LA-UR- 97-3329

An Infrared/Video Fusion System for Military Robotics*

Anthony W. Davis Randy S. Roberts
Group ESA-EPE, Mail Stop J580
Los Alamos National Laboratory
Los Alamos, New Mexico 87544

August 5, 1997

1. Background

Sensory information is critical to the telerobotic operation of mobile robots. In particular, visual sensors are a key component of the sensor package on a robot engaged in urban military operations. Visual sensors provide the robot operator with a wealth of information including robot navigation and threat assessment. However, simple countermeasures such as darkness, smoke, or blinding by a laser, can easily neutralize visual sensors.

In order to provide a robust visual sensing system, an infrared sensor is required to augment the primary visual sensor. An infrared sensor can acquire useful imagery in conditions that incapacitate a visual sensor. A simple approach to incorporating an infrared sensor into the visual sensing system is to display two images to the operator: side-by-side visual and infrared images. However, dual images might overwhelm the operator with information, and result in degraded robot performance. A better solution is to combine the visual and infrared images into a single image that maximizes scene information.

Fusing visual and infrared images into a single image demands balancing the mixture of visual and infrared information. Humans are accustom to viewing and interpreting visual images. They are not accustom to viewing or interpreting infrared images. Hence, the infrared image must be used to enhance the visual image, not obfuscate it.

2. Objectives and Approach

The IR/Video Fusion system is designed to provide meaningful information with a minimum of interpretation on the part of the user. The ultimate goal is to seamlessly extend the operator's vision into the far infrared (8 to 14 micrometer) region, as well as to track the observations for the user. These objectives were approached by implementing three primary features into the system: IR switchover, IR overlay, and object tracking.

* This work was funded under DARPA contract AO 8029, Amendment #6.

- IR Switchover: The IR switchover feature is used to provide raw infrared imagery in the event that the color image is lost. Loss of the color image can occur when the scene is darkened, obscured by smoke, or if the color camera is damaged.
- IR Overlay: To provide infrared imagery while viewing the color image, the IR overlay feature causes areas of the color image which correspond to warm areas in the infrared view to be tinted red. This allows warm areas of the image to be highlighted, but the items are still identifiable in the color image.
- Tracking: The final feature of the system is the ability to track warm objects in the field of view. Using the infrared image, the tracking subsystem generates predicted positions for the objects in each frame. These predicted positions are used to classify each observation from frame to frame to produce continuous tracks, allowing targeting and identification.

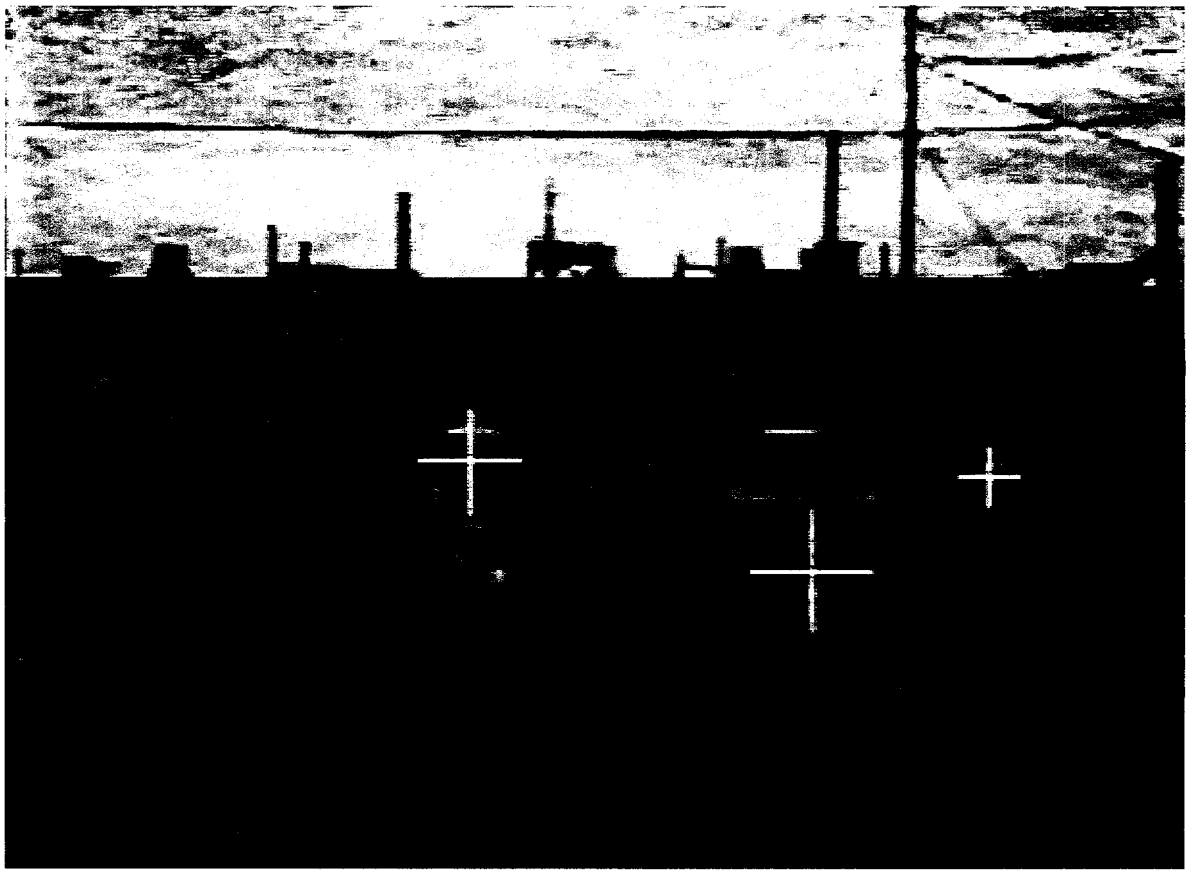

Figure 1. Display of IR Fusion program showing infrared overlay (red tint) and tracking (crosshairs).

3. Hardware

One of the goals during the design of the system was that the hardware should be standard, off the shelf components, and the new features would be implemented in software. The first generation system was developed to assess the camera and processing requirements of such a system, so was built with flexibility in mind.

The far infrared camera used is an Inframetrics model 600 laboratory imaging radiometer (Figure 2). Capable of resolving temperature differences as small as 0.1°C in the 8 to 14 micrometer region, the IR scanner requires liquid nitrogen cooling for the mercury-cadmium-tellurium detector [1]. With the standard lens, the camera has a field of view of 15° vertical by 20° horizontal at a resolution of 200 by 256 pixels. The camera provides a standard RS-170 monochrome video output.

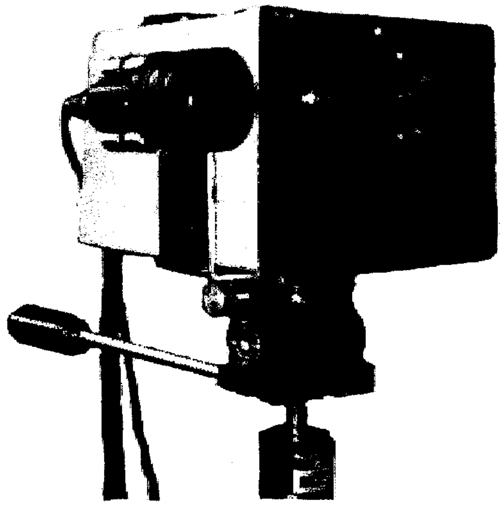
Figure 2. Infrared and Color

For color imaging, a Cohu 8280 remote head CCD camera provides a standard color NTSC output [2]. The imaging head consists of a ½ inch format CCD and a f1.4, 16 millimeter lens. The lens was chosen to match the field of view of the infrared camera (17° vertical by 22° horizontal). The imaging head is mounted as close to the infrared camera as possible to reduce parallax error when overlaying the images. The lens centers of the two cameras are approximately 5.5 centimeters apart.

The video signals from the cameras are digitized by two Matrox Meteor PCI frame grabber cards. These cards take advantage of the high bandwidth of the PCI local bus and provide maximum transfer rates of 42 Mbytes/second [3]. Additionally, the Meteor can push data directly into the video memory of the video output adapter allowing full motion preview of the incoming signals.

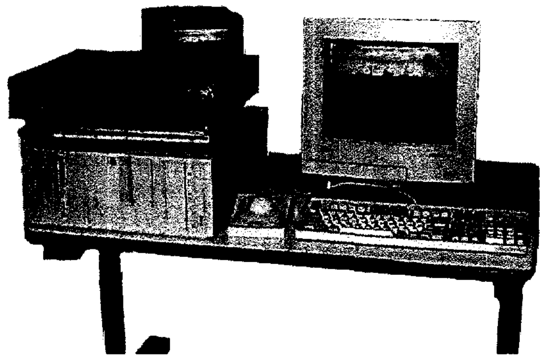
Figure 3. Processing Computer

The current implementation of this system uses a standard Pentium processor running at 166 megahertz for image processing as well as tracking (Figure 3). The computer used is a Micron Millennia Plus with 64 megabytes of RAM running Windows NT 4.0.

4. Software

4.1. Development System

The IR Fusion software was written using the Borland Delphi 2.0 development environment. Delphi, a Pascal based language, allows rapid development of the user interface and is used to make calls to the imaging library. Because Delphi is a compiled language rather than an interpreted language (such as Microsoft Visual Basic), its computational efficiency is great enough to implement the tracking algorithm without degrading overall performance of the system.

The image processing was performed using MIL, the Matrox Imaging Library [4]. This library is a set of Windows dynamic link libraries which can be called from Delphi. The functions in MIL allow control of the digitizer boards as well as basic image processing. MIL can also take advantage of dedicated image processing hardware to improve performance. This can be accomplished by simply recompiling the IR Fusion software for the Matrox Genesis image processing board.

4.2. Image Processing

Once the two images are digitized and available in memory, they are processed to provide the fused image to the operator. Track data from the infrared image is extracted and sent to the tracking subsystem. Five basic operations are performed during image processing: IR Switchover, IR Overlay, Blob Analysis, Automatic Threshold Adjustment, and Image Registration.

4.2.1. IR Switchover

In the event that the image from the color camera is degraded for any reason, the system is designed to shift the display to a raw infrared image. Degradation of the color image can occur when the surroundings are dark or obscured by smoke, or if the color camera is damaged.

A straightforward approach to this problem was employed. Using the MIL functions, a histogram of the color image is created for each incoming frame. The histogram indicates the number of pixels in the picture at each possible intensity (Figure 4). By analyzing the histogram array, the overall contrast of the picture can be determined. A large number of pixels at the same intensity indicates that the picture has low contrast and has probably been degraded (Figure 5). To this end, a simple threshold is used to make the distinction (Table 1). The threshold, set by the user, is typically 3000 pixels in a bin. A 320 by 240 pixel color image has 230,400 pixels in the three color fields. Each pixel has 256 possible intensity values. If any one of the 256 intensities has more

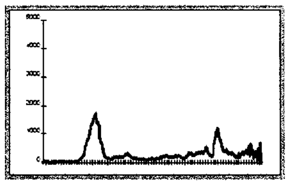

Figure 4. Normal Histogram

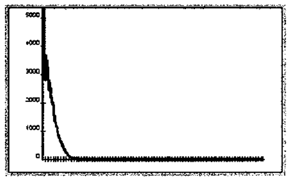

Figure 5. Histogram for a Dark Image

than 3000 pixels assigned to it, the image is identified as degraded, and the raw infrared image is presented to the user in place of the color image with IR overlay. Tracking continues on the infrared data without the color information.

`MimHistogram(ColorImage);`
`MimGetResult(HistArray);`
	Call the MIL Histogram function
	Retrieve the histogram array

`i:=1;`
`bIRView:=FALSE;`
	Initialize the counter
	Initialize the switchover flag

`while (i<= 240) and (bIRView=FALSE) do`	Loop through the array

`begin`
`if HistArray[i]> IRSwitchThresh then`
`bIRView:=TRUE;`
	If any element in the histogram is greater than the threshold, set the Infrared Only flag to true.

`i:=i+1;`
`end;`	

Table 1. IR Switchover Pseudocode.

4.2.2. IR Overlay

During normal operation, the color image is displayed to the user with infrared highlights indicated with red tinting. The infrared image is first shifted by a predetermined amount in both the x and y directions to align it with the color image. The x and y offsets are either set by the user during system calibration, or are determined by the system through an autocalibration algorithm described later. Next, the image is thresholded by setting any pixels with intensity below a user specified level to black (Table 2). The threshold is set by the user by examining the scene and adjusting the value until warm objects are clearly defined and the surroundings are indicated to be cold. The resulting image is then smoothed using a convolve operation and added to the red color field of the color image. The resulting color image is still viewable, but has warm areas indicated with a red tint. Figure 6 shows the two raw images and the fused image. Note the chair is still warm from being recently occupied.

`MimClip(MilImageIR, IRThreshold);`	Set any pixels with values lower than IRThreshold to 0.
`MimConvolve(MilImageIR, M_SMOOTH);`	Smooth the resulting image using M_SMOOTH kernel: $\begin{pmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{pmatrix}/16$
`MimArith(MilImageIR, 2, M_DIV_CONST);`	Reduce the intensity of the image by half.

`MimArith(MilImageColorRed, MilImageIR,`
`M_ADD+M_SATURATION);`
	Add the image to the red field of the color image.

Table 2. IR Overlay Pseudocode.

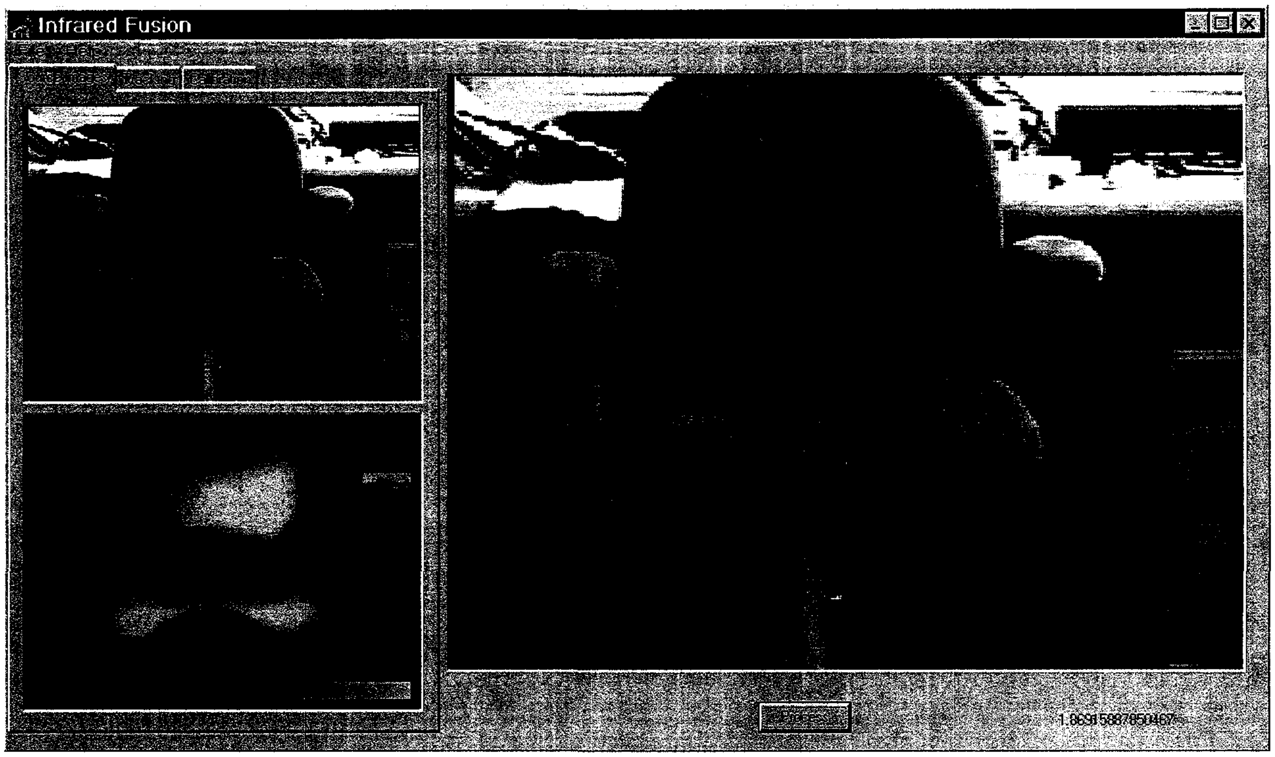

Figure 6. Main Program showing raw color (top left) and infrared (lower left) images, as well as resulting fused image (right). Warm areas indicate the chair was recently occupied.

4.2.3. Blob Analysis

In order to generate a series of points for the tracking algorithm, the infrared image must be segmented into the separate objects of interest. The coordinates of the centroids of these objects can be passed to the tracker as incoming observations.

`MimBinarize(MilImageIR, M_GREATER, IRThreshold)`	Binarize the image using the same IR Threshold as in the IR Overlay section.
`MimOpen(MilImageIR, Repeat)`	Perform an open operation (erosion followed by a dilation) 'Repeat' times.
`MblobCalculate(MilImageIR, MilBlobResult);`	Call the MIL blob analysis function.
`MblobSelect(MilBlobResult, M_DELETE, M_AREA, M_LESS, 10);`	Remove blobs with an area of less than 10 pixels.
`MblobGetResult(MilBlobResult, M_CENTER_OF_GRAVITY, COG);`	Retrieve the centroids of the blobs for use in the tracking algorithm.

Table 3. Blob Analysis Pseudocode.

Blob identification is performed in several stages, the first being the binarization of the infrared image. Each pixel of the infrared image is set to either black or white using the same threshold as in the IR overlay algorithm. Next, the binary image is eroded and dilated to reduce the interconnections between adjacent blobs (Figure 7). The number of times the image is eroded and dilated can be set by the user, and is typically five times. Finally, the Matrox blob analysis library is used to identify and catalog distinct white areas of the image. The resulting array of blob centroids are passed to the tracking algorithm for analysis.

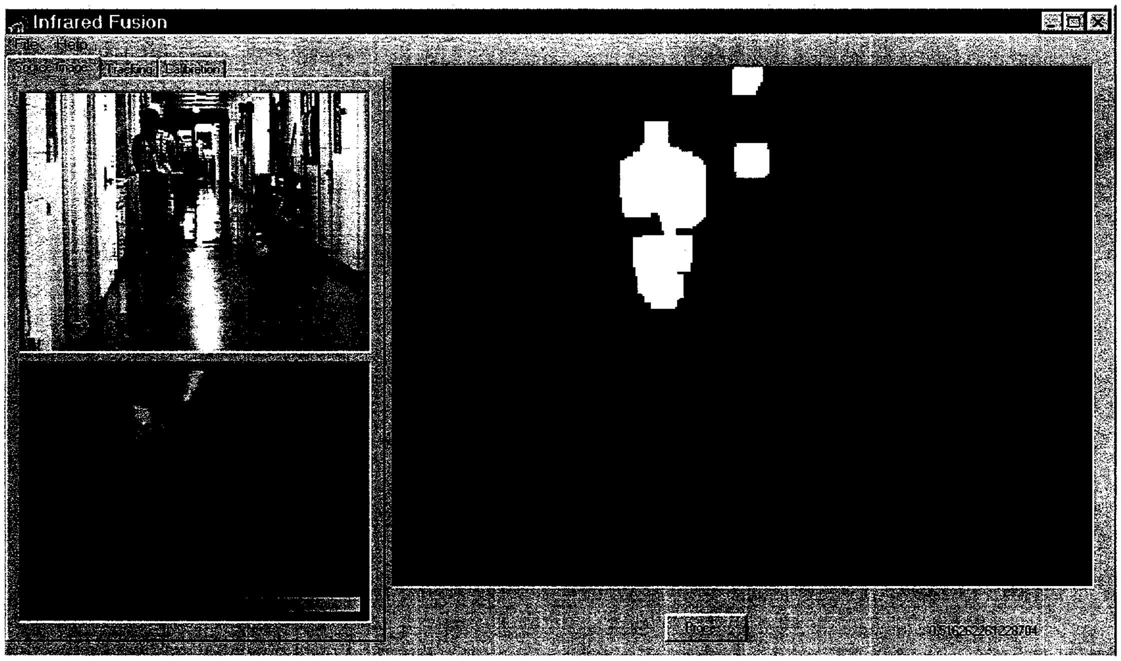

Figure 7. In this special mode, the IR Fusion program shows the internal binarized image used by the blob analysis functions. Note that the person is visible as a single blob, and the warm lights and window are also separate blobs.

4.2.4. Automatic Threshold Adjustment

The threshold used by the IR Overlay algorithm and the Blob Analysis algorithm for determining areas of interest in the infrared image is typically set by the user, and can be varied at run time. Unfortunately, proper adjustment, while critical for stable operation, is often difficult to achieve and maintain. The set center temperature an infrared camera may drift over time, or the ambient temperature may change requiring recalibration. If the threshold is set too low, features in the infrared image may become enlarged and merge together, or ambient noise may cause spurious observation points to be fed into the tracking system. If the threshold is too high, objects in the infrared image may not be observed. For these reasons, a method for automatically setting this threshold is desired.

The recalibration routine (Table 4) is designed to be executed by the user periodically when the system is suspected to be miscalibrated. The function first retrieves the active tracks from the tracking algorithm and places black boxes over any areas designated as active tracks. The remainder of the image is then assumed to be background clutter, and the threshold is set slightly above the average peak. This is accomplished by finding the histogram of the background clutter and setting the threshold to the point where 90% of the pixels have lower intensity.

`TrackObject.GetOfStatus(ActiveT, ActiveArray);`	Retrieve the active tracks from the track object.
`MGraRectFill(MilImageIR, ActiveArray);`	Place filled black rectangles over each active track.
`MimHistogram(MilImageIR);`	Create the histogram for the resulting image.
`MimGetResult(HistArray);`	Retrieve the histogram results.
`i:=1;`	Initialize the counter.
`totalCount:=0;`	Initialize the sum.
`while (totalCount <69120) and (i< BINS) do`	Loop until 90% of the pixels (69120 of 76800) have been counted.
`begin`	
`    totalCount:=totalCount+HistArray[i];`	Add to the count.
`    i:=i+1;`	Increment the counter.
`end;`	
`IRThreshold:=i;`	Set the threshold to the point where 90% of the pixels were found.

Table 4. Automatic IR Threshold Adjustment Pseudocode.

4.2.5. Image Registration

Registration between the color and infrared images is critical to maintain the seamless fusion of the two into a single coherent image. The color and infrared cameras were selected to have very similar fields of view(IR: 20° horizontal by 15° vertical, Color: 22° horizontal by 17° vertical), so only a simple x and y offset is required to align the images. This is typically accomplished by the user when initially calibrating the system. With the infrared image overlaid on the color image, a warm object can be used to observe misalignments. Once aligned, the images should remain aligned unless the cameras are physically reoriented with respect to each other.

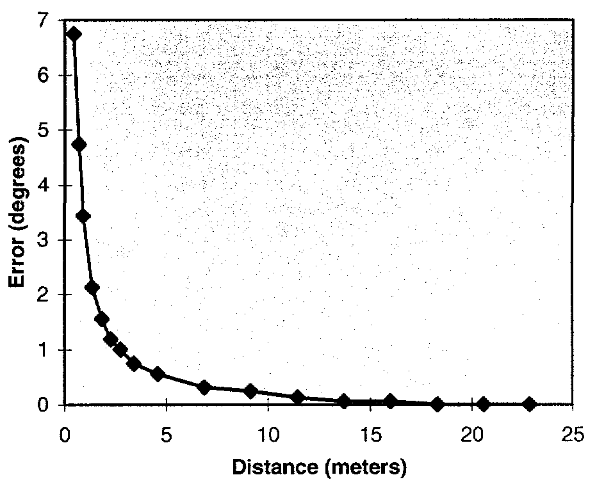

Figure 8. Parallax Error as a Function of Distance

In addition to physical misalignment, parallax error can cause the images to become misregistered. An experiment was conducted to determine parallax error as a function of distance. A small heat and light source was used to calibrate the system at a distance of 30 meters. Then the point source was moved closer to the camera system and the error (in degrees) between the color and infrared cameras was measured. As expected, the error exhibits a inverse relationship with distance (Figure 8). Curve fitting techniques allowed precise measurement of the distance between the two lenses, which is 5.4 centimeters. With this information, the far field of this prototype can be assumed as greater than 5 meters.

To assist the user in calibration, a prototype automatic alignment algorithm was developed using the Matrox pattern recognition library (Table 5). The function, still under development, selects the largest blob in the infrared image, performs an edge detection operation on the image bounded by the blob, and attempts to locate a matching image in the color image. An edge detection operation is also performed on the color image, and thus the pattern matching function matches the edges of the infrared image to the edges of the color image. The returned coordinates of the target match in the color image is used to update the x and y offset values for the program. This function has been shown to be very effective with simple, well defined objects, but is less useful with complex surroundings.

`SelectLargestBlob(sxoff, syoff);`	The function first selects the largest blob using the previously described blob analysis functions.
`MimEdgeDetect(MilColor, M_SOBEL);`	The edges are detected in the color image using the M_SOBEL kernel x and y gradients: $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$
`MimConvolve(MilImageIR, M_SMOOTH);`	The IR image is smoothed.
`MimEdgeDetect(MilImageIR, M_SOBEL);`	The edges in the IR image are detected using the M_SOBEL kernel.
`MimBinarize(MilImageIR, M_GREATER, 70);`	The IR image is binarized with a fixed threshold of 70 (of 255).
`MimBinarize(MilImageColor, M_GREATER, 100);`	The color image is binarized with a fixed threshold of 100.
`MimConvolve(MilImageColor, M_SMOOTH);`	The color image is smoothed.
`MpatAllocModel(MilModel, MilImageIR, sxoff, syoff);`	The IR image around the largest blob is set as a model for the Matrox pattern recognition library.
`MpatSetAcceptance(MilModel, 20);`	The acceptance is set to 20 (in arbitrary units.)
`MpatSetPosition(MilModel, sxoff, syoff);`	An initial search position is set to the position of the largest blob in the IR image.
`MpatFindModel(MilImageTempl, MilModel, MilModelResult);`	The Matrox library is used to find the model in the color image.

Table 5. Auto Alignment Pseudocode.

4.3. Tracking

The multiple target tracking (MTT) system implemented in the IR Fusion prototype is based on a classical approach presented in [5] and simplified in [6]. Classical tracking theory breaks the tracking problem into four stages: Gating, Correlation, Trackfile Updating, and Prediction. After every image frame is processed, the coordinates of the observed blobs as well as the estimated measurement variance are passed to the track object, which associates each observation with its corresponding track, predicts the next observation for each track, and returns the ID numbers for the track corresponding to each observation. The tracking algorithm was implemented as an object in Delphi, and was first tested in a simulator before being applied to the IR Fusion main program.

4.3.1. Gating

The first step in updating the trackfile is to assign any observations to tracks based on some simple rules. Surrounding each track's predicted position, a variance box is calculated with a width of three standard deviations. A track's predicted position and standard deviation are calculated by the Filtering and Prediction stage in the previous iteration. If an observation is the only observation to fall within a gating box of a single track, it is immediately assigned as the next observation for that track. If an observation falls within the gating box of two tracks, it is assigned to the track which has that observation as the only observation, if such a track exists. If multiple observations reside within a one or more tracks, the observations are ignored until the next stage. Observations falling outside of any track gates are designated as new tracks (Figure 9).

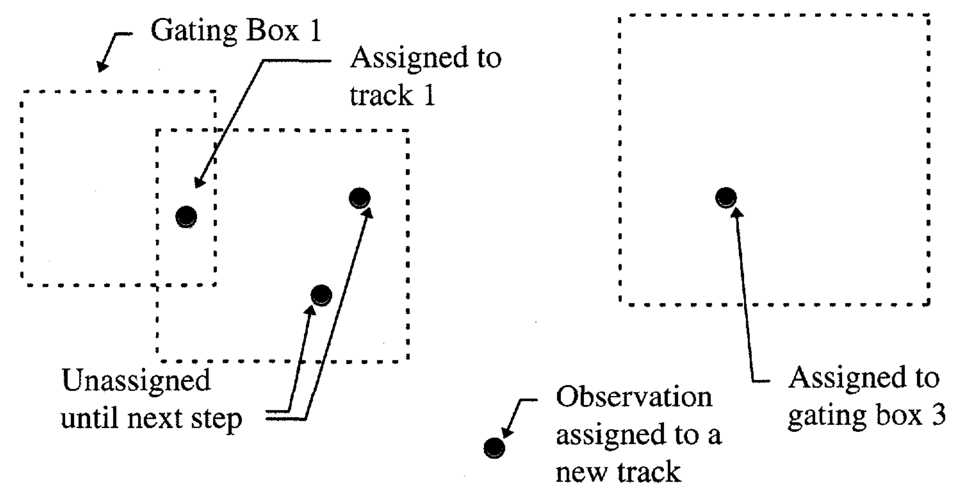
Gating Box 1
Assigned to track 1
Unassigned until next step
Observation assigned to a new track
Assigned to gating box 3

Figure 9. Illustration of Observation Gating

4.3.2. Correlation

Any observations remaining unassigned to a track will require a more careful analysis to determine the proper track to which it should be assigned, if any. If more than one observation is found in any track gate after the Gating is completed, the observation which is closest to the track's predicted position is assigned to the track, and the other observations remain unassigned. Likewise, if an observation lies in the gates of two tracks, the observation is assigned to the track with the closest predicted position.

The distance measurements are made using a normalized statistical distance given in [6]:

$$D = \frac{1}{N}\sum_{i=1}^{N}\frac{\left(X_T(i) - X_C(i)\right)^2}{\sigma_T^2(i) - \sigma_C^2(i)} \qquad \textbf{Equation 1}$$

where $X_T(i)$ is the i^{th} variable from a track, $X_C(i)$ is the i^{th} variable from an observation, N is the number of variables (N=2 in this case, x and y), and $\sigma_T^2(i)$ and $\sigma_C^2(i)$ are the corresponding variances.

4.3.3. Trackfile Maintenance

Once the observations have been assigned to their corresponding tracks, each track status can be evaluated. New tracks are assigned a status of 'Tentative', and will remain 'Tentative' until observations are observed along the track in three consecutive frames. Once a track has been observed in three frames, it is promoted to 'Active' status. 'Active' tracks which are not seen in a frame are given 'Extrapolate' status. An 'Extrapolate' track which is later observed is re-designated as 'Active'. An 'Extrapolate' track which is not seen for five frames is demoted to 'Dead'. In this manner, the state of each track can be evaluated by the calling program.

4.3.4. Filtering and Prediction

The final step in the tracking loop is to predict the next position of each track. This is accomplished using a Kalman filter, specifically the filter described in [5] Section 3.1. The estimated position of each track is represented by two state vectors, generically denoted as $\hat{x}(k|k)$. The vector $\hat{x}(k|k)$ is a 2 by 1 column vector where the first entry is x, or y, and the second entry is the corresponding time rate of change $\dot{x}$ or $\dot{y}$. The Kalman filtering equation used for this application is

$$\hat{x}(k|k) = \hat{x}(k|k-1) + \boldsymbol{K}(k)\left[y(k) - \boldsymbol{H}\hat{x}(k|k-1)\right] \qquad \textbf{Equation 2}$$

where $\boldsymbol{K}(k)$ is the Kalman gain at time k, $\boldsymbol{H}$ is the measurement vector given by $\boldsymbol{H}=[1,0]$, and $\boldsymbol{y}(k)$ is the observation vector. The Kalman gain equation is

$$\boldsymbol{K}(k) = \boldsymbol{P}(k|k-1)\boldsymbol{H}^T\left[\boldsymbol{H}\boldsymbol{P}(k|k-1)\boldsymbol{H}^T + R_C\right]^{-1} \qquad \textbf{Equation 3}$$

where $\boldsymbol{P}(\cdot|\cdot)$ is the estimator covariance matrix, and R_C is the variance of the observation. The filtered covariance matrix is computed by

$$\boldsymbol{P}(k|k) = [\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}]\boldsymbol{P}(k|k-1) \quad \textbf{Equation 4}$$

where $\boldsymbol{I}$ is the identity matrix. Finally, the predicted state vector and covariance matrix are computed by

$$\hat{\boldsymbol{x}}(k+1|k) = \Phi\hat{\boldsymbol{x}}(k|k) \quad \textbf{Equation 5}$$

$$\boldsymbol{P}(k+1|k) = \Phi\boldsymbol{P}(k|k)\Phi^T \quad \textbf{Equation 6}$$

The state transition matrix for both x and y is

$$\Phi_x = \Phi_y = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad \textbf{Equation 7}$$

where T is the time between frames.

This model predicts straight line paths based upon the incoming observation data. Unfortunately, as the track advances, the computed variances become smaller and smaller, to the point that any stray acceleration will cause incoming observations to fall outside the gating box of the track, and the track will be lost. In actual practice, a simple position and velocity model is insufficient to model the behavior of objects moving in front of the infrared camera. A method for compensating for this is described in [7]. The unpredictable acceleration is simulated by adding 'state noise' to the estimator covariance matrix. This effectively limits the minimum size of the gating box, and permits a more flexible model. Thus, Equation 6 changes to

$$\boldsymbol{P}(k+1|k) = \Phi\boldsymbol{P}(k|k)\Phi^T + \boldsymbol{Q} \quad \textbf{Equation 8}$$

where $\boldsymbol{Q}$ is the 'state noise'.

5. Future Research/Applications

The prototype IR Fusion system brought to light many hardware and software problems which must be solved in order to produce a practical tool for surveillance applications. Additionally, as development progressed, many potential applications for the system were discovered.

5.1. Hardware

One difficulty with the prototype system was the difficulty of processing the incoming frames rapidly enough for a smooth display. All image processing was performed by the host computer's main processor, which is not optimized for such an application, resulting in frame rates of approximately 3 frames per second. A better approach is to move the image calculations to a dedicated image processing card installed in the host computer. The Matrox Genesis card is an image processing card based on the Texas Instruments TMS320C80 digital signal processor, and could replace both frame grabbers. Additionally, since MIL supports the Genesis, only minor software revisions would be necessary to implement the external processor.

The Inframetrics Model 600 infrared camera used in the prototype system has several drawbacks which make it undesirable for a most implementations. The camera requires liquid nitrogen cooling, is large and heavy, creates acoustic noise with the scanning mirror, has high power consumption, is relatively fragile, and has low resolution. The camera identified as a possible replacement for the Model 600 is the Amber Technologies Sentinel (Figure 10). The Sentinel weighs only 4.2 pounds and consumes only 6 watts of power [8]. The focal plane of the Sentinel is uncooled, and requires neither liquid nitrogen nor a Sterling cooler. Amber Technologies also offers the Sentinel in component form, without the external case. This allows the imaging array to be mounted separate from the camera control unit, and permits mounting of the color camera in the same case with the infrared camera. The Sentinel operates in the 8-12 micrometer range, and has a resolution of 320 by 240 pixels.

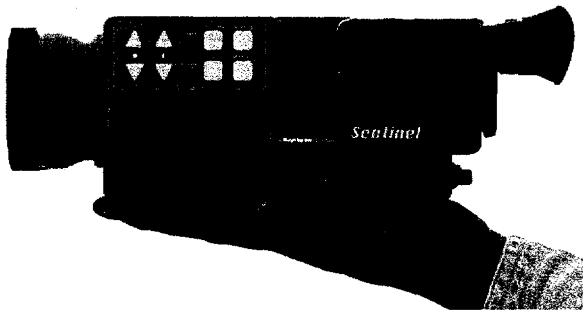

Figure 10. Amber Sentinel

5.2. Algorithms

The algorithms used in the prototype system illustrated the fundamental soundness of the IR Fusion concept, but some improvements are needed before the system could be fielded. Areas for improvement include infrared segmentation and the tracking algorithm.

Segmenting the infrared image is currently accomplished using a fixed threshold which is either set by the user or set by the automatic calibration routine. Unfortunately, this method is very susceptible to maladjustment of the threshold. Even when the threshold is well adjusted, the system occasionally suffers from blob splitting and blob coalescing. Images of people are sometimes split into two blobs, the torso and the legs (Figure 11). In later frames, the image may be recombined. The oscillation between combined and uncombined blobs will cause poor performance in the tracking algorithm. A better algorithm might use a form of adaptive thresholding.

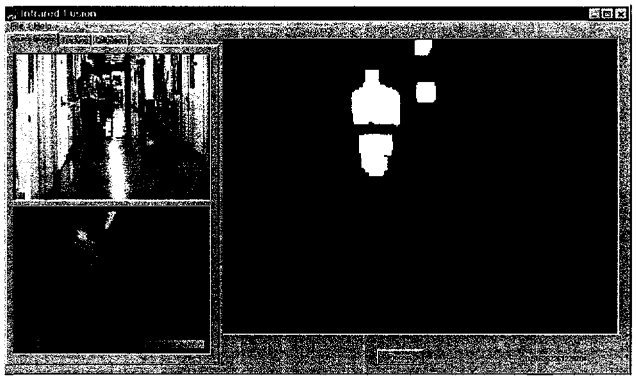

Figure 11. IR Fusion program showing blob splitting. The blob formed for the person has been split between the torso and legs.

The simple position and velocity model used by the tracking algorithm does not well model the real world inputs to the system. The method currently used to modify the model is to add state noise to the covariance matrix, which causes the model to be more flexible. Future work could extend the model to include other variables, such as acceleration and blob size. The addition of range information to the system could also greatly enhance the performance of the tracking system.

5.3. Applications

Many different applications have been proposed for this system beyond the original design for telerobotic sensing. Among these applications are fixed surveillance, military and police aircraft surveillance, and personal military field use.

In a fixed surveillance application, the system could be set up to watch a guarded perimeter and alert the operators to any moving targets in that area. The target could be photographed by the secondary camera, either a normal color camera as in the prototype system, or a low light video camera. The data could then be presented to the operator in a clear manner, and coordinates of the track could be fed into other systems for further use. This is an improvement over current systems which require constant supervision by the user.

Infrared imagery has long been used in military and police surveillance. A special problem presented by aircraft surveillance is that of tracking targets on the ground while the aircraft is in motion. For instance, police helicopters are often tasked to search for people on the ground with the infrared camera and illuminate them with a gimbal mounted searchlight. Holding the searchlight on target while the aircraft is in motion is extremely difficult. Additionally, imagery acquired using the infrared camera is inadmissible in courts of law because they do not operate in the visual range. A color camera with a telephoto lens could be attached to the searchlight and record the actions of people on the ground. The operator of this system would be presented with a series of tracks to select from, and upon selecting a track, the searchlight and color camera could track that object.

Recent reductions in power consumption of cameras and processing equipment has made it possible to reduce the size of the IR Fusion system to a unit capable of being carried in a backpack into the field. Such a system could be used to secure a perimeter in the field rapidly, or to monitor movements of people or equipment in sub-optimal conditions. The unit could be made to provide an overlaid display in a set of goggles worn by the user.

6. Acknowledgments

The authors would like to thank Steve McBurnett (DARPA) and Janine Fales (ESA-EA) for their support of this project.

7. Reference

[1] Inframetrics, *Inframetrics Model 600 Operations Manual,* Bedford, MA, 1983. http://www.inframetrics.com.

[2] Cohu Incorporated, *Installation and Operation Instructions: 8280 NTSC and 8380 PAL Series Composite Output and YC Output Camera Head and Camera Control Unit,* San Diego, CA, 1995. http://www.cohu.com/cctv/index.html.

[3] Matrox Image Processing Group, *Meteor Installation and Hardware Reference,* Matrox Electronic Systems, Ltd., Mooers, NY, 1997. http://www.matrox.com/.

[4] Matrox Image Processing Group, *Matrox Imaging Library User Guide,* Matrox Electronic Systems, Ltd., Mooers, NY, 1997. http://www.matrox.com/.

[5] S. S. Blackman, *Multiple-Target Tracking with Radar Application,* Artech House, Norwood, MA, 1986.

[6] R. S. Roberts, *Radar and IRST Models used in the Distributed AWACS Project,* Los Alamos National Laboratory, LA-UR-97-3060, NM, 1992.

[7] S. Grewal and P. Andrews, *Kalman Filtering: Theory and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[8] Amber, *Sentinel, The Uncooled IR Imaging Advantage,* Amber, A Raytheon Company, Goleta, CA, 1997. http://www.amber-infrared.com.