

Implementation of an Autonomous Aerial Reconnaissance System

Doug Coleman Drew Creel Doug Drinka
Nicholas Holifield William Mantzel Ali Saidi
Michael Zuffoletti

University of Texas at Austin
IEEE Robot Team Aerial Robotics Project

May 29, 2002

ABSTRACT

The University of Texas at Austin IEEE Robot Team has prepared an entry to the 2002 International Aerial Robotics Competition. The team aims to demonstrate autonomous forward flight and navigation between waypoints in 2002, and additional capabilities in 2003 and 2004. This document covers the design decisions and system design for the team's first year in competition. The basic system architecture consists of a primary aerial vehicle and a multi-mode sub-vehicle that communicate via high bandwidth wireless link with the base station. We describe the hardware and software used to autonomously pilot the helicopter, and the vision system. Special emphasis is placed on off the shelf, cost effective solutions.

INTRODUCTION

The task set for the 2002 International Aerial Robotics Competition dictates that graphical reconnaissance must be returned from within a structure with openings 1 meter wide, following the travel of 3 kilometers of waypoints by a team's primary air vehicle. Due to the difficulty of building an autonomous vehicle able to travel 3 kilometers, fit safely through a 1 meter opening, and navigate inside simulated human dwellings, our team chose a primary air vehicle (XCell .60 Gas Graphite Helicopter) to deliver a multimode sub-vehicle into the structure once an opening has been located. Attempts during the year 2002 will involve the first qualifying task: flying waypoints autonomously along a 3 km path.

Aerial Vehicle

Figure 1. XCell .60 in flight

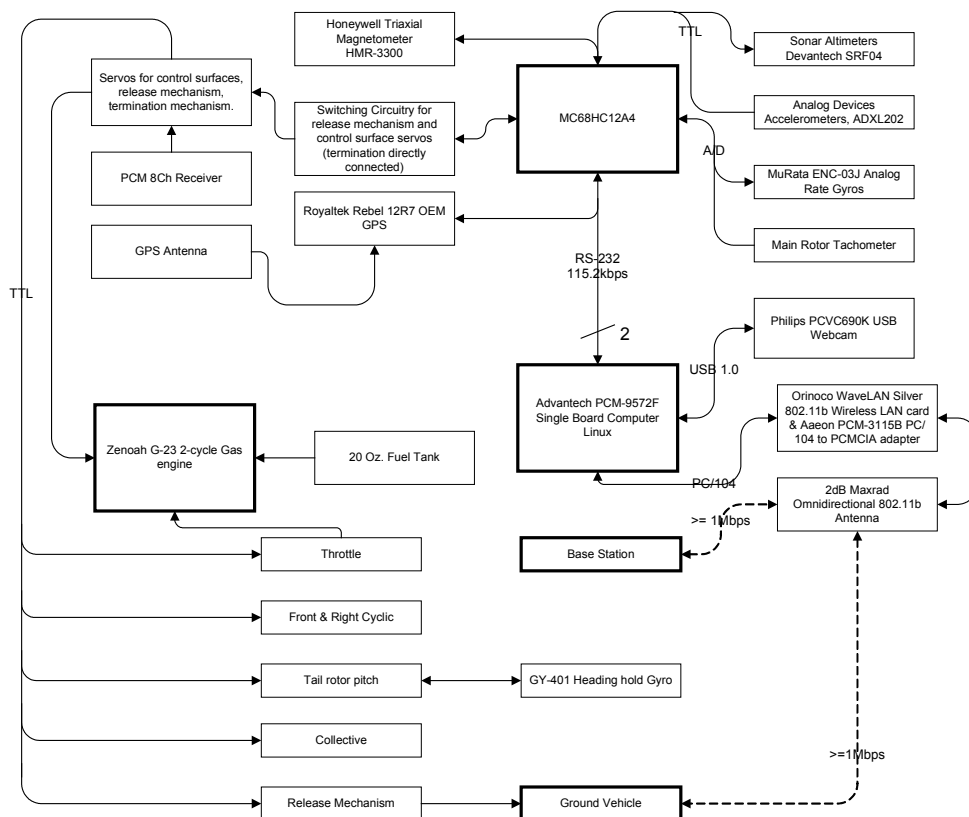


Our team chose an XCell .60 based on recommendations from local hobbyists around Austin, its lifting ability, and its high quality parts. We also contacted helicopter specialist Curtis Youngblood for help with upgrades in order to increase stability and reliability. Specifically, we replaced the horizontal tail fin stabilizer with a larger FAI part, and added a slipper gear in order to control heading orientation during an autorotation. Further upgrades were to replace the plastic main rotor head with an aluminum upgrade, and likewise to replace a plastic collar sleeve on the main shaft with aluminum. These new parts allow for maximum response without the contact points locking up, which is important when the computer calculates only minute adjustments. Our team pilot and Mr. Youngblood flew many unloaded test flights help to break in the helicopter and prepare the gas engine for lifting.

For autonomous flight, the helicopter offers a major advantage over the airplane. With a suitable combination of sensors, the helicopter can move to a GPS waypoint and hover there indefinitely, while an airplane must continue to fly in a pattern around the waypoint and must avoid any obstacles that may be present in an urban setting. Consequently, a helicopter is more apt to complete qualifier two, as it can maintain proximity to the building while locating the windows using sonar and vision.

ONBOARD SYSTEMS

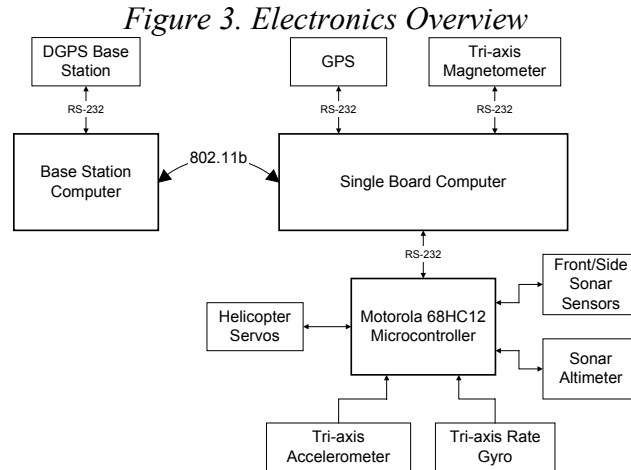
Figure 2. System Overview



Sensor Interfacing

The microcontroller, a Motorola MC68HC812A4, shown in Figure 2 provides the onboard computer's interface to the IMU, sonar, and servos. The IMU is a custom circuit board

with four vertical daughter cards: two accelerometer boards and two gyro boards. It incorporates three orthogonal axes of accelerometers and three orthogonal axes of gyros. The accelerometers update at 110 Hz, and a potentiometer allows for changes in sample rate. The sensors are sampled with interrupts on the microcontroller, which handles the data and passes it through to the onboard computer.



Sonar sensors mounted around the base of the helicopter provide for obstacle avoidance and the mission critical task of detecting open windows. The sonar sensors activate upon receiving a 10 μ s pulse from the microcontroller and send out a 40 KHz sequence of eight bits, and keep the echo pin high until the bit sequence is detected as a reflection. They can detect obstacles from 3 cm out to 3 m. During qualifier two, for which the helicopter must locate an open window, the sonar sensors can ping the windows already recognized by image processing to see if a window is open.

A second type of sonar sensor, a longer-range sonar sensor taken from a Polaroid camera, points down and serves as an altimeter. It is somewhat heavier and more power hungry than an SRF-04 sonar, but it provides a range of 10 m. An accurate height estimate will be essential for takeoff and landing as well as normal flight control.

The microcontroller is interfaced to the helicopter servos for both actuation of control surfaces and data logging. By using timer interrupt synchronized pulse width modulation, the microcontroller can output a discrete angular position value ranging from 0 to 255 to each of the five servos in a sequential order. An additional capability of the microcontroller-servo interface board is position logging during manual flight. By correlating this data with sensor inputs, we can use the system identification toolbox of MATLAB to find the transfer characteristics of the helicopter's control surfaces.

Onboard Computer

We purchased an EBX-format Pentium III 500 MHz single board computer from Advantech to control all higher-level avionics functions. While it's definitely possible to implement the control software necessary to fly a helicopter on a microcontroller, the competition calls for communication with the base station and transmission of pictures. The SBC interfaces seamlessly with our USB webcam and our 802.11b wireless PCMCIA card. The chassis that houses the SBC is 16.5cm wide, and attaches to our mounting platform, which extends out from the sides of the helicopter. This previously proven technique requires few

modifications to the aircraft, and has shown itself to be practical in a winning design as well as in our own flight tests. [1]

Onboard Computer Interfaces

The computer interfaces to the GPS, the tri-axial magnetometer, and the microcontroller via RS-232 and the webcam via USB using a multithreaded program written in C and compiled with gcc in Linux. The flight control program captures data from the sensors and performs the Kalman filtering and PID control as well.

GPS

The Royaltek Rebel 12R7 GPS is our chosen solution, providing 3 m accuracy in practice without a differential beacon. GPS accuracy for the current competition task set is not of the utmost importance, as precedent indicates that a wide swath as far as 140 feet from a waypoint still counts as visiting a given waypoint. Inertial, sonar, and imaging sensors can be used for qualifiers two and three, as no GPS information regarding the structure is available anyway. The 12R7 is low power, lightweight, and inexpensive – the OEM unit and an antenna together were \$160.

Wireless System

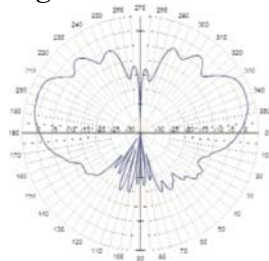
The constraints on the components for our wireless link were numerous. The airborne segment of the system needed to weigh less than 1 kg, be small enough to easily fit on the helicopter, and use only a few watts of power. To accomplish this we chose to use commercially available 802.11b parts. These systems tend to be lighter than their packet radio counterparts and have a much higher bandwidth, which is excellent for sending back pictures from the sub-vehicle.

The downside of 802.11b is the limited range of the high (2.4 GHz) frequency used. Two solutions were proposed to extend the range of the communications to the required 4 km (a worst case 3 km arena plus a 1 km safety margin). The first proposed system consisted of an omni-directional antenna on the helicopter and relied on the base station to track the helicopter with a parabolic antenna. This would provide the 20 dB of gain required to traverse 4 km, but it involves constructing a highly complex electromechanical tracking system.

We decided to use omni-directional low gain antennas both at the base station and on the helicopter. These antennas, while low gain, have a much better vertical beam pattern. This enables the wireless link to stay active regardless of the helicopter's attitude in normal flight. The low gain is augmented with an amplifier manufactured by YDI (YDI AMP2440-I) between the 802.11b card (Orinoco WaveLAN Silver) and the antenna (Maxrad BMAXC24503). The amplifier requires a 12V supply, has type N coaxial connectors, provides 10 dB of transmit gain, and has a 12 dB receive preamp.

The 802.11b card has a Bit Error Rate (BER) threshold of -82 dB for 11 Mb/s. However, we only require minimum link bandwidth, which is 1.0 Mb/s. The BER threshold at that bit rate is -94 dB. The flexible bit rate of the 802.11b card allows for a much more reliable link. [2]

Figure 4 – Vertical Beam Pattern



The Maxrad antenna is a mere 5.25 inches in length, weighs 54 grams, and provides 3 dBi of gain. At left is its vertical beam pattern, indicating a high degree of coverage at moderate angles of elevation. [3]

In order to compute the viability of this setup, we simply plug the relevant numbers into an online calculator such as that offered by Electrocom Distributing, Inc. It estimates that our transmitted power will be 617 mW, which is less than the FCC-allowed 1000 mW, and that given a -94 dB BER threshold our link will be viable and be prone to fail 0.1% of the time. [4]

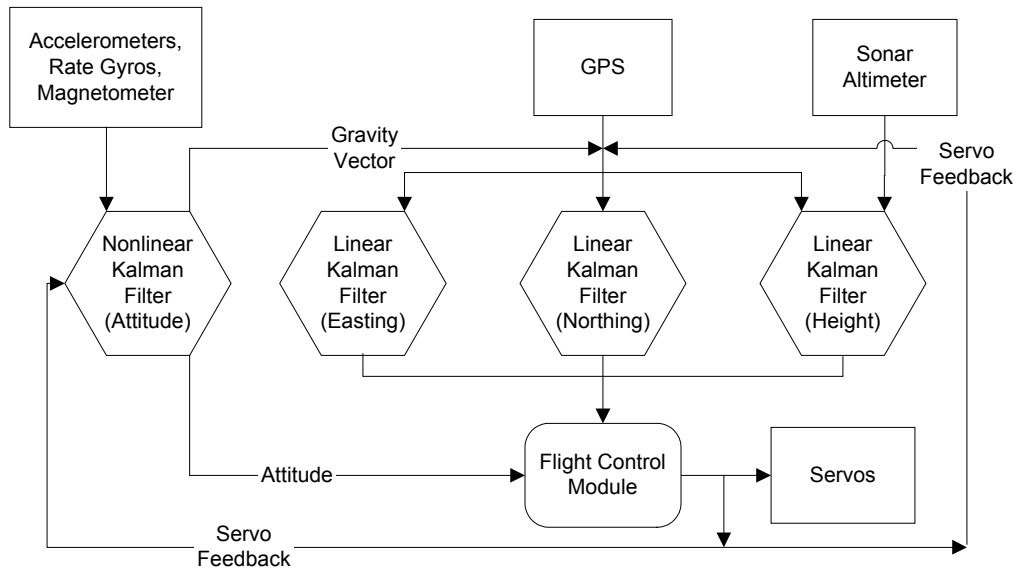
ONBOARD FLIGHT CONTROL SYSTEM

Stability Controller

In order to keep the vehicle stable the flight control algorithm relies on Kalman filtered sensor data as its sole real-world feedback. We decided to use PID controllers to implement our control loops, due to their simplicity and easily tunable parameters. [5]

Kalman Filtering

Figure 5. Kalman Filter Dataflow



Attitude will be obtained with a nonlinear Kalman filter, which handles nonlinear state propagation. After attitude is obtained, the body frame accelerometer readings and sonar altimeter can be transformed to earth frame. Then the gravitational effect can be taken away to obtain a crude acceleration vector. Each axis of acceleration and each axis of GPS position will be fed into one of three Kalman filters for northing, easting, and altitude. Each linear Kalman filter will have the following elements: acceleration bias, velocity, velocity bias, and position. Accelerations will be anticipated with the control term, which is the thrust force resolved into earth coordinates via the previously computed direction cosine matrix. [6][7][8]

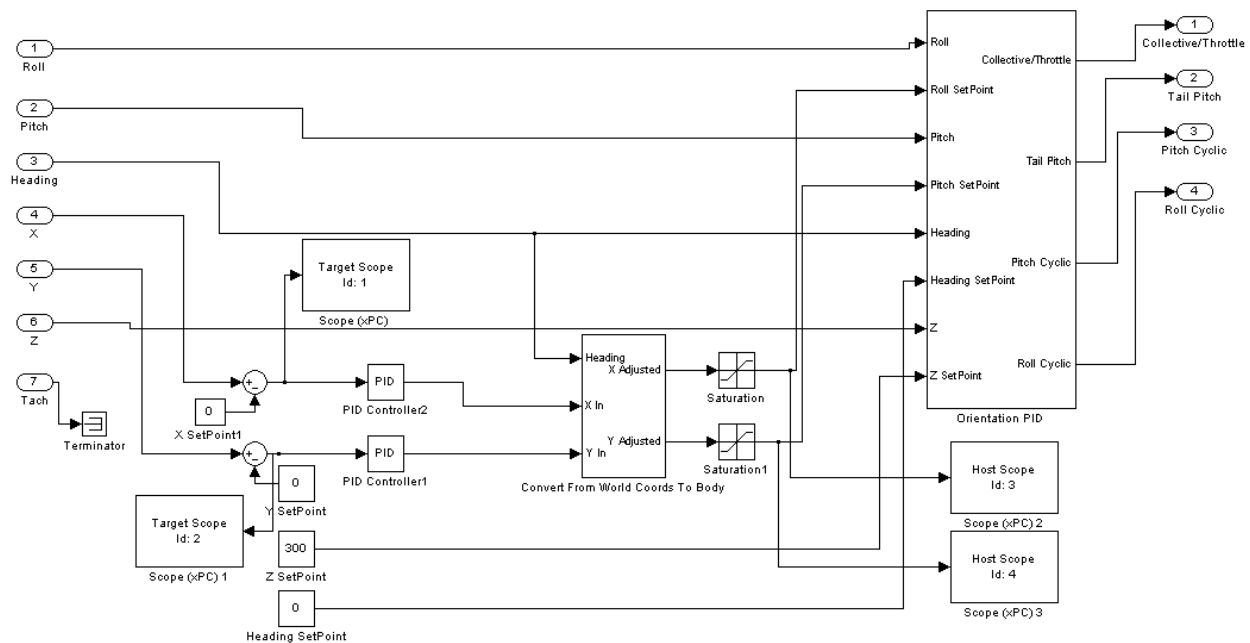
Development

In the final stage of the vehicle's development, the flight control system will exist as a single C program compiled for Linux. This platform will allow the different software components to run as a single multithreaded program with shared memory between systems

already built into the design. However, for development we have chosen to use Simulink (<http://www.mathworks.com/products/simulink>) because it allows visual debugging and rapid code changes.

For simulation purposes, we decided to use the commercial flight simulator RealFlight (<http://www.realflight.com>), a graphical simulator designed for RC hobbyists learning to fly aircraft similar to our air vehicle. It was extended by adding network output and input of simulation data. Our system, shown here as a Simulink model, takes physical data from the simulator and returns control values over the network, effecting PID control loops. This allows us to achieve roughly correct control loops before attempting to fly a physical air vehicle with them.

Figure 6. Simulink PID Loop Development



Simulink has a toolbox, *xPC Target*, which fits a control model and an operating system onto a single 1.44 MB diskette, which is loaded onto a networked computer for simulation. The programmer can include blocks called scopes, as seen in the Figure 5, to give visual output for system variables. Using these scopes, we can see the response of the system in real-time.

While the use of Simulink is purely for testing purposes, it is trivial to translate by hand the algorithms into C. This drastically shortens coding and debugging time because of the clear, correct algorithmic reference provided by Simulink.

Implementation

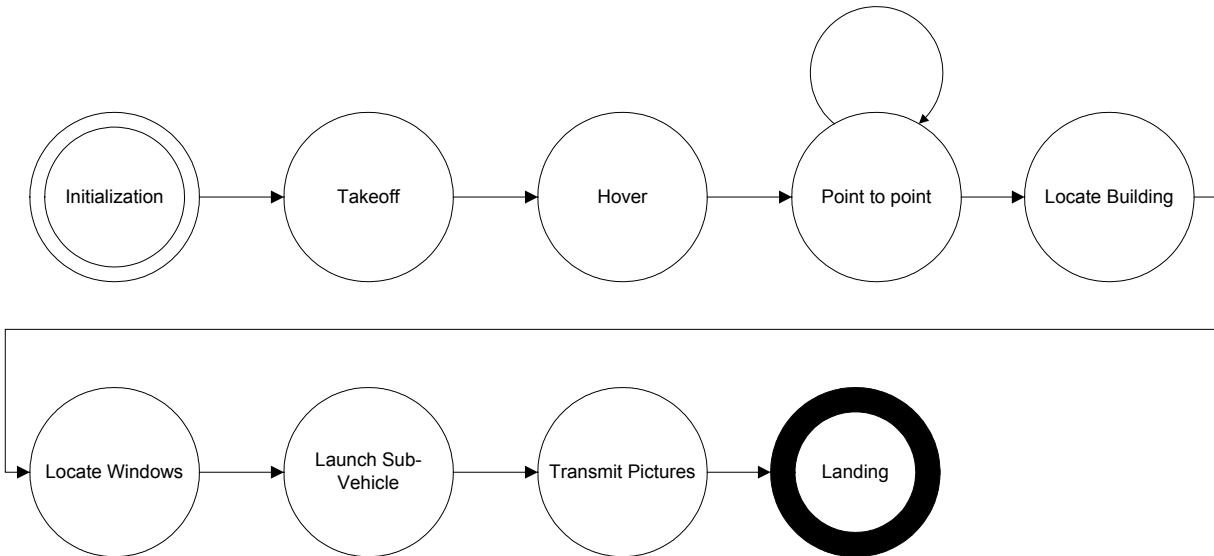
The lowest level of the flight control system drives the helicopter to a desired attitude and altitude. The system uses a PID controller to maintain a commanded attitude and altitude. Simulated results suggest that this relatively simple feedback mechanism is sufficient to achieve an acceptable response.

The next layer accepts latitude, longitude, and altitude as inputs. This layer adjusts the attitude of the vehicle in order to reach the desired point in space by adjusting attitude. Two PID controllers orient the helicopter on map grid coordinates.

High Level Control

The main objective of the flight control system is to accomplish mission objectives. We model the upper level control as a finite state machine, as shown in the figure below. Each state transitions to a sensor error state if at any time the sensors stop responding. In this state, the helicopter sends a notice to the ground station, prompting the human pilot to reassert manual control. If sensors begin functioning again, the control algorithm will return to its previous state and resume execution of the mission, provided manual control has not yet been asserted.

Figure 7. Mission Execution State Diagram

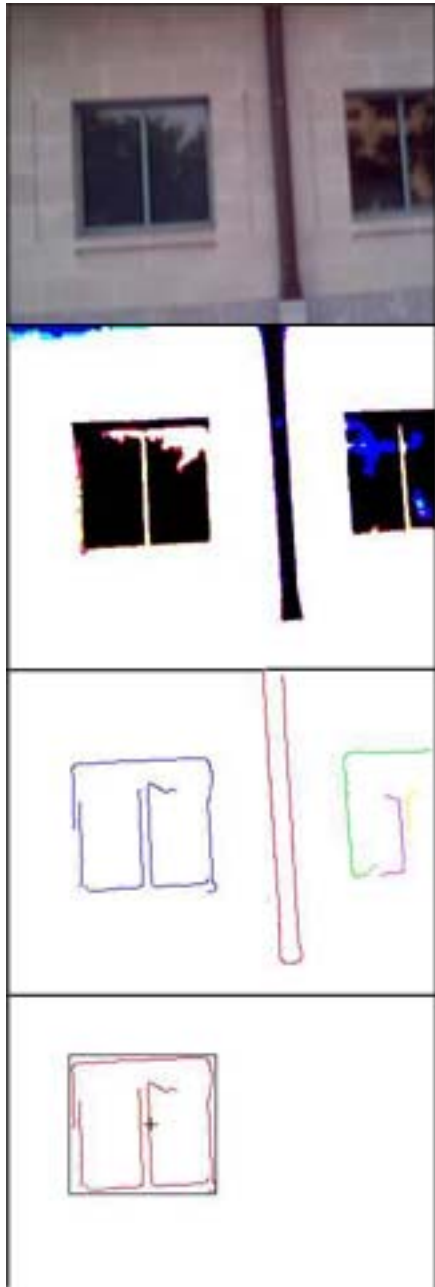


Before the mission, the computer is given a list of waypoints that make up the route. The first task of the autonomous helicopter is to fly through each waypoint in the list. When the computer finds that the waypoint hasn't been reached, it transitions into the "point to point" state. In this state, the computer feeds incremental points along the route to the waypoint into the lower level controller. The computer automatically optimizes the trajectory of the vehicle by moving the goal point as the vehicle nears it. This allows the vehicle to stay near top speed rather than reverting to a near-hover before moving on to the next waypoint.

For qualifiers two and three, in which interaction with a building is necessary, the "locate building" state uses both GPS coordinates and image processing to identify the target building, at which point the walls of the building are scrutinized for a suitable entry point. Once found, the sub-vehicle is launched into the building and the helicopter moves into "transmit picture" state, which entails hovering in a position safely away from the building that allows the best transmission of reconnaissance from the sub-vehicle to the base station.

Computer Vision

Figure 8. From top to bottom: raw image, color hysteresis image, edge detected image, window found



To identify buildings, windows, and the IARC logo, we developed a computer vision scheme based around a Canny edge detector [9], a multi-level color hysteresis loop, and a simple Kalman filter [6]. The algorithm attempts to identify profile shapes with the edge detector and areas of interesting color with hysteresis thresholding. The Kalman filter interprets the output of these two procedures and gives us a reliable probability of the identity of the detected object.

Images are taken from the Phillips PCVC690K USB camera twice per second, at a bit depth of 24 bits (although the practical CCD bit depth is much less than this) and a resolution of 320 by 240 pixels. The raw image, after being converted from the YUV420P color space to the RGB color space, is smoothed with a median filter. This eliminates many “stray” edges found by the edge detector. Next, the filtered image is sent to the Canny edge detector and the colorized hysteresis loop.

The colorized hysteresis loop is a simple hysteresis loop that operates on each color channel of the image. The output is an image containing large patches of color which can be used to confirm the identity of a detected object or for collision avoidance purposes. For example, a large patch of green in the middle of the screen could be interpreted as a tree, and the flight control system may decide to alter the helicopter’s course to avoid the tree.

The Canny edge detector operates on a black and white version of the filtered image and produces an image with single-pixel edges. The Canny detector itself is a two-part process. First, the image is smoothed by Gaussian convolution, producing a gradient magnitude image with strong “ridges” where edges are detected. Then the gradient is eliminated by non-maximal suppression, leaving only the top most points of the ridge and resulting in a single-pixel edge [9].

Once the Canny edge detector produces an image of edges, the edge pixels are grouped into edge sets by a recursive tracking algorithm. This algorithm eliminates some of the smaller gaps in the edges and isolates the edges from each other. Each edge set is converted to a polarized profile image, where the profile contains the angle and distance of

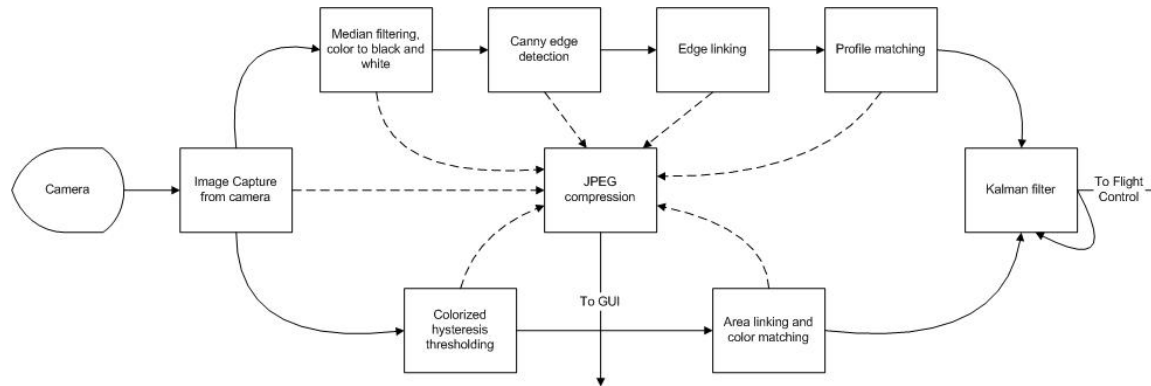
each pixel from the origin. These profiles are then compared against stored profiles for known objects in attempt to match the edges to known shapes.

The matching procedure computes a match percentage for each edge and stored image combination based on the average variance of the edge set profile and the stored image profile. These variances are fed into the Kalman filter, which also uses the information obtained by

hysteresis thresholding to improve image detection capability. The Kalman filter, in addition to combining the two inputs, smoothes the detection output and eliminates discontinuities in multi-frame object recognition [6]. This mitigates the likelihood of identifying an object in one frame and misidentifying it in the next.

Images after any step in the image recognition process can be compressed into JPEG format and sent back to the base station for monitoring purposes.

Figure 9. Computer vision dataflow

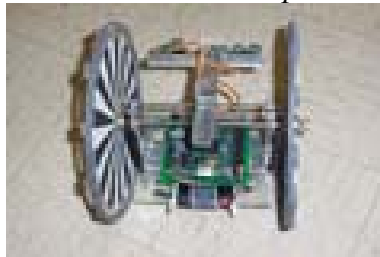


By looking for simple shapes that aren't generally found in nature, such as squares, rectangles, or a large X in a circle, in the case of the IARC logo, we can identify buildings and windows. The flight control software will have a stored profile of the IARC logo that it can use in pattern matching. This will be used to locate the correct building, and then an open window. To locate windows, the flight control software will look for squares and rectangles, and follow up with sonar testing to determine whether they are open or closed.

Sub-Vehicle

This is a summary; for more information on our design efforts for the sub-vehicle, please see the companion paper “Design and Implementation of a Helicopter-delivered Sub-Vehicle.”

Figure 10. Sub-vehicle implementation



The sub-vehicle is a two wheeled design with an onboard x86 PC and microcontroller. Its locomotion consists of DC gear-motors further geared down with pulleys, giving it a top speed somewhat approximate to human walking speed. It has a USB camera identical to that of the air vehicle, and sonar sensors for navigation. It has an integral tube that enables it to be launched using compressed gas from the primary air vehicle. It is anticipated that we will re-

implement this vehicle very differently next year with all our lessons learned, with the goal of competing with it in 2003.

Acknowledgements

We would like to thank:

- Our faculty sponsor, Professor Howard Neal, for all of his help and guidance,
- Curtis Youngblood, for all those helicopter crashes that never happened.
- Our financial sponsors: The UT Austin College of Engineering and Department of Electrical and Computer Engineering, Schlumberger, and Lockheed Martin.

References

[1] M. Musial, U. W. Brandenburg, G Hommel. "MARVIN: Technische Universität Berlin's Flying Robot for the IARC Mellennial Event." <http://pdv.cs.tu-berlin.de/MARVIN/>. 2001.

[2] "Wireless LAN World PC Card." <http://www.agere.com/client/docs/DS02115.pdf>. 2002.

[3] "BMAXC24503 Elevation Cut." http://www.maxrad.com/maxrad_products/broadband/2-6ghz/patterns/mobile_series_24_ghz.pdf. 2002.

[4] "Wireless Network Link Analysis". <http://www.ecommwireless.com/cgi-local/wireless.main.cgi>. 2002.

[5] H. Shim, T. J. Koo, F. Hoffmann, and S. Sastry. "A Comprehensive Study of Control Design for an Autonomous Helicopter." Proceedings of the 37th IEEE Conference on Decision and Control. 1998

[6] Bishop, Gary and Welch, Greg. "An Introduction to the Kalman Filter." http://www.cs.unc.edu/~welch/kalman/kalman_filter/kalman.html. 2002.

[7] Wan, Eric A. and van der Merwe, Rudolph. "The Unscented Kalman Filter for Nonlinear Estimation." <http://cslu.cse.ogi.edu/nse1/ukf/>. 2002

[8] Rogers, Robert M. *Applied Mathematics in Integrated Navigation Systems*. American Institute of Aeronautics and Astronautics. 2000.

[9] Parker, James R. *Algorithms for Image Processing and Computer Vision*. John Wiley and Sons, New York. 1996.