Introduction

During the summer of 1990, the Association for Unmanned Vehicle Systems announced The First International Aerial Robot Competition [1] to be held the following July. When the fall quarter began at Georgia Tech, the School of Aerospace Engineering organized a multi-disciplinary team and attracted industrial sponsors to field an entry to compete. The team included Aerospace Engineering, Civil Engineering, Electrical Engineering, Mechanical Engineering, and the College of Computing. The team varied in size over the course of the year, but averaged around 20 graduate students and 7 faculty advisors.

The competition was held in an outdoor volleyball court, as shown in Figure 1. Two bins, a little less than 2 meters in diameter were positioned on opposite sides of a one meter high central barrier. Six metallic disks 3 inches in diameter (similar to the old 8mm film reels) were randomly placed in one of the bins. The task was for the vehicle to fly to the source bin, locate and retrieve a disk, and then fly to the destination bin and deposit the disk in the bin. The goal was to move as many disks as possible within 3 minutes. The vehicle was required to fly completely autonomously from takeoff until landing with no human intervention.

The basic helicopter platform that the Georgia Tech team chose for use in the competition is shown in Figure 2. This remotely piloted vehicle was heavily modified to meet size and performance requirements. Flying within a fenced volleyball court required rapid accurate sensing of the vehicle's position. Inertial navigation systems were deemed inapplicable and it was decided that a sensor giving the vehicle's location relative to the ground was required for safe flight.

Radio triangulation systems were briefly explored but found to be prohibitively expensive. Since it had already been decided to use an on-board camera for target location, it was suggested that it could be extended to also sense the vehicle's position. A sensitivity analysis completed later as part of the concurrent engineering effort [9] determined that the benefits of two cameras, one devoted to each task, outweighed the added weight penalties. The final deployed configuration consisted of two vision systems. The first camera,

Figure 1: Competition Arena

Figure 2: Vehicle

with a horizontal field of view generated by the use of a mirrored conic, looked out at artificial landmarks placed around the outside of the arena. The second camera utilized a strobe light and downward looking optics to view the disks in the source bin. Since the target detection system used a subset of the software developed for the vehicle position determination system, only the latter will be described in this paper. A more in-depth treatment of the similarities and differences of the two systems can be found in [3].

Two "Stinger" Integrated Vision Systems (IVS) were purchased from Dickerson Vision Technologies, Inc. for the project. The IVS combined the Charge-Coupled Detector (CCD), the digitizer and a Motorola 68000 based microcomputer on one small circuit board. This removed the need for a camera, digitizer, and high bandwidth video link to a microcomputer. The Texas Instruments TC211 CCD produced an image

of 165 rows by 192 columns with 256 levels of gray scale for each pixel. Fixed focal length optics were used to reduce weight, size and cost.

To simplify the image processing task it was decided that the landmarks should appear in the images as bright objects against a dark background, allowing a thresholding technique to be used for landmark recognition. Thresholding allowed rapid grouping of pixels into landmark or background categories. To threshold an image each pixel was compared to a threshold intensity value. If the pixel's intensity was greater than the threshold then the pixel was part of a landmark, otherwise it was part of the background. A region growing algorithm grouped the pixels categorized as landmarks into blobs. A center of mass algorithm was then used to compute the center of the blob (landmark) in image coordinates. Sub-pixel interpolation provided the center of mass locations to 1/256 of a pixel. The process proved accurate to 1/128 of a pixel with the least significant bit fluctuating randomly. The navigation landmarks used were 500 watt incandescent light bulbs. These were found to be brighter than nearly all expected backgrounds. Filters were then used on the camera to attenuate the light to usable levels. The exposure time was allowed to vary from 0.5 msec to 10 msec while tracking the varying lighting conditions. Under normal lighting conditions the exposure time was around 1 msec. These short exposure times were used to reduce image blur resulting from imaging from the moving platform.

The outer control loop, which replaced the human pilot, operated at a frequency of 10 cycles per second. The IVS allowed image acquisition and image processing to occur on-board, while meeting the required 10 hertz rate. The locations of landmarks and targets in the images along with other vehicle telemetry was transmitted via a radio downlink to an Intel 386SX microcomputer equipped with a Cyrix 387SX math coprocessor where the math intensive vector calculations required by the visual triangulation algorithm were executed.

1. Algorithms

1.1 Background

Many methods have been presented in the literature for detecting and growing regions in images. In [8] a method for region extraction based on building a pyramid from reduced resolution versions of the image is presented. Pixels are grouped based on similarity and proximity. In [6] blobs are found based on detecting the edges surrounding them. In [11] the problem is reformulated as a cost function minimization problem using the A^* search algorithm. In [2], four different techniques are presented. The locally operating blob coloring algorithm groups connected pixels into regions based only on local similarities. Semantic region growers use semantic information to help classify pixels and regions. Once regions have been grown to a small size, merging operations occur based on Bayesian decision functions. These functions evaluate the probability that two regions have the same semantic interpretation. Globally operating region growing via thresholding techniques assume bimodal images. A threshold value is chosen such that it falls in the valley between the two peaks on a gray scale histogram of the image. In [5] the Chow-Kaneko technique is presented which can be used when the bimodal assumption fails. This algorithm subdivides the image into rectangular sub-images with the hope that the smaller pieces will be bimodal. Thresholds are then determined for the sub-images as before and image processing uses the corresponding thresholds while processing. A related approach using local gradients and least-squares interpolation is presented in [7]. In [13], objects are segmented based on range information gathered from optic flow algorithms. Local mean thresholding techniques [4, 10] compute a threshold based on a function of the median intensity of the neighboring pixels. Pixels exceeding this threshold are classified as belonging to a region.

1.2 Adaptive Landmark Detection

Background intensities, landmark intensities, and the size of imaged landmarks all varied widely as the helicopter moved within the arena. This resulted in images in which it was difficult to detect landmarks. To overcome these problems a landmark tracking algorithm was developed. This method used a feedforward windowing technique where the locations of landmarks in the current image were remembered. Processing

in the next image was localized to small windows surrounding those image locations. To bootstrap the algorithm, the entire image was processed to find all landmarks. However, during flight, landmarks were unpredictably occluded by both the helicopter and barriers within the arena. This required frequent bootstrapping of the process to reacquire landmarks as they became visible, thereby reducing the accuracy of the resulting position estimate.

To hold image intensities at desired levels, image exposure time was varied in accordance with the perceived intensity of landmarks in a feedforward process. This improved image quality in the diverse lighting conditions experienced during the vehicle's moving back and forth between the lights by allowing the exposure time to track the changing ambient lighting conditions. Since the IVS used a software selectable exposure time, this was a straightforward extension.

To better handle cluttered backgrounds, including areas in which the background intensities exceeded the landmark intensities, an adaptive landmark detection algorithm was developed. The algorithm used local mean thresholding to eliminate the problems inherent with a global threshold. This allowed recognition of lights placed in front of bright backgrounds, such as the sky on a sunny day, as well as dim backgrounds such as a brick wall at night, while providing good rejection of bright background areas. This ability was a result of the algorithm's automatically compensating for varying ambient lighting conditions. The local mean thresholding scheme also automatically handled varying background lighting conditions within a given image.

To improve performance, our implementation used a trailing mean calculation where the median value of the intensities of the last four pixels sampled was used. A pixel whose intensity exceeded twice this median value ($T = 2 * mean$) was classified as being within a region. Figure 4 shows the algorithm in pseudo-code. A static threshold of one and one-half times the mean ($T = 1.5 * mean$) was then used in a region thresholding algorithm to grow the region. Figure 3 shows pseudo-code for a simple region growing algorithm. To reduce execution time our implementation was not recursive. However, it is functionally the same. The intensities of pixels classified in the region were set to zero to prevent duplicate recognition.

1.3 Vehicle Position Determination

The vehicle positioning subsystem was responsible for determining the X,Y location of the helicopter within the arena. Altitude was sensed separately with an ultrasonic altimeter. The on-board IVS reported the locations of any landmarks detected in image coordinates. The task of using these pixel locations to accurately and robustly determine where the vehicle was located was a difficult problem. The adaptive position determination algorithm, shown in pseudo-code in Figure 5 reduced the problem to a solvable form by making several simplifying assumptions.

The first problem was to match each perceived light in the image to a real landmark in the world model. The general assignment problem is exponential. To make this tractable, the system was bootstrapped from a known location. In feedforward mode, estimated vehicle position was based on the last calculated location. Using this estimate, expectations as to where each landmark would appear in the image were generated. These expectations were then used to guide the matching of imaged blobs to landmarks.

When a landmark had been matched to a blob for 10 images in a row, the landmark was flagged as being tracked. For each landmark being tracked, the location of the matching blob in the image was saved and used in a feedforward process. The blob in the next image closest to the saved location was determined. If the distance between the blob and the saved location was less than a threshold value, the blob was assumed to match the landmark. If the threshold was exceeded 5 times in a row, tracking was abandoned. This process helped to maintain the correspondence between landmarks and image blobs during short occlusions of the landmark and reduced erroneous matching of landmarks to background artifacts.

When at least two lights had been matched to landmarks, it was possible through a triangulation technique to determine where the vehicle was located. Lines drawn along the perceived viewing angle on the world map intersected at the point where the camera was located when the image was taken. The

```
GROW_BLOB(row, col, threshold, row_mass, col_mass, count, blob_sum)
BEGIN
    IF image[row][col] > threshold THEN

        /* Add the contribution of this pixel to the center of gravity calculations */
        row_mass = row_mass + image[row][col] * row
        col_mass = col_mass + image[row][col] * col
        count = count + 1
        blob_sum = blob_sum + image[row][col]

        /* Zero this pixel's intensity so it isn't counted again */
        image[row][col] = 0

        /* Try growing the pixel above */
        GROW_BLOB(row - 1, col, threshold, row_mass, col_mass, count, blob_sum)

        /* Try growing the pixel to the right */
        GROW_BLOB(row, col + 1, threshold, row_mass, col_mass, count, blob_sum)

        /* Try growing the pixel below */
        GROW_BLOB(row + 1, col, threshold, row_mass, col_mass, count, blob_sum)

        /* Try growing the pixel to the left */
        GROW_BLOB(row, col - 1, threshold, row_mass, col_mass, count, blob_sum)
    ENDIF
END
```

Figure 3: Grow Blob Algorithm

intersection points for all pairs of lights where the angle of intersection was $30^o \leq \theta \leq 150^o$ were computed. The complicated optics and low resolution of the camera combined to limit the accuracy for which each landmark's angle of incidence could be computed. Small angular errors resulted in greatly amplified error when computing the point of intersection for nearly parallel lines. The intersection points of lines where the angle of intersection was $0^o \leq \theta < 30^o$ or $150^o < \theta \leq 180^o$, (lines within 30^o of parallel) were found to be so error prone as to warrant their exclusion.

To generate the best vehicle position estimate from the resulting cluster of intersection points, the mean X, Y value was computed. During the competition it was expected that there would exist spurious bright spots in the image background. These could originate from photographic lighting as well as reflections from the sun on bright surfaces. It was expected that periodically the system would erroneously label a background artifact as a landmark. This would cause all intersections computed using that landmark to be in error. To keep these points from biasing the position calculation, any points that were more than twice as far from the mean X, Y value as the point nearest to the mean value were discarded as spurious. This process prevented the problem of all points being discarded and was faster to compute than a standard deviation. A final mean X, Y value was then computed using the remaining points and reported as the predicted vehicle position.

2. System deployment

The assembly drawings for the conic used to generate the 360^o field of view are shown in Figure 6. The slope of the cone was chosen to match the camera optics. With the 42^o slope of the cone and a 15^o field of view lens, the system had a $+3^o ... -4.5^o$ vertical field of view. This was the main problem with the system, the susceptibility to vehicle roll and pitch. A wider field of view lens and matching conic would

```
INPUT:    image[maxrow][maxcol]
OUTPUT:   list of blobs

CONST     num_samples = 4
BYTE      queue[num_samples]
WORD      queue_sum
BYTE      queue_mean
BYTE      old_pixel

BEGIN
    /* We need to preload the queue with typical values so we don't get a false hit before scanning real data. */
    /* We will grab values from the image diagonal and assume they are typical */
    FOR i = 1 TO num_samples DO
        queue[i] = image[i][i]
        queue_sum = queue_sum + queue[i]
    ENDDO

    /* now scan the image for blobs */
    FOR row = 1 TO num_rows DO
        FOR col = 1 TO num_columns DO
            IF (image[row][col] > (2 * queue_mean) THEN
                /* pixel is within a blob */
                row_mass = 0
                col_mass = 0
                count = 0
                blob_sum = 0
                GROW_BLOB(row, col, 1.5 * queue_mean, row_mass, col_mass, count, blob_sum)

                /* if the landmark contains at least min_size pixels then remember it */
                IF count ≥ min_size THEN
                    blobs[num_blobs].row = row_mass/blob_sum
                    blobs[num_blobs].col = col_mass/blob_sum
                    num_blobs = num_blobs + 1
                ENDIF
            ENDIF

            /* Don't corrupt our median value by using pixels that have been set to 0 */
            IF (image[row][col] > 0) THEN
                /* remove the oldest pixel from the queue */
                old_pixel = dequeue(queue)
                /* put the new one in */
                enqueue(queue, image[row][col])
                /* subtract the old value from the sum and add in the new pixel */
                queue_sum = queue_sum − old_pixel + image[row][col]
                /* compute the new mean */
                queue_mean = queue_sum/num_samples
            ENDIF
        ENDDO
    ENDDO
END
```

Figure 4: Adaptive Thresholding Algorithm

```
INPUT:    blob locations in image coordinates
OUTPUT:   vehicle X,Y position

BEGIN
    /* create a unit vector from the center of the image to the blob, rotate the vector to compensate for vehicle yaw */
    FOR i = 1 TO num_blobs DO
        blobs[i].vector = make_unit_vector(image_center, blobs[i].image_loc)
        blobs[i].vector = rotate(blobs[i].vector, current_yaw)
    ENDDO

    /* try to match each light we are tracking to a blob */
    FOR i = 1 TO 5 DO
        match light to blob closest to the blob location in the last image
        If nearest blob is farther than a threshold distance away, mark light as unmatched.
    ENDDO

    FOR each remaining light DO
        match light to blob closest to the predicted location
    ENDDO

    FOR each pair of lights DO
        IF (30 ≤ angle of intersection ≤ 150) THEN
            calculate_intersection(line1,line2)
        ENDIF
    ENDDO

    mean_X, mean_Y = mean X,Y value of intersection points

    /* Find the distance to the point closest the mean value */
    closest = ∞
    FOR i = 1 TO num_points DO
        If distance(mean_X, mean_Y, points[i].X, points[i].Y) < closest THEN
            closest = distance(mean_X, mean_Y, points[i].X, points[i].Y)
        ENDIF
    ENDDO

    FOR i = 1 TO num_points DO
        If distance(mean_X, mean_Y, points[i].X, points[i].Y) > 2 * closest THEN
            discard(points[i])
        ENDIF
    ENDDO

    vehicle_X, vehicle_Y = mean X,Y value of remaining intersection points
END
```

Figure 5: Adaptive Position Determination algorithm

have provided more resiliency to this problem. Unfortunately, this was the widest angle lens available at the time.

The conic was manufactured out of brass stock and polished until smooth. It was then silver plated to create a mirror. A threaded hold was drilled in the back for mounting. A smaller hole was drilled completely through the mounting bolt and the center of the conic. A small LED mounted in the point of the conic was visible in the image when energized. When the vision system was started, it energized

the LED and determined it's image location. This automatic method of locating the center of the conic provided some resiliency to rough handling. The center location of the conic was used by the adaptive position determination algorithm while converting pixel locations to angular coordinates. Figure 7 shows the IVS and the conic assembly mounted on a test stand.

Figure 6: Conic Assembly Drawings

Figure 7: Picture of Camera and Conic Assembly

Figure 8 shows the forward payload area of the helicopter. The conic assembly was mounted on top of the support bracket in the top of the picture. The IVS circuit board was mounted in the guide rails visible in front of the bracket.

Figure 8: Forward Payload Area

3. Experimental results

A preliminary static error analysis was completed for the system. The IVS and conic assembly were mounted on a tripod and placed in a mockup of the competition arena. The room used for the testing was only 32 feet wide instead 48 feet so the world model was modified to reflect the actual placement of the landmarks. Figure 9 shows the setup with three of the lights and the camera visible.

Figure 9: Experimental Setup

Figure 10 is a long exposure image taken to show the conic through the camera. Notice that the field of view of the camera has been adjusted so the conic fills the image. Figure 11 shows an image taken with the camera near the source bin. The image exposure time was chosen to match the current landmark detection algorithm exposure time. Figure 12 shows the ground station console for the vision system. The lower rectangle depicts the arena with the current helicopter position represented by the white dot just to the right of the source bin. The upper rectangle depicts the image that the camera is seeing. The dots show where blobs have been found in the image. The numbers to the right are the list of line intersections that have been computed. Those being used in the final mean calculation are marked with a dot to the left of them. The numbers to the right of the arena report the blob locations and the computed vehicle X,Y location. Figure 13 shows the camera having been moved several feet. Notice the blobs have moved in the image and fewer intersections were computed.

Figure 14 is a graph of the difference between the X,Y location reported by the system and the known X,Y values. Samples were gathered at 36 points in a 3 foot by 3 foot grid along the center line between the bins. This process pointed out the necessity of having accurate yaw information. A rotation of the camera by 10^o was found to cause an error of nearly 3 feet. Most of the error seen in this figure can be attributed to difficulty in keeping the camera heading constant while moving it. The second major source of errors is in the smaller arena. By moving the lights 16 feet closer together, some loss of accuracy is to be expected. This is confirmed by Figure 15 which shows that most of the error is actually in the Y dimension.

Figure 10: Long Exposure Image to Show Conic

Figure 11: Typical Exposure Showing Lights

Figure 12: Vehicle Near Source Bin

Figure 13: Vehicle Between Bins

Figure 14: Graph of Total Static Spatial Error

Figure 15: Graph of X-dimension Static Spatial Error

4. Conclusions

The application of an industrial vision system to an airborne navigation system was successful. The integrated vision system replaced the expected camera, radio link, and ground based vision processor with a small on-board unit. The ability to perform visual processing on-board the aircraft resulted in greater reliability through reduced susceptibility to electromagnetic interference.

The use of local mean thresholding reduced the complete acquisition, digitization and image processing cycle to below 50 milliseconds for typical images. Processing time was dependent on the percentage of the image identified as landmarks, but the worst case cycle time with the entire image identified as a landmark was still less than 100 milliseconds.

The adaptive position determination algorithm proved usable even with uncalibrated optics and unsurveyed landmarks. The redundant triangulation technique is applicable to many other situations requiring position determination. Unfortunately the Georgia Tech helicopter entry in the competition never completed the integration phase. Serious problems with the airframe delayed flight testing to the point that the vision system was unable to be flight tested. Static testing without accurate yaw information in the smaller arena provided sufficient confidence that the system would meet it's ± 0.5 foot accuracy design goal.

Vision based position determination is a viable technique for many situations where contact sensing is not available. Using thresholding techniques to recognize landmarks allows for very rapid vision processing. The use of integrated vision systems results in a self-contained sensing package that can output high level, compact information. This eliminates the need for high bandwidth video links and greatly increases reliability.

5. Acknowledgements

The authors would like to thank Mark Gordon, Steve Dickerson, Russ Clark, and Marc Slack for their aid in developing the computer algorithms used in this project, and Dan Schrage for organizing and leading the Georgia Tech team. The Mitre Corporation, Pacific RPV, Georgia Tech Interdisciplinary Programs Office, the Army Aerostructures Directorate, Vigyan Inc., Guided Systems Technologies, TRON-Tek, Watson Sensors, and the Digital Equipment Corporation deserve recognition for their donations of time, equipment and/or funding for this project. Finally the authors wish to thank all the Georgia Tech faculty and students who devoted so many hours of their time to the UAV competition.

6. References

1. "First International Aerial Robotics Competition," *Competition Announcement,* Association for Unmanned Vehicle Systems, 1101 14th Street, N.W., Suite 1100, Washington, D.C., 20005.

2. Ballard, D.H., Brown, C.M. *Computer Vision,* Prentice-Hall, 1982.

3. Baker, N.C., MacKenzie, D.C., Ingalls, S.A., "Development of an Autonomous Aerial Vehicle: A Case Study," *Journal of Applied Intelligence,* to appear.

4. Baratz, M.R., "The IBM 1975 Optical Page Reader", *IBM Journal of Research and Development,* Vol. 12, Pg. 354-363, 1968.

5. Chow, C.K., Kaneko, T., "Boundary Detection and Volume Determination of the Left Ventricle from Cineangiograms," *Computers in Biomedical Research 3,* 1973, Pg. 13-16.

6. Cooper, D.B., "Maximum Likelihood Estimation of Markov-Process Blob Boundaries in Noisy Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-1, No. 4, Oct. 1979, Pg. 372-384.

7. Haralick, R.M., "The Digital Edge," *Proceedings IEEE Computer Society's Conference on Pattern Recognition and Image Processing,* 1981, Pg. 285-291.

8. Hong, T.H. "Compact Region Extraction using Weighted Pixel Linking in a Pyramid," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-6, No. 2, Mar. 1984, Pg. 222-229.

9. Ingalls, S.A. "Application of Concurrent Engineering Methods to the Design of an Autonomous Aerial Robot", Master's Thesis, School of Aerospace Engineering, Georgia Institute of Technology, Nov. 1991

10. Kittler, J., Illingworth,J.,"An Automatic Thresholding Algorithm and Its Performance," *Proc. Seventh Int. Conf. Pattern Recognition*, Vol. 1, Montreal, 1984, Pg. 287-289

11. Martelli, A., "An Application of Heuristic Search Methods to Edge and Contour Detection," *Communications of the ACM*, Vol. 19, Feb. 1976, Pg. 73-83.

12. Perez, A., Gonzalez, R.C., "An Iterative Thresholding Algorithm for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-9, No. 6, November 1987, pp. 742-751.

13. Sridhar, B., Chatterji, G., "Object Segmentation for Helicopter Guidance," *Proceedings, Applications of Artificial Intelligence X: Machine Vision and Robotics,* Vol. 1708, April 1992, pp. 630-640.