Visual Tracking Technologies for an Autonomous Rotorcraft

David E. Cardoze
Thomas R. Collins
Ronald C. Arkin

Mobile Robot Laboratory, College of Computing
and Georgia Tech Research Institute
Georgia Institute of Technology, Atlanta, Georgia, 30332
cardoze@cc.gatech.edu, arkin@cc.gatech.edu, thomas.collins@gtri.gatech.edu

1 Introduction

Visual tracking of moving objects is especially difficult when the observation platform is a small rotorcraft. The complications include limited size constraints, frame-of-reference drift, vibration, and sudden variations in illumination. Emerging commercial-off-the-shelf (COTS) technologies can assist the tracking process, but integration with custom components is yet another consideration in the development of a complete, robust system.

The Autonomous Scout Rotorcraft Testbed (ASRT) is a technology demonstration program of an unmanned aerial vehicle system developed for the Aviation Applied Technology Directorate of the U. S. Army Aviation and Troop Command. The purpose of this program is to demonstrate autonomous operation of an air vehicle for future applications to a wide variety of military systems. A subsystem of ASRT, the Mission Equipment Package (MEP) is explored here in detail, with an emphasis on the visual tracking capabilities. The MEP consists of the sensor and actuator systems required to implement the mission-oriented aspects of the ASRT system - delivery of a humanitarian payload from the air vehicle to a human on the ground.

In order to accomplish the ASRT mission, the operator at a remotely-located groundstation must be presented with real-time stabilized video. The vehicle would normally be flying within 50 feet of the ground, directed to a search area selected from an onscreen map of the local terrain. After the vehicle reaches the search area, the operator then views the stabilized video and designates a potential target. The system is expected to maintain continuous tracking as the vehicle moves to an appropriate standoff distance from the target and completes the mission. At any time, a new target may be designated, or the vehicle may be redirected to an entirely different search area. The mission was to be demonstrated in an area of rolling terrain with forested areas that effectively constitute obstacles at the given altitude for flight.

Clearly, the visual tracking requirements are intertwined with those of the vehicle as a whole. The development of the ASRT vehicle is thus both a research problem and an integration project. The research questions include not only the development of tracking strategies, but also the acquisition of usable visual imagery, the design of a suitable human-computer interface, the organization of software components within a viable software architecture, and the integration of all hardware and software components.

In order to select the appropriate tracking strategy for this problem, we have to consider the following issues. First, the imagery to be processed will be obtained from a moving platform which induces motion in the image sequence and hence makes target tracking more difficult. Second, the system must operate in natural outdoor environments which makes the tracking process more difficult because of dynamic target obscurants like vegetation and dust. Third, the system must work in both day and night scenarios, and therefore it cannot heavily rely on color and other kinds of information that may not be available in low-light environments.

After considering the above issues, we decided to employ a simple motion-based tracking algorithm together with an off-the-shelf image stabilization system as our main tracking strategy.

The organization of the rest of the paper is as follows. In section 2, we describe previous work in target tracking that is closely related to our objectives within the ASRT project. In section 3, we describe the ASRT project and its different subsystems. We focus on how the Mission Equipment Package relates to them. Next, in section 4, we describe the tracking strategies that we implemented, and in section 5, we describe results obtained prior to the demonstration date. In section 6, we describe the system as it was fielded for the demonstration. Finally, in section 7, we present our conclusions.

2 Related Work

Extensive research has been conducted in the area of target tracking. Instead of attempting to provide an exhaustive reference, only representative work that is closely related to our objectives with the ASRT project is reviewed.

David et al [6] have presented a real-time automatic target acquisition and tracking system in the context of dynamic battlefield environments. Their system, like ours, operates on natural environments which makes the tracking process much more difficult because of dynamic target obscurants like vegetation and dust. Their system uses real-time image processing hardware to track multiple tank-sized targets. They use a combination of best-first-search and probability-based algorithms for target tracking.

Samy [16] discusses different approaches to target tracking for natural textured backgrounds in real-time. He presents an overview of feature extraction and texture measures approaches.

Other researchers, including Gilbert et al. [7], and Hager et al. [9] have developed real-time tracking systems using dedicated and non-dedicated hardware, respectively. Gilbert was interested in the problem of missile tracking, while Hager's system is a tool for rapid prototyping and experimentation in research and education. It supports independent tracking agents that exchange information in a hierarchical network.

In addition, Zheng and Chellapa [18] explore the detection of motion from imagery acquired from moving platforms. They address the difficulties created by the relative motion of the imaging device and present a subpixel accuracy image registration algorithm to compensate for it. Their work estimates camera rotation and translations by matching feature points obtained using Gabor wavelet models. After the images are compensated for camera motion, they track the target using a motion-based approach.

Recently, several other methods have been proposed for the tracking of human motion, in both outdoor and indoor environments. These are especially relevant to this research and will be described in greater detail.

Sato et al. [14] fits an articulated model of the human body to the initial image of a sequence and accomplishes tracking by computing pose increments of the body parts between frames. The computed pose increments result from the solution of systems of linear equations with data obtained from several viewpoints. However, this requires the user to manually fit the model to the initial frame of the tracking sequence. The system is targeted at applications in virtual reality, performance-driven computer animation and human-computer interaction.

Cheng and Moura [3] employ a geometric model of the human body together with kinematic constraints to track human walking. They exploit the fact that walking is a periodic motion to guide the tracking process. They determine posture by matching edge information extracted from the image sequence to information on a model walkers. They define a similarity measure to determine the geometrical similarity between model walkers and the one extracted from the data.

Polana and Nelson [12] present an approach to human motion tracking that does not require identification of human parts and relies on the detection of periodic signals on the motion field between successive frames. Their goal is to recognize and classify motion activities of a designated actor in outdoor environments even in the presence of other moving objects. They address the problem of multiple moving objects by computing estimates of the position of the object of interest.

Pfinder [17] is a system for tracking of the human body developed by Wren et al. that uses statistical models of color and shapes. It assumes the scene to be significantly less dynamic than the user, which does not necessarily hold for outdoor environments. They are mainly interested in applications including wireless interfaces, video databases, low-bandwidth coding, virtual reality interfaces and smart rooms.

Finally, Cai, Mitiche and Aggarwal [2] developed a motion-based system

Figure 1: ASRT Vehicle. Figure courtesy of Mark Gordon.

to track human motion in indoor environments. They exploit the structure of indoor environments to allow for the case were the viewing system undergoes non-negligible motion. Applications for the system include surveillance, image compression, and content-based multimedia storage and retrieval.

3 ASRT Project Overview

The Autonomous Scout Rotorcraft Testbed (ASRT) is an unmanned aerial vehicle capable of autonomously flying to a designated search area, then searching for a moving man-sized target, and finally dropping a humanitarian payload before returning to its starting point [5, 8]. The project was funded by the US Army Aviation Technical Directorate as an Advanced Concept Technology Demonstration. The goal of the overall project is to demonstrate autonomous operation of an unmanned air vehicle asset, enabling unmanned systems to look over the hill in a hostile environment and to provide increased situational awareness for an aerial scouting team. The ASRT project is intended to promote new applications for robust flight control laws, integrated mission execution strategies, application of modeling and simulation, and vehicle robotics. In addition, the program helps define current difficult technical barriers, contributes to future full scale manned concepts with tests using unmanned flights of high risk concepts, and advances the state-of-the-art for future autonomous vehicles.

In the ASRT vehicle (figure 1), autonomous flight control is completely included within the onboard systems, but visual processing and certain high-level control functions are performed at a ground station, with wireless datalinks to maintain communication with the vehicle. This is consistent with the intended use of the vehicle, in which it is normally directed between waypoints planned by a ground-based mission planner. Furthermore, remote operator intervention is a required capability of the system, including redirection of vehicle destination, manual camera pointing, designation (and redesignation) of the deployment target, and manual release of the payload.

The onboard components include the flight control computer, the Mission Equipment Package (MEP), wireless communication devices, a differential GPS receiver, as well as transducers and actuators that are integral to the stability of the vehicle itself. Of greatest interest here is the MEP, which consists of the sensor and actuator systems required to implement the mission-oriented aspects of the ASRT system: delivery of a humanitarian payload to a sensed target.

The MEP is tightly coupled to another subsystem, the Mission Planning Computer (MPC). The MPC provides the operator interface, including a point-and-click map of the local terrain allowing intuitive redirection of the vehicle. Since the map is referenced to absolute geographical coordinates, and since the onboard differential GPS system is accurate to approximately a centimeter, hovering at a designated location is limited only by the flight control characteristics of the system in conjunction with environmental conditions. The interface also includes the optional display of real-time video from the onboard camera, again with a point-and-click capability, but in this case functioning as a target-designation indicator.

The Rumbaugh object paradigm [13] was used to organize, plan, and develop the MEP and MPC subsystems and their interactions. In conjunction with the team-oriented IPPD (Integrated Product and Process Development) design approach, object diagrams provide early "roadmaps" that describe subsystem functionality and partition responsibilities among design teams. As the development progresses, these same diagrams are updated to reflect the exact methods and attributes of each subsystem, as well as the messaging that can occur between them.

4 Tracking Strategies

The fielded system is described later in greater detail, but some of the fundamental architectural decisions affected the tracking strategies. In particular, the incorporation of a separate image stabilization system enabled this aspect to be largely decoupled from the tracking process.

Figure 2 shows how these systems work together within ASRT. Video coming from the air vehicle via the video link is sent as input to the Sensar PV-1 real-time image stabilization computer. Stabilized video is available both to the tracking computer and to the MPC computer, which provides the human-computer interface, including target designation. The output from all video-based systems is displayed and archived as necessary.

The Teleos AVP-100 is a real-time tracking system which has the capability to produce motion measurements within a subimage that is selected under software control. The user can select the subimage size and dynamically control the region of interest. Although initially available as a COTS solution to the tracking problem, the Teleos system was never used as anything more than a backup tracking strategy. Parallel development efforts with the Teleos system and those

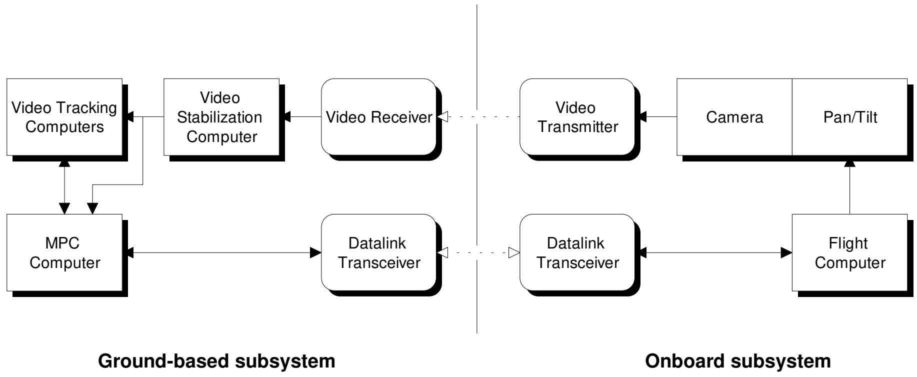

Figure 2: System Architecture

described below resulted in working implementations of each approach, but the Teleos implementation was less reliable for the conditions at the demonstration site.

A motion-based tracking algorithm runs on an SGI Indy workstation. Two different motion algorithms have been developed and tested using imagery that has been stabilized using the PV-1 system at the demonstration test site at Berry College in North Georgia. Both of these algorithms were developed during early stages of the project. The goal of the first method is to track the target in real-time; the second one is concerned with extraction of motion-based objects. Each of these algorithms is presented, along with results, in the following sections.

4.1 Method One: A Real-Time Tracking Algorithm

The goal when developing this algorithm was to achieve real-time target tracking using a non-dedicated workstation. This requirement severely reduces the amount of computation that can be performed between two consecutive frames. In order to meet this constraint, two commitments were made. First, only partial processing of each image frame was conducted (sub-windowing). Second, complex feature extraction or segmentation of the image was avoided.

Using this approach, the operator can dynamically locate the center of the processed sub-window anywhere within the image during the target designation phase. The actual size of the sub-window is dynamically variable and changes according to the amount of motion detected within it. Basically, the algorithm simply computes image differences between two consecutive frames and then tracks the centroid of the difference from frame to frame. The size of the region of interest varies along both spatial dimensions according to the respective variances over the computed image difference.

As customary, the image difference between two frames f, g over a given region is defined as

$$d(i,j) = \mid f(i,j) - g(i,j) \mid$$

where

$$x - \lfloor \frac{n}{2} \rfloor \leq i < x + \lceil \frac{n}{2} \rceil, y - \lfloor \frac{m}{2} \rfloor \leq j < y + \lceil \frac{m}{2} \rceil$$

for an n by m window centered at x, y.

We further define

$$d'(i,j) = \begin{cases} 1, & d(i,j) > \epsilon \\ 0, & otherwise \end{cases}$$

where ϵ is some threshold. This is done to eliminate some of the noise that is always present in images and to obtain a binary image.

Next, we compute the centroid of the window. This is the pair $(\bar{x}, \bar{y})$ where

$$\bar{x} = \frac{1}{nm} \sum_{i=x-\lfloor \frac{n}{2} \rfloor}^{x+\lceil \frac{n}{2} \rceil - 1} \sum_{j=y-\lfloor \frac{m}{2} \rfloor}^{y+\lceil \frac{m}{2} \rceil - 1} d'(i,j) \cdot i, \quad \bar{y} = \frac{1}{nm} \sum_{i=x-\lfloor \frac{n}{2} \rfloor}^{x+\lceil \frac{n}{2} \rceil - 1} \sum_{j=y-\lfloor \frac{m}{2} \rfloor}^{y+\lceil \frac{m}{2} \rceil - 1} d'(i,j) \cdot j.$$

The centroid is interpreted to be an estimate of the center of the moving object, and the center of the window is repositioned to be at $(\bar{x}, \bar{y})$ for the next iteration of the algorithm. Also, the variance along each dimension is computed. These are defined as

$$x_\sigma = \sqrt{\left(\frac{1}{nm} \sum_{i=x-\lfloor \frac{n}{2} \rfloor}^{x+\lceil \frac{n}{2} \rceil - 1} \sum_{j=y-\lfloor \frac{m}{2} \rfloor}^{y+\lceil \frac{m}{2} \rceil - 1} d'(i,j) \cdot i^2 \right) - \bar{x}^2},$$

$$y_\sigma = \sqrt{\left(\frac{1}{nm} \sum_{i=x-\lfloor \frac{n}{2} \rfloor}^{x+\lceil \frac{n}{2} \rceil - 1} \sum_{j=y-\lfloor \frac{m}{2} \rfloor}^{y+\lceil \frac{m}{2} \rceil - 1} d'(i,j) \cdot j^2 \right) - \bar{y}^2}.$$

The size of the window along the x-axis for the next iteration is set to be $1.5\, x_\sigma$, and similarly the size along the y-axis is set to be $1.5\, y_\sigma$. This is done so that the size of the window will increase or decrease according to the amount of motion detected within it. The size of the window along either dimension is restricted to be within a range specified by the user (in our implementation the minimum along either dimension was set to be 10 pixels and the maximum 70). This allows the user to limit the minimum and maximum size of the tracking window.

If the number of non-zero valued pixels inside the window falls below some threshold (10 pixels is the default), the target is declared lost and the window is placed at the center of the image with maximum size along both dimensions.

4.2 Method Two: Extracting Motion-based Objects

As noted earlier, the previous algorithm does not recognize or identify its targets; it only follows movement. We now describe an algorithm developed by Zheng and Chellapa [18] used for extraction of motion-based objects. It starts from the observation that more can be done than just thresholding to reduce noise.

The key point is that if a point in an image is moving, then it is very likely that some or all of its neighbors are moving too. Thus we can discard isolated points in the difference image, if we are willing to ignore very small targets. They also point out that the threshold value should not be a constant. Instead it should vary accordingly to the local contrast of the image. As the contrast increases, the threshold should also increase. Thus they classify a point as a moving point if

$$min\{\overline{d(i,j)}, d(i,j)\} > max\{\gamma D(i,j), \epsilon\}$$

where

$$\overline{d(i,j)} = \frac{1}{N_\omega} \sum_{i,j \in \omega} d(i,j),$$

$$D(i,j) = sup_{i,j \in \omega}\{f(i,j)\} - inf_{i,j \in \omega}\{g(i,j)\},$$

γ is the relative threshold,
ω is the neighborhood of (i,j), and
N_ω is the number of points in ω.

In other words, a given point is penalized for having neighbors with low values in the difference image and for being in a high-contrast area.

This algorithm tries to detect the moving object on each image. Let Ω be the points of the object in a given image. Thus, if we consider three consecutive images f_1, f_2 and f_3, then we could detect $S_1 = \Omega_1 \cup \Omega_2$ by two consecutive applications of the Zheng-Chellapa criteria. There is a problem though, since points that belong to an object tend to have similar intensities, at least locally. In practice S_1 is not detected but rather a subset of it. The subset not usually detectable is the one where the points that happened to move to a location that was previously occupied by a point with the same or very similar intensity value. Thus formally, using the previously described method to detected moving points, the set detected is

$$\bar{S_1} = S_1 - S_1'$$

where the displacement from Ω_1 to Ω_2 is assumed to be (u_1, v_1) and

$$S_1' = \{(i,j) \mid (i,j) \in \Omega_1 \cap \Omega_2 \;\; and \;\; f_1(i,j) = f_1(i-u_1, j-v_1)\}$$

Similarly

$$\bar{S_2} = S_2 - S_2'$$

and

$$S_2' = \{(i,j) \mid (i,j) \in \Omega_2 \cap \Omega_3 \;\; and \;\; f_2(i,j) = f_2(i-u_2, j-v_2)\}$$

where we assume that the displacement from Ω_2 to Ω_3 is (u_2, v_2)

It is easy to see that if $(u_1, v_1) = (u_2, v_2) =$ (u, v) then

$$(i,j) \in S_1' \Longleftrightarrow (i+u, j+v) \in S_2'$$

and that

$$(\bar{x_2}, \bar{y_2}) = (\bar{x_1} + u, \bar{y_1} + v)$$

where $(\bar{x_2}, \bar{y_2}), (\bar{x_1}, \bar{y_1})$ are the centers of mass of f_1 and f_2.

Now that we have all the definitions, we can go back to the algorithm. The first thing to do is to compute $\overline{S_1}$. Next, estimate Ω_1 by taking the closure of $\overline{S_1}$. The closure is constructed recursively from $\overline{S_1}$ by connecting any two points within it that share a common coordinate. Once the closure has been computed, translate f_1 by (u,v) calling it f_1', and compute its difference with f_2. Finally, a point (i,j) in the translated closure is determined to be a point of the moving object in image f_1 if its value in the new difference image is below some threshold, i.e., $\mid f_1'(i,j) - f_2(i,j) \mid \leq \epsilon$ and either it belongs to $\overline{S_1}$ or at least one of its 3x3 neighbors is in Ω_1.

In other words, what this algorithm does is the following. It computes the difference between f_1 and f_2, and attempts to filter out as much noise as possible. Then it computes the closure of this difference to try to make up for the holes due to similar gray level of moving points. Next it computes the difference between f_2 and f_3, to estimate the displacement from f_1 to f_2. Then it shifts f_1 by that displacement, and finally looks for points in the closure that yield zeros in the new difference image.

The assumptions that this algorithms make about the object, more precisely the motion of its center of mass and the relative position among points on it, suggest that this algorithm should work reasonably well with rigid bodies, but if the object being tracked is non-rigid, then the assumptions made do not hold any longer. The assumption that the displacement from Ω_1 to Ω_2 is (u_1, v_1) and ${S_1}' = \{(i,j) \mid (i,j) \in \Omega_1 \cap \Omega_2 \ \ and \ \ f_1(i,j) = f_1(i - u_1, j - v_1)\}$ is equivalent to saying that we are dealing with one rigid body in purely translational motion, or with more than one rigid body all with the same translational motion. This algorithm was implemented to determine its performance when the object being tracked is a human walking (a deformable, non-rigid object), and thus its assumptions do not strictly hold.

5 Results

Both algorithms have been run on video imagery of a human walking in outdoors environments, including the demonstration site chosen for the ASRT Project. The video was stabilized using the SENSAR system before being fed to the algorithms which were run on a non-dedicated SGI Indy Workstation. The results obtained are presented below.

5.1 Method One

This real-time algorithm worked well as long as the SENSAR system could provide stabilized imagery without introducing considerable translations from one frame to the next (i.e., saccadic motion). In addition, the algorithm worked best when the area occupied by the target was neither too large nor too small.

Figure 3: Sequence obtained using real-time tracking algorithm.
This sequence illustrates the performance of the algorithm against a relatively static background. It proceeds in time from right to left and from top to bottom. The detected target is shown within the box.

If the area was too big, about 1/3 or more of the pixels on each dimension, the high frequency noise introduced by the moving grass and leaves caused the algorithm to lose the target. On the other hand, the algorithm could not extract the target from the background if its size was too small. This occurred when the target size was less than about 1/10 of the total number of pixels on each dimension. Figures 3 and 4 show typical results obtained using this algorithm.

5.2 Method Two

The current implementation of this motion-based object tracking algorithm on the SGI Indy workstation, does not run in real-time. The user can select the position to start tracking from within the image, and then the algorithm collects a sequence of frames (six in our implementation), and then processes them. The processing is further limited to a subwindow within the image (30 pixels along

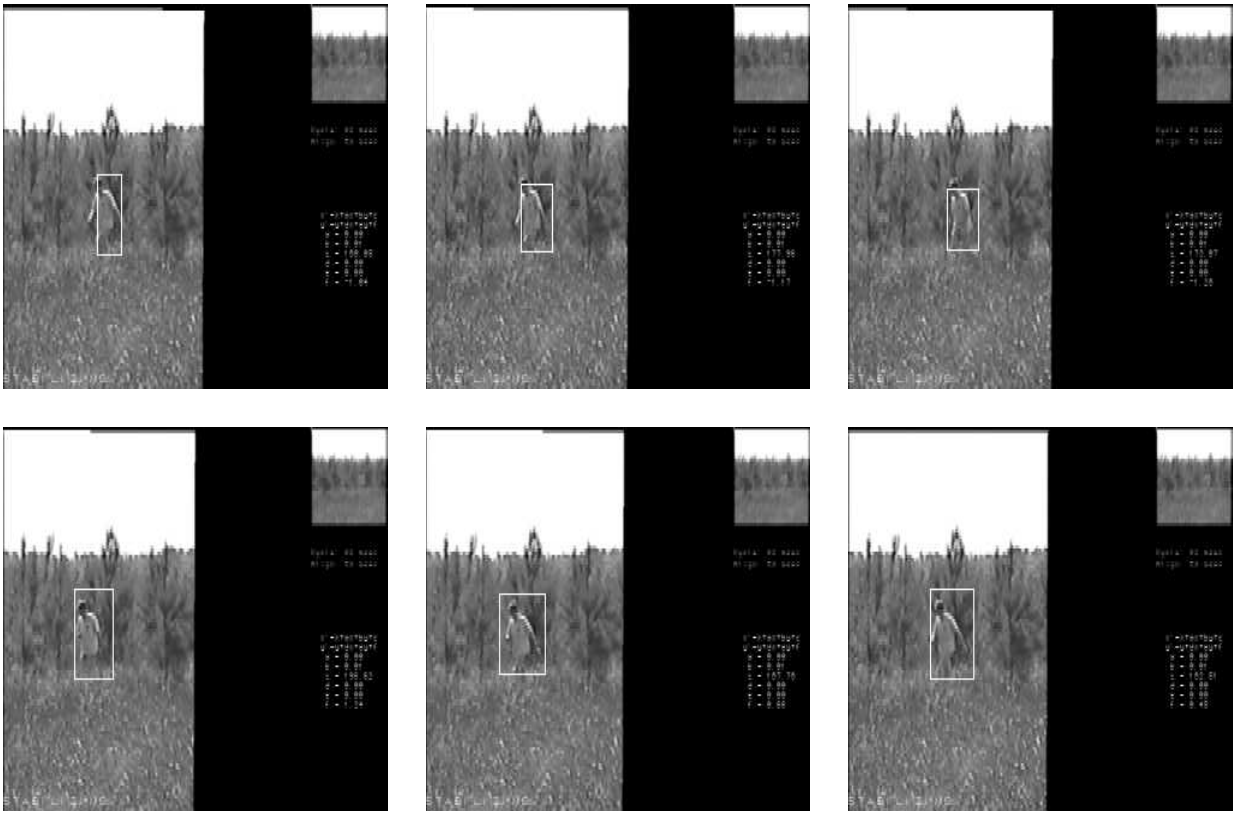

Figure 4: Another sequence obtained using real-time tracking algorithm. This sequence taken at the Berry College demonstration site illustrates the algorithm's performance against obscurants. The sequence proceeds in time from right to left and from top to bottom. Notice how the size of the interest area defined by the white rectangle changes according to the size of the target in the frame.

both dimensions). In these experiments, it was observed that the algorithm tends to detect parts of the target, even though sometimes it also selects sections of the environment. This behavior, however, can be easily understood from the description of the algorithm, since it assumes that there is only a single moving object. Figures 5 and 6 show typical results obtained using this algorithm.

Both methods were tested on the same set of 30 image sequences taken from video samples from the demonstration site. Each sequence consisted of 6 frames. For the first method, the number of frames where the algorithm kept all or more than half of the target within the center of the processed sub-window were counted as tracked successfully. The algorithm tracked 175 of the 180 frames. The fact the the sequences consisted of only 6 frames each may have had an influence on this result, since it may take more frames for the effect of

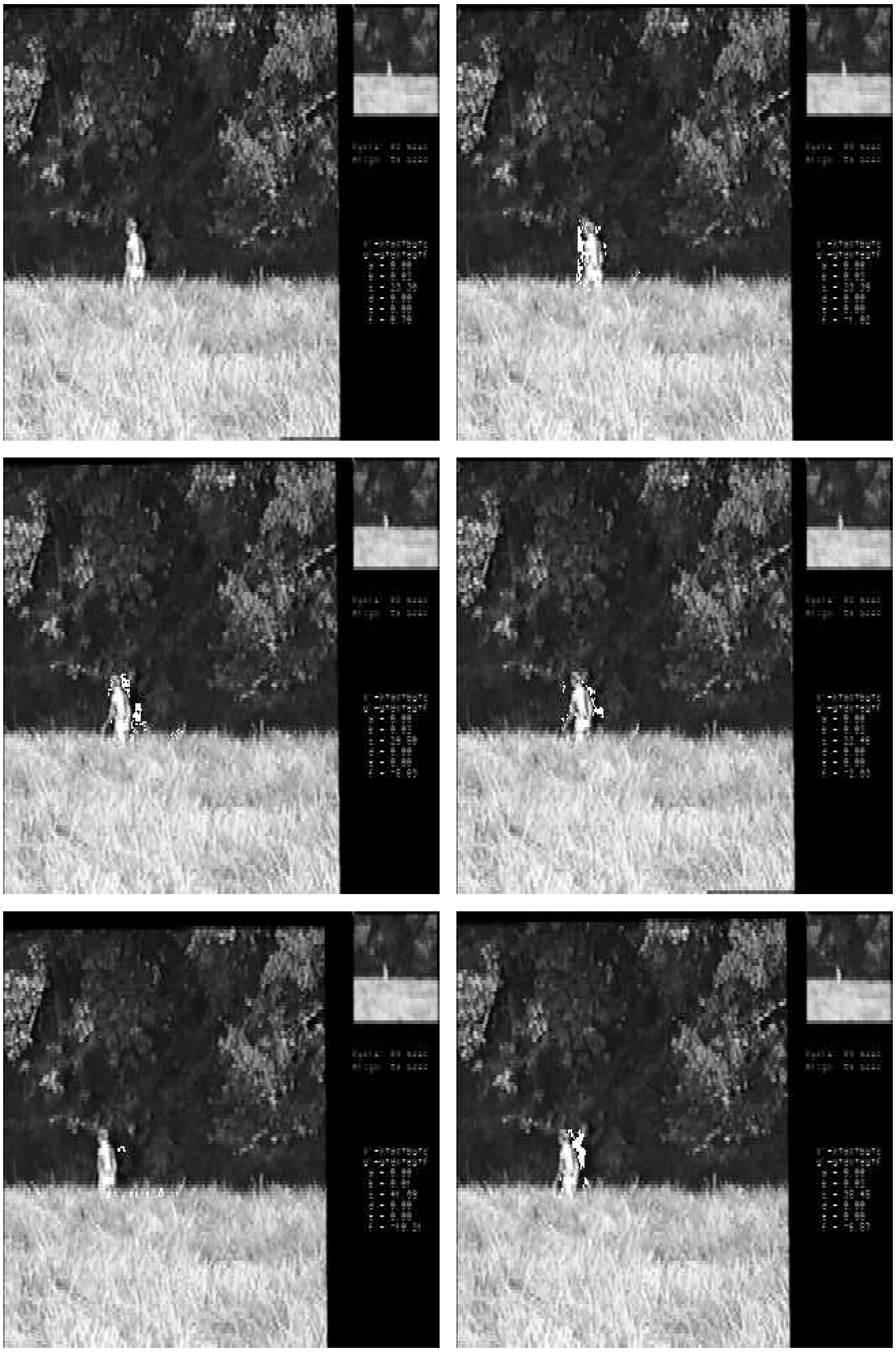

Figure 5: Sequence obtained using Zheng-Chellapa algorithm.
This sequence, also taken at the demo site, proceeds in time from top to right to left and from top to bottom. Notice how features (moving points highlighted as white areas) are not extracted sometimes due to high similarity between frames. Occasionally features from the background get extracted as well.

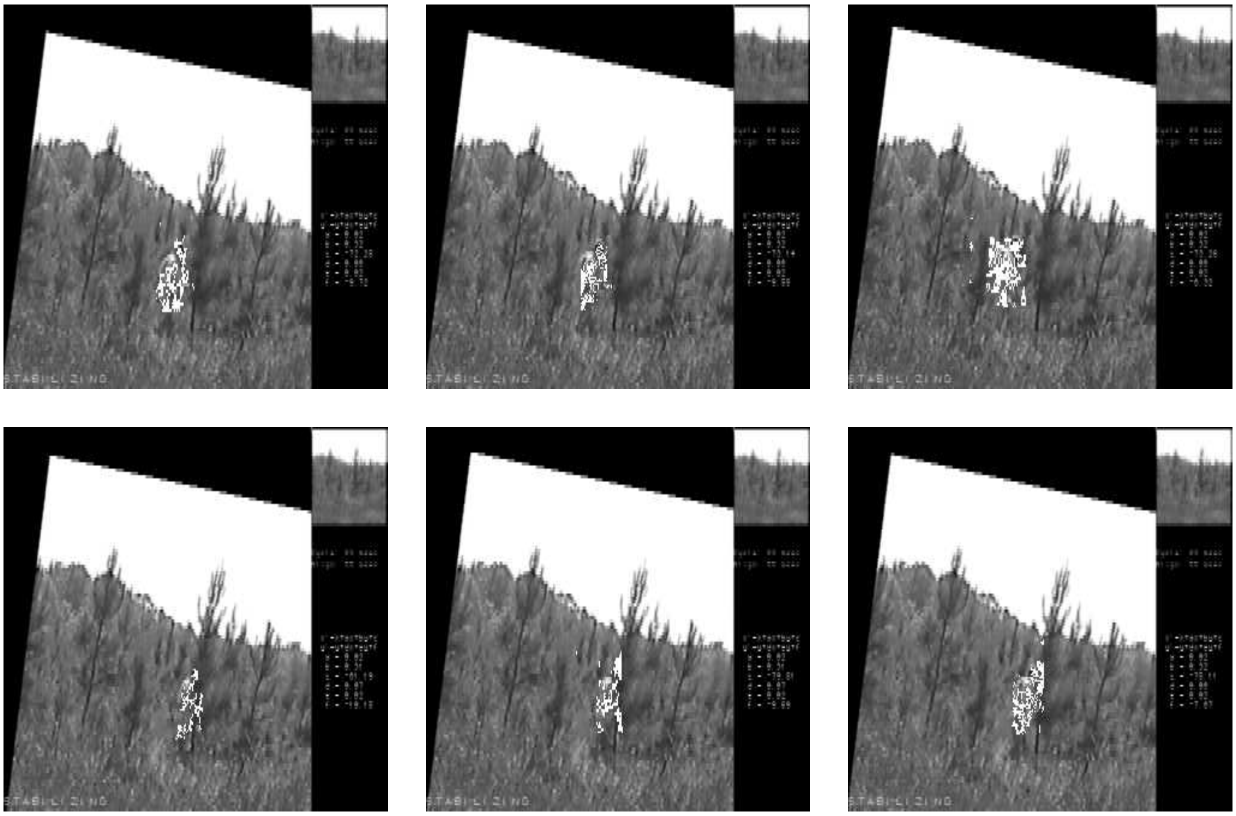

Figure 6: Another sequence obtained using Zheng-Chellapa algorithm.
This sequence, also from the demo site, illustrates the performance of our implementation against obscurants. Notice how regions of the background environment are selected by the algorithm due to their constant motion in the wind.

the background motion to affect the algorithm. Also, only image sequences where there were no abrupt discontinuities in camera motion were used.
For the second algorithm, 170 frames (of the 180 used) extracted features that corresponded to the target, but 84 frames also extracted extraneous features from the environment. This last result is to be expected since the algorithm cannot differentiate target motion from background motion.

6 Fielded UAV System

The project goals required the actual implementation of the tracking strategies in a hardware configuration that could be transported to remote sites. This included not only the vehicle itself, but all ground support hardware as well.

Although several sites were used during the development and test process, the site chosen for final test and demonstration was at Berry College in Rome, Georgia, about 70 miles northwest of Atlanta. Part of this site was recently-cultivated farmland used by the agricultural program of the college, but it also included a large river which served as a convenient geographical boundary, as well as the requisite amount of dense forest and varied terrain.

Another significant factor was the harsh operating environment for any equipment that would be carried onboard the vehicle. Not only are high vibration levels a normal phenomenon associated with rotorcraft, but they also can be especially severe during the course of development of an experimental vehicle. Payload limitations ensure that power supplies are minimal and power regulation is only as good as the most demanding component requires. This section describes the approach taken to developing the fielded system and the results that were obtained.

6.1 MEP Hardware

A major design consideration for ASRT was using COTS (commercial off-the-shelf) technology wherever possible, to demonstrate that most technology elements were already available. This, of course, did not preclude the in-house development of key components, especially where COTS technology was absent, expensive, unreliable, or otherwise undesirable. For ASRT, this applied to flight control software, and (as has been described) partly to tracking.

The MEP (Figure 2) has been divided into two distinct subsystems because of the physical placement of the various components. One of the subsystems includes the onboard components, which acquire data and relay it across wireless links. The other subsystem is part of the ground station, where most of the signal-processing and decision-making functions are performed.

The onboard subsystem consists of interchangeable cameras (one for daylight operation and one for low-light operation) and a pan-tilt platform for positioning. Because of severely limited payload capabilities of the small vehicle, the pan-tilt mechanism was a minimal implementation with poor dynamic response characteristics and with no provision for mechanical image stabilization. Pan-tilt commands are relayed through the flight-control computer, which has total control over the wireless datalink to the ground station.

The ground portion of the MEP consists of the wireless video receiver, the video stabilization computer, and the visual tracking computers, which are closely associated with the MPC computer and the wireless datalink.

The main distinction between the ASRT visual tracking system and other tracking applications is the harsh operating environment of a small rotorcraft. The mission requirements of hovering while searching or tracking a target precluded the option of a fixed-wing aircraft. Vibration is significant on any rotorcraft, and on small airframes it can be severe. An example is shown in Figure 7. This is a normal NTSC video image, with interlaced frames at 30 frames per second. Note the disparity between the odd and even lines in the image, due to vibration during the elapsed time between subframes. Apart from this

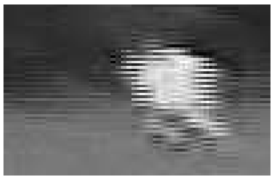

Figure 7: Enlarged region of typical image acquired during flight. Vehicle vibration is manifested as a noticeable horizontal shift between successive lines of interlaced video.

blurring that occurs within a single frame, there is the additional difficulty in recognizing image features from one frame to the next and properly correlating them.

The ASRT ground station consisted of the remainder of the MEP components, as well as the MPC, as illustrated in greater detail in Figure 8. Since it was impossible to include electronic or mechanical stabilization on the vehicle itself, a COTS computer-based solution was adopted. The Sensar PV-1 sys-

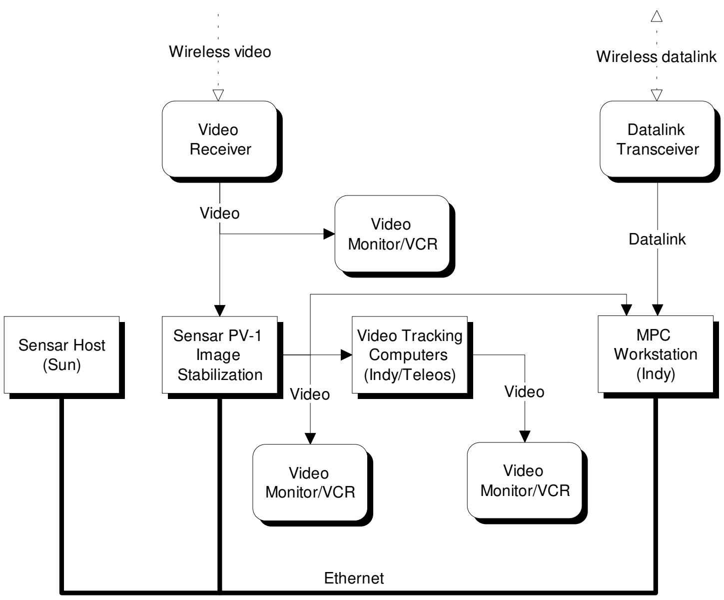

Figure 8: ASRT Ground Station

tem is a commercial implementation of pyramidal image processing techniques developed at Sarnoff Laboratories [15]. This system possesses dedicated special-purpose hardware to construct image pyramids. The image alignment process computes optical flow using local correlation estimates on these structures and then constructs a parametric model to minimize the difference between images.

Another COTS system was considered to implement the actual visual tracking. The Teleos AVP-100 tracking system uses texture-based algorithms developed originally by Nishihara [11]. While early testing of this system was promising, the techniques described here gradually surpassed the Teleos in terms of reliability. Also, eventually support was no longer provided by the manufacturer, and it was thus no longer an option. Either or both of the two tracking computers (Teleos AVP-100 or Indy workstation) could operate simultaneously with the stabilized video as their input signal. At any point in the video processing, it was possible to view and videotape the signal. A video switcher was used to implement this, which also facilitated displays on external monitors for demonstrations.

A feedback control path is provided to the onboard pan/tilt controller unit for the camera. This enables the tracking algorithm to repoint the camera to the estimated center of the target, independently of the vehicle motion. The limited range of motion is a constraint, and large variations in vehicle attitude can cause loss of target tracking, but the additional flexibility afforded by the pan/tilt controller minimizes the frequency of such occurrences.

6.1.1 MEP Software

The MEP software system is made up of several processes that provide target tracking capabilities, image stabilization and communication capabilities with the MPC system. Central to the MEP software system is the MEP Server. This process handles all communication among the MEP processes. The MEP Server is a state-machine that receives messages and broadcasts them to other processes depending on its current state. Some of these messages can cause the server to change its current state. Target tracking is provided by two processes. One of them is the motion based algorithm described in section 4.1 which runs on the Indy workstation, and the other is the Teleos tracker that runs on the Teleos system. Image stabilization is provided by the SENSAR system. Below we describe these components in more detail.

MEP Server: At any given moment, the server is in one of the following states: **START**, **MEDIA**, **DESIGNATE**, and **TRACK**. When the MEP server is in each state the various components are affected differently. In the **START** state, no video is displayed and no tracking is done. In the **MEDIA** and **DESIGNATE** state, video from the vehicle is displayed and control of the pan/tilt is enabled. Target designation is allowed in the **MEDIA** states however. Finally, when the server is in the **TRACK** state, tracking is performed and target designation is allowed while manual pan/tilt control is disabled.

Motion Tracker: The motion tracker displays video coming from the vehicle, allows target designation by the user, and performs motion-based tracking. All interactions with other MEP processes are done through the server. The motion tracker itself is implemented as a state-machine which at any given moment is in one of the following states: **IDLE**, **MEDIA**, **DESIGNATE**, and **TRACK**. While in the **IDLE** state the motion tracker listens to the server for messages requesting transitions to either the **MEDIA** or the **DESIGNATION** states. Also no video is shown and no tracking is performed. In the **MEDIA** state the tracker shows video from the helicopter and listens to the server for messages indicating transitions to either the **IDLE** of **DESIGNATE** states. As with the MEP server, there is currently an instantaneous transition from the **MEDIA** to the **DESIGNATE** state. In the **DESIGNATE** state target designation by the user is allowed. As soon as the user designates a target, the motion tracker sends a message to the server indicating the designation as well as the location of the target and enters the **TRACK** state. Once in this state, the tracker shows tracks and displays the region of interest containing the target. If necessary the tracker sends requests to the server to pan/tilt the camera to keep the target within the field of view. Also while in this state, messages from the server can cause transitions to the **IDLE**, **MEDIA** or **DESIGNATE** states. In addition, the server can receive messages to SENSAR indicating that a pan/tilt motion (saccade) just occurred. Using this information the tracker can reposition the region of interest.

Teleos Tracker: The Teleos tracker uses the AVP-100 tracking capabilities. The AVP-100 tracks by separating the target from the background based upon the relative motion. The target to be tracked is the largest moving object in the image, a 2D projection of the 3D environment, it favors closer as well as physically larger objects. It does not differentiate based upon the rate of movement although, depending on the various parameter settings, there is a lower motion threshold below which a particular object can no longer be tracked.

The Teleos tracker has two main functions. First, it provides an interface between the user and the AVP-100 system so that he can set the tracking parameters. The other function is to process the tracking data coming from the AVP-100. This data is tested for validity, filtered, and used to automatically vary some of the control parameters in real-time. Finally, the Teleos tracker is responsible for handling communication with the MEP server.

SENSAR system: Image stabilization is provided by the SENSAR system. The stabilization code runs in the PV-1 while the host is used to download the code, initiate the process in the PV-1, and communicate with it. Most of the code used for both the host and the PV-1 was provided by the manufacturer. Some modifications were made to both the PV-1 code and the host code in order to be able to communicate with the MEP server. This way SENSAR can inform the server when a saccade occurred, and the server can request a pan/tilt saccade to the SENSAR system on behalf of one of the target trackers.

There is a user-interface process that controls the display of video coming from the vehicle, allows user designation of a target, and controls the pan/tilt device. This process runs in the MPC system and it shares areas of memory with MPC processes. It is through this shared memory the MEP and the MPC systems communicate. These areas contain data related to pan/tilt control, the location and orientation of the helicopter, and the estimated location of the target. The user-interface process can receive requests from the user to allow or disallow the display of video from the helicopter, to allow or disallow the display of video coming from the helicopter, to allow designation of a target, and to pan or tilt the camera. When one of these requests is received the interface sends a request to the server to transition to the appropriate state and/or perform the requested actions.

6.2 Field Results

Since the ASRT program also included aspects of vehicle design and flight control, it was inevitable that only a limited number of flight tests would be available for the visual tracking system. Furthermore, of necessity these tests would occur only in the latest stages of the program after a stable controlled platform was available. Any earlier deployment of the full MEP system would have jeopardized the investment in equipment.

Early in the system development cycle, flight conditions were simulated by manual vibration of the camera, observing targets from a tower. In this and most other tests, the image stabilization computer and all other relevant hardware was included as it would be in the fielded system – the only difference was the absence of the vehicle itself. This strategy ensured that the tracking system was fully integrated long before flight tests were conducted. Integration with the MPC was also performed at this stage, enabling point-and-click designation from the operator control station.

As soon as the rotorcraft was available for even limited manually-piloted flights with the MEP hardware onboard, in-flight tracking tests were performed. Over the course of this testing, flights took place at three different locations and at various times of the day, providing a range of image conditions. The flight characteristics of the system are somewhat different with a human pilot. Our experience was that both the attitude control and the ability to hold position was generally worse under manual operation. This made the manually-piloted flights a good worst-case scenario for tracking. These tests also provided the opportunity to experiment with the optimal selection of panning angle as the target begins to approach the limits of the field of view.

Largely due to this testing approach, the system successfully tracked targets during its first full test under autonomous flight control, albeit only for short periods of stable flight during which it was possible to maintain reasonable attitude and pointing control. Later this same day, unrelated problems with the manual safety override system resulted in serious damage to the vehicle, and additional tracking tests were never performed in fully autonomous flight.

Figure 9 shows a frame of raw video from the onboard camera during hover close to the ground. The person in view was tracked and displayed at the MPC

Figure 9: Raw video from hovering vehicle during a period of active target tracking.

computer, as shown in Figure 10. The inset video (in the lower left corner of the screen) at this instant is the stabilized version of the raw video shown in Figure 9. As is typical for the stabilized video, the inset image includes an augmented background area (in black) to accommodate the rotating frame of reference experienced by the raw video as the vehicle rotates in the roll direction. The bounds of the stabilized image are expanded to include all four corners of the rotated video as long as continuous stabilization is attained. If frame-to-frame correlation is lost by the stabilization computer (as will occur during a sudden movement of the vehicle), the image will reset to zero degrees of rotation as the image bounds reduce to those of the raw video.

7 Discussion and Conclusions

Since this visual tracking problem was inherently an application-specific one that also involved a high degree of systems integration, it is necessary to view the results in the context of the complete fielded system. While other tracking methods may perform better in different circumstances (and, in particular, in more carefully controlled environments), these particular methods performed remarkably well over a range of conditions. In addition, these methods seemed to be compatible with the key hardware elements, including the camera, pan/tilt

Figure 10: Mission planner display during target tracking.

mechanism, airframe (structural and dynamic characteristics), videolink, and hardware stabilization computer.

Experience with this and similar applications has shown that any significant amount of parameter "tweaking" to improve performance will likely result in a fragile algorithm tuned to specific imagery. The algorithms described here have relatively few such parameters, and many of them can be set based upon analysis of "hard" conditions, including image size and resolution. More complex algorithms tend to rely on a larger number of critical parameters. Model-based tracking algorithms are a good example of this. If the objects to be tracked are well understood and generally viewed from expected orientations, a model-based tracker can easily be the best performer. But any deviation from the norm can result in poor tracking performance.

Therefore, the simplicity of the real-time tracking method (Method One) works in its favor with regard to robustness. The events that cause it to fail are not unlike those that might cause difficulties even for a human being trying to perform the same tracking. These circumstances include a tracked object that becomes very small or very nearly fills the field of view, as well as an extreme and sudden visual shift. We have seen that the adverse effect of this visual shifting is mitigated somewhat by careful control of the pan/tilt platform, especially when knowledge of deliberate camera motion is passed to the vision tracking algorithm, allowing it to recalculate the expected center of the tracked object. Furthermore, sudden shifts will become less common in this application as aerial vehicle control technology improves, allowing the vehicle to become the primary camera pointer.

The Zheng-Chellapa algorithm (Method Two) proved to be reasonably robust, as well. It is slightly more "clever," resulting in a higher degree of adaptation to image characteristics, but it makes relatively few assumptions about the image and has only a simple model of the object being tracked (as a non-deformable, rigid object). Still, even this straightforward approach to thresholding and target modeling was complex enough to sacrifice real-time performance, and it was also possible to fool it with background motion. Consequently, there was little reason to favor it over the real-time tracking method, especially as long as the camera hardware and image stabilization computer could minimize brightness/contrast variation and positional shifts.

Acknowledgments

This research was funded by the U.S. Army AATCOM at Fort Eustis, VA under Contract #DAAJ02-94-C-0021. Additional funding for the Mobile Robot Laboratory is provided by DARPA, the Office of Naval Research, and the National Science Foundation.

References

[1] Burt, P., "Smart Sensing within A Pyramid Vision Machine", *Proceeding of the IEEE*, Vol. 76, No. 8, pp. 106-144, August 1988.

[2] Cai, Q., Mitiche, A., Aggarwal J.K., "Tracking Human Motion in an Indoor Environment", *International Conference on Image Processing*, Los Alamitos, CA, pp. 215-218, 1996.

[3] Cheng, J-C, Moura J., "Tracking Human Walking in Dynamic Scenes", *IEEE International Conference on Image Processing*, Los Alamitos, CA, pp. 137-40, 1997.

[4] Clark, M., "Robot-based real-time motion tracker", *Proc. SPIE Symposium on Advances in Intelligent Robotics Systems*, pp. 1-7, 1989.

[5] Collins, T., Cardoze, D., and Gerber, D., "Object-Oriented Development of an Integrated Processor System for an Unmanned Aerial Vehicle", *Proc. AUVS '95 Symposium*, Washington DC, July 1995.

[6] David, P., Balakirsky, S., and Hillis, D., "A Real-Time Automatic Target Acquisition System", *Proc. AUVS '90 Symposium*, Dayton, OH, pp. 1-16, July-Aug. 1990.

[7] Gilbert, A., Giles, M., Flachs, G., Rogers, R., Hsun, Y., "A Real-Time Video Tracking System", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAM1-2, No. 1, pp. 47-56, January 1980.

[8] Gordon, M. and Schrage, D., "Development Of The Georgia Tech Autonomous Scout Rotorcraft Testbed", *Proc. AUVS '95 Symposium*, Washington D.C., July 1995.

[9] Hager, G., Toyama, K., "A Framework for Real-Time Window-Based Tacking Using Off-The-Shelf Hardware", DRAFT Documentation For Version 0.95 alpha, Yale University, Department of Computer Science, October 16, pp. 1-10, 1994.

[10] Kluge, K. and Thorpe, T., "Explicit Models for Robot Road Following", *Proc. 1989 IEEE International Conference on Robotics and Automation,* May 1989, Scottsdale, AZ, pp. 1148-54.

[11] Nishihara, H.K., Thomas, H.J., Huber, E., "Real-time tracking of people using stereo and motion", *Proceedings of the SPIE - The International Society for Optical Engineering*, Vol. 2183, pp. 266-73, 1994.

[12] Polana, R. and Nelson, R., "Low Level Recognition of Human Motion", *Motion of Non-Rigid and Articulated Objects Workshop Proceedings*, Los Alamitos, CA, pp. 77-82.

[13] Rumbaugh, J., Blaha, M., Premerlani, W. et al., Object-Oriented Modeling and Design, Englewood Cliffs, NJ: Prentice Hall, 1991.

[14] Sato, A., Kawada, S., Osaki, Y. and Yamamoto, M., "3D Model-Based Tracking of Human Actions from Multiple Image Sequences", *Systems and Computers in Japan*, Vol. 29, No. 8, pp. 48-56, 1998.

[15] Pyramid Vision 1 Alignment/Disparity Library User's Guide, SENSAR Inc, 1994.

[16] Samy, R., "Real-Time Tracking of Target Moving on Natural Textured Background", *SPIE Vol. 595 Computer Vision for Robots*, pp. 142-150, 1985.

[17] Wren, C., Azarbayejani, A., Trevor, D., Pentland, A., "Pfinder: Real-Time Tracking of the Human Body", *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, Los Alamitos, CA, pp. 51-56, 1996.

[18] Zheng, Q. and Chellapa, R., "Motion Detection In Image Sequences Acquired From A Moving Platform", *Proc. IEEE International Conference On Acoustics Speech and Signal Processing*, Minneapolis, MN, April 27-30, pp. 201-204, 1993.