Detection and Tracking of External Features in an Urban Environment Using an Autonomous Helicopter

Srikanth Saripalli and Gaurav S. Sukhatme
Robotic Embedded Systems Laboratory
University of Southern California, USA
srik,gaurav@robotics.usc.edu

Luis O. Mejías and Pascual Campoy Cervera
DISAM-Universidad Politécnica de Madrid
Madrid, Spain
lmejias,campoy@estii.upm.es

Abstract—We present the design and implementation of a real-time vision-based approach to detect and track features in a structured environment using an autonomous helicopter. Using vision as a sensor enables the helicopter to track features in an urban environment. We use vision for feature detection and a combination of vision and GPS for navigation and tracking. The vision algorithm sends high level velocity commands to the helicopter controller which is then able to command the helicopter to track them. We present results obtained from flight trials that demonstrate our algorithms for detection and tracking are applicable in real world scenarios by applying them to the task of tracking rectangular features in structured environments.

Index Terms—Autonomous aerial vehicles, feature tracking, Kalman filter

I. INTRODUCTION

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 [1]. This ability can be extended to urban environments where vision can be used for navigation and obstacle avoidance [2]. 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 are very well understood. 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) makes the detection and tracking task suitable for vision-based state estimation and control.

We combine vision with low-level control to achieve precise autonomous vision-based feature tracking for an unmanned model helicopter. The vision-based system described here acts as an overall controller sending navigational commands to a low-level controller which is responsible for robust autonomous flight and tracking. The result is an algorithm for vision-based tracking of features using an autonomous helicopter in a structured 3D environment.

In our experiments the helicopter (Figure 1) is initialized in hover at an arbitrary location. It is required to autonomously locate and recognize features which make up a rectangular window, align with it and track it in successive frames.

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

In the following section the vision sub-system and image processing algorithms are described. Section 3 provides a description of the window recognition and tracking approach. Section 4 describes the control approach used by the helicopter. Experimental results and future work are presented thereafter.

II. RELATED WORK

Our current work aims to bridge two communities by integrating visual feature detection and tracking with autonomous aerial vehicle control. Autonomous aerial vehicles have been an active area of research for several years. Autonomous model helicopters have been used as testbeds to investigate problems ranging from control, navigation, path planning to object tracking and following. Several teams from MIT, Stanford, Berkeley and USC have had an ongoing AFV project for the past decade. The reader is referred to [3] for a good overview of the various types of vehicles and the
algorithms used for their control. Recent work has included autonomous landing [4], [5] aggressive maneuvering [6] and pursuit-evasion games [7].

Most of the previous work in the detection or tracking of object of interest in the scene can be classified into three categories, i.e. Correlation-based tracking, Optical flow-based tracking and Feature-based tracking. Feature-based techniques [8], [9] that detect motion or use motion to extract features of the target, rely on the constraint that the movement of the target is small between successive images allowing the object to be tracked using its spatio-temporal continuity. Constraining the movement to ensure overlap between two successive images puts restrictions on speed of the object.

Gilbert et. al. [10] proposed statistical classification-based approach for moving object detection. The optical flow-based tracking [11] schemes are capable of giving accurate location of moving object but these schemes are computationally expensive and are ideal when combined with a tracking algorithm based on phase difference. These techniques are computationally efficient and are ideal when combined with a tracking algorithm for finding structured known features in an environment.

III. THE TESTBED AND THE EXPERIMENTAL TASK

Our experimental testbed, the AVATAR (Autonomous Vehicle for Aerial Tracking And Reconnaissance) [13] 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 a three single-axis gyroscopes provide 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 carried via 2.4 GHz wireless Ethernet. Autonomous flight is achieved using a behavior-based control architecture [4].

IV. VISUAL PREPROCESSING

Our vision system consist of two main tasks running in a client-server architecture. The server task, 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.

In order to improve the performance of the system, the entire image is not processed. In the 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 into the category of window-based tracking [14] 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 image processing algorithm is described below in three parts; thresholding, segmentation and square finding.

A. Segmentation and Thresholding

The purpose of this stage is to mainly extract the color that characterizes the object of interest. Such a segmented image is then converted to greyscale by thresholding. In the greyscale image template matching is performed for feature recognition. The equation used to convert a color image to greyscale image is given by [15]

\[
Y = 0.21267 \times R + 0.715160 \times G + 0.072169 \times B \quad (1)
\]

where R,G,B are the red, green and blue image channels.

The formulation for threshold based segmentation is as follows:

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

\[
\begin{align*}
\text{for } i & \leftarrow N1 \text{ to } N2 \\
Y_i &= 0.21267 \times I(i)_r + 0.715160 \times I(i)_g + 0.072169 \times I(i)_b \\
&\quad \text{if } Y_{low} < Y_i < Y_{up} \\
&\quad \{ I_i = C_f \\
&\quad \text{else} \\
&\quad \{ I_i = C_b \\
\end{align*}
\]

where \( N1 \) and \( N2 \) are the limits of the local search area \( C_f \) and \( C_b \) the values for foreground and background intensities. \( Y_{low} \) and \( Y_{up} \) are the lower and upper thresholds; usually, \( Y_{low} = 0.7 \times Y_c \) and \( Y_{up} = Y_c \). \( Y_c \) is the grey scale projection of the target color.

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.

B. Square finding

The next step identifies those geometric features that are candidates for features to be tracked, in our case a window. This algorithm takes a binary image, finds the contours and approximates it using the Douglas-Peucker algorithm [16]. A square is extracted using the convexity and angle between the vectors of the approximated contour. This algorithm is used for the first few frames for detection of the window. Once a window is detected, the matching algorithm (described in the following section) 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 fast forward flight.

V. OBJECT RECOGNITION AND TRACKING

A. 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. Template matching occurs 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:

\[
\zeta = \frac{\sum_{\hat{x}}^{w-1} \sum_{\hat{y}}^{h-1} T(\hat{x}, \hat{y})I(x + \hat{x}, y + \hat{y})}{\sqrt{\sum_{\hat{x}}^{w-1} \sum_{\hat{y}}^{h-1} T^2(\hat{x}, \hat{y}) \sum_{x}^{W} \sum_{y}^{H} I^2(x + \hat{x}, y + \hat{y})}}
\]  

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.

B. Window Tracking

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. Based on a second order kinematic model for the tracked object we model the equation of the target as a linear system described by:

\[
x_{k+1} = Ax_k + Bu_k + w_k
\]  

where \( w_k \) is the random process noise and the subscripts 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:

\[
Z_k = H_k X_k + u_k
\]  

where \( H_k \) is known and \( u_k \) is random measurement noise. A second order Kalman filter is used for tracking the target. The filter is formulated as follows. 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 \( R \). The Kalman filter operates as follows:

\[
\hat{X}_k = A \hat{X}_{k-1} + Bu_k + K(Z_k - H \hat{X}_{k-1})
\]

\[
K = P_k H^T (R + H P_k H^T)^{-1}
\]

\[
P_k = (I - K H) P_{k-1}
\]
matrix $R$, and that it is not correlated with the process noise. The system dynamics are given by

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

where $a_{k-1}$ is a random time-varying acceleration and $T$ is the time between the steps $k$ and $k-1$. The state propagation and update equation for the discrete Kalman filter are given by [17]

$$
\begin{align}
\hat{X}_k(-) &= \Phi_{k-1}\hat{X}_{k-1}(-) \\
P_k(-) &= \Phi_{k-1}P_{k-1}(+)\Phi_{k-1}^T + Q_{k-1} \\
S_k &= H_kP_k(-)H_k^T + R \\
K_k &= P_k(-)H_k^TS_k^{-1} \\
P_k(+) &= (I_n - K_kH_k)P_k(-) \\
\hat{X}_k(+) &= \hat{X}_k(-) + K_k(Z_k - H_k\hat{X}_k(-))
\end{align}
$$

In the above equations, the superscript T indicates matrix transposition. $S$ is the covariance of the innovation, $K$ is the gain matrix, and $P$ is the covariance of the prediction error. Also we distinguish between estimates made before and after the measurements occur; $X_k(-)$ is the state estimate that results from the propagation equations alone (i.e., before the measurements are considered) and $X_k(+)\$ is the corrected state estimate that accounts for measurements. $P_k(-)$ and $P_k(+)\$ are defined similarly.

The output of the filter (i.e., current position of the window in the image plane) is used as an error signal $e$ for controlling the velocity of the helicopter. Figure 3 taken during helicopter flight is a representative picture of the detection and tracking process. The smaller circle in the picture represents the features selected by the user and the bigger circle represents the output of the tracker.

![Fig. 3. Detection and tracking process](image)

VI. CONTROL ARCHITECTURE

The AVATAR is controlled using a hierarchical behavior-based control architecture. The control architecture used for the AVATAR is shown in Figure 4. The low-level behaviors have been extensively described in previous work [4], we give a brief summary below and focus on the behaviors specific to the vision-based lateral control problem.

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 (Inertial Measurement Unit) to actuate the tail rotor. 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.

The low-level and short-term goal behaviors roll, pitch, heading, altitude and lateral control behaviors are implemented with proportional controllers (The altitude control behavior is implemented as a proportional-plus-integral(PI) controller). For example the roll control behavior reads in the current roll angle from the IMU and outputs a lateral cyclic command to the helicopter.

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 lateral control behavior is further split into two sub-behaviors, hover control, velocity 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 state when the helicopter should move laterally to a desired GPS waypoint. 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.

Once the features has been located the velocity control sub-behavior takes over from the hover control sub-behavior. It is implemented as a PI controller. The integral term is added to reduce the steady state error. The helicopter starts to servo towards the features. The velocity control sub-behavior is shown in Equation 11 where $\tau$ is the lateral 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.

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

3975
VII. EXPERIMENTS

Experimental trials were performed to test the performance of the vision system with combined flight control. These set of flights were performed at Del Valle Urban Search and Rescue training site in Santa Clarita, California.

A. Experimental results

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 takes control aligning the helicopter to the features being tracked. Four flight trials were performed, the results of which are summarized below. Videos and actual data recorded from the flight trials can be downloaded from http://robotics.usc.edu/~srik/icra05/

The sequence of events during autonomous control can be traced 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. 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. 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 5(a) shows the velocity commands generated by the vision algorithm in the image plane correlated with the helicopter position. Note that these signals have been normalized between -1m/s and 1m/s. Figure 5(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 5(c) and (d) show the path taken by the helicopter while tracking the features. Good 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 5 (a) and (c) where a corresponding change in velocity shows a proportional change in position as expected. In figures 5(c) and (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).

VIII. CONCLUSION AND FUTURE WORK

We have presented an algorithm and experimental results for tracking structured features in an environment using an autonomous helicopter. Data from several flight trials show that our algorithm performs acceptably. We currently process the images at 15fps, we believe that with an independent vision processor the system will work at frame rate (30Hz). Furthermore we intend to make the vision algorithm more robust and reliable. Ideally our tracking algorithm would be able to locate features anywhere within the image at any point in time. But limitations like inter frame velocity or helicopter vibrations, background changes, light and luminance affects directly the matching process. Any further effort towards improving the reliability and robustness should be spent in optimizing the matching algorithm. Since this system is intended to work mainly in outdoor environments, we believe a comprehensive study of all the environmental factors like ambient light in the environment, brightness, shadows etc should be considered in the vision algorithm for making it more robust. Future work involves simultaneously controlling height and lateral velocity.

ACKNOWLEDGMENT

This work is supported in part by DARPA under grant DABT63-99-1-0015 as part of the Mobile Autonomous Robot Software (MARS) program. The authors would like to thank Alan Butler (our pilot), 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


Fig. 5. a) Normalized Error signal (m/s) b) Location of features in the image c) Helicopter position in meters (UTM coordinates - Easting) d) Helicopter position in meters (UTM coordinates - Northing)


