Vision-based Autonomous Landing of an Unmanned Aerial Vehicle

Srikanth Saripalli[1], James F. Montgomery[2] and Gaurav S. Sukhatme[3]

[1,3]srik|gaurav@robotics.usc.edu, [2]monty@robotics.jpl.nasa.gov

[1,3]**Robotic Embedded Systems Laboratory**
Robotics Research Laboratory
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0781

[2]**Jet Propulsion Laboratory**
California Institute of Technology
Pasadena
Los Angeles
CA 91109

Abstract

We present the design and implementation of a real-time, vision-based landing algorithm for an autonomous helicopter. The helicopter is required to navigate from an initial position to a final position in a partially known environment based on GPS and vision, locate a landing target (a helipad of a known shape) and land on it. We use vision for precise target detection and recognition. The helicopter updates its landing target parameters based on vision and uses an on board behavior-based controller to follow a path to the landing site. We present results from flight trials in the field which demonstrate that our detection, recognition and control algorithms are accurate and repeatable.

1 Introduction

Unmanned Aerial Vehicles are indispensable for various applications where human intervention is impossible, risky or expensive e.g. hazardous material recovery, traffic monitoring, disaster relief support etc. A helicopter is highly maneuverable versatile platform. It can take off and land vertically, hover in place, perform longitudinal and lateral flight as well as drop and retrieve objects from otherwise inaccessible places. But the high maneuverability of helicopters comes at a significant cost; they are unstable and dangerous to fly. This shortcoming can be remedied by an unmanned autonomous helicopter, since eliminating the pilot from the control loop decreases the chances of endangering human life. Also the size of the helicopter can be reduced which effectively reduces the cost of the helicopter, while increasing its maneuverability.

For an unmanned helicopter to successfully function, *autonomous landing* is a crucial capability. The structured nature of landing makes it suitable for vision-based state estimation and control. The vision problem that we consider here is a special case of the ego-motion problem where all the feature points lie on a planar surface (in this case the landing pad) [1]. We present an algorithm for vision-based autonomous landing of a model helicopter in an unstructured 3D environment. The helicopter is required to autonomously locate and recognize a helipad of dimensions 122cm x 122cm, align with it and land on it. We present results based on flight data from field tests which show that the algorithm is able to land the helicopter on the helipad repeatably and accurately. On an average the algorithm landed the helicopter to within 40 cm position accuracy and to within 7^o in orientation as measured from the center of helipad and its principal axis respectively.

(a) AVATAR

(b) AVATAR landing on a helipad

Figure 1: AVATAR (Autonomous Vehicle Aerial Tracking And Reconnaissance)

Vision-based robot control has been an active topic of research in the past few years [2, 3, 4]. In [5] a real time computer vision system is presented for tracking a landing target but no autonomous landing was reported. In [6, 7] the autonomous landing problem was decoupled from the problem of vision-based tracking. [8] discusses a vision-based solution to safe landing in unstructured terrain. Several vision-based servoing techniques have been implemented for autonomous control of helicopters [9], but none of them have focused on the landing problem [10]. The problem of autonomous landing is particularly difficult because the inherent instability of the helicopter near the ground [11]. Also since the dynamics of a helicopter are non-linear only an approximate model of the helicopter can be constructed [12].

2 The Test-bed and Experimental Task

Our experimental test-bed AVATAR (Autonomous Vehicle Aerial Tracking And Reconnaissance) [13] is a gas-powered radio-controlled model helicopter fitted with a

PC-104 stack augmented with several sensors (Figure 1). A Novatel RT-20 DGPS system provides positional accuracy of 20cm CEP(Circular Error Probable, i.e. the radius of a circle, centered at the true location of a receiver antenna, that contains 50% of the individual position measurements made using a particular navigational system). A Boeing CMIGTS-II INS unit with three axis accelerometers and three-axis gyroscopes provides the state information to the on-board computer. The helicopter is equipped with a color CCD camera and an ultrasonic sonar. The ground station is a laptop that is used to send high-level control commands and differential GPS corrections to the helicopter. Communication with the ground station is carried via 2.4 Ghz wireless Ethernet and 1.8Ghz wireless video. Autonomous flight is achieved using a *behavior-based* control architecture [14]. This is discussed further in Section 4.

The overall landing strategy is as follows. Initially the helicopter is in *search* mode. The vision algorithm (described below) scans for the landing target. As soon as it detects the landing target the state estimation algorithm sends commands to the helicopter controller. This mode is called *object-track* mode. When the helicopter is above the landing target the vision-based controller commands the helicopter to land. This is called the *land* mode. Figure 2 shows a flow-chart of the algorithm. Next, we describe the vision and state estimation algorithms.

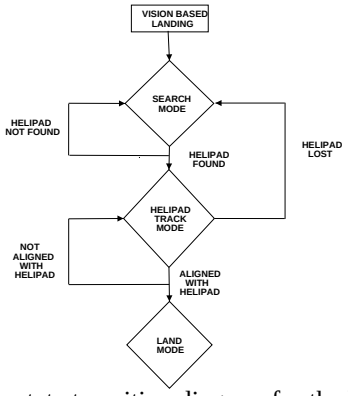


Figure 2: The state transition diagram for the landing task

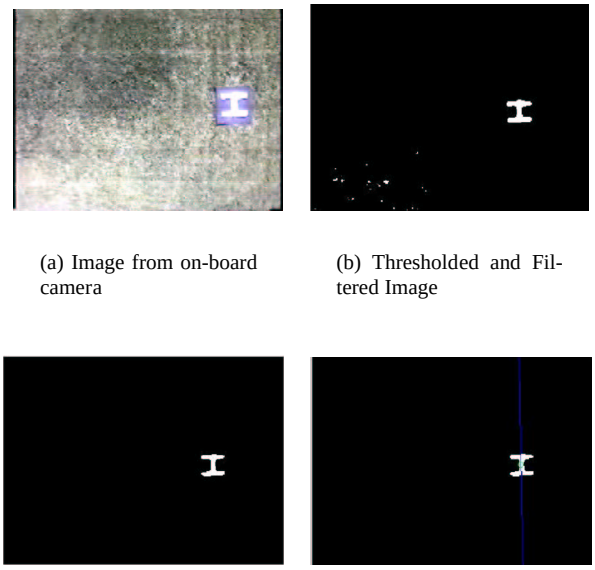
3 Vision Algorithm

The vision algorithm is described below in three parts; preprocessing, geometric invariant extraction, object recognition and state estimation.

3.1 PREPROCESSING

The goal of this stage is to locate and extract the landing target. Figure 3 (a) shows an aerial view of the helipad used in our experiments.

(a) Thresholding and Filtering. Thresholding converts the color image to a binary image. The image obtained from the camera is noisy and the frame grabber is



(a) Image from on-board camera

(b) Thresholded and Filtered Image

(c) Segmented Image

(d) Final Image

Figure 3: Image processing results. All images are taken in-flight from a downward-pointing camera on the helicopter

of low quality, hence we work with binary images to reduce the computational cost and increase the effectiveness of the algorithm. The image is first converted to gray-scale by eliminating the hue and saturation information while retaining the luminance. This is accomplished by the following equation [15]

$$Y = 0.299 * R + 0.596 * G + 0.211 * B \qquad (1)$$

where R,G,B represent the red, green and blue values in the image respectively. The thresholding algorithm must produce a binary image which preserves the landing target but effectively removes most of the other data from the image. A robust implementation is to threshold the image at a fixed percentage (80%) between the minimum and the maximum gray levels. Figure 3(b) shows the image after thresholding. A 7×7 *Median-filter* is applied to the subsequent image for removing noise and to preserve the edge details effectively. Median-filters have low-pass characteristics and they remove additive white noise [15]. They preserve the edge sharpness [16] in an image and are particularly suitable for the recognition of geometric objects such as the helipad.

(b) Segmentation and Connected Component Labeling. The image obtained after thresholding and filtering may consist of objects other than the helipad. In this step the various regions of interest are identified and labeled. The image is scanned row wise until the first pixel at a boundary is hit. All the pixels which belong to the 8-neighborhood of the current pixel are marked as belonging to the current object. This operation is continued recur-

sively until all pixels belonging to the object are counted. A product of this process is the area of the particular object in pixels. Objects whose area is less than a particular threshold ($\leq$ 80 pixels) are discarded. Similarly objects whose area is $\geq$ 700 pixels are discarded. The remaining objects are our ROI(regions of interest) and are candidates for the landing target (Figure 3 (c)).

3.2 Invariant Moments

Geometric shapes possess features such as *perimeter, area, moments* that carry sufficient information for the task of object recognition. Such features can be used as object descriptors, resulting in significant data compression, because they can represent the geometric shape by a relatively small feature vector and are ideally suited for the present task. Based of the geometric features of an object one can calculate a set of descriptors which are invariant to rotation, translation and scaling. These shape descriptors are widely used in optical character recognition and pose estimation. One such class of descriptors [17] is based on the moments of inertia of an object. The $(p+q)^{th}$ order moment of an image $f(x,y)$ is given by

$$m_{pq} = \sum_i \sum_j i^p j^q f(i,j) \tag{2}$$

where the indices *i, j* correspond to the coordinate axes *x, y* respectively.

The center of gravity of the object is specified by

$$\bar{x} = \frac{m_{10}}{m_{00}} \qquad \bar{y} = \frac{m_{01}}{m_{00}} \tag{3}$$

The central moments of an object are the moments defined about the center of gravity and are given by

$$\mu_{pq} = \sum_i \sum_j (i-\bar{x})^p (j-\bar{y})^q f(i,j) \tag{4}$$

where the indices *i, j* correspond to the coordinate axes *x, y* respectively. The normalized central moments, denoted by η_{pq}, are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}{}^{\gamma}} \tag{5}$$

where

$$\gamma = \frac{p+q}{2} + 1 \qquad for \quad p+q = 2,3,... \tag{6}$$

Normalized central moments can be employed to produce a set of invariant moments. The three lower-order invariants $\phi_1, ...\phi_3$ are given in terms of the second and third order central moments [17] by

$$\phi_1 = \eta_{20} + \eta_{02} \tag{7}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{8}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{9}$$

$\phi_1, ...\phi_3$ are scale, rotation and translation invariant. Object eccentricity is given by

$$\epsilon = [\frac{\mu_{02}\cos^2\theta + \mu_{20}\sin^2\theta - \mu_{11}\sin 2\theta}{\mu_{02}\sin^2\theta + \mu_{20}\cos^2\theta - \mu_{11}\cos 2\theta}]^2 \tag{10}$$

Definition: *Object orientation is defined as the angle between the major axis of the object and the x-axis*
It can be derived by minimizing the function

$$S(\theta) = \sum \sum_{(i,j)\in\Re} [(i-\bar{x})\cos\theta - (j-\bar{y})\sin\theta]^2 \tag{11}$$

where (i,j) belong to $\Re$ which is the space representing the image. Minimizing $S(\theta)$ gives the object orientation θ as

$$\theta = \frac{1}{2}\arctan(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}) \tag{12}$$

3.3 Object Recognition and State-Estimation

Initial trials with test data showed that the first, second and third moments of inertia were sufficient to distinguish between the landing target and other objects present in the image (Equations (7),(8),(9)). The algorithm was calibrated offline using a set of images collected in prior flights. The calibration values stored were the mean values of the moments of inertia. During actual flight the moments of inertia of each frame are calculated and compared to the calibration values. If they lie within a tolerance of $\pm 10\%$ of the stored values then the object (in this case the helipad) is said to be recognized and the algorithm proceeds to the next step of state estimation.

The state estimation algorithm calculates the x-y coordinates and orientation of the landing target relative to the helicopter. The heading is calculated using Equation 12, while the x-y coordinates of the landing target are calculated using Equation 3. These state estimates are sent to the helicopter controller.

4 Control Architecture

The AVATAR is controlled using a hierarchical behavior-based control architecture. Briefly, a behavior-based controller [18] partitions the control problem into a set of loosely coupled behaviors. Each behavior is responsible for a particular task. The behaviors act in parallel to achieve the overall goal. Low-level behaviors are responsible for robot functions requiring quick response while higher-level behaviors meet less time critical needs. The behavior-based control architecture used for the AVATAR is shown in Figure 4.

At the lowest level the robot has a set of reflex behaviors that maintain stability by holding the craft in hover. The *heading control* behavior attempts to hold the desired heading by using data from the IMU to actuate the tail rotor. The *altitude control* behavior uses the sonar to control

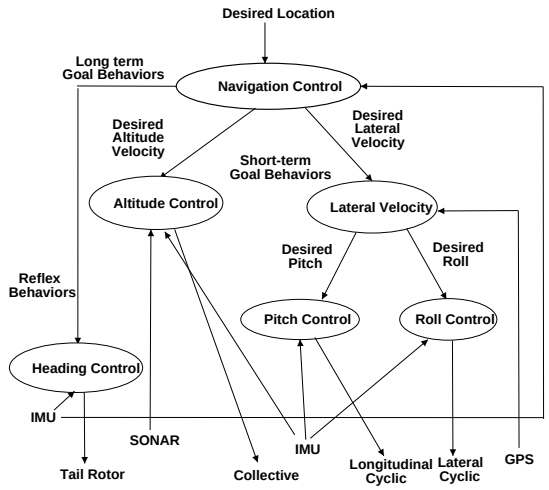


Figure 4: AVATAR Behavior-Based Controller

the collective and the throttle. The *pitch and roll control* behaviors maintain the desired roll and pitch angles recieved from the *lateral control* behavior. The *lateral motion* behavior generates desired pitch and roll values that are given to the *pitch and roll control* behaviors to move to a desired position. At the top level the *navigation control* behavior inputs a desired heading to the *heading control*, a desired altitude to the *altitude control* and a desired lateral velocity to the *lateral control* behavior. A key advantage of such a control algorithm is to build complex behaviors on top of the existing low level behaviors.

The low-level and short-term goal behaviors *roll, pitch, heading, altitude and lateral control* behaviors are implemented with proportional controllers.

The long-term goal behavior *navigation control* is responsible for overall task planning and execution. If the heading error is small, the *navigation control* behavior gives desired lateral velocities to the *lateral velocity* behavior. If the heading error is large, the *heading control* behavior is commanded to align the helicopter with the goal while maintaining zero lateral velocity.

The *altitude control* behavior is further split into three sub-behaviors, *hover control, velocity control and sonar control*. The *hover control* sub-behavior is activated when the helicopter is either flying to a goal or is hovering over the target. This sub-behavior is used during the object recognition and object tracking state when the helicopter should move laterally at a constant altitude. The hover controller is implemented as a proportional controller. It reads the desired GPS location and the current location and calculates the collective command to the helicopter. This is shown in Equation 13 where τ is the collective command sent to the helicopter servos, $f(\theta_{lat}, \theta_{lon})$ is a function of the current latitude and longitude $f(\theta_{dlat}, \theta_{dlon})$ is a function of the desired latitude and the longitude, K_p is the proportional gain. The function f converts a given latitude and longitude to the corresponding distance in meters from a surveyed point.

$$\tau = K_p(f(\theta_{dlat}, \theta_{dlon}) - f(\theta_{lat}, \theta_{lon}))$$

Once the helipad has been located and the helicopter is aligned with the helipad the *velocity control* sub-behavior takes over from the *hover control* sub-behavior. It is implemented as a PI controller. An integral term is added to reduce the steady state error. The helicopter starts to descend till reliable values are obtained from the sonar. The *sonar control* sub-behavior takes over at this point until touchdown. This is also implemented as a PI controller. The *velocity control* sub-behavior is shown in Equation 14 where τ is the collective command sent to the helicopter servos, v is the current velocity v_d is the desired velocity, K_p is the proportional gain and K_i is the integral gain.

$$\tau = K_p(v_d - v) + K_i \int (v_d - v)dt \qquad (14)$$

The *sonar control* sub-behavior is shown in Equation 15, where τ is the collective command to the helicopter servos, x is the current position, x_d is the desired position, K_p is the proportional gain and K_i is the integral gain.

$$\tau = K_p(x_d - x) + K_i \int (x_d - x)dt \qquad (15)$$

5 Experimental Results and Discussion

The helicopter is initially commanded to autonomously fly toward the helipad based on GPS [*search mode*]. Once the helipad is in view , the controller switches to vision-based control [*helipad-track mode*]. If for any reason the helicopter loses sight of the landing pad, the controller switches back to *search mode*. Once in *helipad-track mode* the low-level control behaviors on the helicopter receive commands from the vision controller. The vision system sends orientation, velocity forward and velocity right commands with respect to the image coordinate frame to the helicopter controller. The commands are then converted into velocity-north and velocity-east commands based on the current GPS and heading. The navigational control behavior takes these lateral velocity and heading commands and sends the appropriate commands to the low-level behaviors for the control of the helicopter.

Trial	Total flight time	Landing time	$\delta\theta$
1	306 s	108 s	5°
2	156 s	63 s	15°
3	316 s	112 s	0°
4	308 s	106 s	3°
5	178 s	62 s	10°
6	186 s	87 s	10°
7	194 s	66 s	2°

Table 1: Data from Flight Tests

When the helicopter is oriented with the helipad it starts descending [*land mode*]. At this juncture the helicopter is

controlled by the *velocity control* sub-behavior. If it descends to a height of 3 meters or less the *sonar control* is activated. From this point onwards the helicopter's altitude is regulated by sonar, till it lands.

Image Processing	*CPU time*
Image Acquisition	$\approx 20\%$
Thresholding and Filtering	$\approx 12\%$
Segmentation	$\approx 40\%$
Component Labeling	$\approx 7\%$
Hu's Moments of Inertia	$\approx 10\%$
GUI and displaying images	$\leq 11\%$

Table 2: Computational Cost for Image Processing at 10 frames per second

A total of seven test flights were conducted. The data obtained are shown in Table 1. The final average orientation error ($\delta\theta$) is approximately 7°. The computational cost for image processing is shown in Table 2. The time taken for computing the moments of inertia is only 10% of the total time. Hence, if the landing target has a well defined shape, the vision algorithm is computationally inexpensive. In the future we plan to test whether the same results could be obtained if we implemented our algorithm on noisy data, without filtering. Because of the limited bandwidth from the wireless video transmitter we were able to process only 10 frames per second.

Total No of Frames	12060
Landing Pad observed in	6034
Actual landing Pad present	5832

Table 3: Errors in the Object Recognition Algorithm

Table 3 shows the accuracy of the algorithm used. The data were obtained from approximately 12000 frames during the seven flight trials. Each flight was of a duration of approximately 3 minutes. Out a total number of 12060 frames processed, the landing pad was present in 6034 frames while it was detected in 5632 frames. The algorithm showed a false positive in 202 out of 6034 frames which gives an error rate of 3.36%. The moments of inertia are invariant to rotation, scaling and translation but vary when the plane in which the image lies is continually changing. The helicopter pitches and rolls in flight, which changes the image plane; this distorts the image which results in false positives. In the future we plan to integrate measurements from the IMU with the vision controller to nullify the effects caused by the roll and pitch motion.

Table 4 shows the results averaged over the seven flight trials. We were able to control the heading of the helicopter remarkably well. During the landing phase, the downward velocity is always restricted to a maximum of 0.2 m/sec. This can be seen from Figure 5(a). This was implemented for a smooth descent trajectory, as well as for safety purposes. The trajectory of the craft during descent for a representative trial is shown in Figure 5(c). Although initially there are some variations in height, the helicopter descends smoothly during the later part. For the helicopter to finally land it has to overcome ground effect and turbulence. This can be seen in Figure 5(a), when the downward velocity reaches 0.6 meters/second. The difference between the orientation of the helipad and the helicopter for a representative trial is shown in Figure 5(b). The controller is able to maintain the orientation of the craft in-line with the helipad.

The average position error after landing was 40 cm from the center of the helipad. This value is calculated as the distance from the center of the helipad to the center of the helicopter after landing. This error is small when compared to the size of the landing pad and the helicopter. Presently the camera is statically mounted below the undercarriage of the helicopter pointing down. Depending on the height of the craft even a small inclination in the craft causes a large change in the horizontal distance on the ground, making it difficult to track the landing target precisely. Mounting the camera on a gimbal would solve the problem. Also precise control of the helicopter near the ground is difficult because of the air-cushion developed by the downward thrust from the main rotor of the helicopter.

Mean time to land	73 s
Mean autonomous flight time	234 s
Mean error in orientation	6^o
Standard Deviation in orientation	5^o
Mean error in position	40 cm

Table 4: Average Results from Flight Tests

6 Conclusion and Future Work

We have presented the design and implementation of a real-time vision-based system for detecting a landing target and a controller to autonomously land a helicopter on the target. The vision algorithm is fast, robust and computationally inexpensive. It relies on the assumptions that a.) the landing target has a well-defined geometric shape and b.) all the feature points of the landing target are coplanar. Since we chose a landing target composed of polygons and the helicopter keeps the camera roughly perpendicular to the ground, these two assumptions were justified.

Data from seven flight trials show that our algorithm and landing strategy works accurately and repeatably. The helicopter achieved autonomous landing to within 40 cm positional accuracy and 7° orientation accuracy measured relative to the helipad. In the future we plan to integrate measurements from the IMU with the algorithm described here to nullify the effects caused by the roll and pitch motion thereby improving the detection of the landing target.

In the future we plan to focus our attention on the problem of safe and precise landing of the helicopter in un-

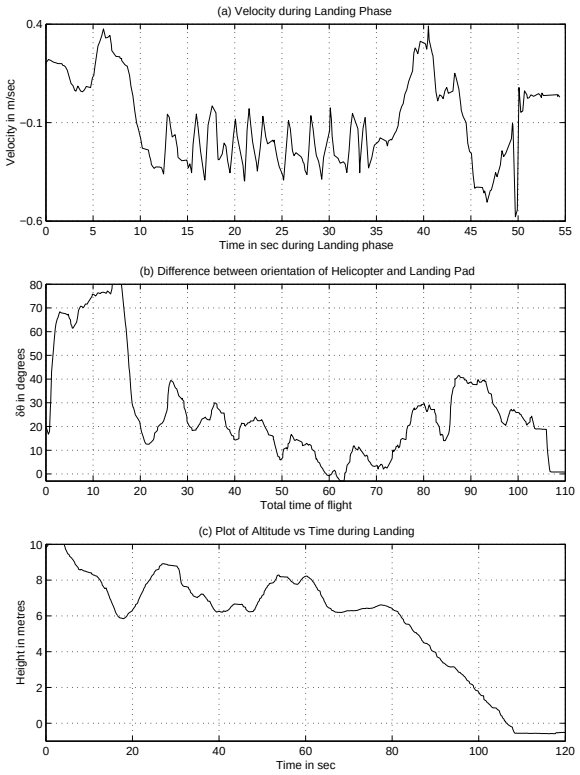


Figure 5: Performance of the Vision Algorithm in Conjunction with the Landing Controller

structured harsh 3D environments. The applications of such a system are enormous; from space exploration to target tracking and acquisition.

7 Acknowledgments

The authors gratefully acknowledge the following individuals for their support: Gerald Bohne, Doug Wilson, Kale Harbick, David Naffin, Sanjeev Koppal, Sajid Siddiqi and Boyoon Jung. This work is supported in part by NASA under JPL/Caltech contract 1231521 and by DARPA under grant DABT63-99-1-0015 as part of the Mobile Autonomous Robotic Software (MARS) program.

References

[1] F.Dellart, S.Seitz, C.Thorpe, and S.Thrun, "Structure from motion without correspondence," Tech. Rep. CMU-RI-TR-99-44, Robotics Institute, Carnegie Mellon University, December 1999.

[2] Yi Ma, Jana Kosecka, and Shankar S. Sastry, "Vision guided navigation for a nonholonomic mobile robot," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 521–537, June 1999.

[3] A.Lazanas and J.-C. Latombe, "Landmark-based robot navigation," in *Proceedings of the Tenth National Conference on Artificial INtelligence*, 1992, pp. 816–822.

[4] S.Hutchinson, G.D.Hager, and P.I.Corke, "A tutorial on visual servo control," in *IEEE Transaction on Robotics and Automation*, October 1996, vol. 12(5), pp. 651–670.

[5] Courtney S. Sharp, Omid Shakernia, and S.Shankar Sastry, "A vision system for landing an unmanned aerial vehicle," in *In Proceedings of IEEE International Conference on Robotics and Automation*, 2001, pp. 1720–1728.

[6] O.Amidi, *An Autonomous Vision-Guided Helicopter*, Ph.D. thesis, Robotics Institute, Carnegie Mellon University, 1996.

[7] Ryan Miller, Bernard Mettler, and Omead Amidi, "carnegie mellon university's 1997 international aerial robotics competition entry," in *International Aerial Robotics Competition*, 1997.

[8] Pedro J.Garcia-Padro, Gaurav S.Sukhatme, and J.F.Montgomery, "Towards vision-based safe landing for an autonomous helicopter," *Robotics and Autonomous Systems*, 2000, (accepted, to appear).

[9] O.Amidi, T.Kanade, and K.Fujita, "A visual odometer for autonomous helicopter flight," *Robotics and Autonomous Systems*, vol. 28, no. 185-193, 1999.

[10] Bruno Sinopoli, Mario Micheli, Gianluca Donato, and T. John Koo, "Vision based navigation for an unmanned aerial vehicle," in *In Proceedings of IEEE International Conference on Robotics and Automation*, 2001, pp. 1757–1765.

[11] Omid Shakernia, Y.Ma, T. John Koo, and S.Shankar Sastry, "Landing an unmanned air vehicle:vision based motion estimation and non-linear control," in *Asian Journal of Control*, September 1999, vol. 1, pp. 128–145.

[12] Andrew R. Conway, *Autonomous Control of an Unstable Helicopter Using Carrier Phase GPS Only*, Ph.D. thesis, Stanford University, March 1995.

[13] USC Autonomous Flying Vehicle Homepage, "http://www-robotics.usc.edu/˜avatar," .

[14] J.F.Montgomery, *Learning Helicopter Control through 'Teaching by Showing'*, Ph.D. thesis, University of Southern california, May 1999.

[15] R.Gonzalez and R.Woods, *Digital Image Processing*, Addison-Wesley, 1992.

[16] Ioannis Pitas, *Digital Image Processing Algorithms*, Prentice-Hall, 1993.

[17] M.K.Hu, "Visual pattern recognition by moment invariants," in *IRE Transactions on Information Theory*, 1962, vol. IT-8, pp. 179–187.

[18] Maja J. Mataric, "Behavior-based control: Examples from navigation, learning and group behavior," *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architecture for Physical Agents*, vol. 9, no. 2-3, pp. 67–83, 1997.