frame using Equation 10, then translate the target location to the local frame.

6. Set the weight of each particle to

$$w^{[n]} = \frac{1}{N} \tag{39}$$

Now the that filter is initialized, it can be updated every time a new measurement comes in using the following procedure:

1. Generate $\tilde{n}$ new particles using the procedure described previously.

2. Set the likelihood of each of these particles to $\frac{1}{N}$ where N is the previous number of particles.

3. Update the the value of N

$$N_{new} = N_{old} + \tilde{n} \tag{40}$$

4. For each particle, n, determine the expected measurement

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix}_{expected}^{[n]} = h(x^{[n]}, x_{uav}, 0) \tag{41}$$

where h is the measurement equation described in Section V.A.

5. Evaluate the likelihood of each particle given the latest measurement

$$p(y_k|x_k^{[n]}) = \eta \exp\left\{ -\frac{1}{2}\left(\begin{bmatrix} \phi \\ \theta \end{bmatrix} - \begin{bmatrix} \phi \\ \theta \end{bmatrix}_{expected}^{[n]}\right)^T \Sigma^{-1} \left(\begin{bmatrix} \phi \\ \theta \end{bmatrix} - \begin{bmatrix} \phi \\ \theta \end{bmatrix}_{expected}^{[n]}\right) \right\} \quad (42)$$

where η is a normalizing constant that ensures that p is a valid probability density, and the noise covariance Σ is given by

$$\Sigma = \begin{bmatrix} \sigma_\phi^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \quad (43)$$

6. Update the weight of each particle using Equation 30.

7. Normalize all the weights using the following formula

$$w_k^{[n]} = \frac{w_k^{[n]}}{\sum_{\hat{n}} w_k^{[\hat{n}]}} \quad (44)$$

This step is not strictly necessary, but it does prevent weights from growing smaller and smaller as time goes on.

Finally, once the target leaves the UAV's field of view, perform a resampling step. First reset N to its original value. Then do the following N times

1. Choose one of the particles n with probability proportionate to $w_k^{[n]}$.

2. Add the chosen particle to the new set.

3. Replace the chosen particle in the old set (i.e. particles may be picked more than once).

VI. Simulation Description

In order to validate the fusion system in this paper prior to flight testing it onboard a UAV, a Matlab simulation was designed. The simulation of the UAV is quite simple. It is assumed that the UAV will fly a raster scan pattern about the flying area at a constant speed and altitude. It is also assumed that the UAV will be flying with wings level while over the target area. The reason for this simplifying design choice was so that the simulation could focus on the sensors and the fusion system without worrying too much about UAV dynamics. If a more faithful vehicle dynamic simulation is required, a simulation such as the one in Ross *et al.*[2] can be used. In order to simulate noise in the vehicle's GPS measurements, the fusion system only has access to a noise-corrupted version of the vehicle's position.

VI.A. Range Sensor Model

The first sensor that must be modeled is the range sensor. The range sensor is expected to provide measurements of the form

$$z = h_{uav} - h_{terrain} + v_z \quad (45)$$

where v_h is zero mean Gaussian noise. The variance of the noise will depend on what type of sensor is being used. A laser has an accuracy less than meter, whereas optical flow will be much nosier. For the results presented in Section VII.B, a standard deviation of 1 meter was used.

The simulated range sensor used actual terrain elevation information from the United States Geological Survey (USGS) National Map Seamless Server.[23] The data from USGS was converted to a ArcInfo ASCII Grid data file. This file contains a grid of elevation values at 1/3 arcsecond (about 10 meter) increments. Given this information, a simulated range measurement can be created by the following procedure:

1. Determine latitude, longitude, and altitude of the UAV.

2. Determine which grid cell that the UAV is inside of.

American Institute of Aeronautics and Astronautics

3. Calculate the true range

$$r = h_{uav} - h_{terrain} \qquad (46)$$

4. Generate a random number, $v \sim N(0, \sigma_h)$.

5. Generate the corrupted measurement

$$z = r + v \qquad (47)$$

VI.B. Camera Model

The camera model is meant to simulate a generic color camera with a resolution of 640x480. The simulation is meant to supply the fusion system with a noise corrupted pixel location of the target, similar to the results the image processing program described in Section IV would return. The camera model takes as an input the actual world location of a target (as determined by a handheld GPS unit) and the world location of the UAV (from the UAV simulation). Using this information and values for the camera intrinsic parameters described in an image processing book such as Trucco and Verri,[21] the camera model determines the pixel coordinates of the target through a series of coordinate transformations.

For the purposes of the simulation, assume that both the UAVs position and the target's position are expressed in a north-east-down coordinate frame with its origin at the lower right corner of th flying area. If this is not the case, they can be transformed using equations such as those found in Stevens and Lewis.[22] Given the position of the target and the UAV in the local NED frame, the location of the target in the UAV's NED frame is given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{NED} = \begin{bmatrix} n \\ e \\ d \end{bmatrix}_{target} - \begin{bmatrix} n \\ e \\ d \end{bmatrix}_{UAV} \qquad (48)$$

Next, the target's location is rotated into the UAV's aircraft-body coordinate (ABC) frame by the following equation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{ABC} = \mathbf{R}^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{NED} \qquad (49)$$

where $\mathbf{R}$ is given in equation 11. If the simplifying assumption is made that the UAV is flying wings level with no angle of attack, the equation for $\mathbf{R}^{-1}$ takes the much simpler form

$$\mathbf{R}^{-1} = \begin{bmatrix} \cos(\psi_y) & \sin(\psi_y) & 0 \\ -\sin(\psi_y) & \cos(\psi_y) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (50)$$

where ψ_y is the yaw of the UAV.

Now that the point is expressed in the UAV's coordinate frame, it must undergo a nonlinear transformation into pixel coordinates.[21] The transformation is given by

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{M_{int}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{ABC} \qquad (51)$$

where $\mathbf{M_{int}}$ is the camera intrinsic parameters matrix given by

$$\mathbf{M_{int}} = \begin{bmatrix} -\frac{f}{s_x} & 0 & o_x \\ 0 & -\frac{f}{s_y} & o_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (52)$$

American Institute of Aeronautics and Astronautics

where (o_x, o_y) is the pixel location of the image center, s_x and s_y are the size of a single pixel (in millimeters) in the horizontal and vertical directions respectively, and f is the focal length of the camera (in millimeters). The pixel coordinates of the image point are then given by

$$x_{im} = \frac{x_1}{x_3} \tag{53}$$

$$y_{im} = \frac{x_2}{x_3} \tag{54}$$

Finally, to simulate inaccuracy in the pixel coordinates, zero mean Gaussian noise is added to x_{im} and y_{im}.

VII. Results

VII.A. Target Identification

The target identification subroutine was tested using flight data collected onboard a UAV equipped with a webcam. The UAV platform is based upon a modified SIG Kadet Senior model aircraft (Figure 3). The webcam used is a Logitech QuickCam with a resolution of 640x480. More information on the UAV platform can be found in Sinsley *et al.*[3, 4]

An example image from the webcam is shown in Figure 4(a). Figure 4(b) shows the results of passing this image through the membership functions "low hue or high hue", i.e. it finds all pixels with a red hue. White pixels are definitely red, black pixels are definitely not red, and gray pixels are in between. Note that a number of pixels that don't appear to be red in the original image, are nonetheless marked as red. This is due to poor lighting. The saturation function takes care of this; Figure 4(c) shows the same image passed through the saturation membership function, now all the "washed out" pixels are excluded. Figure 4(d) shows the results of passing the two images through a fuzzy and. Notice that this image looks a lot like the saturation image. This is because for most pixels the saturation confidence is lower than the hue confidence. Since a fuzzy "and" is a minimum, the lower value dominates. If this had been a well lit image with a lot of colors, hue would have dominated the calculation instead. Finally, Figure 4(e) shows the final processed image with the three red barrels circled.

The accuracy of the barrel detection routine was fairly good for such a simple routine. In a sequence of 105 images, with an average of 2 barrels per image, there were about 10 false alarms and 10 missed detections. The missed detections are not a very serious concern since most barrels appear in multiple images, and are usually detected in at least one of those images. The false alarms are a greater concern because they can start false tracks (i.e. start to track the location of an object that does not exist) or they can incorrectly update an existing track. The best way to deal with this is to check the fuzzy confidence of a detection to determine whether or not to accept it.

The biggest problem with the barrel detection routine is its speed. Using Matlab on a computer with a 1.8 GHz processor, the routine took an average of 2 seconds to process a single image. Probably the biggest reason for this slow speed was the use of Matlab. In Ross *et al.*[2] a similar routine running in c++ using Intel's OpenCV computer vision library[24] achieved real time performance. The use of fuzzy logic is also most likely slower than the hard thresholding method used in the reference.

VII.B. Terrain Mapping

The terrain mapping function has so far been tested in simulation as described in Section VI.A. To evaluate the performance of the terrain mapping, the UAV flew a raster scan flight pattern about the flying area and recorded range measurements 10 times a second. The generated map after one complete flight pattern was then compared to terrain data given in the USGS Seamless Data Distribution System.[23]

Figure 5(a) shows the map generated by the terrain mapping system. Figure 5(b) shows the map as found in the USGS database. The units of both figures are meters from the reference origin. Qualitatively the two maps are similar, but the estimated one is much rougher. To quantify the performance of the mapping both maps were divided into 10 m grids, and the difference in height between the two maps for each grid point was calculated. The mean square error was found to be 0.54 meters squared. This is sufficient accuracy for most tasks. In terms of efficiency, a single update takes an average of 0.006 seconds in Matlab, with a worst-case update time of 0.045 seconds. Therefore, update frequency will be more a function of the sensor update time than the computing time.

Figure 3. SIG Kadet Senior UAV

VII.C. Target State Estimation

The target state estimation routine was tested using the simulation described in Section VI.B. Both the extended Kalman filter and the particle filter were tested by taking measurements 10 times a second as the UAV made a single pass over the target. At the end of the pass, the position estimate was compared to the actual position to determine the accuracy of the method.

Figure 6 shows a time history of the EKF for a single pass over the target. The solid lines show the x (north), y (east), and z (height) errors of the estimation. The dash-dot lines represent one standard deviation from the mean (as represented in the diagonal elements of the state covariance matrix). Notice that all three errors approach zero as the number of measurements increase, as is expected. Unfortunately, many of the errors are more than one standard deviation from the mean. This is most likely due to the non-linear nature of the system.

Figure 7(a) shows the initial particle distribution for the particle filter. The x represents the actual barrel location, the dots are particle locations, and the circle represents the weighted (by likelihood) average of the particle locations. The units are in degrees latitude and longitude. Figures 7(b) through 8(b) shows the time history of the particle distributions at one second increments. Notice that as time goes on the average gets closer and closer to the actual location until the circle is right on top of the x. Figure 8(c) shows the final particle distribution after a re-sampling step takes place. Note that the distribution collapses into just a few particles. This is a common problem with particle filters, known as sample impoverishment.[8] This can be a problem for tracking a maneuvering target because the sample set lacks the diversity to account for a maneuver. It is not such a big problem in this case since the target is stationary.

Table 1 shows a comparison of the performance of the two filters. The mean values are the average over 100 runs. The run times are for Matlab running on a 1.4 GHz processor. As is expected, the EKF is much faster than the particle filter (although the particle filter would still be suitable for real-time implementation if it was coded in c++ as opposed to Matlab). Also the particle filter is slightly more accurate than the EKF. For this simple system it appears the EKF is a better choice due to its efficiency. In a more complicated system, however the particle filter may yield better accuracy.

Table 1. Comparison of the two filters

	Mean Runtime (seconds)	Mean Error (meters)
EKF	6.3×10^{-4}	1.1
Particle Filter	0.17	0.78

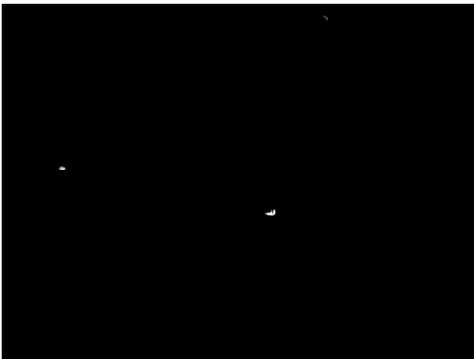American Institute of Aeronautics and Astronautics

(a) Original image
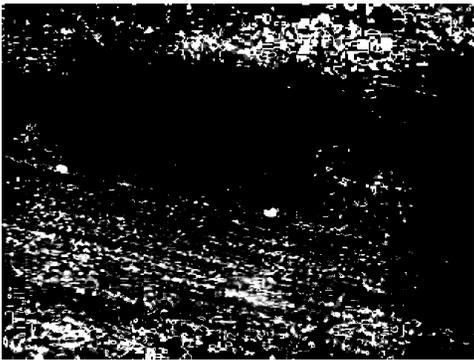
(b) Image passed through hue membership functions

(c) Image passed through saturation membership functions

(d) Image passed through hue and saturation membership functions

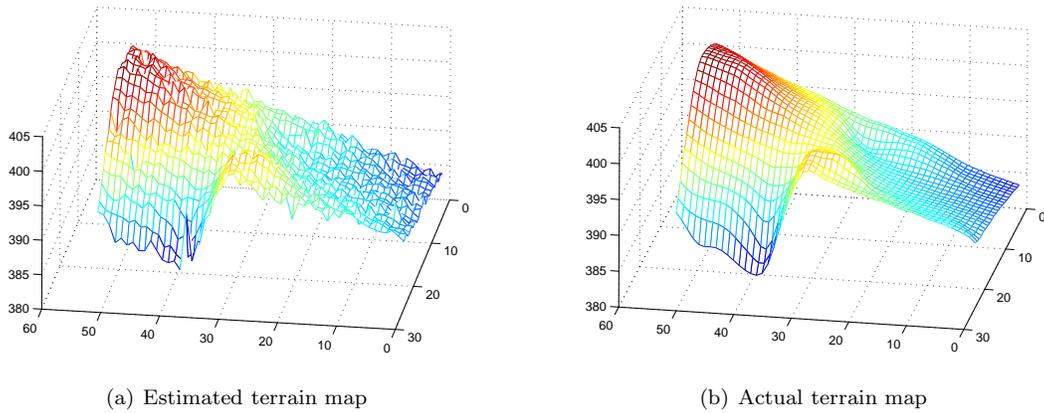(e) Image with red barrels identified

Figure 4. Image processing results

American Institute of Aeronautics and Astronautics

(a) Estimated terrain map

(b) Actual terrain map

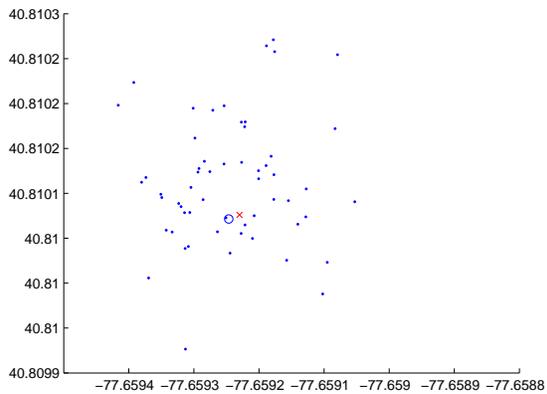**Figure 5.** **Comparison of the two terrain maps (units are meters relative to the reference origin**
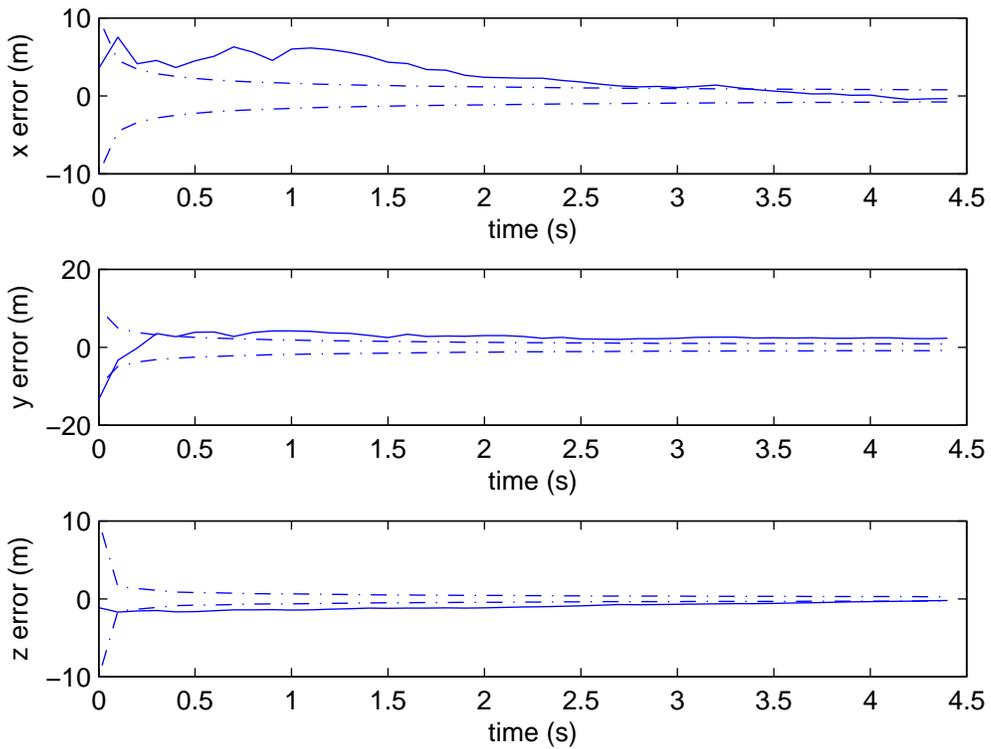
Figure 6. **Time history of EKF errors**

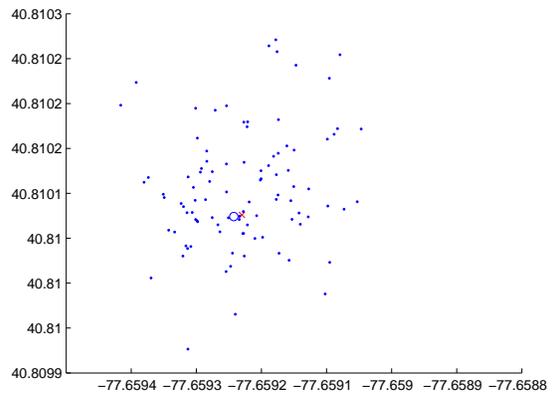American Institute of Aeronautics and Astronautics
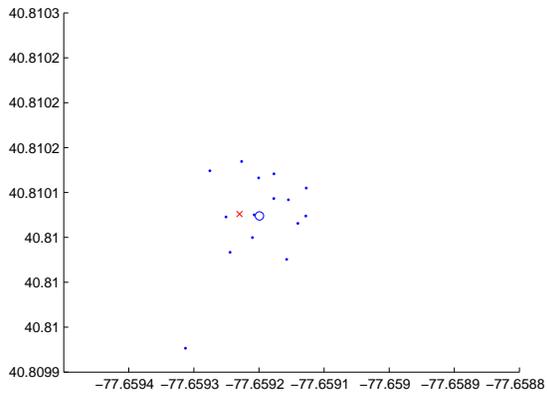
(a) Initial particle distribution
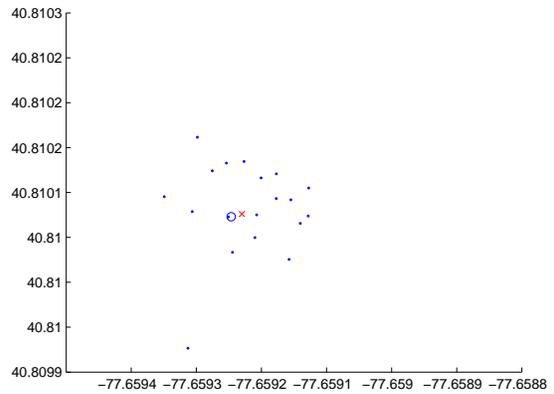
(b) Particle distribution after 1 second
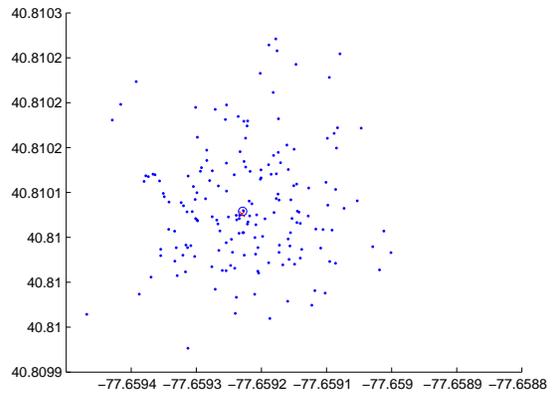
(c) Particle distribution after 2 seconds
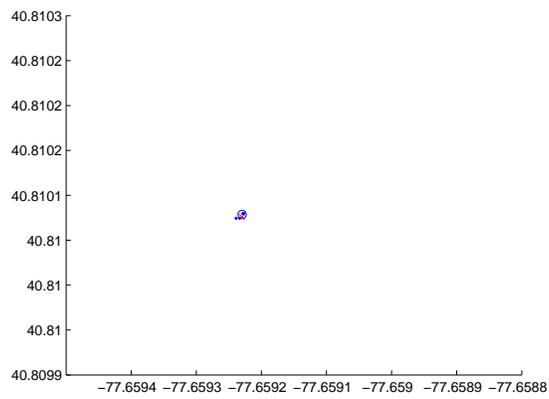
(d) Particle distribution after 3 seconds

Figure 7. Time history of particle distribution (units are degrees latitude and longitude)

American Institute of Aeronautics and Astronautics
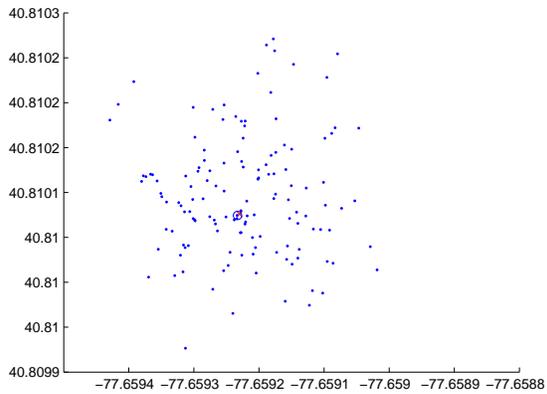
(a) Particle distribution after 4 seconds

(b) Particle distribution after 5 seconds

(c) Final particle distribution after re-sampling

Figure 8. Time history of particle distribution (continued)

American Institute of Aeronautics and Astronautics

VIII.   Conclusions

This paper has presented a novel method for fusing data from a range sensor and a camera in order to accurately locate items on the ground. Using fuzzy logic, red barrels have been detected from a sequence of aerial images with both the false alarm rate and the missed detection rate under five percent. Using simulated range data, a terrain map with a mean square error of less than a meter has been generated. The terrain map has been fused with camera data using both an extended Kalman filter (EKF) and a particle filter. Over 100 runs the mean error of the EKF was 1.1 meters and the particle filter was 0.78 meters.

Acknowledgment

References

[1]Geiger, B. R., Horn, J. F., Sinsley, G. L., Ross, J. A., Long, L. N., and Niessner, A. F., "Flight Testing a Real Time Implementation of a UAV Path Planner Using Direct Collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 6, Nov/Dec 2008.

[2]Ross, J. A., Geiger, B. R., Sinsley, G. L., Horn, J. F., Long, L. N., and Niessner, A. F., "Vision-Based Target Geolocation and Optimal Surveillance on an Unmanned Aerial Vehicle," *AIAA Guidance, Navigation, and Control Conference*, Honolulu, HI, August 2008.

[3]Sinsley, G. L., Miller, J. A., Long, L. N., Geiger, B. R., Niessner, A. F., and Horn, J. F., "An Intelligent Controller for Collaborative Unmanned Air Vehicles," *IEEE Symposium Series in Computational Intelligence*, Honolulu, Hawaii, April 2007.

[4]Sinsley, G. L., Long, L. N., Niessner, A. F., and Horn, J. F., "Intelligent Systems Software for Unmanned Air Vehicles," *46th AIAA Aerospace Sciences Meeting*, Reno, NV, Jan 2008.

[5]Hanford, S. D., Janrathitikarn, O., and Long, L. N., "Control of Mobile Robots Using the Soar Cognitive Architecture," *Journal of Aerospace Computing, Information, and Communication*, Vol. 6, No. 2, 2009.

[6]Long, L. N., Hanford, S. D., Janrathitikarn, O., Sinsley, G. L., and Miller, J. A., "A Review of Intelligent Systems Software for Autonomous Vehicles," *IEEE Computational Intelligence for Security and Defense Applications Conference*, Honolulu, Hawaii, 2007.

[7]Stover, J. A., Hall, D. L., and Gibson, R. E., "A Fuzzy-Logic Architecture for Autonomous Multisensor Data Fusion," *IEEE Trans. Ind. Electron.*, Vol. 43, 1996, pp. 403–410.

[8]Simon, D., *Optimal State Estimation*, John Wiley and Sons, 2006.

[9]Ristic, B., Arulampalam, S., and Gordon, N., *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech House, 2004.

[10]Johnson, A. E., Klumpp, A. R., Collier, J. B., and Wolf, A. A., "Lidar-Based Hazard Avoidance for Safe Landing on Mars," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 6, November-December 2002.

[11]Leal, J., Scheding, S., and Dissanayake, G., "Stochastic Simulation in Surface Reconstruction and Application to 3D Mapping," *IEEE International Conference on Robotics and Automation*, Washington, D.C., May 2002.

[12]Pachter, M., Ceccarelli, N., and Chandler, P. R., "Vision-Based Target Geo-Location Using Feature Tracking," *AIAA Guidance, Navigation and Control Conference and Exhibit*, No. AIAA 2007-6863, Hilton Head, South Carolina, August 2007.

[13]Monda, M. J., Woolsey, C. A., and Reddy, C. K., "Ground Target Localization and Tracking in a Riverine Environment from a UAV with a Gimbaled Camera," *AIAA Guidance, Navigation and Control Conference and Exhibit*, No. AIAA 2007-6747, Hilton Head, South Carolina, August 2007.

[14]Campbell, M. E. and Wheeler, M., "A Vision Based Geolocation Tracking System for UAV's," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, No. AIAA 2006-6246, Keystone, Colorado, August 2006.

[15]Iba, Y., "Population Monte Carlo Algorithms," *Transactions of the Japanese Society for Artificial Intelligence*, Vol. 16, No. 2, 2001, pp. 279–286.

[16]Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*, The MIT Press, Cambridge, Massachusetts, 2006.

[17]Ong, L., Upcroft, B., Ridley, M., Bailey, T., Sukkarieh, S., and Durrant-Whyte, H., "Decentralized Data Fusion with Particles," *Australian Conference on Robotics and Automation*, 2005.

[18]"SICK laser range sensors," http://www.sick.com/group/EN/home/Pages/Homepage1.aspx.

[19]"AGL Laser Altimeter," http://www.cloudcaptech.com/piccolo_accessories_laser_altimeter.shtm.

[20]Marlow, S. Q. and Langelaan, J. W., "Local Terrain Mapping for Obstacle Avoidance using Monocular Vision," *AHS Unmanned Vehicles Specialist's Forum*, 2009.

[21]Trucco, E. and Verri, A., *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, Upper Saddle River, NJ, 1998.

[22]Stevens, B. L. and Lewis, F. L., *Aircraft Control and Simulation*, John Wiley and Sons, 1992.

[23]"USGS Seamless GIS Data," http://seamless.usgs.gov, Last accessed on August 3, 2008.

[24]"OpenCV 1.0," Available at: http://www.intel.com/technology/computing/opencv, Last accessed on August 3, 2008.