SK^E Y^E BALL: REAL-TIME VISION SYSTEM FOR AN AUTONOMOUS MODEL AIRPLANE

Danko Antolovic, Chatham College, Pittsburgh, Pennsylvania 15232
Bryce Himebaugh, Steven D. Johnson, Indiana University, Bloomington, Indiana 47405

1. Introduction to the Sk^e y^e ball Vision Project

Sk^e y^e ball is a radio-controlled aircraft developed to host research and educational projects in autonomous aviation (Figure 1) [1]. The Sk^e y^e ball Vision Project, described here, has the objective of developing a tracking system that follows a shape in a relatively simple two-dimensional scene, in live video. This situated vision system is to be integrated into a larger robotic navigation system used to steer the semi-autonomous model airplane into holding pattern above a selected feature on the ground.

Vision systems for aerial robotics are a topic of active current research, and the reader is directed to references [7] to [14] for a very rough survey of the field.

1.1 History of the Vision System

The Sk^e y^e ball vision was first envisioned as a subsystem implemented on a microcontroller chip. It soon became obvious that a fast (and not too costly) implementation of the early processing stages was needed: vision became an ASIC-cum-microprocessor system, and it is still such a system today [2].

The development has gone through two distinct phases. The first phase yielded a laboratory prototype: the hardware was built from vendor evaluation boards, and the processors were a Xilinx XC4010 FPGA and a Motorola MC68332 (See Figures 2 & 3). Data were shared through an SRAM on the common bus. This architecture required considerable data copying, and the 25 MHz MC68332 processor was rather too slow for the task. Nevertheless, the system was capable of (slow) object tracking, moving a camera on a simple gimbal.

Figure 1. Skeyeball

0-7803-7844-X/03/$17.00 © 2003 IEEE

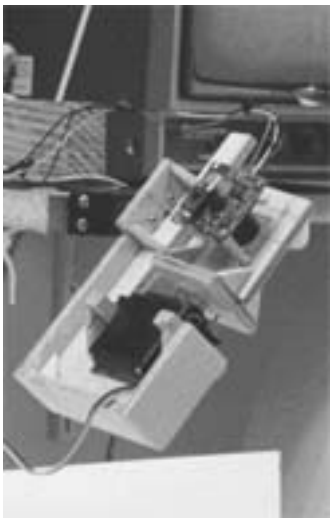

Figure 2. Camera Gimbal, Phase 1

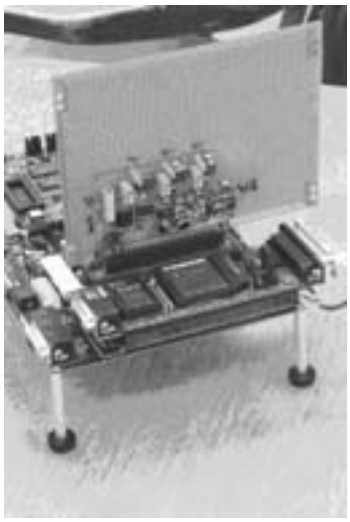

Figure 3. System Prototype, Phase 1

Figure 4. Aerial View of a Target Overflight

We obtained some realistic footage by flying the airplane with the immobile camera. The laboratory prototype was capable of detecting target features in overflight sequences, but tracking an object reliably at flight speeds was very problematic. Figure 4 shows a typical aerial view: the plane casts its shadow next to the bright square target (a brightly colored blanket on the grass).

The first phase gave us good insights into the minimal requirements for an airworthy vision system. The second (current) phase is described in this article. Its major architectural elements are a 90 MHz Motorola ColdFire microprocessor, a pair of synchronous XC4010 Xilinx FPGA's, and a dual-port RAM for shared data.

The fundamental vision algorithm has not seen much change over time, except for the addition of the threshold calculation in the second phase (See Section 2.1) - the improvement has been in the increased speed. Much greater modifications had to be made to the data flow, to take advantage of the dual-port memory and better bus architecture.

An electronic 3X zoom was added in the second phase, which works by restricting the vision field to the central one-ninth, at the appropriately higher resolution.

Finally, the system was given radio-controlled power-up and reset, and a manual mode in which the camera is moved by an operator on the ground. Figures 5 and 6 show the equipment built in the second phase: three circuit boards, an upgraded camera gimbal and the power switch. Figure 6 also shows the radio and TV links connected to the vision system.

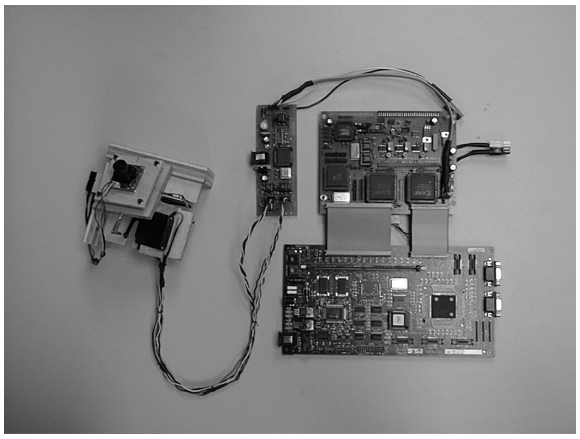

Figure 5. Circuit Boards and Camera Gimbal, Phase 2

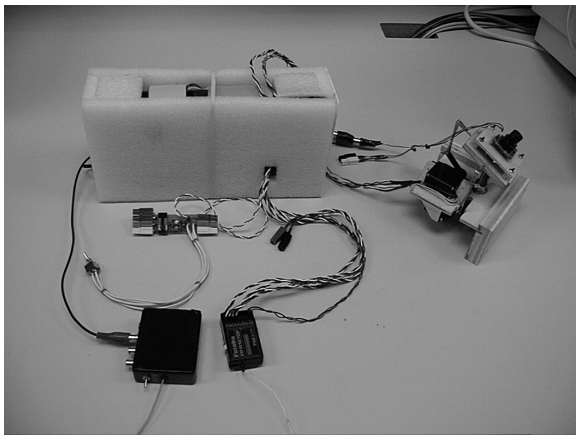

Figure 6. Vision System and Peripherals, Packaged For Test Flight

We have tested the airworthy implementation of the vision system, both in the lab and in the air. See Section 3.2 for the account of the test flights, and Section 3.3 for the description of tracking speed measurements.

2. Functional Overview of the Vision System

2.1 Vision Methodology

As stated in the introduction, the goal was to build a vision system that will follow an object in a relatively simple scene, in live video. We used a two-pronged approach to object tracking, taking into account the motion of the scene and the graphic "signature" of the object (see Figure 7).

This approach was motivated by the fact that object recognition is computationally intensive, and impossible to accomplish on a frame-by-frame basis with the available hardware. Nevertheless, the vision system had to operate fast enough not to allow the object to drift out of the field of vision.

The cycle of tracking steps is illustrated in Figure 8: recognize the small dark object as the target, obtain its offset from the center of the vision field, and move the camera to bring the object to the center. The vision system takes a still frame and bases its calculation on it. In the meantime, the object will change location, perhaps even drift out of the field. By the time it is calculated, the displacement vector may well be irrelevant. To address this possibility, the drift of the scene is tracked frame by frame, and the camera moves to compensate for it. The center of the field moves along with the target, and the displacement vector, when available, will still be meaningful.

The motion detector works by calculating what is in effect a normalized dipole moment of a pair of consecutive video frames. It cannot detect local motion within the image, only the integrated value of spatial displacements. For simple drift motion, this is an adequate algorithm, with the caveat that it is sensitive to appearance of new objects on the periphery, which it also interprets as motion. The importance of drift compensation decreased, however, with the use of a much faster processor in the second phase of the project (see Section 1.1).

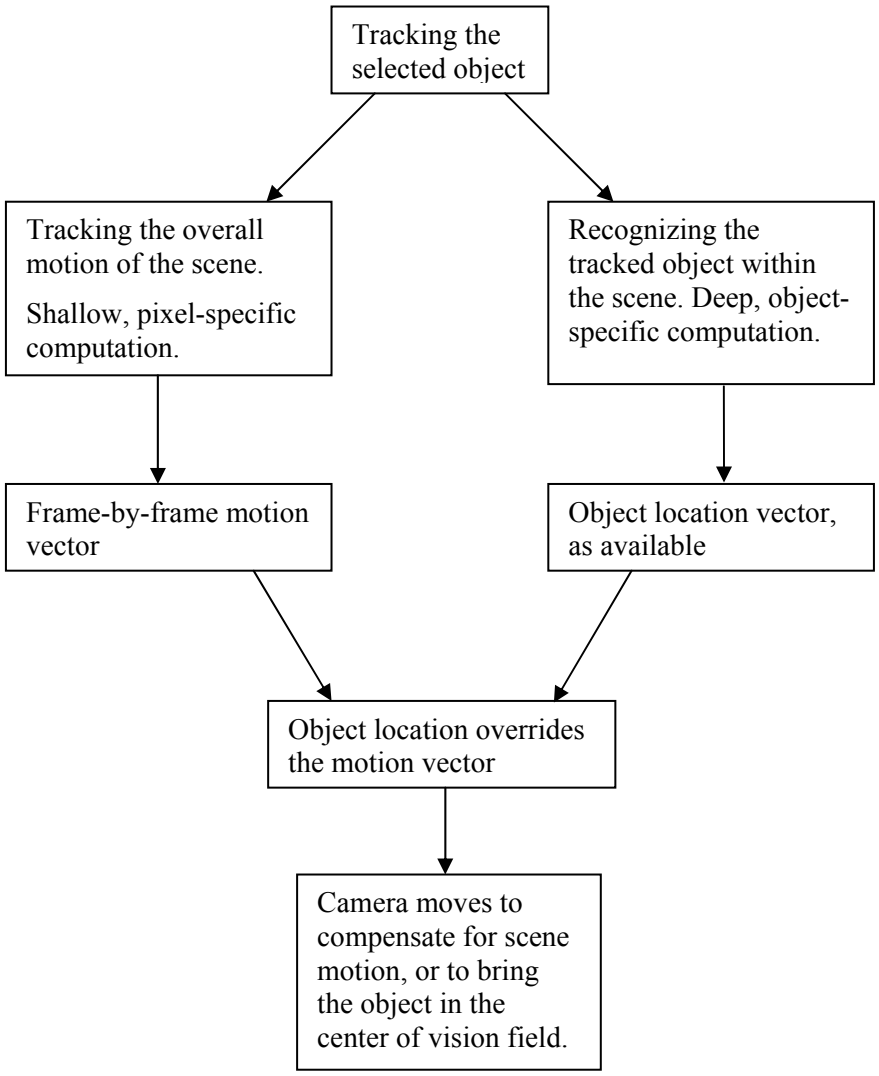

Figure 7. Method Overview

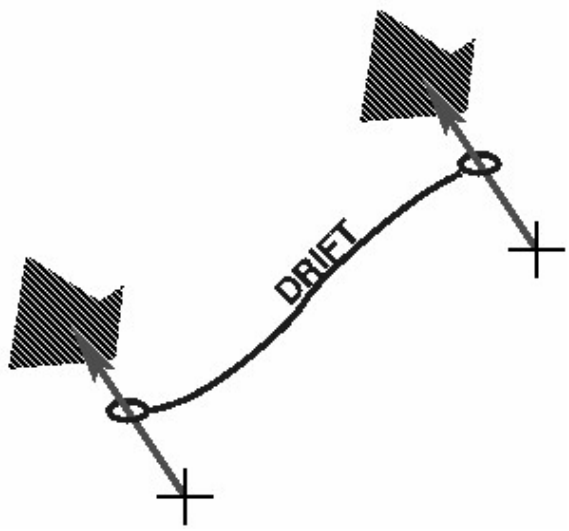

Figure 8. Drift Compensation

Object recognition typically requires several stages in which redundant and non-essential visual information is parsed out, until one is left with a selection of well-defined objects, also referred to as the features of the scene. In our case, the stages are: thresholding, edge detection, segmentation into connected components, clean-up and signature/location calculation.

Figure 9 gives a functional overview of the vision system, where the progressively thinner arrows signify the reduction in bulk of the visual information. This is the typical "funnel" of the vision problem, leading from simple computations on large volume of data to complex ones on a small volume, yielding a number or two as the result. Let us illustrate the magnitude of this reduction: At 30

frames per second, 483 lines per frame, and 644 byte-sized pixels per line, raw camera output amounts to 9.33 Mbytes/sec. At the end of the process, the vision produces a few dozen bytes, about ten times per second. Actual recognition rate depends on the contents of the image.

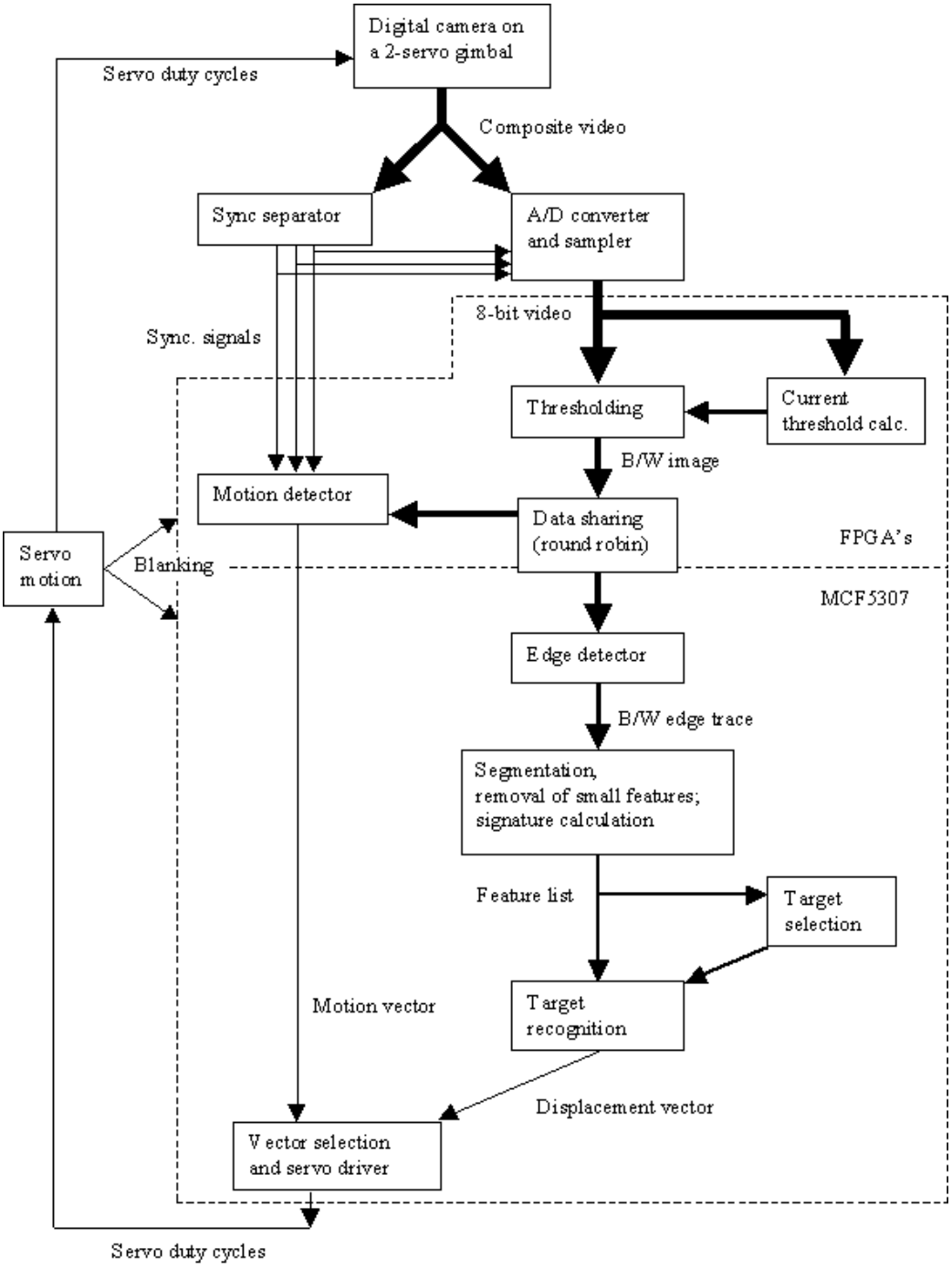

Figure 9. Architecture and Data Flow

Thresholding refers to the usual conversion of grayscale to black and white. Threshold is obtained by simple analysis of the grayscale histogram, which is assumed to be bimodal, separating the pixels into features and background (see [2], sec. 9.1). Both image thresholding and the threshold calculation are performed in hardware, on the frame-by-frame basis.

Edge detection is rendered simple by the preceding thresholding step: a pixel is defined as an edge point if it is black in color, and has between 2 and 7 black neighbors. This step is implemented in code.

Segmentation: edge points are logically grouped into connected threads or loops, and each connected component is assumed to represent a distinct feature (object) in the scene. The output is a collection of arrays, each containing the edge coordinates of one feature. We use a two-pass algorithm described by Lumia et al. [3] (Algorithm 3 in [3]). For each image line, adjacent edge points in that and the previous line are labeled as belonging to the same connected component. Since the components can have an arbitrarily complex topology, line-by-line labeling may result in assigning several labels to the same component. These labeling equivalences are resolved by means of a depth-first graph search.

Identification: to identify a target object within the scene, we used the principal second moments (also known as the moments of inertia in the mechanics of rotation of rigid bodies [4]) as the "signature" of each object. Second moments are invariant under rotations and translations, fairly easy to calculate, and work well in simple scenes.

2.2 Biological Parallels

It is not entirely surprising that certain functional analogies should develop between robotic perception, such as this real-time vision system, and perception in animals [5]. For example, camera motion based on feature recognition bears a similarity to the (involuntary) saccadic motion of the eyes, the one-shot movement that brings a detail of interest into the center of the vision field. Eyes' saccades are fast, have large amplitude (as large as the displacement of the object of interest), and they are ballistic movements, i.e. not corrected along the way. Such motion is compatible with a need for speed over precision: if the object identification is computationally intensive, the result is used to full extent, and as fast as possible [6].

Furthermore, once the camera motion has been initiated, image sampling must be suppressed until the camera has moved to the new location. Otherwise, the result of the tracking step would not be registered, and the next cycle of feature recognition would receive stale data as input. This suppression resembles saccadic blanking in human/animal vision [5]. It is well known that the sensitivity of the optic nerve is suppressed while a saccade is in progress, and it is plausible that the purpose is to prevent visual confusion in biological systems as well.

3. Project Status

3.1 Capabilities and Limitations

We have tested the vision system on the laboratory bench, by using a rotating table (see Section 3.3), and in the air, by deploying it on the Skeyeball airplane.

In the lab tests, the vision system tracks targets reliably under translation and rotation, and in the presence of several shapes introduced as interference. Excessive skew breaks off the tracking, since the vision algorithm makes no provision for it, but an oblique view of ca. 20 degrees is still acceptable. Likewise, occlusion is interpreted as a change of shape, and the target is lost when partially occluded (e.g. by drifting beyond the edge of the vision field). These limitations are obvious consequences of the vision methodology described in Section 2.1 and Figure 8.

The tracking and motion detection work equally well with the zoom engaged. As expected, zoom makes the system more reliable in tracking small targets, at the expense of limiting the field of vision to the central one-ninth.

3.2 Test Flights

The tracking system has been deployed in three sets of test flights of the Skeyeball aircraft so far, amounting to about ten flights in all. In the first two sets of flights, the pilot controlled the vehicle with verbal instructions from a ground operator monitoring the video feed. The tracking component

was activated once the target, a white blanket, came into view. This was a clumsy control arrangement and it proved difficult to coordinate the test. In addition, we experienced substantial radio interference in the radio signal, but we believe that this interference only affected the air-to-ground transmission, not the on-board image processing. The tracking subsystem exhibited unmistakable tracking behavior for short periods (one or two seconds at approximately 150 feet) that the pilot was able to keep the target within the sweep range of the camera gimbal. Tracking was apparently also disrupted by the view of the landing gear.

In the third, most recent set of flights, we tested the system with a gimbal with increased range of motion, which was under manual control of the ground operator when not tracking. Although the target was much easier to locate, evidence of aerial tracking remains inconclusive. Further development and testing are planned for the summer of 2003.

Additional system features are needed for more extensive test flights. Enhanced air-to-ground communication will improve diagnostic data logging, as would a redundant image processing system on the ground workstation. However, both of these alternatives are somewhat compromised by EMI interference, which can be reduced but is unlikely to be eliminated entirely in this inexpensive system.

3.3 Measurements of the Tracking Speed

The Sk^{e}y^{e}ball airplane flies within a range of speeds and altitudes; our fixed-camera flights have ranged from 18 to 60 mph, with a typical speed of ca. 35 mph. Likewise, target-overflight altitudes have been from 60 to 320 ft. Consequently, the line of sight to the target feature changes direction relative to the body of the airplane, with certain angular velocity, and the vision system must be able to keep up with it. In the test flights, the target passed through the vision field of the fixed camera in time intervals ranging from 0.8 to 4.2 seconds, depending on the altitude and velocity of the airplane.

In order to obtain a quantitative measure of the vision's tracking abilities, we have constructed a test rig - a rotating table with features to track. The vision system locks successfully onto the (largest) presented feature and the camera turns following the rotation of the table. The rotation speed is gradually increased, until the tracking breaks off or target acquisition becomes impossible. Figure 10 shows the experimental setup: the camera gimbal, the rotating table with two features, and the TV screen showing the camera's view.

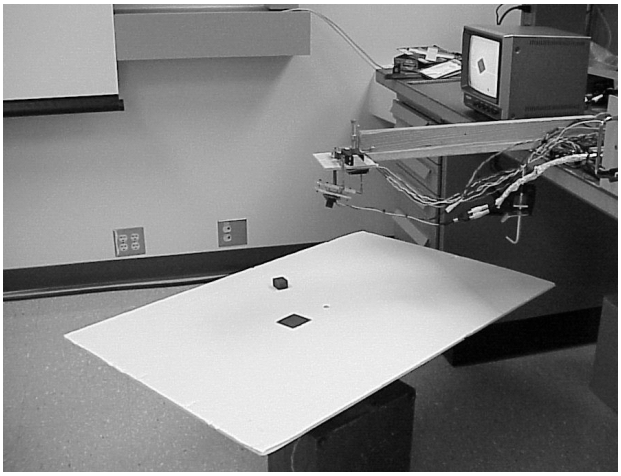

Figure 10. Laboratory Set-Up for the Tracking-Speed Measurements

Figure 11 shows the geometry of the setup. The speed of the table's motor was regulated by applying variable voltage, and the angular speed of the table was measured as the time needed for ten turns. A simple formula relates the table's angular speed, ω , to the camera's sweeping speed, θ :

$$\theta = \frac{\omega}{\sqrt{1+\left(\frac{d}{r}\right)^2}}$$

where d is the elevation of the camera above the table, and r is the distance of the target feature from the center of the table.

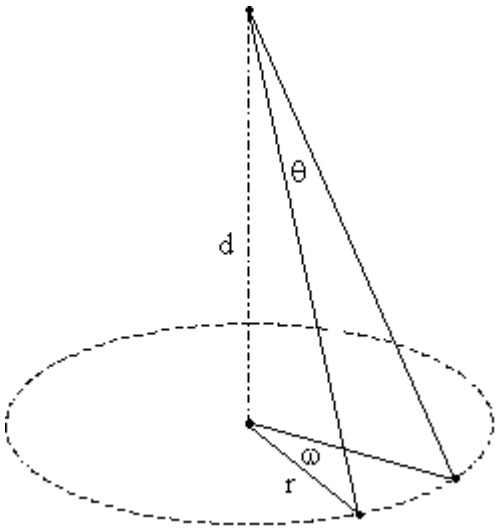

Figure 11. Table Speed vs. The Camera's Sweep

At d = 26.5 cm, and r = 10 cm, we found that the tracking was still reliable with the camera sweeping an arc at the maximum speed of: θ_{max} = 45 degrees/second. The camera's field of vision is ca. 47 degrees high and ca. 60 degrees wide, which puts this vision system within the range of speeds required to keep up with the overflight speeds that were quoted above.

Of course, tracking speed depends on the complexity of the scene. These measurements were performed with two or three shapes, plus an intermittently visible edge of the table. The scene observed in a real flight is richer in features, but at least for grassland and trees, features tend to have low contrast and disappear below the threshold, which is set dynamically, to single out high-contrast targets.

4. Hardware Architecture

Figure 12 is an overview of the architecture of the vision system. It shows all the signals pertaining to the flow of data from the camera, through the processors and back to servo motors, although it omits some peripheral details (see [2]).

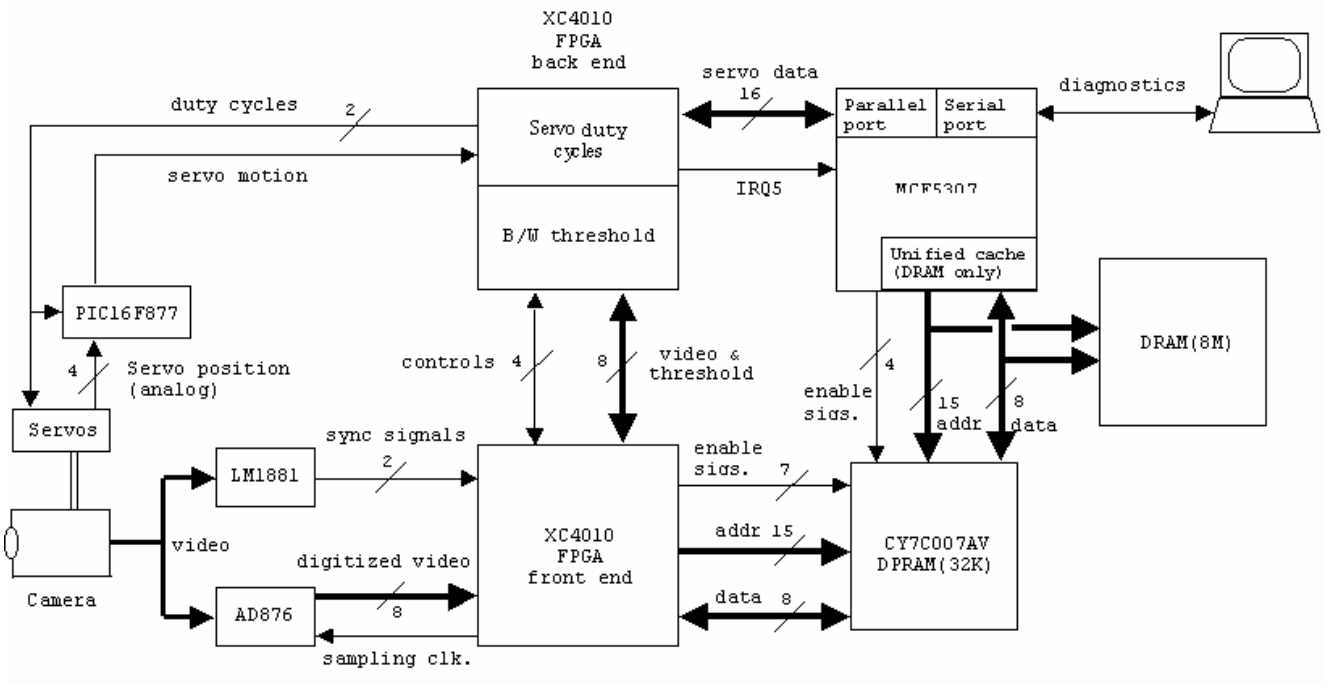

Figure 12. Architecture of the Vision System

Camera: the "eye" of the system is a small digital camera, producing grayscale (non-color) NTSC video signal. The camera is mounted on a gimbal driven by two servo motors, with a 50-degree range of motion in each direction, and is permanently focused on infinity.

Sync separator: the three NTSC synchronization signals are extracted from the video by means of a sync separator, LM1881 by National Semiconductor, mounted on a prototyping board along with supporting circuitry.

A/D converter: we use Analog Devices' AD876, which is a pipelined 10-bit converter. It is mounted on the same proto board, with supporting circuitry for reference voltages.

Sampling control: digitization (frame-grabbing), thresholding and threshold calculation, motion detection and zoom are implemented as digital designs on a synchronous pair of Xilinx XC4010 FPGA's, running at 33.3 MHz. Start-up configuration is done with two Atmel's AT17LV configuration ROMs.

Object recognition: the entire object recognition is implemented as code, running on a 90 MHz Motorola MCF5307 ColdFire integrated microprocessor. We use a commercial evaluation board, Arnewsh's SBC5307, with 8 megabytes of DRAM, start-up flash ROM, expansion bus and communication ports.

Image memory: the two processors share image data through a 32K dual-port SRAM, CY7C007AV by Cypress Semiconductor. Data access is implemented as a round-robin procedure, with the objective of speeding up high-volume data transfer in the early stages of the vision process.

Servo driver: driver for the servo motors that move the camera is implemented on one of the two FPGA's. Motion feedback from the servos is generated by a PIC16F877 microprocessor, on the basis of servos' analog position signals.

5. Summary Remarks

Real-time perception can be envisioned as a funnel in which the data volume is reduced, but the algorithmic complexity increases. Typically, there will be several stages with fairly different breadth/depth ratio.

This is intrinsically not a problem amenable to single-architecture processing. Of course, a speed tradeoff is in principle always possible, but engineering considerations such as power consumption and heat dissipation place a very real limit on that approach. We believe that it is better to use several processor architectures, each suitable for a different stage of the perception process. Configurable microchips make this goal both realistic and appealing.

Robotic perception is also a problem in embedded computing. Requirements imposed by the small model airplane are a bit on the stringent side, and one can envision a much more relaxed design for an assembly line or security system. However, the need for perception is naturally the greatest in mobile robots. In such applications the vision system will always have to be compact and autonomous, because it bestows autonomy on a mobile device whose primary function is something other than carrying a vision system around.

Animal vision cannot be separated from cognitive functions and motor coordination, and this must be true for robotic vision as well. How much "intelligence" is built into high-level processing of visual information depends on the ultimate objectives: for example, searching for a three-dimensional shape in a complex panorama is a problem different from that of hovering above a prominent feature on the ground.

In terms of steering and motor coordination, biological parallels are relevant. It is known that inertial motion sensors play a large role in the gaze control of mobile animals. Since the input from the motion sensors is simpler, and the processing presumably faster, this sensory pathway provides the supporting motion information much faster than can be obtained by visual processing. The airplane may very well benefit from an eventual integration of its vision and attitude/motion sensors.

Visual perception is an ill-posed problem, and examples of functioning compromises may be more valuable than exact results. A few pitfalls notwithstanding, we believe that a synthesis of computational, physiological and engineering knowledge will be necessary for the eventual development of reliable and versatile perception systems.

References

[1] Johnson, Steven D., Bryce Himebaugh, Skeyeball Project Home Page. http://www.cs.indiana.edu/hmg/skeyeball.

[2] Antolovic, Danko, 2001, Development of a Real-Time Vision System for an Autonomous Model Airplane, Indiana University Computer Science Department Technical Report No.557, http://www.cs.indiana.edu/Research/techreports/TR557.shtml

[3] Lumia, Ronald, Linda Shapiro, Oscar Zuniga, 1983, A New Connected Components Algorithm for Virtual Memory Computers, Computer Vision, Graphics and Image Processing, v. 22, pp. 287-300.

[4] Goldstein, Herbert, 1980, Classical Mechanics, 2nd Edition, Addison-Wesley.

[5] Wandell, Brian A., 1995, Foundations of Vision, Sinauer.

[6] Rodieck, R.W., 1998, The First Steps in Seeing, Sinauer.

[7] Amidi, Omead, Takeo Kanade, James Ryan Miller, April 1998, Vision-based autonomous helicopter research at Carnegie-Mellon Robotics Institute 1991-1997, American Helicopter Society International Conference, Heli, Japan,. Paper no. T7-3, http://www.ri.cmu.edu/pubs/pub 923.html

[8] Jones, H., E. Frew, B.Woodley, S. Rock, 1998, Human-robot interaction for field operation of an autonomous helicopter, Proceedings of SPIE Vol. 3525 Mobile Robots XIII and Intelligent Transportation Systems, Technical Conference on Mobile Robots XIII. http://sun-valley.stanford.edu/~heli/info.html.

[9] Nordberg, Klas, Patrick Doherty, Gunnar Farnebäck, Per-Erik Forssén, Gösta Granlund, Anders Moe, Johan Wiklund, October 2002, Vision for a UAV helicopter, Proceedings of IROS'02, workshop on aerial robotics, Lausanne, Switzerland.

[10] Sharp C., O. Shakernia, S. Sastry 2001, A vision system for landing an unmanned aerial vehicle, http://robotics.eecs.berkeley.edu/~cssharp/SSSicra2001 submit.pdf.

[11] Vaughan, R. T., G. S. Sukhatme, F. J. Mesa-Martinez, J. F. Montgomery, 2000 Fly spy: Lightweight localization and target tracking for cooperating air and ground robots, Distributed Autonomous Robotic Systems, 4, pp. 315–324.

[12] Bagnell, James, Jeff Schneider, 2001, Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods, Proceedings of the International Conference on Robotics and Automation 2001, IEEE, http://www.ri.cmu.edu/pubs/pub_3791.html

[13] Ettinger, S. M., M. C. Nechyba, P. G. Ifju, M. Waszak, October 2002, Vision-Guided Flight Stability and Control for Micro Air Vehicles, International Conf. on Intelligent Robots and Systems, http://www.mil.ufl.edu/~nechyba/mav/

[14] Saripalli, Srikanth, James F. Montgomery, Gaurav S. Sukhatme, May 2002, Vision-based autonomous landing of an unmanned aerial vehicle, IEEE International Conference on Robotics and Automation, Anthony A. Maciejewski, ed.