Obstacle Detection for Small Autonomous Aircraft Using Sky Segmentation[*]

Timothy G. McGee, Raja Sengupta, and Karl Hedrick
AINS Center for Collaborative Control of Unmanned Vehicles
University of California, Berkeley
2105 Bancroft Way, Berkeley, CA 94720
{tmcgee@me, raja@path, khedrick@me}.berkeley.edu

***Abstract* – A vision-based obstacle detection system for small unmanned aerial vehicles (UAVs) is presented. Obstacles are detected by segmenting the image into sky and non-sky regions and treating the non-sky regions as obstacles. The feasibility of this approach is demonstrated by using the vision output to steer a small unmanned aircraft to fly towards an obstacle. The experiment was first verified in a hardware in the loop (HIL) simulation and then successfully implemented on a small modified remote control plane using a large inflatable balloon as the obstacle.**

Index Terms – UAV, sky segmentation, obstacle avoidance, support vector machine.

I. Introduction

Small unmanned aerial vehicles (UAVs) have the potential to act as low cost tools in a variety of applications such as traffic monitoring, border patrol, search and rescue, and surveillance. During recent years, new technologies including the development of miniature inertial sensors and avionics packages have increased the functional capabilities of UAVs. However, limitations in areas such as robust obstacle detection still prevent many applications from becoming realities. For the class of small UAVs considered here, those weighing 5 to 15 kg, the sensors currently being investigated for obstacle detection include lidar, miniature radar, and computer vision. Lidar has been successfully demonstrated on larger unmanned helicopters [1], although the size and weight of these systems prevent them from being used on smaller vehicles. Recent advances in miniature radar are promising for smaller UAVs [2], although this technology is still limited and provides only minimal angular information. With the current limitations of lidar and radar, the size and low power constraints of miniature cameras make computer vision a logical choice for small UAVs. This work explores a computationally inexpensive method of obstacle detection and avoidance strategies which use computer vision as the sensor.

There has been a variety of previous work that has demonstrated the use of computer vision on small unmanned aircraft. Several groups have used downward looking cameras to aid in autonomous helicopter landing by either identifying safe landing sites [3] or calculating position and orientation to a landing target of known size and shape [4,5]. Artificial markings have also been used to demonstrate target following and landmark navigation [6,7].

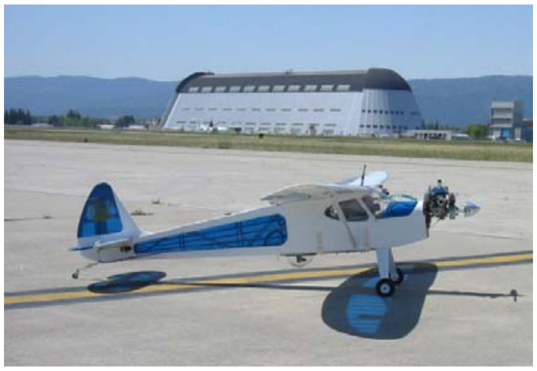


Fig. 1 UC Berkeley fixed wing UAV testbed.

Road following using computer vision has been demonstrated using a fixed wing aircraft [8], and horizon detection has been demonstrated as an attitude sensor for low-level control of a micro air vehicle [9]. Biomimetic approaches to computer vision have also successfully used optical flow to perform terrain following and have been studied for obstacle detection [10]. Despite the variety of vision-based capabilities demonstrated on UAVs, there is a noticeable absence in the literature of airborne systems that have successfully used vision to detect and avoid obstacles in their flight paths.

The paradigm for vision-based obstacle detection is the use of multiple images to find correspondences between low-level features. The geometry of computer vision required to calculate distance to these features using triangulation is well understood and has been studied extensively [11]. Real-time stereo vision systems that utilize two rigidly mounted cameras are commercially available, although the depth resolution of these systems is severely limited by the baseline separation of the cameras. A commonly proposed approach to overcome this problem is to use a sequence of images from a single camera [12]. Although recent work has demonstrated the calculation of rotation and translation between uncalibrated images in real time [13], analysis reveals that depth estimation from forward motion images is very poor at the center of the field of view [14]. Because small fixed wing UAVs must maintain a minimum forward velocity, it is most important to detect obstacles in the center of the field of view, making obstacle detection from forward motion video difficult.

[*] This work is supported by the ONR grant #N00014-03-C-0187, SPO #016671-004 and a NDSEG Fellowship for Mr. McGee.

In order to overcome the difficulties of detecting obstacles using triangulation of low-level features, a detection system is presented which identifies obstacles by classifying an image into sky and non-sky regions. The non-sky regions are then treated as obstacles. This approach is similar to the one presented in [15] that used color to find drivable road for a ground vehicle, and to [17] that utilized probabilistic hidden Markov trees to perform sky/ground segmentation to detect the horizon for micro air vehicle control. In order to demonstrate the feasibility of this method, sky segmentation is demonstrated for a fixed wing unmanned aircraft using both a hardware in the loop (HIL) simulation and the actual aircraft shown in Fig. 1. During these tests, turn-rate commands were used to steer the aircraft towards the obstacle in order to keep the obstacle in the field of view.

II. Computer Vision

A. Sky Segmentation

The classification of the image into sky and non-sky was performed using a support vector machine (SVM) [18,19,20], that classified each pixel based on its color in the YCrCb color space, after smoothing the image with a Gaussian filter to reduce the effects of noise. A support vector machine was chosen for classification since it is inherently a binary classifier and is easily scalable to include more properties that include texture in addition to color. Although texture information would make the classification more robust over a wider variety of conditions [17], color alone proved to be sufficient for the experiments presented.

SVM classification is based on separating hyper-planes. For any set of linearly separable data, there exist values of **a** and b such that any data point **x** in the data set can be classified by the equation:

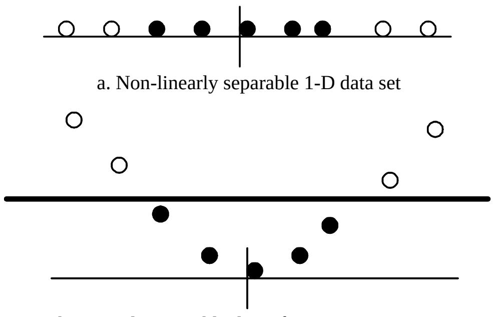
$$y(\mathbf{a}\cdot\mathbf{x}+b)>0 \quad (1)$$

where y = ±1 depending on which of the two classes the data point belongs too. When dealing with a finite set of data, Eq. 1 can be expressed as:

$$f(\mathbf{x})=sign\left(\sum_{1}^{N} w_i(\mathbf{x}\cdot\mathbf{x_i})+b\right) \qquad \sum_{1}^{N} w_i=0 \quad (2)$$

where the $\mathbf{x_i}$ values are typically a small subset of the example points and f(**x**) = ±1 provides a classification of the test point **x**.

While Eq. 2 provides a simple method for classifying a point in a set of linearly separable data, many data sets of interest are not linearly separable. However, by mapping the data set into a higher dimensional space using a map $\phi(\mathbf{x})$, a higher dimensional separating hyper-plane can be found in the higher dimensional space. A simple example of this is illustrated in Fig. 2. The set of one dimensional points in the dataset are not linearly separable, although by mapping the points to a higher two-dimensional space through the map $\phi_{example}(z) = (z,z^2)$, the points become linearly separable.


a. Non-linearly separable 1-D data set

b. Linearly separable data after mapping to 2-D

Fig. 2 Example of mapping to higher dimension

After mapping the data to a higher dimension, Eq. 2 takes the form:

$$f(\mathbf{x})=sign\left(\sum_{1}^{N} w_i(\phi(\mathbf{x})\cdot\phi(\mathbf{x_i}))+b\right) \qquad \sum_{1}^{N} w_i=0 \quad (3)$$

Since the classification in Eq. 3 depends on the dot product of the mapped data points, the mapping function $\phi(\mathbf{x})$ does not need to be found explicitly. Instead, only the kernel function is required:

$$K(\mathbf{x},\mathbf{x_i})=\phi(\mathbf{x})\cdot\phi(\mathbf{x_i}) \quad (4)$$

Combining Eq. 3 and 4 yields the final form of the classification function for the SVM:

$$f(\mathbf{x})=sign\left(\sum_{1}^{N} w_i K(\mathbf{x},\mathbf{x_i})+b\right) \qquad \sum_{1}^{N} w_i=0 \quad (5)$$

There are a variety of common support vector kernel functions which yield good results. For the sky classification, a polynomial kernel of the form:

$$K(\mathbf{x},\mathbf{x_i})=(\mathbf{x}\cdot\mathbf{x_i}+1)^d \quad (6)$$

was implemented using the Torch machine learning library for C++ [20] with d = 8.

For an explanation of the derivation of the w_i and b values and which $\mathbf{x_i}$ values are used, and further information on various kernel functions the reader should refer to one of the references listed [18-20].

B. Horizon Detection

Because the horizon is the projection of ground plane at infinity on the camera plane, extruding the horizon line perpendicular to the image plane creates a plane parallel to ground through the aircraft as shown in Fig. 3. Thus, objects below the horizon are below the aircraft and out of the flight path of the aircraft, while objects above the horizon pose a threat. One limitation of using sky segmentation for detecting obstacles is that objects whose maximum heights equal to the altitude of the aircraft will not protrude above the horizon, and will thus be not be detectable.

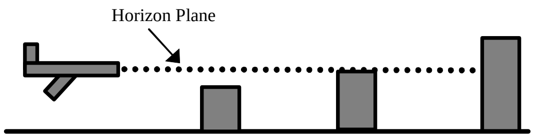


Fig. 3 Location of obstacles with respect to horizon

The horizon was found by searching for the line that best divided the binary sky segmented image into sky and non-sky regions. The first step was to apply standard erosion and dilation to the binary segmented image to remove any small sections of misclassified pixels. The border between sky and non-sky regions was then found by smoothing the binary image and classifying all pixels with values near 0.5 as boundary pixels. The horizon detection was then performed using the OpenCV [21] implementation of the Hough Transform on the border image. This method utilizes the fact that a line can be represented by the equation:

$$x\cos\theta + y\sin\theta + \rho = 0 \tag{7}$$

where (x,y) are the coordinates of the points on the line, θ is the rotation of the line, and ρ is the perpendicular distance from the line to the origin. Thus, each line in (x,y) space can be represented by a point in (ρ,θ) space. Conversely, each point in the (x,y) space represents a curve in (ρ,θ) space, which correspond to all of the possible lines that pass through that point. Thus, for each point in (x,y) space, a vote is placed in each bin of a discretized (ρ,θ) space that corresponds to all possible lines that pass through that point. Bins in (ρ,θ) space that receive a large number of votes correspond to probable lines.

Because the binary image from the sky classification can contain multiple lines if there are obstacles or if there is improper classification of water or light colored roads as sky, all of the possible lines in (ρ,θ) space receiving a minimum number of votes were considered as possible horizons. The best horizon line was then chosen as the candidate line which minimized the cost function:

$$J(\rho,\theta) = \sum_{i,j} e(\mathbf{x}_{ij})$$

$$e(\mathbf{x}_{ij}) = \begin{cases} 0 \text{ if } f(\mathbf{x}_{ij}) = 1, \ (i,j) \text{ above horizon} \\ \alpha \text{ if } f(\mathbf{x}_{ij}) = -1, (i,j) \text{ above horizon} \\ 1 \text{ if } f(\mathbf{x}_{ij}) = 1, (i,j) \text{ below horizon} \\ 0 \text{ if } f(\mathbf{x}_{ij}) = -1, (i,j) \text{ below horizon} \end{cases} \tag{8}$$

where f(**x**) is the classification output, 1 for sky, -1 for non-sky, $\mathbf{x}_{ij}$ is the color vector from the pixel at (i,j) in the image, and α is a positive constant. Thus, J(ρ,θ) is a weighted sum of all of the pixels above the candidate horizon classified as non-sky and all of the pixels below the candidate horizon classified as sky. The scaling factor, α, with a value greater than unity was found to improve performance of the algorithm.

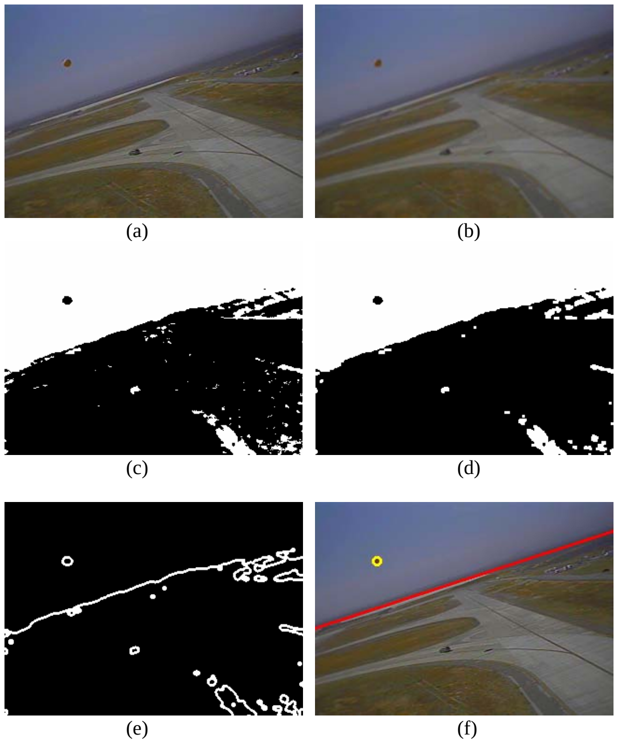


Fig. 4 Vision algorithm:
a)original image. b) smoothed image
c) binary segmented image d)erosion and dilation
e)sky/non-sky border f)horizon and obstacle found

C. Object Detection

Once the horizon was found, all non-sky pixels above the horizon could be treated as obstacles in the path of the aircraft. Since a single obstacle was used for testing, and there were often a few non-sky pixels close to the horizon, the obstacle was determined as the connected group of non-sky pixels with the greatest perpendicular distance to the horizon.

D. Output of the Vision System

The final output of the vision system was the angle of the horizon, and the image coordinates of the obstacle. The complete vision algorithm is summarized in Fig. 4.

III. Lateral Control

In order to demonstrate the integration of the vision-based obstacle detection with the control of a small fixed wing aircraft, a simple lateral controller was designed to autonomously steer the aircraft at the detected obstacle. The dynamics of the aircraft can be modelled by the standard unicycle model:

$$\begin{aligned} \dot{x} &= U_{TAS}\cos\theta + V_{wx} \\ \dot{y} &= U_{TAS}\sin\theta + V_{wy} \\ \dot{\theta} &= \omega_{cmd} \end{aligned} \tag{9}$$

where (x,y) is the position of the aircraft in world-fixed coordinates, U_{TAS} is the true airspeed of the aircraft, θ is the heading of the aircraft, V_w is the wind speed, and ω_{cmd} is the commanded turning rate. The goal of the controller was to drive the angle between the balloon heading (θ) and the angle to the obstacle (ϕ) to zero as shown in Figure 5. This control objective was achieved using a simple proportional controller, setting the turn-rate proportional to the lateral angle to the balloon:

$$\omega_{cmd} = K\varepsilon = K(\phi - \theta) \tag{10}$$

The lateral angular error to the obstacle was found from the distance of the centroid of the obstacle to the center of the image along the horizon (x_l) as shown in Figure 6. This lateral distance was calculated from the image coordinates of the obstacle and the angle of the horizon:

$$x_l = x_o \cos\theta - y_o \sin\theta \tag{11}$$

Thus the final control law became:

$$\omega_{cmd} = K\varepsilon = -\hat{K}x_l = -\hat{K}(x_o \cos\theta - y_o \sin\theta) \tag{12}$$

IV. Experimental System

The sky segmentation for the obstacle detection algorithm was developed for implementation on a Sig Rascal model aircraft (Fig. 1). This aircraft has a wingspan of 2.8 m, an empty weight of 5.5 kg, and a gross weight of 10 kg. This allows 4.5 kg for fuel, avionics, and payload. The low-levelaircraft control is performed by the Piccolo avionics package, shown in Fig. 7 which is available commercially from Cloud Cap, Inc, and weighs 212 g [16].

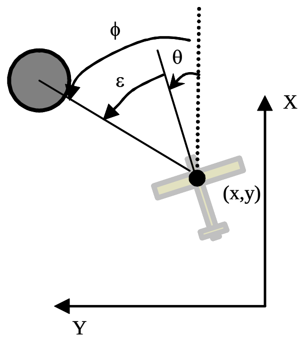


Fig. 5 Control strategy

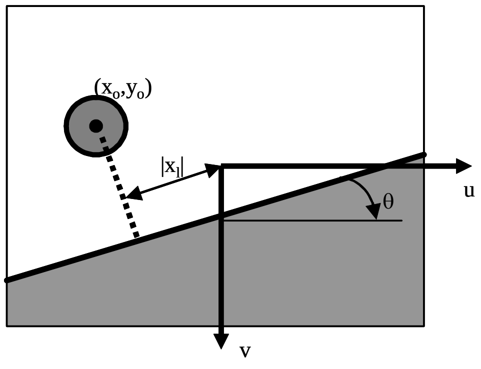


Fig. 6 Lateral distance to obstacle

Fig. 7 Piccolo Avionics Package

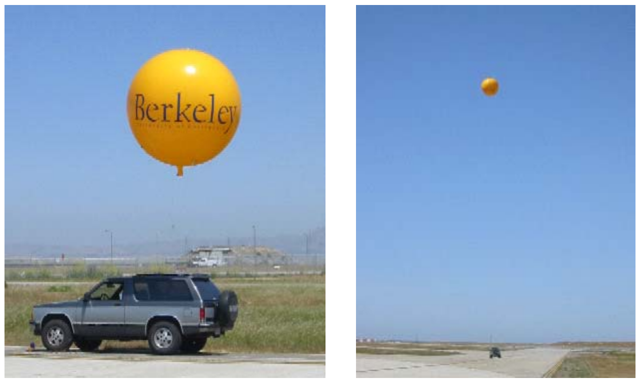


Fig. 8 Balloon Obstacle

The vision system consists of a forwarded looking CCD camera with 320x240 pixel resolution and a 62° field of view that is mounted to the left wing of the aircraft. Vision processing is performed onboard the aircraft using a PC104 stack with a 700 MHz Pentium III processor. The PC104 communicates with the Piccolo avionics via a serial port. The current setup allows the PC104 to send turn-rate, altitude, airspeed, and waypoint commands to the Piccolo and receive aircraft telemetry including GPS coordinates, altitude, airspeed, and estimated roll, pitch, and yaw angles.

In order to verify the integration of the system before an actual flight test, the vision algorithm and control method were implemented using a hardware in the loop simulation. The actual flight avionics and vision processing hardware were used in the HIL tests. The aircraft dynamics were replaced by a high-dimensional nonlinear simulation provided with the Piccolo avionics package. The camera outputs were simulated using the Vega software package.

During the experimental test on the actual aircraft, a 2.6m diameter helium balloon tethered at a height of 40m was used as an obstacle as shown in Fig. 8. In order to avoid the balloon, a GPS receiver was placed below it, and its coordinates were sent to the aircraft. When the aircraft came within 100 m of the balloon, a 15°/sec turn-rate was given to steer away from the balloon.

V. RESULTS

The obstacle and horizon detection were first tested using video recorded during two days of manual flight testing of the aircraft. The support vector machine was then trained using video from the first day of flying and tested using video from the second day. The vision system was able to correctly find the horizon and balloon in about 90% of the images that the balloon was in view. The main causes of error were found to be the similarity in color of the sky and the road in the low resolution cameras and variations in the color output of the camera images resulting from its autocalibration to changes in light intensity as the plane pitched up.

The combined vision and control algorithm was able to run at an average of approximately 2 Hz on real images. The majority of the processing time was required by the sky classification, the Hough, and the horizon ranking. SVM classification required 120ms to classify the 320x240 image using 8 support vectors. The Hough Transform and horizon ranking required between 200 and 600 ms per image depending on the number of candidate lines returned.

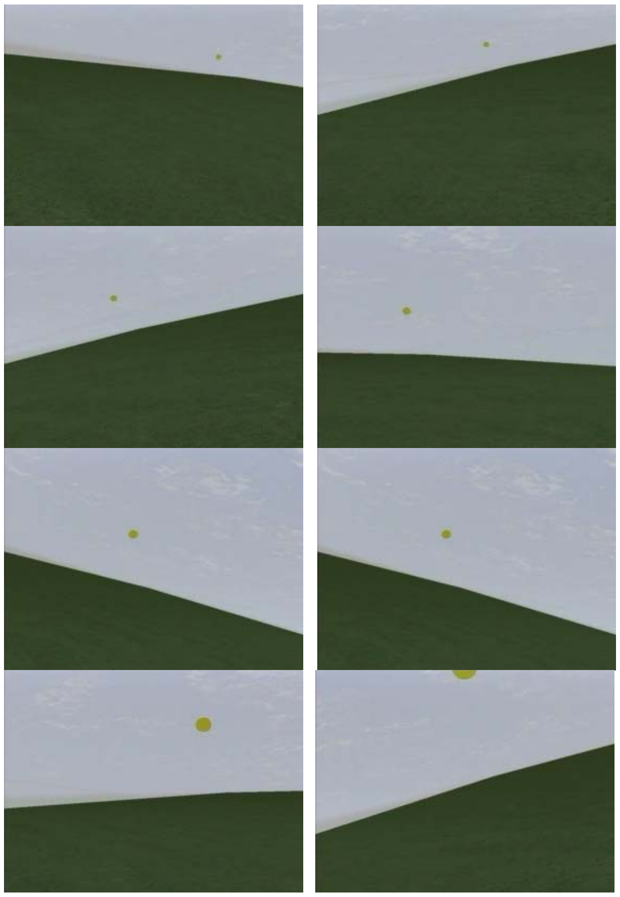


Fig. 9 View of plane during HIL simulation

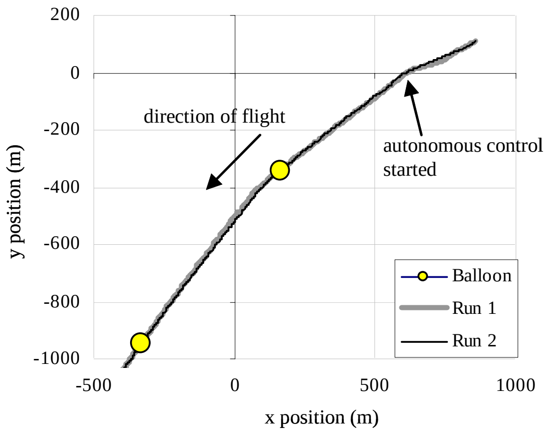


Fig. 10 Tracking results in HIL simulation

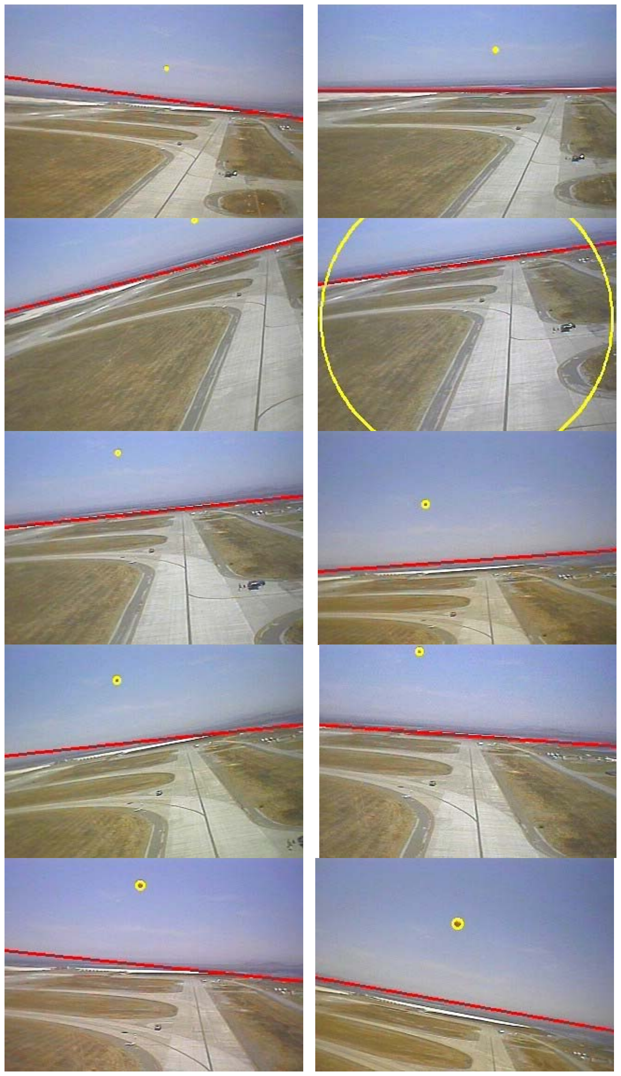


Fig. 11 View of plane during flight test

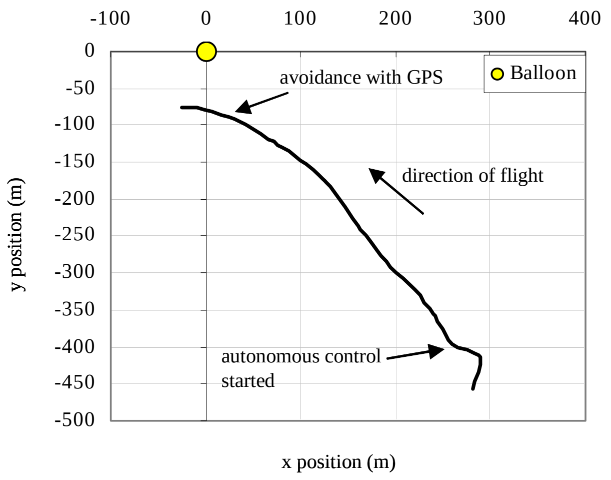


Fig. 12 Tracking results on actual aircraft

Although the speed of the vision algorithm was adequate for these experiments, the processing times could be reduced by using a faster processor or by optimizing the code for speed. The horizon and obstacle detection could also be improved by tracking the horizon and any obstacles between frames if the processing time between frames were made small enough.

During these tests, the aircraft was able to keep each balloon in its field of view until that balloon was passed, at which time the next balloon came into view. The images in Fig. 9 and illustrate the small oscillations experienced by the aircraft as it steered to center a balloon in the image. The tracking results on the actual aircraft are shown in Fig. 11-12. In Fig. 11, the red line represents the calculated horizon, and the yellow circle is the location of the balloon. In the third frame, the balloon leaves the field of view resulting from pitch oscillations.

VI. Conclusions

A vision-based system for obstacle detection was proposed that detects obstacles by classifying the image into sky and non-sky regions and treating non-sky regions as obstacles. This algorithm was tested in both hardware in the loop simulation and on an actual small unmanned aircraft to steer the aircraft at the obstacle. This obstacle detection method has the advantage of being able to detect obstacles in a single image. The main disadvantages of the algorithm are that it can only detect obstacles above the horizon that are viewed with sky in the background. While this disadvantage prevents obstacle detection through sky segmentation from being a universal obstacle detection method, there are places in which UAVs would encounter single large obstacles such as billboards above a highway or tall buildings which could be avoided by detecting them against the sky.

Acknowledgment

The authors would like to acknowledge NASA Ames Research Center and T.J. Forsyth for providing us access to the Moffett Air Field for flight testing. We would also like to recognize the contributions of the rest of the flight operations team, including Xiao Xiao, Eric Frew, Aram Soghikian, Jack Tisdale, Allison Ryan, Steve Jackson, Mike Morimoto, Stephen Spry, Zuwhan Kim, Marco Zennaro, Sivakumar Rathinam, and Anouck Girard, in the development, maintenance, and operations of our UAV system.

References

[1] N. Vandapel, R. Donamukkala, and M. Hebert, “Experimental results in using aerial LADAR data for mobile robot navigation”, *The 4th International Conf. on Field and Service Robotics*, 2003.
[2] R. Fontana, et al, “An ultra wideband radar for micro air vehicle applications”, *Proc. of the IEEE Conf. on Ultra Wideband Systems and Technologies*, 2002.
[3] P. Garcia-Pardo, G. Sukhatme, and J. Montgomery, “Toward vision-based safe landing for an autonomous helicopter”, *Robotics and Autonomous Systems*, vol. 38, 2002.
[4] O. Shakernia, Y. Ma, T.J. Koo, J. Hespanha, and S. Sastry, “Vision guided landing of an unmanned air vehicle”. *Proc. of the 38th Conf. on Decision and Control*, 1999.
[5] S. Saripalli, J. Montgomery, and G. Sukhatme, “Vision-based autonomous landing of an unmanned aerial vehicle” *Proc. of IEEE International Conf. on Robotics and Automation*, 2002.
[6] S.M. Rock, E.W. Frew, H.L. Jones, E. LeMaster, and B. Woodley, “Combined CDGPS and vision-based control of a small autonomous helicopter”, *Proc. of the American Control Conference*, 1998.
[7] J. Kim, S. Sukkarieh, “Airborne simultaneous localisation and map building”, *Proc. of IEEE International Conf. on Robotics and Automation*, 2003.
[8] E. Frew, et al, “Vision-based road-following using small autonomous aircraft”, *Proc. of the IEEE Aerospace Conference*, 2004.
[9] S.M. Ettinger, M.C. Nechyba, P.G. Ifju, and M. Waszak, “Vision-guided flight stability and control for micro air vehicles”, *Proc. of IEEE International Conf. on Intelligent Robots and Systems*, 2002.
[10]G.L. Barrows, J.S. Chahl, and M.V. Srinivasan, “Biologically inspired visual sensing and flight control”, *Aeronautical Journal*, vol. 107, pp. 2134-2140, March 2003.
[11]R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge, UK: Cambridge University Press, 2000.
[12]Z. Zhang, R. Deriche, O. Faugeras, and Q. Luong, “A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry”, *Artificial Intelligence*, vol. 17, pp. 87-119, 1995.
[13]D. Nister, “Preemptive RANSAC for live structure and motion estimation”, *Proc. of the Ninth IEEE International Conf. on Computer Vision*, 2003.
[14]B. Bhanu, S. Das, B. Roberts, and D. Duncan, “A system for obstacle detection during rotorcraft low altitude flight”, *IEEE Trans. on Aerospace and Electronic Systems*, vol. 32, no. 3, July 1996.
[15]P. Chaturvedi, E. Sung, A. Malcom, and J. Ibanez-Guzman, “Real-time identification of drivable areas in a semi-structured terrain for an autonomous ground vehicle”, *Proc. of SPEI*, 2001.
[16]“Piccolo User’s Guide”, Cloud Cap Technology, Inc.
[17]S. Todorivic and M. Nechyba, “Sky/ground modelling for autonomous MAV flight”, *Proc. of IEEE International Conf. on Robotics and Automation*, 2003.
[18]C. Burges, “A tutorial on support vector machines for pattern recognition”, *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.
[19]D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Upper Saddle River, NJ: Prentice Hall, 2003.
[20]Torch3 Machine Learning Library, *www.torch.ch/*
[21]OpenCV Library, *www.intel.com/research/mrl/research/opencv/*