Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea · May 21-26, 2001

A Vision System for Landing an Unmanned Aerial Vehicle

Courtney S. Sharp Omid Shakernia S. Shankar Sastry

Department of Electrical Engineering & Computer Science
University of California Berkeley, Berkeley, CA 94720
{cssharp,omids,sastry}@eecs.berkeley.edu

Abstract

We present the design and implementation of a real-time computer vision system for a rotor-craft unmanned aerial vehicle to land onto a known landing target. This vision system consists of customized software and off-the-shelf hardware which perform image processing, segmentation, feature point extraction, camera pan/tilt control, and motion estimation. We introduce the design of a landing target which significantly simplifies the computer vision tasks such as corner detection and correspondence matching. Customized algorithms are developed to allow for real-time computation at a frame rate of 30Hz. Such algorithms include certain linear and nonlinear optimization schemes for model-based camera pose estimation. We present results from an actual flight test which show the vision-based state estimates are accurate to within 5cm in each axis of translation and 5 degrees in each axis of rotation, making vision a viable sensor to be placed in the control loop of a hierarchical flight management system.

1 Introduction

Computer vision is gaining importance as a cheap, passive and information-rich source complementing the sensor suite for control of Unmanned Aerial Vehicles (UAV). A vision system on board a UAV typically augments a sensor suite that might include Global Positioning System (GPS), Inertial Navigation Sensors (INS), laser range finders, a digital compass and sonar [2, 12, 14]. The design of any real-time vision system is a daunting task: It involves a systematic integration of hardware, low level image processing (such as segmentation and feature extraction); multiple view geometry (such as pose and structure estimation) and synthesis of real-time controllers.

Because of its structured nature, the task of autonomous landing is well-suited for vision-based state estimation and control and has recently been an active topic of research [5, 8, 9, 10, 11, 14, 15]. In [7] a technique is presented for estimating the pose relative to a known object given a scaled orthographic projection model of a camera. In [15], the use of vanishing points of parallel lines on a landmark is proposed for the purpose of estimating the location and orientation of a UAV relative to a landing pad. Since their technique relies on vanishing points of parallel lines, their algorithm is most sensitive to noise and gives the worst

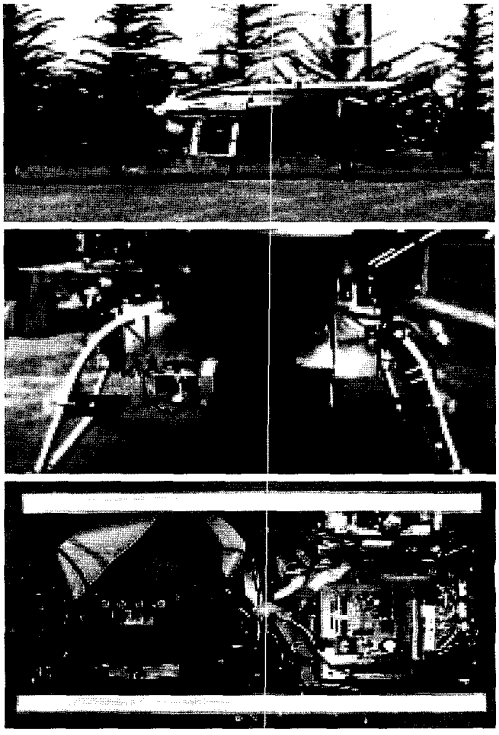

Figure 1: Showing the Berkeley UAV testbed: Yamaha R-50 helicopter (top), mounted pan/tilt camera and computer box (middle), and on-board navigation and vision computers (bottom).

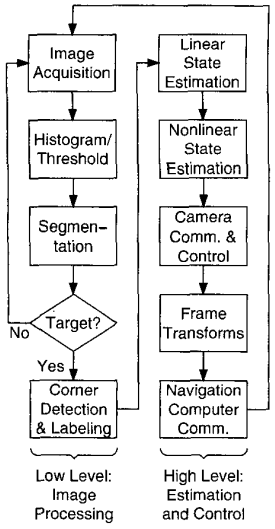

Figure 2: Vision system software flow-chart: image processing followed by estimation and control.

pose estimates when it matters the most: when the UAV is directly over the pad.

In this paper, we introduce a design and implementation of a real-time vision system for a rotor-craft UAV which estimates its pose and speed relative to a known landing target at 30Hz. Our vision system uses *customized* vision algorithms and *off-the-shelf* hardware to perform in real-time: image processing, segmentation, feature point extraction, camera control, as well as both linear and nonlinear optimization for model-based pose estimation. Actual flight test results on our UAV testbed show our vision-based state estimates are accurate to within 5cm in each axis of translation and 5 degrees in each axis of rotation, making it a viable sensor to be placed in the control loop of a hierarchical flight management system [6, 12].

Paper Outline: Section 2 contains a detailed description of each component of our vision system. Section 3 presents system integration details and experimental results from real flight experiments, and Section 4 gives concluding remarks and directions for future research.

2 Vision System Software Design

Our vision system software consists of two main stages of execution: image processing and state estimation, each with a sequence of subroutines. Figure 2 shows a flow-chart of the algorithm.

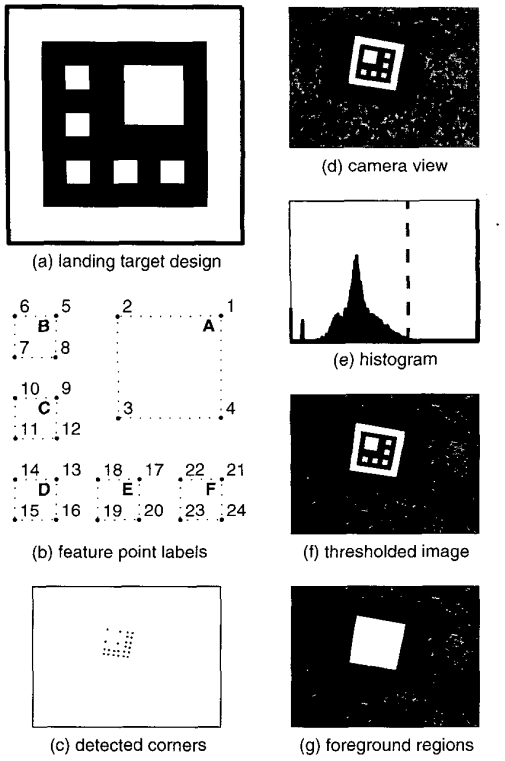

Figure 3: Landing target design and image processing. The target design (a) is made for simple feature labeling (b), robust feature point extraction (c), and simplified image segmentation (d-g).

2.1 Image processing

Our goal for image processing is to locate the landing target then extract and label its feature points. This process includes: (1) thresholding the grayscale image to a binary one, (2) segmenting the landing target out of the background, (3) detecting the corners in landing target, and finally (4) labeling those corners. In order to simplify the image processing, the design of the landing target must make it easy to identify and segment from the image background, provide distinctive feature points, simplify feature labeling, and allow for algorithms that can execute in real-time using off-the-shelf hardware.

Figure 3(a) shows our landing target design. The connected white border enclosing the target simplifies segmenting it from the background, given that landing target lies on a dark background and that no other connected white region completely encloses it. Figure 3(b) shows the feature point labeling on the corners of the

interior white squares of the landing target. We choose corner detection over other forms of feature point extraction because it is simple, robust, and provides a high density of feature points per image pixel area. We choose squares over other n-sided polygons because they maximize the quality of the corners under perspective projection and pixel quantization. Moreover, our particular organization of the squares in the target allows for straight-forward feature point matching invariant of Euclidean motion and perspective projection, as to be shown in the section "Feature Labeling." Color tracking was also explored as a cue for feature points, however we found color tracking to not be robust because of the variability of outdoor lighting conditions.

(1) Thresholding. The thresholding algorithm must produce a binary image such that the black and white regions of the landing target are preserved and that the primary image background is connected and black. We found the algorithm with the most acceptable compromise between best-case quality and overall robustness to be one of the simplest: threshold the image based on a fixed percentage between the min and max gray levels. Figure 3(d) to Figure 3(f) give results of this thresholding scheme tested on a real image input from the on-board camera.

(2) Segmentation. Given a binary image as in Figure 3(f), our segmentation stage must separate the landing target from the background and return the interior squares. We segment the landing target via two consecutive passes of a standard connected components labeling algorithm [3] based on *4-connectivity*. The first pass identifies the background as the largest black component (see Figure 3(g)). The second pass identifies the landing target as the single foreground region with seven white components and one black component.

(3) Corner Detection. The corner detection problem we face is highly structured: We need to detect the corners of 4-sided polygons in a binary image. The structured nature of the problem allows us to avoid the computational cost of a general purpose corner detector.

The fundamental invariant in our corner detector is that *convexity* is preserved under Euclidean motion and perspective projection. This implies that for a line through the interior of a convex polygon, the set of points in the polygon with maximal distance from each side of the line contain at least two distinct corners of the polygon.

To find two arbitrary corners of a 4-sided polygon, we compute the perpendicular distance from each edge point to the vertical line passing through the center of gravity of the polygon. If there is more than one point with maximal distance on a side of the line, we choose the point which is farthest from the center of gravity. We then find the third corner as the point of maximum distance from the line connecting the first two corners. Finally, we find the fourth corner as the point of the polygon with maximum distance to the triangle defined by the first three corners. Figure 3(c) shows the output of our corner detection algorithm on a sample image.

(4) Feature Labeling. Our feature labeling algorithm is based on the following property. For some plane $\mathcal{P}$ containing points $q_1, q_2, q_3 \in \mathbb{R}^3$, consider the smallest magnitude angle θ between vectors $(q_2 - q_1)$ and $(q_3 - q_1)$, where $\theta > 0$ is CCW when viewing some surface side of $\mathcal{P}$. Given any image of points q_1, q_2, q_3 taken from the same side of $\mathcal{P}$, if θ' is the corresponding angle in the image of those points, then $\text{sign}(\theta') = \text{sign}(\theta)$. This property guarantees that correspondence matching based on clockwise ordering of feature points will work for any given image of those feature points.

Our feature labeling algorithm consists of two steps: first, each square is uniquely identified based on its center of gravity, then the corners within each square are identified as in Figure 3(b). We label the squares of the target starting with the identification of square D. For each square, we compute the vectors between its center to the centers of the other squares. We identify square D as the square with two pairs of collinear vectors. We identify the remaining squares by ordering them from square D. Finally, we order squares associated with the collinear vectors from square D by their distance from it. To complete the feature labeling process, we identify the corners of each square in a similar manner.

2.2 Pose Estimation

(1) Geometry of Planar Features. Given the labeled feature points, estimating the UAV state is the so-called model based camera pose estimation problem from computer vision. We apply both linear and nonlinear optimization algorithms toward this problem. The linear optimization algorithm is globally robust but sensitive to noise. The nonlinear optimization algorithm requires adequate initialization but is more robust to noise. Thus we solve the camera pose estimation problem by first solving the linear problem and using those results to initialize the nonlinear algorithm.

The equation relating a point in the landing pad

coordinate frame to the image of that point in the camera-head frame is given by the equation

$$\lambda_i \mathbf{x}_i = APgq_i, \tag{1}$$

where $\lambda_i \in \mathbb{R}$ is an unknown scale, $\mathbf{x}_i \in \mathbb{R}^3$ is the homogeneous coordinates of the feature point in the image plane, $A \in \mathbb{R}^{3\times3}$ is the camera calibration matrix, $P = [I\ 0] \in \mathbb{R}^{3\times4}$ is the projection matrix, $g \in SE(3)$ is the homogeneous representation of the Euclidean motion between the landing pad coordinate frame and the camera-head frame, and $q_i \in \mathbb{R}^4$ is the homogeneous representation of the point in the world. Using a *calibrated* pinhole model for the perspective projection of the camera, without loss of generality we set $A = I_{3\times3}$. Then the scale term λ_i is given by

$$\lambda_i = e_3^T Pgq_i, \tag{2}$$

where $e_3 = [0\ 0\ 1]^T \in \mathbb{R}^3$. Equations (1) and (2) together imply the following constraint

$$(\mathbf{x}_i e_3^T - I)[R\ p]q_i = 0. \tag{3}$$

We have knowledge of each q_i from the geometry of the designed landing target. The corner detection algorithm extracts each feature point $\mathbf{x}_i$ in the image frame. The feature labeling algorithm associates each $\mathbf{x}_i$ with its corresponding q_i. From this data, we need to recover the camera pose $Pg = [R\ p]$ where $R \in SO(3)$ is the rotation and $p \in \mathbb{R}^3$ is the translation from the landing target to the camera head.

(2) Linear Optimization. Since all the feature points lie on the the plane of the landing target, without loss of generality we may chose the inertial coordinate frame such that $e_3^T q_i = 0$ for all i. Taking $q_i = [q_{i1}\ q_{i2}\ q_{i3}\ 1]^T$, and $[R\ p] = [r_1\ r_2\ r_3\ p]$, equation (3) implies

$$\left(\mathbf{x}_i e_3^T - I\right)[r_1\ r_2\ p] \begin{bmatrix} q_{i1} \\ q_{i2} \\ 1 \end{bmatrix} = 0. \tag{4}$$

Since the above equation is linear in $[r_1\ r_2\ p]$, we can reorganize equation (4) into vector form for each feature correspondence pair $(\mathbf{x}_i, q_i)$ and stack all the equations to get

$$F \begin{bmatrix} r_1 \\ r_2 \\ p \end{bmatrix} = 0, \tag{5}$$

where $F = [F_1^T \cdots F_n^T]^T \in \mathbb{R}^{2n\times9}$, n is the number of feature points on the landing target, and

$$F_i = \begin{bmatrix} q_{i1} & 0 & -q_{i1}\mathbf{x}_{i1} & q_{i2} & 0 & -q_{i2}\mathbf{x}_{i1} & 1 & 0 & -\mathbf{x}_{i1} \\ 0 & q_{i1} & -q_{i1}\mathbf{x}_{i2} & 0 & q_{i2} & -q_{i2}\mathbf{x}_{i2} & 0 & 1 & -\mathbf{x}_{i2} \end{bmatrix},$$

where $\mathbf{x}_i = [\mathbf{x}_{i1}\ \mathbf{x}_{i2}\ 1]^T \in \mathbb{R}^3$.

By a slight modification of a well known result on the planar structure from motion problem (see, for example Weng [13]), it can be shown that if there are at least 4 features points such that no three are collinear, then $\text{rank}(F) = 8$. However, due to noise in corner detection, in practice F is always full rank. Hence we compute the least squares estimate of the null space of F by applying standard SVD techniques to compute the singular vector $\left[\tilde{r}_1^T\ \tilde{r}_2^T\ \tilde{p}^T\right]^T \in \mathbb{R}^9$ corresponding to the smallest singular value of F. Given the "positive-depth constraint" that the landing pad is in front of the camera, if necessary we negate the singular vector result of the SVD computation to ensure that $\tilde{p}_3 \geq 0$.

Next, we solve for scale of the translation $p \in \mathbb{R}^3$ by computing the scale factor on the vector $\left[\tilde{r}_1^T\ \tilde{r}_2^T\ \tilde{p}^T\right]^T$ that gives $\tilde{r}_1$ and $\tilde{r}_2$ unit norm; namely, $p = 2\tilde{p}/(\|\tilde{r}_1\| + \|\tilde{r}_2\|)$.

Finally, we solve for the rotation matrix R from the estimates $\tilde{r}_1$ and $\tilde{r}_2$ in two steps. First we project the matrix $[\tilde{r}_1\ \tilde{r}_2\ 0] \in \mathbb{R}^{3\times3}$ onto the group of orthogonal matrices $O(3)$ by computing the SVD of $[\tilde{r}_1\ \tilde{r}_2\ 0] = U\Sigma V^T$ and setting $R = UV^T$. To ensure that R is a rotation matrix, we need $\det(R) = 1$. Thus, if our SVD computation yields $\det(UV^T) = -1$, we flip the sign of the third column vector of R, which is equivalent to setting $R = UV^TQ$, where $Q = \text{diag}(1, 1, -1) \in \mathbb{R}^{3\times3}$.

In practice, the linear algorithm described above is quite noisy because it estimates 9 parameters for a system of equations with 6 degrees of freedom. However, the linear estimate is close enough to the true solution to serve as a good initialization for the nonlinear optimization technique.

(3) Nonlinear Optimization. The nonlinear optimization algorithm minimizes the *reprojection error* $G = [G_1^T \cdots G_n^T]^T \in \mathbb{R}^{2n}$ where $G_i \in \mathbb{R}^2$ is given by

$$G_i = (\mathbf{x}_i e_3^T - I)[R\ p]q_i, \tag{6}$$

where the last row is ignored from the right hand side matrix. The rotation matrix is parameterized by ZYX-Euler angles θ_1, θ_2, θ_3, where $\theta_i \in (-\pi, \pi]$ and $R = e^{\hat{e}_3\theta_3}e^{\hat{e}_2\theta_2}e^{\hat{e}_1\theta_1}$. The estimation parameters for the nonlinear optimization are $\beta = [\theta_1\ \theta_2\ \theta_3\ p_1\ p_2\ p_3]^T$.

We apply the the vector form of the Newton-Raphson method to iteratively solve for β:

$$\beta_{n+1} = \beta_n - k_n(D_\beta G|_{\beta_n})^\dagger\ G(q, y, \beta_n) \tag{7}$$

where k_n is an adaptive step size, $D_\beta G$ is the Jacobian of G with respect to β, and $(D_\beta G|_\beta)^\dagger$ is the Moore-Penrose pseudo-inverse of $D_\beta G|_\beta$.

We symbolically calculate the Jacobian for the estimation parameters and numerically evaluate its value at runtime. β_0 is initialized by the result of the linear estimate or by a recent nonlinear estimate, if available. We adaptively select k_n to guarantee $\|G(q, y, \beta_n)\|$ monotonically decreases for successive n.

For any rotation matrix R, there exist two congruent Euler angle parameterizations such that $\delta = [\theta_1, \theta_2, \theta_3, p_1, p_2, p_3]$ and $\gamma = [\pi + \theta_1, \pi - \theta_2, \pi + \theta_3, p_1, p_2, p_3]$ are equivalent parameterizations for $[R\ p]$. When $\cos(\theta_2) \neq 0$ it is direct to check by symbolic computation that each iteration of (7) produces equivalent results for both parameterizations; that is

$$D_\beta G|_\delta = (D_\beta G|_\gamma)\,\mathrm{diag}(1, -1, 1, 1, 1, 1). \qquad (8)$$

Thus, iterations of β_n in (7) step through equivalent rotations R_n regardless of the particular Euler parameterization.

The nonlinear algorithm outperforms the linear algorithm because it optimizes over only the necessary 6 parameters of the transformation. However, due to the nonlinearities there are many local minima. Hence, the algorithm is highly sensitive to initialization and thus is only useful given a decent initialization from the linear algorithm.

3 System Integration and Results

(1) Hardware. As part of the BErkeley AeRobot (BEAR) project [1], our UAV testbed is a Yamaha R-50 helicopter (see Figure 1) on which we have mounted:

- *Navigation Computer*: Pentium 233MHz Ampro LittleBoard running QNX real-time OS – responsible for low level flight control [12]
- *Inertial Measurement Unit*: NovAtel MillenRT2 GPS system (2cm accuracy) and Boeing DQI-NP INS/GPS integration system
- *Vision computer*: Pentium 233MHz Ampro LittleBoard running Linux – responsible for grabbing images, vision algorithms and camera control
- *Camera*: Sony EVI-D30 Pan/Tilt/Zoom camera
- *Framegrabber*: Imagenation PXC200
- *Wireless Ethernet*: WaveLAN

The framegrabber captures images at 30Hz, which currently sets the upper-bound on the rate of our vision estimates. The inter-relationship of this hardware as it is mounted on the UAV is depticted in Figure 4.

(2) Camera Control. Proper control of the pan/tilt camera can increase the range of motion of the UAV while keeping the landing target in the field of view of the camera-head. The PTZ camera we use

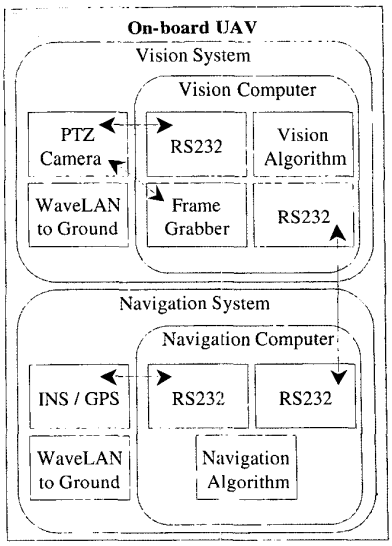

Figure 4: Organization of hardware on the UAV.

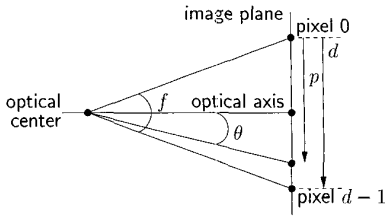

Figure 5: Geometry of pan/tilt with respect to optical center and image plane. f is the field of view in radians, d is the width in pixels, p is the location in pixels of the target projected onto the axis, and θ is the rotation required to center the target.

has an internal controller which can be commanded to relative or absolute pan/tilt locations via a serial link. By convention, positive pan moves the target left in the image, and positive tilt moves the target down in the image. For our camera, the axes of rotation for pan and tilt coincide with its optical center. For a particular axis in the image, as in Figure 5, the angle between a point on the axis and the center of the axis is given by

$$\theta = h(p, d, f) = \mathrm{atan2}\left(p - \frac{d}{2}, \frac{d}{2}\cot\left(\frac{f}{2}\right)\right), \qquad (9)$$

where f is the field of view of the axis in radians, d is the width of the axis in pixels, and p is the location in pixels of the target projected onto the axis.

Now, let the desired location of the target be at (x_0, y_0), the target be at (x_1, y_1), the image width and height be l_x and l_y, and the horizontal and vertical field of view be f_x and f_y. Then, using equation (9), the amount of pan and tilt necessary to move the target to the desired location is given by

$$\begin{aligned} \theta_{pan} &= h(x_1, l_x, f_x) - h(x_0, l_x, f_x) \\ \theta_{tilt} &= h(y_1, l_y, f_y) - h(y_0, l_y, f_y), \end{aligned}$$

where θ_{pan} and θ_{tilt} are relative to the current pan/tilt state.

(3) Frame Transformations. As shown in Figure 6, the geometry of the coordinate frames and Euclidean motions involved in the vision-based state estimation problem are labeled as: (a) Landing target, (b) Landing pad, (c) Camera head, (d) Camera base, (e) UAV, (f) Inertial frame. We use the notation $g_{ba} \in SE(3)$ to denote the Euclidean motion (translation and rotation) of coordinate frame b with respect to frame a. The transformations g_{ef} and g_{fb} are used only to evaluate the state estimates of the vision algorithm and are not used by the vision algorithm directly.

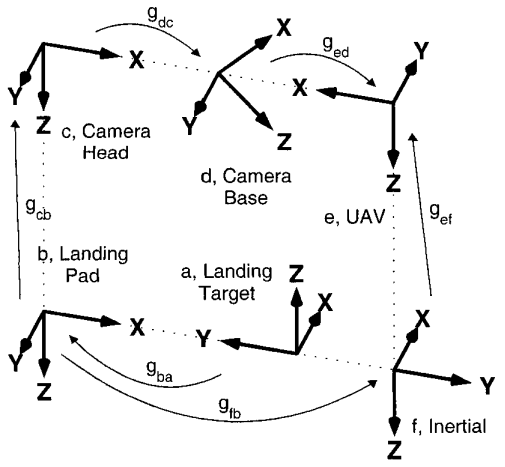

Figure 6: Geometry of the coordinate frames and Euclidean motions involved in the vision-based state estimation problem.

(4) Software. LAPACK (Linear Algebra PACKage) and BLAS (Basic Linear Algebra Subprograms) [4] are used in the vision algorithms for standard matrix operations. A multi-resolution approach is used in our segmentation algorithm to reduce processing time, effectively performing segmentation on a 160×120 image. Then the edges of the interior squares of the landing target are calculated from the original 320×240 image. We have developed a "ground station" which communicates through wireless ethernet to the onboard vision computer and displaying the current state of the vision system through a Java-based GUI (shown in Figure 7).

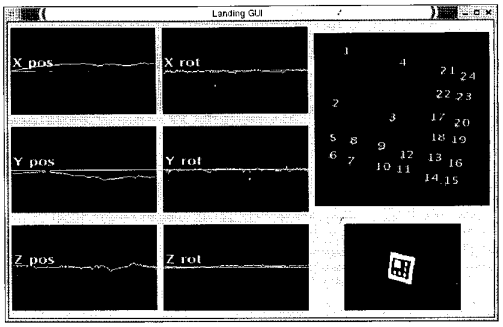

Figure 7: Ground station Java-based visualization GUI showing position and rotation estimates, extracted feature points and labels, and the binary image view from the camera.

(5) Flight Test Results. For the flight test, the UAV hovered autonomously above a small, stationary landing pad with the vision system running in real-time. While the vision system did not affect control of the UAV, state estimates from the navigation system were synchronously gathered for later comparison.

Figure 8 shows the results from the flight test, comparing the output of the vision-based state estimation algorithm with the INS/GPS measurements accurate to 2cm. All plots show the state of the UAV with respect to the landing pad. The vision estimates are more noisy, but otherwise follow the INS/GPS measurements. Also, errors in the internal and external camera calibration parameters marginally affects some of the estimates – the x-position and z-rotation, in particular. While logging a fraction of the original images in real-time, the Linux write caching system caused noticeable, synchronized gaps in each data plot.

Figure 9 shows the root mean squared error of each estimated state parameter. The position estimates are within 5cm accuracy. The x- and y-position estimates should perform better than the z-position estimate. That is, the same amount of translation along the x- or y-axis causes more detectable change in the image than along the z-axis, the optical axis of the camera. However, this is not apparent in our plots due to the noted calibration error. The rotation estimates are all within 5 degrees accuracy. As expected, the z-rotation

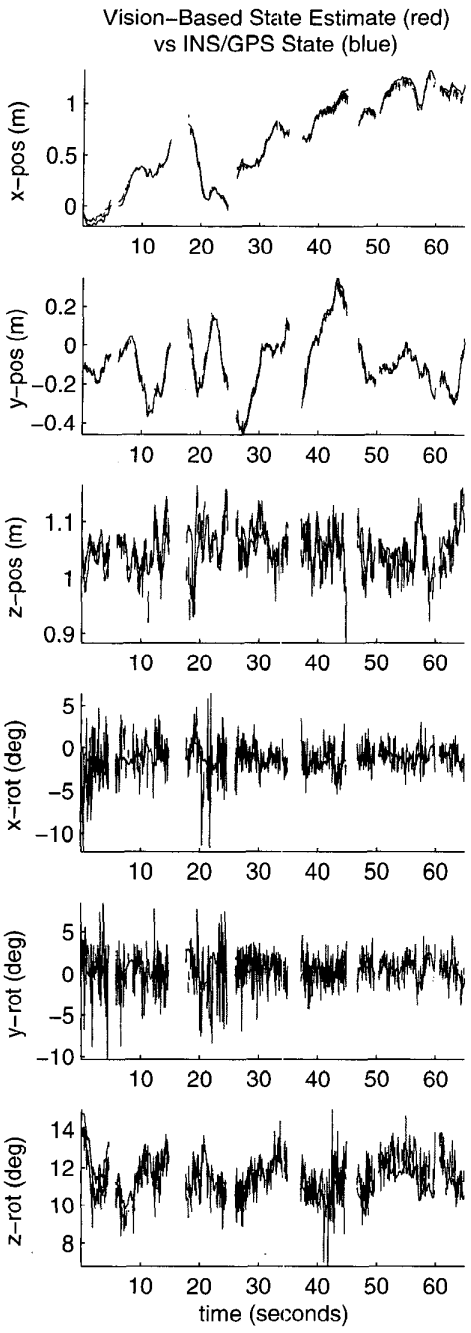

Figure 8: Flight test results: the vision estimates are more noisy but otherwise follow the INS/GPS state of the UAV relative to the landing pad. A calibration error is most notable in the estimates of x-position and z-rotation.

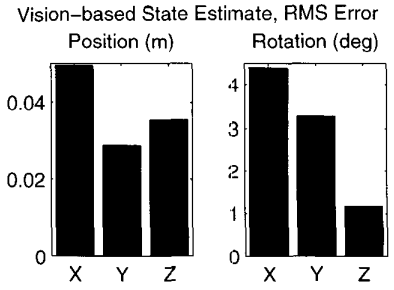

Figure 9: The RMS error of state estimates for each axis of translation and rotation. Position error is within 5cm. Rotation error is within 5 degrees.

"Low Level Image Processing"	
Segmentation pass 2, edge detection (low res)	42%
Threshold, segmentation pass 1	30%
Image acquisition	8%
Corner detection (high res)	6%
Histogram estimation	4%
Feature labeling	<1%
Total	90%

"Other Processing"	
Nonlinear optimization	5%
Bit-encoding binary image for GUI	2%
Linear optimization	2%
Frame transformations	1%
Misc.	<1%
Total	10%

Table 1: Computational cost of each stage of processing, measured from actual program execution. Image processing is expensive, consumed by segmenting the landing target from the image.

estimate significantly outperforms the other two. That is, the same amount of rotation about the z-axis causes more detectable change in the image than the other axes. Overall, the vision-based state estimates are accurate enough to be used in the closed-loop of a high-level feedback controller for landing in a hierarchical flight management system [6, 12].

Table 1 shows the computational cost of each stage of processing in our system, measured from actual program execution. In particular, we see that almost all of our computational time is spent on image processing. In turn, most of the image processing is spent on the segmentation algorithms for extracting the landing target. The conclusion is that any further effort toward improving computational efficiency should be spent optimizing the image segmentation algorithms.

4 Conclusion and Future Work

In this paper, we have presented the design and implementation of a real-time vision system to land a UAV onto a landing target. Some aspects of our system perform particularly well. Our feature labeling is computationally inexpensive and extremely robust to noise. The camera control algorithm performed well given the dynamic limits of the pan/tilt actuators of our camera. Given a good threshold for a grayscale image, our segmentation algorithm never failed to extract the landing target. With adequate initialization, our estimates from nonlinear optimization proved to be robust to noise. However, the thresholding and corner detection algorithms have room for improvement.

Flight test results show the vision based motion estimates are accurate to within 5cm in each axis of translation and 5 degrees in each axis of rotation. The estimates are sufficiently accurate to allow our vision sensor to be placed in the control loop of a hierarchical flight management system [6, 12]. For future research, we will use our vision system to land an UAV onto a moving platform which simulates the motion of a ship deck, as shown in Figure 10.

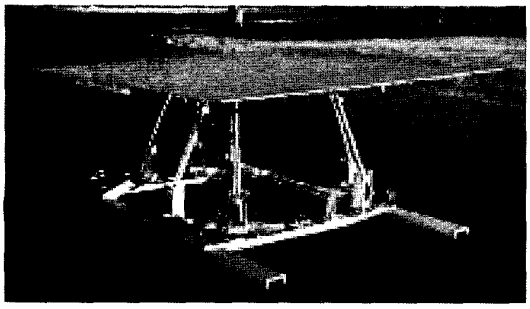

Figure 10: Programmable 6 DOF landing platform to simulate watercraft dynamics. To be used in future work.

Acknowledgments

The authors would like to thank the BErkeley AeRobot team, especially Hyunchul Shim and René Vidal. This research was supported by ONR grants N00014-00-1-0621 and N00014-97-1-0946 and ARO MURI grant DAAH04-96-1-0341.

References

[1] BErkeley AeRobot (BEAR) Project homepage. `http://robotics.eecs.berkeley.edu/bear`.

[2] M. Bosse, W.C. Karl, D. Castanon, and P. DeBitetto. A vision augmented navigation system. In *IEEE Conference on Intelligent Transportation Systems*, pages 1028–33, 1997.

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

[4] LAPACK, BLAS Homepages. `http://www.netlib.org/{lapack,blas}`.

[5] I. Kaminer, A. M. Pascoal, W. Kang, and O. Yakimenko. Integrated vision/inertial navigation system design using nonlinear filtering. *To Appear: IEEE Transactions on Aerospace and Electronics.*

[6] T.J. Koo, F. Hoffmann, H. Shim, B. Sinopoli, and S. Sastry. Hybrid Control of Model Helicopter In *Proc. of IFAC Workshop on Motion Control,* pages 285-290, Grenoble, France October 1998.

[7] D. Oberkampf, D.F. DeMenthon, and L.S. Davis. Iterative pose estimation using coplanar feature points *Computer Vision and Image Understanding* 63(3):495–511, May 1996.

[8] A. Petruszka and A. Stentz. Stereo vision automatic landing of VTOL UAVS. In *Proceedings of Assoc. Unmanned Vehicle Syst. Int.*, pages 245–63, 1996.

[9] F.R. Schell and E.D. Dickmanns. Autonomous landing of airplanes by dynamic machine vision. *Machine Vision and Applications*, 7:127–134, 1994.

[10] O. Shakernia, Y. Ma, T.J. Koo, J. Hespanha, and S. Sastry. Vision guided landing of an unmanned air vehicle. In *Proceedings of CDC*, Phoenix, Arizona, December 1999.

[11] O. Shakernia, Y. Ma, T.J. Koo, and S. Sastry. Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control. *Asian J. of Control*, 1(3):128–145, September 1999.

[12] D.H. Shim, H.J. Kim, and S. Sastry. Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicles. In *Proceedings of AIAA Conf. on Guidance, Navigation and Control*, Denver, 2000.

[13] J. Weng, T.S. Huang, and N. Ahuja. *Motion and Structure from Image Sequences.* Springer-Verlag, 1993.

[14] S. Werner, S. Furst, D. Dickmanns, and E.D. Dickmanns. A vision-based multi-sensor machine perception system for autonomous aircraft landing approach. In *Proc. of the SPIE - The International Society for Optical Engineering*, vol 2736, pages 54–63, Orlando, FL, USA, 1996.

[15] Z.F. Yang and W.H. Tsai. Using parallel line information for vision-based landmark location estimation and an application to automatic helicopter landing. *Robotics and Computer-Integrated Manufacturing*, 14(4):297–306, 1998.