COVER SHEET

"This is a preprint of an article published in Journal of Field Robotics Vol 23. Issues 3-4.PP:185-199, http://www.interscience.Wiley.com/.

Visual Servoing of an Autonomous Helicopter in Urban Areas Using Feature Tracking

Luis Mejías, Srikanth Saripalli, Pascual Campoy and Gaurav S. Sukhatme

Abstract— **We present the design and implementation of a vision-based feature tracking system for an autonomous helicopter. Visual sensing is used for estimating the position and velocity of features in the image plane (urban features like windows) in order to generate velocity references for the flight control. These visual-based references are then combined with GPS-positioning references to navigate towards these features and then track them. We present results from experimental flight trials, performed in two UAV systems and under different conditions that show the feasibility and robustness of our approach.**

Index Terms— **Unmanned Aerial Vehicle, feature tracking, visual servoing, autonomous helicopter**

I. Introduction

Our goal is to build vision-guided autonomous flying robots. Vision allows such robots to serve as intelligent eyes-in-the-sky suitable for numerous applications including law enforcement, search and rescue, aerial mapping and inspection, and movie making. Vision for flight control encompasses a broad spamming of vision-based object detection and tracking, optical position estimation, inertial navigation, GPS, and non-linear system modeling. An autonomous helicopter is highly suitable for tasks like inspection, surveillance and monitoring. The ability of the helicopter to fly at low speeds, hover, fly laterally and perform maneuvers in narrow spaces makes it an ideal platform for such tasks. Electric power companies use helicopters to inspect towers, transmission lines and other defects [Campoy et al., 2000]. This ability can be extended to urban environments where vision can be used for navigation and obstacle avoidance [Hrabar and Sukhatme, 2003]. One can also envisage tasks such as inspection and surveillance, where the helicopter is required to recognize some set of features and track them over time. This ability is particularly useful in urban structured environments where the features that are to be tracked have many similar geometric properties. Vision provides a natural sensing modality for feature detection and tracking. In many contexts (e.g. urban areas, airports) the structured nature of features (e.g. windows) facilitates vision-based state estimation and control.

We combine vision with low-level control to achieve precise autonomous vision-based feature tracking for an unmanned autonomous helicopter. The vision-based system described here acts as an overall controller sending navigation commands to a flight controller which is responsible for autonomous control of the helicopter. The result is an overall algorithm for vision-based tracking of features using an autonomous helicopter in structured 3D environment. Our tests have been performed on two platforms. The first platform AVATAR [Montgomery, 2000] at the University of Southern California and the second COLIBRI [COLIBRI, 2005] at Universidad Politécnica de Madrid. In the experiments explained later the helicopter [Figures 1 and 2] is initialized in hover at an arbitrary location. A user selects feature from the ground control unit (in this case windows on a building) which the helicopter should align to and track in successive frames.

II. Related Work

Recently there has been significant interest in small autonomous flying vehicles in the robotics community. An early autonomous navigation system for a model-scale helicopter (the Hummingbird) was reported in [Conway, 1995]. The unique feature of this system was the sole use of GPS as the navigation sensor replacing the Inertial Measurement Unit. In [Jun et al., 1999] a system based on integration of the onboard INS and GPS was used to produce accurate position and velocity estimates. The

Luis Mejías and Pascual Campoy are with the Computer Vision Group at Universidad Politécnica de Madrid, Madrid, Spain. Email: lmejias, campoy@etsii.upm.es. Srikanth Saripalli and Gaurav S. Sukhatme are with the Robotic Embedded Systems Laboratory, Center for Robotics and Embedded Systems, University of Southern California, Los Angeles, USA. E-mail: srik,gaurav@robotics.usc.edu.

Fig. 1. The Autonomous Vehicle Aerial Tracking and Reconnaissance (AVATAR)

Fig. 2. The COLIBRI autonomous helicopter during a flight trial

autonomous helicopter reported in [Amidi, 1996], [Miller et al., 1997], had a combination of vision and GPS for navigation capability. The onboard DSP-based vision processor provided navigation information such as position, velocity and attitude at an acceptable delay (on the order of 10ms), which was combined with GPS and IMU data for accurate attitude and position measurements. The reader is referred to [Saripalli et al., 2003b], [Shim et al., 1998], [Johnson, 1980], [Conway, 1995], [Montgomery, 1999] for an overview of the various types of vehicles, theory and the algorithms used for their control. Recent work has included autonomous landing [Saripalli et al., 2003a], [Shakernia et al., 1999], aggressive maneuvering of AFVs (Autonomous Flying Vehicles) [Gavrilets et al., 2002], and pursuit-evasion games [Vidal et al., 2002].

In [Bosse, 1997], a vision augmented navigation system is discussed for autonomous helicopter control which uses vision in-the-loop to control a helicopter. A notable vision-based technique used in autonomous helicopter control, is the visual odometer [Amidi et al., 1998], which provides accurate navigational information (position and velocity) which is combined with inertial measurements. In [Wu et al., 2005] vision is used as additional sensor and fused with inertial and heading measurements for control of an unmanned rotorcraft. In [Garcia-Pardo et al., 2001] a vision-based solution is given for safe-landing site detection in unstructured terrain where the key problem is for the onboard vision system to detect a suitable place to land, without the aid of a structured landmark such as a helipad. Recent work on autonomous landing using vision and inertial sensing is described in [Merz et al., 2004]. Previously we have shown a real time computer vision system for tracking a landing target and have successfully coupled it with a helicopter controller to achieve landing [Saripalli et al., 2003a]. Work on window tracking for one dimensional visual control of an unmanned autonomous helicopter is reported in [Mejias et al., 2005]

III. Autonomous Helicopter Testbed

The experimental testbed, AVATAR (Autonomous Vehicle for Aerial Tracking And Reconnaissance) [Montgomery, 2000], [AVATAR, 2005] is a gas-powered radio-controlled model helicopter fitted with a PC-104 stack augmented with sensors. A Novatel RT-2 DGPS system provides positional accuracy of 2 cm 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). An ISIS-IMU unit with three single-axis accelerometers and three single-axis gyroscopes provides rate information to the onboard computer, which is fused using a 16 state kalman filter. 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 via 802.11b. Autonomous flight is achieved using a *behavior-based* control architecture [Saripalli et al., 2003a].

The COLIBRI helicopter testbed is based on a gas powered model helicopter twin stroke engine with 52cc and 8 hp, fitted with a Xscale based

flight computer augmented with sensors (GPS, IMU, Magnetometer also fused with a kalman filter). For vision processing it has a VIA mini-ITX 1.25GHz computer onboard with 512Mb RAM, wireless interface, and a firewire color camera for acquiring the images. Both computers run Linux. The ground station is a laptop used to send high-level control commands to the helicopter. Is also used for visualization of image data and communications with the onboard image processing algorithm. Communication with the ground station is via 802.11g wireless Ethernet.

IV. VISUAL PREPROCESSING

We present an image-based velocity references approach to visually control the displacement of a helicopter and two vision approaches to detect and track features on buildings. The visual control approach has been tested in two stages. In the first approach, the helicopter autonomously detects features (in this case windows) and based on the output of the vision algorithm, is commanded to laterally align with the window. On a higher level a user select (on a GUI) the detected windows which the helicopter should align to. The altitude of the helicopter is maintained independently using GPS. In the second approach the user selects a window to track and the visual controller sends lateral as well as vertical displacement commands to the helicopter, in effect controlling both the lateral position as well as the altitude.

Both systems have in common the same visual processing architecture, that consists of two main tasks running in a client-server architecture. The *server* task (image processing), which runs on board the helicopter, extracts the features and performs tracking. The *client* task consists of a high level graphical user interface that sends higher level commands to the helicopter and also logs the data.

A. AVATAR: feature detection and tracking

In order to improve the performance of the system, the entire image is not processed. In a general case, image processing needs to be performed over the entire image to extract the desired features, but this task requires high speed processing or special purpose hardware in order to work at frame rates (30Hz), but for the task of feature tracking not all pixels in the image are of interest. Thus the computational cost can be reduced if only a local area of the image is processed, our approach falls on the category of *window-based tracking* [Haralick and Shapiro, 1992] techniques. Our algorithm measures the match between a fixed-size window with features with an initially picked template along a sequence of images. The position of the local search window is first located in the same position of the template, then is successively updated in the previous successful matches. The vision system consists of five stages. The feature detection algorithm is described below in three stages; thresholding, segmentation and square finding. Once the feature is detected the tracking is performed in two stages: template matching and window tracking, both of which are described later in this section.

1) Segmentation and Thresholding: The purpose of this stage is to extract the color that characterizes the object of interest. Such a segmented image is then converted to grayscale by thresholding. Next template matching is performed for feature recognition. The Equation used to convert a color image to grayscale is given by [OpenCV, 2005]

$$Y = 0.21267 * R + 0.715160 * G + 0.072169 * B \quad (1)$$

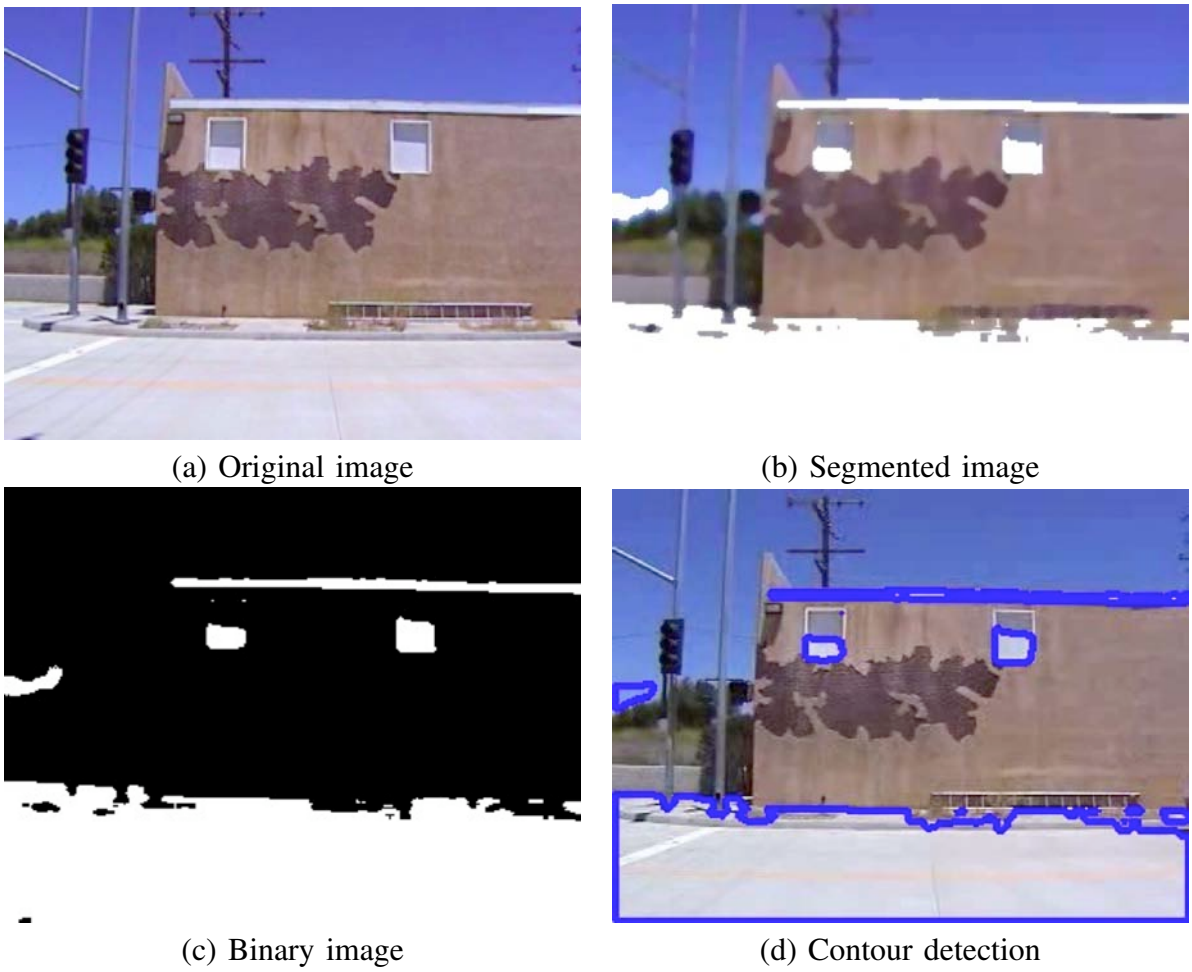
where: R,G,B are the red, green and blue image channels. The formulation for threshold based segmentation is as follows:

Let be I the image with component I_r, I_g, I_b, respectively

Algorithm 1: SEGMENTATION()

$$\begin{cases} \textbf{for } i \leftarrow N1 \textbf{ to } N2 \\ Y_i = 0.21267 * I(i)_r + 0.715160 * I(i)_g + 0.072169 * I(i) \\ \begin{cases} \textbf{if } Y_{low} < Y_i < Y_{up} \\ I_i = C_f \\ \quad \textbf{else} \\ I_i = C_b \end{cases} \end{cases}$$

where:
$N1$ and $N2$ are the limits of the local search area
C_f and C_b the values for foreground and background
Y_{low} and Y_{up} are the lower/upper thresholds,
usually, $Y_{low} = 0.7 * Y_c$ and $Y_{up} = Y_c$. Y_c is the gray scale projection of the target color using (1). Next the segmented image is thresholded to produce a binary image where the object of interest is represented by 1's and the background with 0's.



(a) Original image (b) Segmented image

(c) Binary image (d) Contour detection

Fig. 3. Sequence of the image preprocessing operations.

2) Square finding: This is intended to identify those geometric features that are candidates for features to be tracked, in our case a window. This algorithm takes a binary image (Figure 3c), finds the contours and approximates it. The detected contours are represented by continuous lines composed of one or more line segments (Polylines). An algorithm for reducing the points in a polyline to produce a simplified polylines is applied. The Douglas-Peucker (DP) algorithm [Douglas and Peucker, 1973] is used for this purpose. This algorithm works from the top down by starting with a crude initial guess at a simplified polyline, namely the single edge joining the first and last vertices of the polyline. Then the remaining vertices are tested for closeness to that edge. If there are vertices further than a specified tolerance, $\epsilon > 0$, away from the edge, then the vertex furthest from it is added the simplification. This creates a new guess for the simplified polyline. Using recursion, this process continues for each edge of the current guess until all vertices of the original polyline are within tolerance of the simplification.

More specifically, in the DP algorithm, the two extreme endpoints of a polyline are connected with a straight line as the initial rough approximation of the polyline. How well it approximates the whole polyline is determined by computing the distances from all intermediate polyline vertices to that (finite) line segment. If all these distances are less than the specified tolerance ϵ, then the approximation is good, the endpoints are retained, and the other vertices are eliminated. However, if any of these distances exceeds the ϵ tolerance, then the approximation is not good enough. In this case, we choose the point that is furthest away as a new vertex subdividing the original polyline into two (shorter) polylines.

A rectangle is extracted using the convexity and angle between the vectors of the approximated contour. This algorithm is used for the first few frames, once a window is detected, the matching algorithm

takes over to track the window independently, even when the detection algorithm fails. This is done for robustness reasons where detection might be impossible during forward flight. Figure 3 summarizes the early preprocessing sequence.

3) Template Matching: The matching process starts by selecting a patch of 40x40 pixels around the location of the target chosen by the user. This patch is successively compared with the local search window of 100x100 in the grey scale image. This local search area is first located in the same position of the patch. It is then updated using the location of the previous successful match. The template matching is performed by measuring the similarity between the patch and the features in the local search area in successive image sequences. The output of this process is a quantitative measure of similarity which is converted to image coordinates. We use the *normalized cross correlation* which is defined by:

$$\zeta = \frac{\sum_{\hat{x}}^{w-1}\sum_{\hat{y}}^{h-1} T(\hat{x},\hat{y})I(x+\hat{x},y+\hat{y})}{\sqrt{\sum_{\hat{x}}^{w-1}\sum_{\hat{y}}^{h-1} T^2(\hat{x},\hat{y})\sum_{\hat{x}}^{w-1}\sum_{\hat{y}}^{h-1} I^2(x+\hat{x},y+\hat{y})}} \quad (2)$$

In the above Equation w and h are boundaries of the local area, $I(x,y)$ and $T(x,y)$ represents the image and template intensities, respectively.

4) Kalman Filter: Once a suitable match between the target template and the features in the image is found a Kalman filter is used to track the feature positions in the image sequence over time. The inputs to the kalman filter are the x and y coordinates (in pixels units) given by the template matching algorithm. The outputs are the estimates of these coordinates in the next frame. Based on a second order kinematic model for the tracked object we model the Equation of the target as a linear system described by:

$$X_{k+1} = AX_k + Bu_k + w_k \quad (3)$$

where w_k is random process noise and the subscripts on the vectors represent the time step. X_k is the state vector describing the motion of the target (its position p, velocity v and acceleration a). The measurement vector at time k is given by

$$Z_k = H_k X_k + \nu_k \quad (4)$$

where H_k is known and ν_k is random measurement noise. A second order Kalman Filter is used for tracking the target. The filter is formulated as follows. Suppose we assume that the process noise w_k is white, zero-mean, Gaussian noise with a covariance matrix Q. Further assume that the measurement noise is white, zero-mean, Gaussian noise with a covariance matrix R, and that it is not correlated with the process noise. The system dynamics in general form is $\hat{X}_k = \phi_{k-1}\hat{X}_{k-1} + w_k$ expanding the Equation is

$$\begin{bmatrix} p_k \\ v_k \\ a_k \end{bmatrix} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{k-1} \\ v_{k-1} \\ a_{k-1} \end{bmatrix} + w_k$$

where a_{k-1} is a random time-varying feature acceleration, v_{k-1} is the velocity and p_{k-1} its position. T is the time between the steps k and $k-1$. For this task the feature position is only taken into account reducing the state vector to $\hat{X}_k = [p_k] = [x\ y]^T$ The state propagation and update Equation for the discrete Kalman filter are given by [Kalman, 1960]

$$\hat{X}_k(-) = \Phi_{k-1}\hat{X}_{k-1}(-) \quad (5)$$

$$P_k(-) = \Phi_{k-1}P_{k-1}(+){\Phi_{k-1}}^T + Q_{k-1} \quad (6)$$

$$S_k = H_k P_k(-){H_k}^T + R \quad (7)$$

$$K_k = P_k(-){H_k}^T {S_k}^{-1} \quad (8)$$

$$P_k(+) = (I_n - K_k H_k)P_k(-) \quad (9)$$

$$\hat{X}_k(+) = \hat{X}_k(-) + K_k(Z_k - H_k\hat{X}_k(-)) \quad (10)$$

In Equations 5 to 10, the superscript T indicates matrix transposition. Φ is the transition matrix. S is the covariance of the innovation, K is the gain matrix, and P is the covariance of the prediction error. Equation 5 is state estimate extrapolation. Equation 6 is the state covariance extrapolation. Equation 8 is the Kalman gain matrix. Equations (9)(10) are the state covariance update and the state estimate update, respectively. Also we distinguish between estimates made before and after the measurements occur. $\hat{X}_k(-)$ is the state estimate that results from the propagation Equations alone (i.e., before the measurements are considered) and $\hat{X}_k(+)$ is the corrected state estimate that accounts for measurements. $P_k(-)$ and $P_k(+)$ are defined similarly.

The output of the Kalman filter ($\hat{X}_k(+)$) is used as an error signal *e* for controlling the velocity of

the helicopter. The derivation of the visual references from the feature position estimation is similar to Equation 13. The Figure 4 taken during flight summarizes the detection and tracking process. The small circle represent the feature picked by the user and the bigger circle represents the output of the Kalman filter.

Fig. 4. Detection and tracking process

B. COLIBRI: feature tracking and visual control

Fig. 5. Window being tracked during flight

The tracking algorithm consists of a Lucas-Kanade tracker [Lucas and Kanade, 1981], [Tomasi and Kanade, 1993]. The Lucas-Kanade algorithm (which is a Gauss-Newton gradient descent non-linear optimization algorithm) is derived as follows.

$$\sum w(x,y)I_xI_yu+\sum w(x,y)I_y^2v=-\sum w(x,y)I_yI_t \quad (11)$$

$$\sum w(x,y)I_x^2u+\sum w(x,y)I_xI_yv=-\sum w(x,y)I_xI_t \quad (12)$$

The goal of the Lucas-Kanade algorithm is to minimize the sum of squared error between two images, the template and the image warped back onto the coordinate frame of the template. The value $w(x,y)$ is the Gaussian window, I_x and I_y the pixel values, I_t is the current grey level intensity image, u and v are components of optical flow field in x and y coordinates respectively.

An implementation [Bouguet Jean Yves, 1999] of this algorithm is used to tracks 4 points corresponding to four corners of a window. A grey level version of the captured image is used for this purpose. Two of the four points (corners) mentioned before are initialized by the user. First the user picks two points (opposite corners) then the remaining two points are searched and detected in the neighborhood of the first two. The criteria to find these remaining two points are based on the eigen values, grey level values, corner image location, etc. The central point of these four corners is then used as a basis for the visual reference.

During visual processing since the detection and tracking of objects occurs in image space, the natural output from such an algorithm is velocity in image space. This represents the control inputs to the helicopter in terms of velocity in the body-coordinate frame.

1) Lateral Visual Reference: Once the object of interest is located, its location in the image is used to generate the visual references to the flight control (Figure 6). If the camera is located approximately at the center of mass of the helicopter ($Xhc = 0$) the angles β and α will coincide. Given the fixed kinematic relationship between the camera and the helicopter the task of visual servoing consists of making the angles zero that produces the alignment of the vehicle with the target if the value of i is $\frac{w}{2}$.

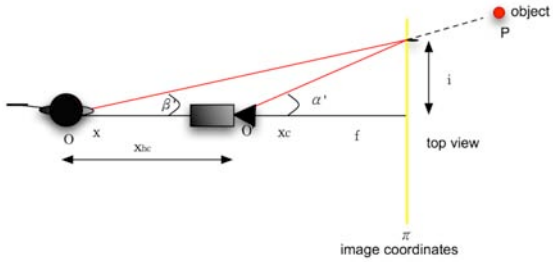


Fig. 6. Helicopter-Camera reference: top view

Based on the above diagram i is the value that makes the references higher when the object is far

from the center and minimum when the helicopter is in the image center. The lateral visual signal that commands the helicopter laterally is given by Equation 13.

$$vy_{ref} = 2\frac{(i - \frac{w}{2})}{w} \tag{13}$$

where w is the image width and i is the horizontal component of the object location in the image.

2) Vertical Visual References: Using the same approach mentioned above and taking as reference the scheme (Figure 7) the task of visual servoing can be accomplished if the value of j is $\frac{h}{2}$.

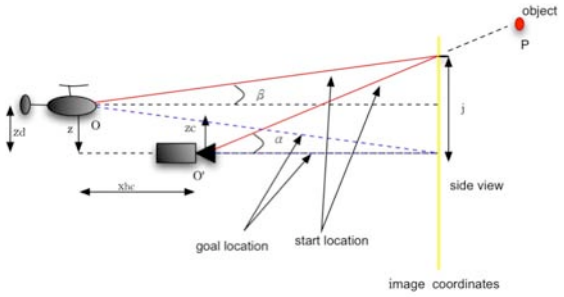


Fig. 7. Helicopter-Camera reference: lateral view

Based on that, the vertical visual signal that commands the helicopter vertically is given by Equation 14.

$$vz_{ref} = 2\frac{(j - \frac{h}{2})}{h} \tag{14}$$

where h is the image height and j is the vertical component of the object location in the image.

Before the references are sent to the flight controller a low pass filter is applied. The general form of the filter is the following

$$X = (1 - k) \times X^{t-1} + k \times X^{t} \tag{15}$$

where $X = [vy_{ref}\ vz_{ref}]^T$, the vector X^t encodes the current values. The value of k has been empirically chosen to be equal to 0.9.

V. Control Architectures

A. AVATAR flight control architecture

The AVATAR is controlled using a hierarchical behavior-based control architecture. The behavior-based control architecture used for the AVATAR is shown in Figure 8. The low-level behaviors have been extensively described in previous work [Saripalli et al., 2003a], we give a brief summary below and focus on the behaviors specific to the vision-based lateral control problem.

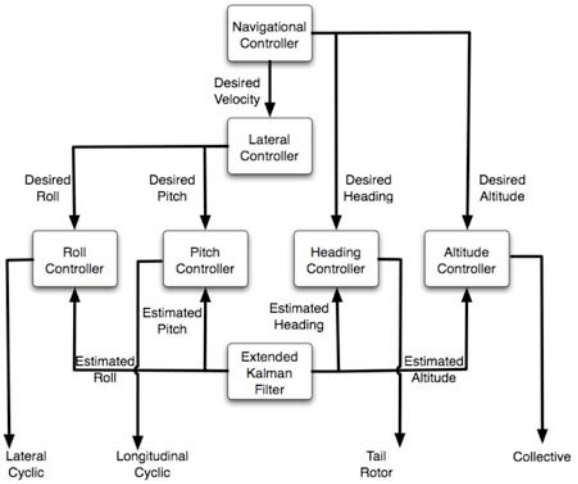


Fig. 8. AVATAR behavior-based controller

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 kalman filtered heading to actuate the tail rotor. The *altitude control* behavior uses the sonar, GPS and IMU to control the collective and the throttle. The *pitch and roll control* behaviors maintain the desired roll and pitch angles received from the *lateral velocity* behavior. The *lateral velocity* behavior generates desired pitch and roll values that are given to the *pitch and roll control* behaviors to achieve a desired lateral velocity. At the top level the *navigation control* behavior inputs a desired heading to the *heading control*, a desired altitude or vertical velocity to the *altitude control* and a desired lateral velocity to the *lateral control* behavior. A key advantage of such a control algorithm is the ability to build complex behaviors on top of the existing low level behaviors, without changing them.

B. COLIBRI flight control architecture

The overall scheme of the flight controller is shown in the Figure 9. The controller is based on a decoupled PID control in which each degree of freedom is controlled separately based on the assumption that the helicopter dynamics are decoupled. The attitude control stabilizes the helicopter in hover maintaining the desired roll, pitch and

heading. The velocity and position controllers are responsible to generate the appropriate references to the attitude controller in order to maintain a desired velocity or position, respectively.

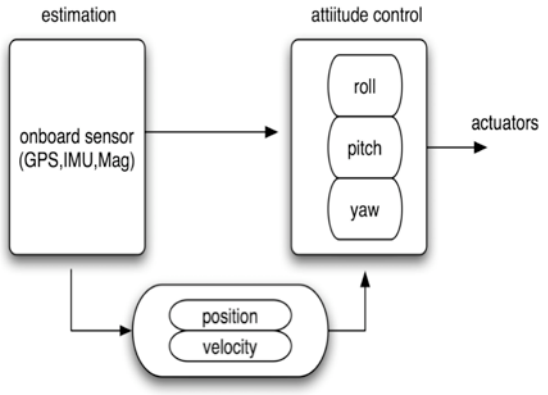


Fig. 9. Flight controller scheme

The attitude controller is implemented as a proportional-plus-derivative (PD) control. The heading is implemented with a proportional-plus-derivative-plus-integral control, the respective output signal are given according to the following Equations

$$\delta_{lat} = K_p(\phi_d - \phi) + K_d\frac{(\phi_d - \phi)}{dt} \tag{16}$$

$$\delta_{lon} = K_p(\theta_d - \theta) + K_d\frac{(\theta_d - \theta)}{dt} \tag{17}$$

$$\delta_t = K_p(\psi_d - \psi) + K_i\int(\psi_d - \psi)dt + K_d\frac{(\psi_d - \psi)}{dt} \tag{18}$$

where: δ_{lat}, δ_{lon} and δ_t are the lateral cyclic, longitudinal cyclic and collective tail commands, respectively. The values K_p, K_i and K_d are the proportional, integral and derivative gains associated with each controller. The position and velocity controller are implemented in the same way according with same control law. These values were obtained for autonomous navigation and hover. These gains will be reviewed in the future to tune them for a best visual servoing task.

VI. Experimental Results

Two sets of experiments were performed to test the vision system with combined flight control. These were conducted in two different locations and experimental conditions. The first set of flights were performed at Del Valle Urban Search and Rescue training site in Santa Clarita, California, using the AVATAR helicopter. The second set of tests were performed on two different days to validate the vision system with the flight control. This set of flights were done in a closed urban area at the ETSII campus in Madrid, Spain using the COLIBRI helicopter.

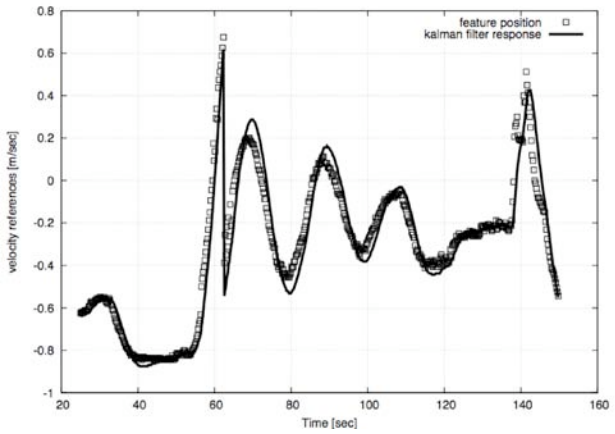
A. Experimental Results at Del Valle Urban Search and Rescue training site in Santa Clarita, California

This section presents the results obtained during flight trials. In our experiments the helicopter is commanded to fly autonomously to a given GPS waypoint. The vision algorithm then takes control aligning the helicopter to the features being tracked. Four flight trials were performed, the results of which are summarized below.

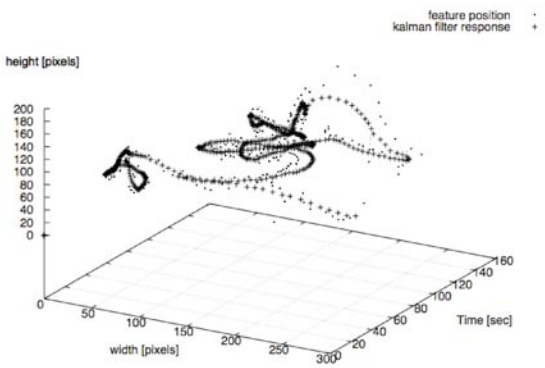
The sequence of events during autonomous control is as follows: The helicopter starts in autonomous hover at some GPS location. It is commanded to GOTO a particular GPS location which is in the vicinity of the features to be detected and tracked (dashed line segment in Figures 10(c)(d)). The features in these set of experiments happened to be rectangular windows. As soon as the helicopter detects these features the controller on the helicopter switches from GPS-based control to vision-based control (solid line segment in figures 10(c)(d)). The Kalman filter-based tracker commands the low level controller on the helicopter. The helicopter then tracks the window in successive frames, and produces the necessary velocity commands to the controller such that it can hover facing the window. The object of interest in these experiments was a window which was 4 meters away from the GPS waypoint.

Figure 10(a) shows the velocity commands generated by the vision algorithm in the image plane, solid line. This signal is the output of Kalman filter as mentioned in IV-A.4. Note that for lateral control only the y component of this signal is taken and normalized between -1m/s and 1m/s. Figure 10(b) shows the location of the features in the image plane. Both the output of the raw correlation based feature tracker and the Kalman filter are shown.

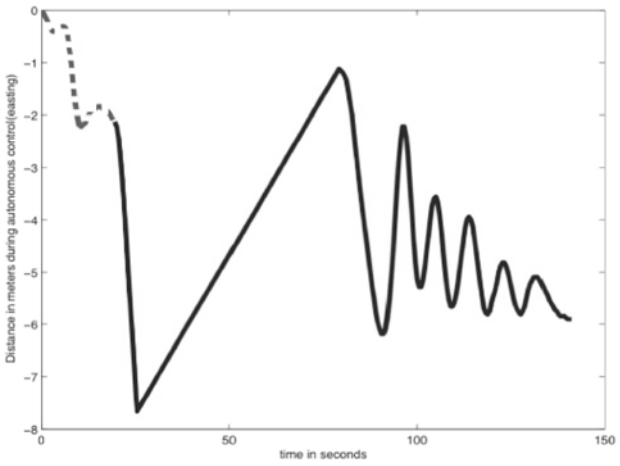
Figures 10(c) and 10(d) show the path taken by the helicopter while tracking the features. Good



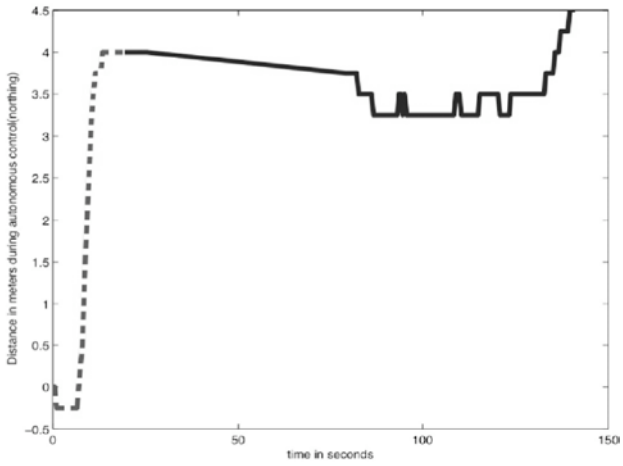
a) Error signal (lateral velocity reference) vs time during vision-based lateral control



b) Position of the feature in the image plane vs time during vision-based lateral control



c) Helicopter lateral displacement vs time during flight trial



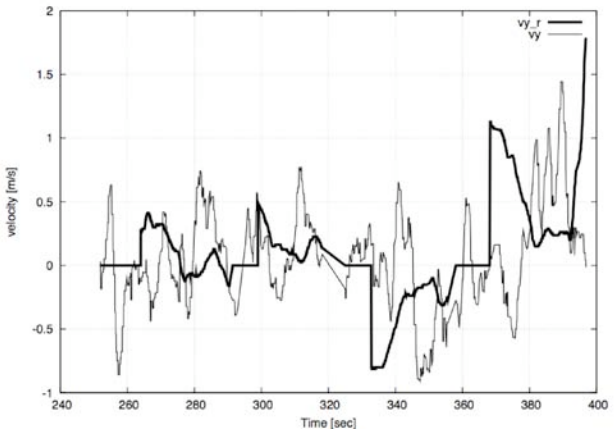
d) Helicopter longitudinal displacement vs time during the flight trial

Fig. 10. a)Normalized error signal (m/sec) b)Location of the features in the image c)Helicopter position in meters (UTM coordinates-Easting) d)Helicopter position in meters (UTM coordinates-Northing)

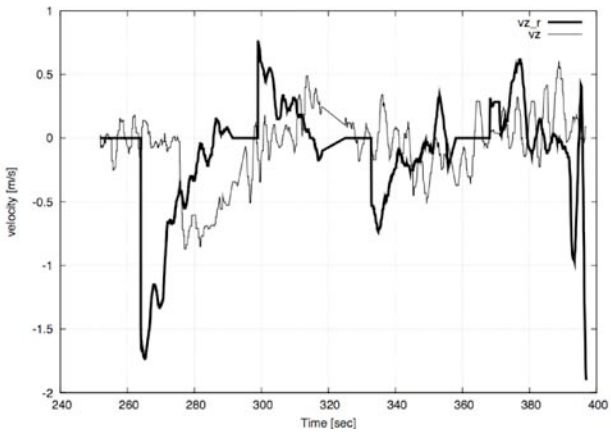
correlation can be seen with respect to the desired commands by the vision system and the path of the helicopter. This can be noticed by comparing Figures 10(a) and 10(c) where a corresponding change in velocity shows a proportional change in position as expected. In figures 10(c) and 10(d) the solid line represents the time when the helicopter is controlled based on GPS and the dashed line represents vision-based control. The oscillations which are seen in the graph are due to the Proportional-Integral (PI) control used. In the future we plan to test an Proportional-Derivative (PD) control to dampen the oscillations. Also a very large time period of oscillation can be seen (around 10 seconds) which is to be expected since we command only small changes (+/- 1m/sec). In these experiments the vision processing task was ran on the same flight control computer. The current image processing rate for this set of experiments was 15fps, we believe that with an independent vision processor the system will work at frame rate (30Hz).

B. Experimental Results at ETSII campus in Madrid, Spain

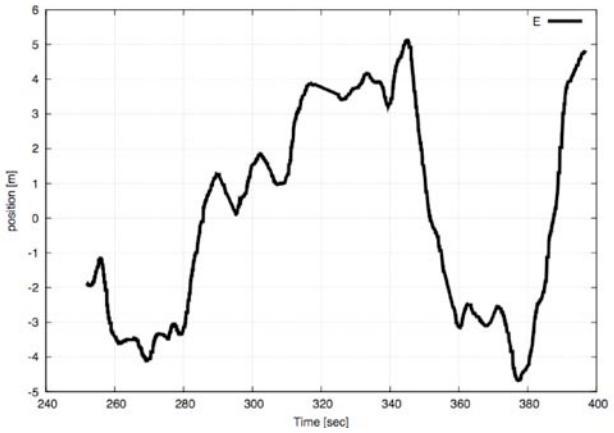
A total of seven experimental trials on two different days were performed to validate the vision system with the flight control. This section presents the results obtained during these trials. In the experiments the helicopter is commanded to fly autonomously to a given GPS way-point and then the vision algorithm takes control aligning the helicopter to the feature being tracked. From the set of experiments the two most relevant experimental results are presented here. Videos and more information from the flight trials can be downloaded from



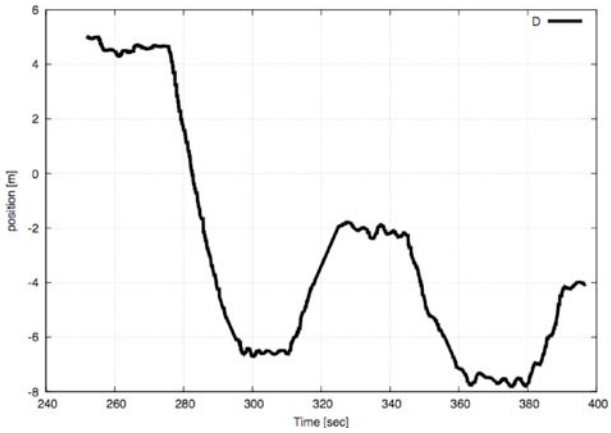
a) vy and visual references vs time during visual servoing task



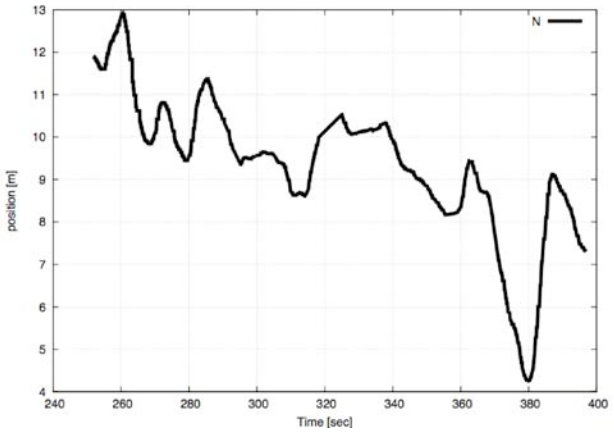
b) vz and visual references vs time during visual servoing task



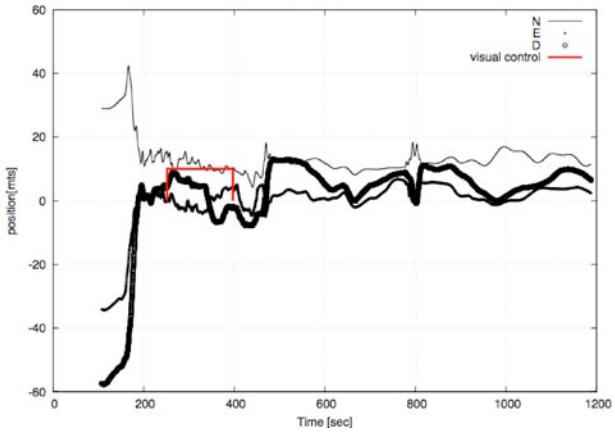
c) Helicopter position (UTM-easting) vs time during visual servoing task



d) Helicopter position (UTM-down) vs time during visual servoing task



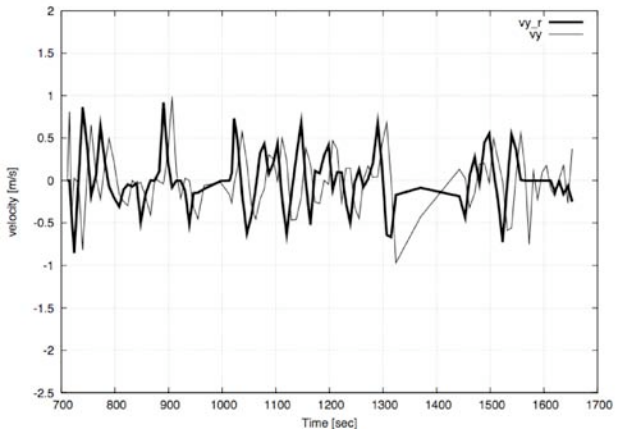
e) Helicopter position (UTM-northing) vs time during visual servoing task



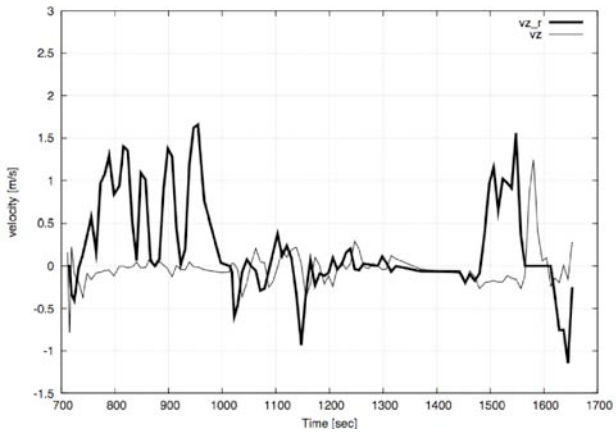
f) Helicopter general log for position (UTM coordinates) during the entire flight trial
visual control is high when vision takes control

Fig. 11. Performance of the system during sixth flight trial. Subfigures a) to e) refers to vision based control, subfigure f) shows the helicopter displacements during the entire flight trial. Position is shown in meters in the UTM coordinates system NED.

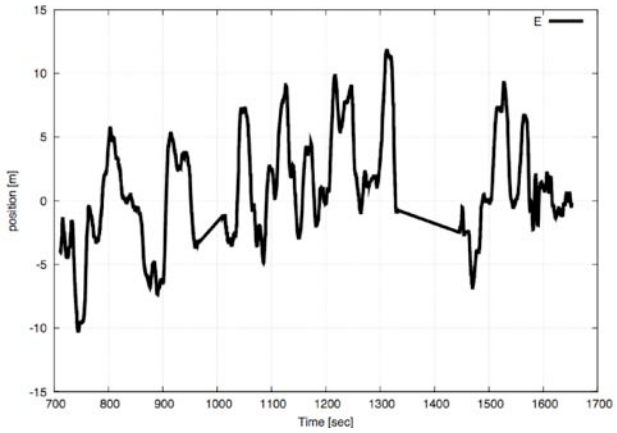
`http://www.disam.upm.es/colibri`. Figures (11) and (12) depict the results for these two trials. The data taken from the helicopter are shown in Figures 11(f) and 12(f). The visual task occurs at $250 < t < 400$ sec and $700 < t < 1650$ sec, respectively.



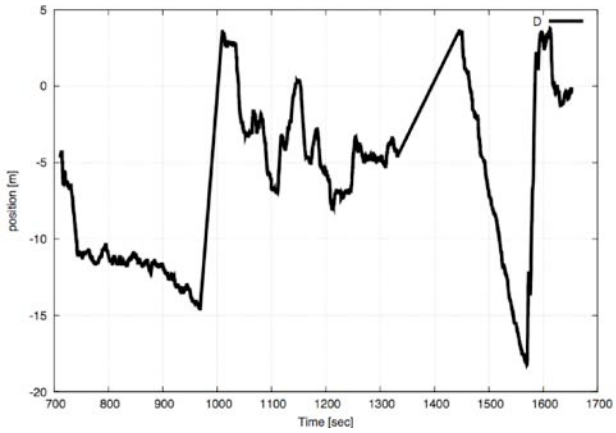
a) vy and visual references vs time during visual servoing task



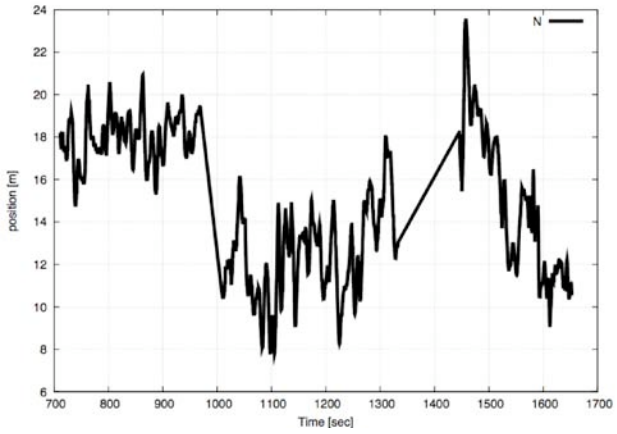
b) vz and visual references vs time during visual servoing task



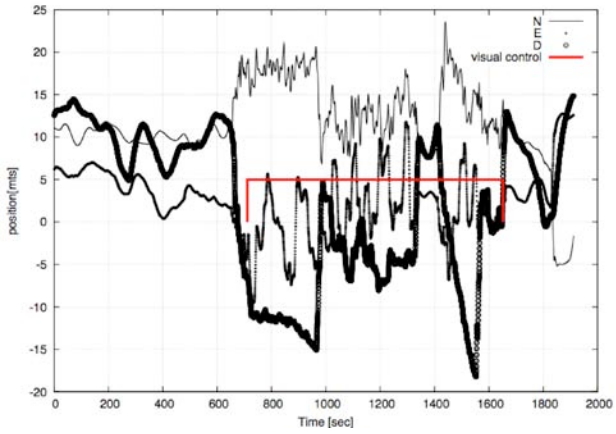
c) Helicopter position (UTM-easting) vs time during visual servoing task



d) Helicopter position (UTM-down) vs time during visual servoing task



e) Helicopter position (UTM-northing) vs time during visual servoing task



f) Helicopter position (UTM coordinates) during the entire flight trial.
visual control is high when vision takes control

Fig. 12. Performance of the system during seventh flight trial. Subfigures a) to e) refers to vision based control, subfigure f) shows the helicopter displacements during the entire flight trial. Position is shown in meters in the UTM coordinates system NED.

Figures 11(a-e) and Figures 12(a-e) show the location of the helicopter and the references involved while the visual control is performed. Figures 11(a) and 11(b) show the visual references with respect to the helicopter velocity in the lateral and vertical direction. Figure 11(b) correlates the vertical visual

references (vz_r) and the helicopter velocity (vz). This shows the correspondence between two signals that causes the displacements in altitude (down) observed in Figure 11(d). Analogously Figure 11(a) and 11(c), correlates the velocity references (vy_r) with the helicopter velocity (vy) and lateral displacement (east). Velocity references are given in body frame. For this particular task a negative vy_r causes negative evolution in the lateral position of the helicopter[1]. Negative values vz_r will cause the helicopter to moves up (positive evolution in D) and positive to move down.

In general the vision-based controller is able to command the requisite velocities that are required for the helicopter to successfully track the features (in our case windows). From the Figures 11(a-b) and Figures 12(a-b) it can be seen that when there is a noticable change in the targets position in the image plane the algorithm produces the requisite visual reference commands to the velocity controller on the helicopter. This then causes a change in the velocity of the helicopter and subsequently a change in the position of the helicopter such that it is able to align and track the target. Although we do not know the global coordinates of the target, we can see that the reference commands converge to zero which implies that the helicopter has the target in the center of its image coordinate frame.

The behavior observed in the longitudinal velocity in Figures 11(e) and 12(e) is undesirable but understandable since during the task vx_r is fixed to zero which does not guarantee that the helicopter keeps its longitudinal position, since we are controlling velocity not position. A combination of the two approaches, i.e, position and velocity control will cause a better general behavior.

The rate at which the visual processing sends command to the flight controller was one of the most critical parameters. We have noticed that the higher the vision system sends commands to the flight control, the smoother is the resulting helicopter behavior.

That is expected since vision sends input reference commands (in term of sample rates) to a high level controller. The low sample rate cause long reaction time and poor perturbation rejection. In these experiments special emphasis was placed on the speed. In the current version the visual processing rate is 20 fps. The fact that the tracking algorithm only tracks 4 points allows the vision system to run at this frame rate.

[1]Note that the lateral and longitudinal positions correspond to easting and northing in these particular set of experiments. This is because the helicopter was flying with its nose pointing towards north. This is not true in all cases.

VII. Conclusion and Future Work

We have experimentally demonstrated an approach to visually control an autonomous helicopter. Such a capability is very important for aerial robotics. Our strategy can be extrapolated to many tasks that require visual servoing for UAV. Such tasks require the vision algorithm to command the UAV towards features of interest when GPS has dropouts (usually in urban areas) or to track a target. Our algorithm assumes that the UAV can accept lateral, longitudinal and altitude velocity commands and does not depend on the inherent dynamics of the UAV. Most of the UAV in production or research use (both fixed wing and rotorcraft can take velocity commands as input) [DARO, 1998]. This was experimentally demonstrated by performing vision-based window tracking tasks on two different platforms at different locations and different conditions. Although the vision algorithms differed in their structure depending on the nature of the task, the control inputs to the helicopter controller remained the same. Both visual processing stages were tested in outdoor environments and track features even with a 10^o change in roll. However its believed that the segmentation algorithm could be sensitive to strong changes in ambient light, brightness, shadows, etc. For this reason a more detailed study of all the environmental factors should be made and included in future approaches.

In this specific task we observed experimentally that the performance of the system decreases when the frame-rate goes below 15 fps. In order to increase the reliability and robustness of the system for future developments a more comprehensive study about high performance visual servoing will be carried out.

Future work includes an extensive set of test of the vision approach and its integration with the flight control. The goal is to unify the approach in order to make it common to different helicopter platform. Additionally we plan to tune the gains of the flight control algorithm in the COLIBRI helicopter to improve the performance. The next step is to apply

this approach to the inspection of power lines in a real scenario.

ACKNOWLEDGMENT

This work is sponsored in part by NSF grants IIS-0133947, CNS-0325875, CNS-0331481, the caltech/JPL PRDF program and by the Spanish Science and Technology Ministry under the project CICYT DPI2004-06624. The authors would like to thank Alan Butler and Jorge Leon for support with flight trials. We also thank Capt. Larry Collins and Capt. Brian Lefave (Urban Search and Rescue Task Force 103), Los Angeles Fire Department for access to the test site.

REFERENCES

[Amidi, 1996] Amidi, O. (1996). *An Autonomous Vision-Guided Helicopter*. PhD thesis, Robotics Institute, Carnegie Mellon University.

[Amidi et al., 1998] Amidi, O., Kanade, T., and Fujita, K. (1998). A visual odometer for autonomous helicopter fight. In *Proceedings of the Fifth International Conference on Intelligent Autonomous Systems (IAS-5)*.

[AVATAR, 2005] AVATAR (2005). The Autonomous Flying Vehicle Project http://www-robotics.usc.edu/˜avatar.

[Bosse, 1997] Bosse, M. (1997). A vision augmented navigation system for an autonomous helicopter. Master's thesis, Boston University.

[Bouguet Jean Yves, 1999] Bouguet Jean Yves (1999). Pyramidal implementation of the lucas-kanade feature tracker. Technical report, Microsoft Research Labs.

[Campoy et al., 2000] Campoy, P., Garcia, P. J., Barrientos, A., Cerro, J., Aguirre, I., Roa, A., Garcia, R., and Munoz, J. M. (2000). An autonomus helicopter guided by computer vision for overhead power cable inspection. In *5th International Conference in Live Maintenance*, Madrid, Spain.

[COLIBRI, 2005] COLIBRI (2005). The colibri project http://www.disam.upm.es/colibri.

[Conway, 1995] Conway, A. R. (1995). *Autonomous Control of an Unstable Helicopter Using Carrier Phase GPS Only*. PhD thesis, Stanford University.

[DARO, 1998] DARO (1998). Unmanned aerial vehicles annual report. Technical report, Defense Airborne Reconnaissance Office (DARO), Pentagon, Washington DC.

[Douglas and Peucker, 1973] Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122.

[Garcia-Pardo et al., 2001] Garcia-Pardo, P. J., S.Sukhatme, G., and Montgomery, J. F. (2001). Towards vision-based safe landing for an autonomous helicopter. *Robotics and Autonomous Systems*, 38(1):19–29.

[Gavrilets et al., 2002] Gavrilets, V., Martinos, I., Mettler, B., and Feron, E. (2002). Control logic for automated aerobatic flight of miniature helicopter. In *AIAA Guidance, Navigation and Control Conference*, number AIAA 2002-4834, Monterey, CA, USA.

[Haralick and Shapiro, 1992] Haralick, R. M. and Shapiro, L. G. (1992). *Computer and Robot Vision*, volume 2. Addison-Wesley.

[Hrabar and Sukhatme, 2003] Hrabar, S. and Sukhatme, G. S. (2003). Omnidirectional vision for an autonomous helicopter. In *IEEE International Conference on Robotics and Automation*, pages 558–563, Taiwan.

[Johnson, 1980] Johnson, W. (1980). *Helicopter Theory*. Princeton University Press, Princeton, New Jersey.

[Jun et al., 1999] Jun, M., Roumeliotis, S. I., and Sukhatme, G. S. (1999). State estimation of an autonomous flying helicopter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1346–1353, Kyongju, Korea.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Trans ASME*, 82(D):35–45.

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th IJCAI*, pages 674–679, Vancouver, Canada.

[Mejias et al., 2005] Mejias, L., Saripalli, S., Sukhatme, G., and Campoy, P. (2005). Detection and tracking of external features in a urban environment using an autonomous helicopter. In *IEEE International Conference on Robotics and Automation*, pages 3983–3988, Barcelona, Spain.

[Merz et al., 2004] Merz, T., Duranti, S., and Conte, G. (2004). Autonomous landing of an unmanned helicopter based on vision and inbertial sensing. In *International Symposium on Experimental Robotics*, Singapore.

[Miller et al., 1997] Miller, R., Mettler, B., and Amidi, O. (1997). Carnegie mellon university's 1997 international aerial robotics competition entry. In *International Aerial Robotics Competition*.

[Montgomery, 1999] Montgomery, J. F. (1999). *Learning Helicopter Control Through 'teaching by showing'*. PhD thesis, University of Southern California, Los Angeles, CA.

[Montgomery, 2000] Montgomery, J. F. (2000). The usc autonomous flying vehicle (afv) project: Year 2000 status. Technical report, Institute for Robotics and Intelligent Systems Technical Report IRIS-00-390,.

[OpenCV, 2005] OpenCV (2005). Open source computer vision library http://www.intel.com/research/mrl/research/opencv/.

[Saripalli et al., 2003a] Saripalli, S., Montgomery, J. F., and Sukhatme, G. S. (2003a). Visually-guided landing of an unmanned aerial vehicle. *IEEE Transactions on Robotics and Automation*, 19(3):371–381.

[Saripalli et al., 2003b] Saripalli, S., Roberts, J. M., Corke, P. I., Buskey, G., and Sukhatme, G. S. (2003b). A tale of two helicopters. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 805–810, Las Vegas, USA.

[Shakernia et al., 1999] Shakernia, O., Ma, Y., Koo, T. J., and Sastry, S. S. (1999). Landing an unmanned air vehicle:vision based motion estimation and non-linear control. In *Asian Journal of Control*, volume 1, pages 128–145.

[Shim et al., 1998] Shim, H., Koo, T. J., Hoffmann, F., and Sastry, S. (1998). A comprehensive study of control design for an autonomus helicopter. In *Proceedings of IEEE Conference on Desicion and Control*, volume 4, pages 3653–3658, Florida, USA.

[Tomasi and Kanade, 1993] Tomasi, C. and Kanade, T. (1993). Shape and motion from image streams - a factorization method. *Proc. of the National Academy of Sciences of the United States of America*, 90(21):9795–9802.

[Vidal et al., 2002] Vidal, R., Shakernia, O., Kim, H. J., Shim, D., and Sastry, S. (2002). Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669.

[Wu et al., 2005] Wu, A. D., Johnson, E. N., and Proctor, A. A. (2005). Vision-aided inertial navigation for flight control. *AIAA Journal of Aerospace Computing, Information and Communication*, 2:348–360.