Vehicle Real-time Attitude Estimation System (VRAES)

John N. Sanders-Reed

SVS R&D Systems, Inc.
26 Meadow View Rd., Sandia Park, NM 87047
JNS@SVSINC.COM

ABSTRACT

A Vehicle Real-time Attitude Estimation System (VRAES) is described which uses multiple optical sensors to detect and track moving targets. The pointing and shape information from each optical sensor is combined in real-time to estimate the 3-space location and attitude (orientation) of the target. This paper describes the system architecture, the algorithm, and the real-time implementation. Test results indicate that the system can operate in real-time, on a wide range of data and provide precise estimates of position and orientation.

Keywords: Attitude, Real-time, Position estimation, Tracking, 3D, Triangulation

1. INTRODUCTION

Passive optical sensors, such as telescope mounted cameras, offer several benefits over active sensors for tracking and imaging applications. The benefits include higher resolution imaging, simpler systems, and lower power. Furthermore, passive systems do not broadcast their location and hence can be utilized in a more covert manner. However, when observing distant targets, passive optical systems provide essentially “angles only” pointing information. It is difficult to obtain range information, and hence absolute 3-space position information for a target observed with a single passive optical sensor. Techniques for obtaining range from a single passive sensor fall into two categories. If the target is relatively close, conventional auto-focus based techniques can be utilized. At longer ranges, one can determine range for a resolved target if the target dimensions are known. However, if the target dimensions are unknown, or if the target is sufficiently far away that it is unresolved, then range information cannot be obtained from a single passive optical sensor.

This paper describes a Vehicle Real-time Attitude Estimation System (VRAES) for determining target orientation, in real-time, using multiple ground based optical sensors. The baseline system described here utilizes fixed location azimuth / elevation (AzEl) mounts with various telescopes and electronic camera systems. These front-end systems are placed at known locations in order to detect and track target vehicles in flight. Video imagery from each camera is processed in real-time to perform target detection and feature extraction (shape parameters to be used for location and attitude estimation). The detection reports from each sensor are then passed to a central processor for real-time 3-space position and attitude estimation. While the system can act simply as an attitude estimation enhancement to existing tracking systems, it also has the capability to perform real-time tracking and thus provide a complete processing system.

The proof-of-concept system described in this paper consists of a non real-time workstation version to show overall system capability and accuracy, and a single front-end processor system to demonstrate real-time processing capability. This paper describes the basic algorithm and data flow developed to meet the VRAES requirements, the real-time image processing hardware used in the front end, the test and validation techniques, and results establishing the performance of the system.

ALGORITHM DESCRIPTION

Processing Stages

The VRAES processing chain consists of a series of processing steps, each dependent on the results of the previous stages. The first natural break point in describing the algorithms is to divide the processing into that which is done independently at each sensor, and that which is done at a central processor using inputs from all of the sensors. The individual sensors are responsible for image acquisition, target detection, feature extraction, and tracking. The central processor combines the inputs from the sensors to perform a 3-space position estimation and the attitude estimation (figure 1).

Figure 1. VRAES system diagram

Another way of thinking of the problem is that the sensors and their processors perform the high data rate image handling functions in order to extract a few important image features. These can then be sent over a low bandwidth link to the central processor which operates on a small volume of data from each sensor. The critical features which are passed from the sensor to the central processor are the sensor location, the sensor line-of-sight to the target (target position in azimuth and elevation, also known as the first moments of the target), and the target second moments, corresponding to moments-of-inertia. These will be described in more detail later.

In the following sections, the detection and feature extraction, tracking, 3-space position estimation, and attitude estimation will be examined in more detail.

Detection and Feature Extraction

Automatic Threshold Selection

The highest volume data processing is to extract target information from the raw image data. Since portions of the image near the edges may be obscured or corrupted (e.g. by a bar code along one edge for time stamp and pointing information), a moveable and sizable window is used to allow the operator to select a subregion of the image containing the target for processing. This window can be moved and sized by the operator interactively throughout the processing. The window not only allows the operator to select a region of the image containing only target and nominal background, it also reduces the number of pixels actually used in processing.

The basic detection is performed using a threshold detection algorithm. The algorithm allows detection of both targets above and below the mean background level. In order to operate as a real-time system, the algorithm must incorporate automatic threshold selection. Since the backgrounds can vary significantly from frame to frame, the threshold must be dynamic. In order to determine the threshold, a histogram of the current image is formed. The histogram is a plot in which the x-axis is the pixel grey scale level (e.g. 0 to 255), and the y-axis is the number of occurrences in the image of that pixel value. In almost all cases, even with highly structured backgrounds, the background will form the largest peak in the histogram. The thresholding algorithm finds the peak value of the histogram and assigns the pixel value of the peak as the estimate of the image mean.

The algorithm determines the noise level of the background by assuming a Gaussian noise distribution about the background mean, μ . A Gaussian distribution may be described by $A = A_0 e^{-\frac{(x-\mu)^2}{4\sigma^2}}$, where μ is the location of the peak and σ

is the width parameter. At the point where the amplitude A has fallen to A_0/e^2 , the width is 4σ . Thus the algorithm scans out from the peak value of the histogram in both directions until the pixel frequency is less than $1/e^2$ of the peak frequency. This width is divided by 4 to obtain an estimate of σ . While the assumption of a Gaussian background is often far from accurate (such as in the case of bands of clouds), the approach stills gives a robust estimate of the mean, and a useful estimate of the magnitude of the background variation.

The histogram approach to estimating the background mean and standard deviation is preferred to an arithmetic calculation, both because it is faster and because it is more robust. Consider the case of a large bright object, filling perhaps a third of the image, and a relatively uniform background. The arithmetic mean will be severely skewed upward because of the inclusion of the many bright target pixels. Similarly, the arithmetic standard deviation will be much larger than the true background standard deviation. The histogram provides a simple method of ignoring the target pixels, thus leading to a more robust estimation of the background statistics. Values obtained using the histogram of an entire image have been compared with arithmetic values obtained by selecting a region of the image containing no targets. Agreement to within a few percent was usually obtained.

The operator selects a detection threshold (T) by specifying the number (α) of standard deviations above or below the background mean ($T = \mu \pm \alpha\sigma$). Separate values of α are used for the upper and lower threshold, and the operator may adjust these values interactively during the mission. The default value is 6, corresponding to a 6σ target. The operator may also select to only detect bright targets, only dark targets, or both bright and dark targets.

Detection

Once the threshold is set, the image is scanned for pixels that are above the upper threshold or below the lower threshold. Before declaring a detection, these candidate pixels are compared with a list of bad pixels in the image plane. Once the pixel passes this test it is declared a detection. Adjacent pixels are clustered to form objects. The basic clustering algorithm [1] can distinguish multiple complex shaped objects, such as interdigitated objects. However, in certain instances, particularly the VRAES application, one wishes to force all detections into a single object and compute statistics on this single object. While the workstation version of the algorithm allows either mode of operation, the real-time algorithm always assumes a single object and clusters all detections into one object.

The detection algorithm includes a size threshold which allows rejection of targets with fewer than a specified number of pixels, or more than a specified number of pixels. Again, the real-time implementation is a bit less flexible. The real-time implementation has no upper size limit (not relevant to the VRAES application anyway). However it can reject isolated pixels or small clusters, before including them in the overall detection.

Feature Extraction

The critical target parameters for the VRAES application are the target location (target centroid or first moments), and the target shape parameters used for attitude estimation. The shape parameters used at this point are the second moments [2]. These are the parameters the real-time implementation computes. In addition, the workstation implementation also produces the outline shape of the object, the minimum and maximum extents in the image horizontal and vertical dimensions, the object peak and total intensity, and the number of saturated pixels in the object. The intensity parameters are used for radiometric calibration and signature analysis applications, while the outline shape can be a useful operator aide.

The shape moments (first and second moments) can use either the pixel values for intensity weighted moments, or simply the position, for silhouette weighted moments. The silhouette weighted moments appear to be most appropriate for the VRAES application.

The actual moments are computed in the standard way. The first moments are:

$$x_0 = \sum_{i=\text{target}} x_i I_i / \sum_{i=\text{target}} I_i \quad , \quad (1a)$$

$$y_0 = \sum_{i=\text{target}} y_i I_i / \sum_{i=\text{target}} I_i \quad . \quad (1b)$$

The I_i values are either the pixel intensity (intensity weighted moments) or 1 (silhouette moments). The second moments are:

$$\sigma_{xx} = \sqrt{\frac{\sum_{i=t \text{ arg et}} x_i^2 I_i}{\sum_{i=t \text{ arg et}} I_i} - x_0^2} \quad , \quad (2a)$$

$$\sigma_{yy} = \sqrt{\frac{\sum_{i=t \text{ arg et}} y_i^2 I_i}{\sum_{i=t \text{ arg et}} I_i} - y_0^2} \quad , \quad (2b)$$

$$\sigma_{xy} = \sqrt{\frac{\sum_{i=t \text{ arg et}} x_i y_i I_i}{\sum_{i=t \text{ arg et}} I_i} - x_0 y_0} \quad . \quad (2c)$$

Note that in equation (2c), one must move the sign of the argument of the square root outside the square root. Thus if the argument is negative, take the square root of the absolute value, and then apply the minus sign in order to obtain a negative σ_{xy} .

Both the real-time and the workstation versions of the software are written such that the inclusion of higher moments at a later time will be a trivial modification. This may prove useful for more sophisticated attitude estimation or eventually for object classification.

Position Calibration

Once the object has been detected and the features extracted, position calibration can be performed. Both the workstation and the real-time version convert pixel positions and sizes to angular positions and sizes, using mount pointing information together with pixel size and focal length, or Field-Of-View (FOV) and number of pixels. A setup flag indicates whether azimuth increases clockwise or counter-clockwise, and in fact, the system can operate with an astronomical right ascension and declination (RaDec) mount as well.

The current frame time (used for tracking and position estimation) and the mount pointing information (azimuth and elevation) are encoded in each image frame. For digital imagery, this information can be encoded in a frame header. Alternatively, for analog data such as RS-170 video, the pointing information can be encoded using a bar code along one edge of the imagery.

Frame-to-Frame Correlation and Tracking

The workstation version supports simultaneous tracking of up to 50 targets in the field of view, with some on linear and some on curving trajectories [1]. However, the VRAES application tracks a single target (and in fact forces all detections into a single target), so the real-time implementation forces the track on the single target in the field of view. A track history is maintained, containing the time history for each of the parameters of the object (e.g., first and second moments). This data is passed to an output routine which can either write to an ASCII file or to a communications link.

3-Space Position Estimation

Time Base Alignment

The 3-space position estimation (and the attitude estimation) routine assumes a set of simultaneous observations of the target vehicle, obtained from multiple sensors. If all of the cameras in the VRAES are frame locked, the central processor will receive detection reports from each camera corresponding to observations at the same time. However, in many applications this is not the case. In real-time applications this can occur because the cameras may be free running and not frame-locked (so observations could occur up to 1/2 frame time apart), or the cameras are running at different frame rates. Alternatively, only selected frames from each camera may be processed. For whatever reason, it is often necessary to extrapolate position and shape parameters to a common time base before performing the 3-space position estimation. In this case one camera is specified as the time base, and observations from other cameras are interpolated to the frame time of the time base camera.

For this proof-of-concept demonstration the time-base alignment was run in "analysis" mode using the workstation implementation (because only a single set of front end real-time image processing hardware was available). In this case imagery from each camera was processed separately, and then the data was merged to perform the position estimation. Because the video frame grabbing capability of the workstation could only grab 2-3 frames per second out of the 30 frame per second video stream, and because there was no synchronization between image streams, the alignment problems were much more severe than would be encountered in a real-time system (two image streams could be out of synch by as much as 1/4 second, or 7 frame times). In a real-time system, a simple linear extrapolation routine would probably be adequate for time-base alignment. However, for the proof-of-concept demonstration we chose to utilize a cubic spline interpolation to estimate the position and shape parameters.

The time base alignment is performed using a cubic spline fit of the parameter as a function of time. Thus for example, a cubic spline fit of the azimuth position versus time is performed and the coefficients are stored. The same procedure is repeated for the elevation and for each of the shape moments. The cubic spline coefficients are then used to estimate the position or shape at the time corresponding to frame times of the time base camera. The result is an estimate of the position and shape at the same time from each camera.

Multi-Sensor Least Square Triangulation

Two angles only sensors (such as cameras) observing a common target can be used to triangulate the target position in 3-space. If more than two cameras are used the problem is over specified and becomes a least squares problem. In this case one could weight the contribution of various sensors based on some estimate of the quality of the observation. For our derivation we used a right handed coordinate system (Figure 2), with the azimuth angle θ increasing clockwise from the positive y-axis. The positive z axis, representing positive elevation, is up, out of the paper. The elevation angle ϕ increases in the usual manner.

Figure 2. 3-Space coordinate system

If the subscript t refers to the target, and the subscript i refers to the i^{th} sensor, then the azimuth angle of the target from the sensor is given as:

$$\tan \theta_i = \frac{x_t - x_i}{y_t - y_i} \quad . \quad (3)$$

Equation (3) gives a set of simultaneous equations if multiple sensors observe the target, allowing one to solve for x_t and y_t . The resulting least squares matrix equation is:

$$\begin{bmatrix} \cdot & \cdot \\ x_i - y_i \tan \theta_i & \cdot \\ \cdot & \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ 1 & -\tan \theta_i \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} \quad . \quad (4)$$

This can be solved with standard least squares techniques. The z position of the target can be obtained by averaging the z position estimate from each sensor:

$$z_t = r_i \tan \phi_i + z_i \quad , \quad (5)$$

where the horizontal separation distance r_i is given by

$$r_i = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2} \quad (6)$$

If a target is on the line between two sensors, no position information will be available concerning the location of the target along that line. This is easily seen for the case of only two sensors observing a target. In this case $\tan\theta_1 = \tan\theta_2$ and the matrix in equation (4) has a determinant of zero, or in other words, no inverse.

Attitude Estimation

The information content of an image, or of a target within an image, can be completely described in many ways. The simplest method is simply the image itself, the list of all pixel locations and their values. A common alternative representation is the Fourier representation. The Fourier description can be attractive if one can isolate target features in frequency space and usefully describe the object with a small subset of the Fourier coefficients. In this case the use of a limited frequency range can dramatically reduce the data volume without seriously limiting the information content. This is a key principal behind many lossy image compression techniques. Depending on the application, other features can be used to represent the target image in a much more compact notation than the pixel value enumeration. Target outline maps provide one such representation, but are typically limited to simply the outline information and are subject to degradation in the presence of background clutter or sensor noise. Object moments, like the Fourier description, can be used to provide a complete representation of the target image. However, a truncated set of image moments can be computed much more efficiently than the truncated Fourier description. Thus if the first few image moments provide sufficient description of the relevant image/target features, image moments can provide a very attractive representation [3].

Two approaches to attitude estimation were considered. One approach called for the use of a bank of outline filters, to be matched against observed target outlines. The filter bank could be trimmed to reduce the number of filters that had to be compared with the target outline, based on previous target attitude and velocity vector. The second approach was the use of object shape moments [2, 3-6] for the attitude estimation. The use of the object moments does not require nearly as complete a target detection as the outline filter bank, and hence will work under lower signal to noise conditions. The use of the object moments also requires less a priori target information.

Object moments are easy features to extract from an image and lend themselves to implementation in real-time hardware. Furthermore, moments provide a robust and scaleable approach to object classification or attitude estimation. For these reasons, the object moments approach was chosen over the outline filter bank approach. Our initial investigation has used only the second moments (described previously) for attitude estimation. However, little additional computation is required to compute higher order moments. Thus an easy upgrade path is available to add higher order moments (and hence more target detail) as the need arises. Finally, the use of object moments requires much less a priori knowledge and target specific setup than the use of outline filters. In the case of a missile, no a priori knowledge is required, as demonstrated in the algorithm performance section.

The second moments of an object, as observed from each sensor, are used to generate two points in 3-space, and the angles between these points are computed as pitch and yaw. Two angles only points are generated at each sensor:

$$(x_0, y_0) + (\sigma_{xx}, \sigma_{yy}) \quad (7a)$$

$$(x_0, y_0) - (\sigma_{xx}, \sigma_{yy}) \quad (7b)$$

where (x_0, y_0) is the object centroid, and σ_{xx} and σ_{yy} are two of the three second moments of the object. The 3-space position estimation technique described in the previous section is then used to combine these points from each sensor and generate two 3-space points. The pitch ϕ , is computed as:

$$\tan \phi = \frac{\Delta z}{\sqrt{\Delta x^2 + \Delta y^2}} \quad (8)$$

where the separations are the linear distances between the two moment generated points. The sign of the third second moment, σ_{xy} is used to initially determine the sign the pitch, with a final revision based on the target velocity. If the y component of velocity is negative, the sign of σ_{xy} is assigned to the pitch, while if the y component is positive, the opposite sign is used.

The yaw angle ξ is computed as:

$$\tan \xi = \frac{\Delta x}{\Delta y} \quad (9)$$

Again, the velocity is used to provide a final modification to the yaw. If the y component of velocity is positive, 180° is added to the yaw.

At this time, no estimation of target roll angle is performed. Also, as indicated in the algorithm performance section, the foregoing algorithm works very well with a symmetric object such as a missile. Interpretation and use of second moments as the sole attitude estimation parameters, in the case of an aircraft having wings and tail, presents some problems. In this case, the second moments should be augmented by the use of additional parameters, such as third moments, or the location of extreme points of the object.

REAL-TIME IMPLEMENTATION

The front end sensor processing, consisting of target detection, feature extraction, and tracking requires special purpose real-time image processing hardware in order to achieve the real-time processing requirement. Because it is envisioned that the front end processing will be performed locally at each sensor, it is important that the front end processing hardware be capable of operating in a harsh environment. Since there will be one set of front end hardware per sensor, it is also important that the system not be too expensive or complex.

The front end sensor processing system (figure 3) described here uses a single DataCube MV200 image processing card with a host 68040 single board computer, both residing in a VME chassis. An optional DataCube DigiColor video input board provides additional color to grey scale mapping flexibility to enhance certain targets. This system, based on the commercial Opti-Trak™ 100 real-time video tracker, performs all of the threshold selection, target detection, feature extraction, and track history maintenance functions for a single sensor. In addition, if desired, the system can provide mount control for closed loop tracking.

The frame-by-frame detection reports from each sensor are sent over a communications channel to a central processor which performs the position and attitude estimation. As documented in the algorithm performance section below, real-time performance for the central processor can be achieved using a UNIX workstation (Silicon Graphics Indigo2). Thus the remainder of this section will focus on the high data rate front end processing.

Although the differences between the workstation version of the front end algorithm and the real-time system were noted in the algorithm description section, a brief summary of the real-time front end algorithm is given here for completeness. The automatic threshold selection is identical to the workstation version, using the histogram to estimate the background mean and standard deviation. An operator controlled window is supported for selecting a subregion of the image for actual detection processing. The real-time version supports both dark and bright target selection with independent thresholds. Unlike the workstation version, the real-time detection algorithm clusters all detected pixels into a single target. However, it does have the ability to reject isolated pixels or small clusters. Target features computed in the real-time version include the target moments through the second moments. These include the zero moment for total number of pixels, the first moments for position estimation, and the second moments for attitude estimation. The outline shape and peak intensity are not computed in the real-time algorithm. Unlike the workstation version, the real-time algorithm only supports a track history for a single target. However, the real-time version also includes the capability to provide closed loop tracking by providing error signals to the mount controller. Since the remainder of the algorithm (3-space position estimation and attitude estimation) runs on the central workstation processor, it is the workstation algorithm already described.

The proof-of-concept real-time front end is implemented for a single sensor and contains an X-windows operator control interface, allowing the operator to interactively adjust the bright and dark target thresholds and the detection window size and location, throughout a mission. Depending on the nature of the data and control links between the front end processor and the central workstation, these features may not be available for a field system. In addition to the DataCube MV200 and the 68040 host single board computer, the system used a DataCube DigiColor image acquisition board for input. The primary reason for this was to provide a higher quality image grabber than the one present on the MV200. In addition to the better quality acquisition module, the DigiColor also allows the system to provide a color to grey scale mapping in order to enhance the target characteristics relative to the background. This feature is useful for all scenes and could be particularly beneficial for a cooperative target.

Figure 3. Front-end real-time processing

ALGORITHM PERFORMANCE

Testing Categories and Data

The algorithm described in the previous two sections has been tested for basic performance of all major components and the ability of the components to work together as a system. It has also been tested for precision of the results, for robustness, using a number of data sets, and for execution speed to verify real-time capability. In the following sections, each of these categories will be discussed and results presented.

The primary data used for algorithm verification consisted of two video tapes, one containing field data of real objects, and the other containing simulation data. The field data contained a number of launch sequences and reentry sequences for a missile. Some of these sequences contained a left edge bar code with time and mount pointing information. However, sensor location for the field data was unknown, so the multi-sensor triangulation routines could not be tested with the field data. The field data was used to test the front end detection, feature extraction, and position estimation routines, as well as the bar code interpretation routine.

The simulation data consisted of image data for a missile launch and for an aircraft. There were 6 image sequences for each vehicle, as seen from 3 different sensors. For each vehicle there were three daylight sequences and three night time or fog sequences. The three daytime sequences were simultaneous observations from 3 separate sensors, whose locations were known, thus allowing testing of the triangulation and attitude estimation routines. Similarly, the night time or fog sequences were three simultaneous observations of each vehicle from the three sensors. Time, mount pointing, and actual vehicle 3-space position were recorded across the top of each frame of data using alphanumeric characters. Unfortunately, many of the characters were blurred beyond recognition, leading to some pointing and timing errors. Both vehicles were moving along the line $x=0$, in the $-y$ direction. The missile was launched from the origin ($x=y=0$) and gained elevation throughout the flight, while the aircraft slowly descended throughout the flight. The pitch for the missile was given as a constant 32° and for the aircraft as a constant -9° . No yaw information was given, but based on the flight pattern it was assumed to be 0° for both vehicles.

Testing for the 3-space position estimation and attitude estimation was done exclusively on the workstation, using the simulation data. The workstation was only able to capture 2-3 frames per second from the 30 frame per second (fps) image stream, and images were obtained independently from each image sequence. The workstation was a Silicon Graphics Indigo2 which captured the incoming video as 24-bit red-green-blue image data. The image processing routines operate on grey scale

data so it was necessary to convert the RGB data to grey scale. On the workstation this could be done using any mapping desired, from equal weighting of each component, to the use of a single RGB component. For the simulation data, the red component was used in order to highlight the red alphanumeric display in each image, and to reduce the blue sky background. The real-time DataCube system used the DigiColor board to perform the same color to grey scale mapping function. In the future, more sophisticated color to grey scale mappings can be easily implemented, in order to highlight target features.

Algorithm Precision and Accuracy

Basic single sensor two dimensional position and tracking accuracy have been extensively tested on a wide range of data (primarily collected at the MIT Lincoln Laboratory Experimental Test Site located on White Sands Missile Range, NM). Two types of tests were performed. In one test, a star was imaged in the center of the field of view and then the mount was moved so that the star moved around the focal plane. The computed star position, using the centroid location and mount pointing information, remained constant to sub-pixel accuracy. In fact, it was possible to resolve the mount shaft encoder increments. The absolute accuracy achieved in this test was 1 μ rad. The second test involved tracking several closely spaced objects while the mount was jerking in random directions, so as to cause apparent motion of the objects on the focal plane of up to 1/8 of the field of view. This is a critical test of the angular position calibration routine because a failure to perform precise angular calibration will result in either broken tracks or a cross association between targets. The limiting factor was determined to be mount stiffness, or the precision between actual camera line-of-sight and the mount encoder readings.

The 3-space position estimation and attitude estimation rely both on the accuracy of the single sensor angles only position estimation and the least square triangulation routine. The attitude estimation also relies on the adequacy and accuracy of the shape moments. Position and attitude estimation were performed using simulation data for a missile and for an aircraft. Graphs of position and velocity versus time are shown in figure 4, 5. The target location in the y and z directions was found to be within 2 meters of the simulation "truth" location for essentially the entire flight path. For much of the time the error was less than 1 meter. The x position estimate was found to have considerably more error than the other two dimensions, especially while the target was near the y=0 position. This is a geometry induced error. During this time the target is essentially directly between 2 of the sensors, which thus provide no information concerning the relative distance between the sensors (as discussed following equation 6). These results were obtained despite the errors in timing and pointing information introduced due to the smeared alphanumeric display.

The attitude estimation suffered from the same geometry problem as the position estimation: near y=0 the estimates had significant errors but as the target moved out along the -y axis and the viewing geometry became more favorable, the estimates improved. The missile data gave an estimate of the pitch that, except for the first 2 seconds, was always within 2 degrees of the truth pitch. The cross axis orientation, or yaw, was found to have more error, due to its reliance on the x position estimation. Nevertheless, after the first 6 seconds of flight the yaw was found to be within 10 degrees of the expected 0 degree yaw, and within 10 seconds the error was approximately 5 degrees.

As indicated in the algorithm description section, the aircraft attitude estimation should benefit significantly from the use of additional information, such as the fact that the target is indeed an aircraft, the use of higher order moments, or other shape descriptors. Nevertheless, the yaw for the aircraft was found to be within 5 degrees of the expected 0 degree yaw once the aircraft was out of the poor geometry regime. The pitch was found to be within 10 degrees of the correct truth value for essentially the entire duration. For the latter half of the observations, the pitch was within 5 degrees of truth.

Figure 4. Missile Position and Orientation

Figure 5. Aircraft Position and Orientation

Systems Integration and Completeness

We have processed both field data and simulation data, and demonstrated end-to-end processing. This includes decoding time stamp and pointing information from a left edge bar code in the field data, the ability to produce target history files from each individual sensor, and then bring the separate sensor data together to produce 3-space position and attitude estimates. Determination of the line-of-sight from a sensor to a target using the left edge bar code and the target position within the field of view was performed using field data from a missile test flight (figure 6). The 3-space position and attitude demonstration was performed using simulation data from three simulated sensors. Each image stream was processed and the results used to produce the position and attitude estimates, as described in the previous section.

Figure 6. Line-of-site using bar code pointing data and target position within field-of-view.

The software has been written from the beginning in a modular form [1], such that the processing modules do not need to worry about their data sources and sinks. Thus the introduction of a new data format, with a left edge bar code for time and pointing data, was accommodated by a simple front end decoding algorithm. The software has in the past been expanded to handle 4 new image formats beyond the original format first used with the software. In each case, the addition of a new format was accomplished with less than 8 hours of development work. Similarly, the various intermediate results and final output are simply function calls to the processing modules, and whether the I/O is to an ASCII file or to a UNIX socket connection for direct communication to another process, is invisible to the core routines. Because prerecorded data was used with only a single real-time front end processor, the processing results from each "sensor" had to be stored as temporary files until all of the front-end processing was complete. The modular approach just described allows these temporary files to be replaced with a communications link, without affecting performance of the rest of the system.

Algorithm Robustness

The front end detection and tracking algorithms have been tested in numerous applications over the past several years. They have demonstrated the ability to handle a wide range of targets, backgrounds, multiple target scenarios, and unusual target trajectories. Tested applications include multiple unresolved objects, resolved bird imagery (target is changing shape and direction), protein motion analysis, and of course the current attitude estimation work. Over 100,000 frames of image data have been successfully processed.

The detection and tracking algorithm has also been tested on all of the field data sequences (missile) supplied for this project, both launch and reentry, and has demonstrated the ability to detect and track the objects as long as they have been resolved. Detection and tracking was also successfully demonstrated using all of the simulation data: both the day and night/fog sequences for a missile and for an aircraft. The only data for which the position and attitude estimation could be performed was the simulation data. This was because the simulation data specified the sensor location for multiple views of the same object. The position estimation was demonstrated to be highly accurate for both targets, as described previously, except where the target was directly between two sensors. The attitude estimation for the missile was also demonstrated to be accurate, using only the second moments and no a priori information about the target whatsoever. The aircraft attitude estimation was not as good as might be desired, but it is expected that accuracy can be improved by the use of higher moments, or a priori knowledge about the nature of the target.

Execution Speed

The basic tracker, prior to commencement of the VRAES project, had been demonstrated to work with image data at up to 250 frames per second. The tracker has currently been modified for the VRAES program to include the threshold selection technique described in the algorithm description section, as well as the object moment computation for shape description. The modified tracker has demonstrated real-time frame-by-frame threshold determination, centroid tracking, and computation of the second moments. The operator has the ability to interactively adjust the threshold (CFAR) parameter, as well as the track gate size.

The 3-space position estimation and attitude estimation take place on a central workstation, utilizing the position and shape information from each sensor. The system is currently running on a Silicon Graphics Indigo2 workstation. The 3-space position and attitude estimation for 59 frames of data from each sensor was performed in 0.06 seconds. This included reading the target files from each sensor, from disk, performing the time base alignment between sensors, the position estimation, the attitude estimation, and writing the results out to a disk file. A real-time system which accepts detection reports from sensors over a communications link, and which does not require the full cubic spline interpolation for time base alignment, should execute even faster than the current system. Therefore it appears that the central processor computations can easily be performed in real-time, and that additional sensors could be included with no performance penalty.

SUMMARY

This study has demonstrated the feasibility of a system for the real-time detection, tracking, 3-space position estimation, and attitude estimation of flying targets, using multiple optical sensors. A system architecture and processing algorithms have been developed and demonstrated, both for their ability to operate under a range of target and background conditions, and for their ability to run in real-time. The system described by this work includes not only the basic functions required, but also the ability to perform closed loop tracking. The described system is a highly modular system which can be expanded to meet a variety of needs, including object classification and identification, as well as radiometric analysis.

The basic detection algorithm is a threshold detection algorithm using automatic threshold selection on a frame-by-frame basis, and allowing detection of both bright and dark targets. The 3-space position estimation uses angles only target centroid location data from each sensor, in a least squares triangulation method. The attitude estimation combines angles only target moments from each sensor in order to estimate pitch and yaw of the target. Accuracy of the 3-space position estimation is within 1 meter at a range of 5000 meters, and the attitude estimation routine provides pitch within 2 degrees and yaw within 5 degrees for a missile.

ACKNOWLEDGMENTS

This work was sponsored under Army contract DAAD07-95-C-0117 as a Phase I Small Business Innovative Research (SBIR) project. The author would also like to acknowledge valuable help from Dr. David Gibson of the MIT Lincoln Laboratory.

REFERENCES

1. J.N. Sanders-Reed, Transportable Algorithm TestBed (TATB) - A software architecture for real-time target detection, tracking, and signature estimation, Proc. SPIE, **2468**, April 1995. pp 80-91.
2. M.-K. Hu, Visual pattern recognition by moment invariants, IRE Trans. Inform. Theory, **IT-8**, Feb. 1962. pp 179-187.
3. A.P. Reeves, R.J. Prokop, S.E. Andrews, F.P. Kuhl, Three-Dimensional Shape Analysis Using Moments and Fourier Descriptors, IEEE Trans. Pattern Analysis and Machines Intelligence, **10** 6 Nov. 1988. pp 937-943.
4. A.P. Reeves, The general theory of moments for shape analysis and the parallel implementation of moment operations, Purdue University Tech. Rpt. TR-EE 81-37, Oct. 1981.
5. S.A. Dudani, K.J. Breeding, R.B. McGhee, Aircraft identification by moment invariants, IEEE Trans. Comput. **C-26**, Jan. 1977. pp 39-46.
6. M.R. Teague, Image analysis via a general theory of moments, J. Opt. Soc. Amer. **70** 8, Aug. 1980. pp 920-930.