Automatic Rapid Updating of ATR Target Knowledge Bases

by
Barton S. Wells
Frederick L. Beckner

Final Report on Contract DAAH01-99-C-R077
Cyberdynamics, Incorporated

June 17, 1999

U.S. Army Aviation and Missile Command
Redstone Arsenal, Alabama

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	17 Jun 99	Final 17 Dec 98 - 17 Jun 99

4. TITLE AND SUBTITLE
Automatic Rapid Updating of ATR Target Knowledge Bases

5. FUNDING NUMBERS
C DAAH01-99-C-R077

6. AUTHOR(S)
Barton S. Wells
Frederick L. Beckner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Cyberdynamics Incorporated
1860 Embarcadero Road, Ste. 155
Palo Alto, CA 94303-3362

8. PERFORMING ORGANIZATION REPORT NUMBER
CYB-9901

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Aviation & Missile Command
AMSAM-AC-RD-A
Redstone Arsenal, AL 35898-5200

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The feasibility of creating a software system to perform automatic rapid updating of ATR target knowledge databases is investigated. Methods of comparing infrared images with CAD model renderings, including object detection, feature extraction, object alignment, match quality evaluation, and CAD model updating are researched and analyzed. A GUI-based software application created to demonstrate object detection, feature extraction, and automatic image alignment is described. An improved method of edge detection based on directional masks and second derivatives of pixel intensities is given. A conceptual software system for rapid updating of ATR knowledge databases is described. A technique for the comparison of images based on line features is discussed. Based on this research it is found that the development of an automatic rapid updating system is feasible.

14. SUBJECT TERMS
edge detection, image matching, match quality metric, CAD model rendering, image alignment, object detection, masks, feature extraction, line features

15. NUMBER OF PAGES
27

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Automatic Rapid Updating of ATR Target Knowledge Bases

Overview

This report contains the results of a study of the feasibility of implementing an automatic rapid target updating system (ARTUS) for use with Army missile guidance systems based on 2D infrared target images. These systems operate by matching the infrared image from a missile guidance sensor with predicted infrared images derived from a database of CAD models of a number of different possible targets. Such a system can be adversely affected by variability in the target geometry due to the presence of externally carried objects such as fuel tanks, supply crates, etc, not contained in the CAD model. The adverse effects of such objects could be minimized if there were a way to rapidly modify the CAD model to reflect the presence of such objects based on images obtained from reconnaissance sensors, in effect tailoring the CAD model to match specific targets.

The feasibility of creating such a software system was studied by identifying the different required steps, and by studying the feasibility of each step. The required steps are:

a) to automatically locate a target in an infrared image,
b) compare it to a selection of CAD models,
c) determine which CAD model most closely matches the target in the image,
d) determine the differences between the target in the image and the selected CAD model, and
e) modify the CAD model appropriately to match the target in the image as best possible.

A software application, titled Ctr (for Cyber Target Recognition) was created to test these ideas.

The studies of the feasibility of this goal involved: reviewing existing in-house object detection algorithms, collecting CAD model and infrared imaging data, creating software to read these data, creating algorithms to view these data, creating image processing algorithms to detect objects, aligning these objects from within each type of image, investigating techniques for image match quality

measurement and rapid updating of the CAD models to reflect differences between the models and infrared images.

An ARTUS system was deemed to be feasible using techniques investigated during our Phase I work. It was found that object detection could be performed satisfactorily on the infrared image data and on renderings of the CAD models. Line features were able to be found in both images, and could be visibly correlated. A technique adapted from other work at Cyberdynamics was conceptually designed to align and measure the match quality of infrared images with CAD model renderings, as well as a method to find the differences and rapidly update the CAD models to reflect these differences.

Phase I Work

The following is a list of the work accomplished during Phase I. In this phase we:

1. Reviewed in-house object detection research. Minor research had been done in the past within Cyberdynamics on object detection within color photographs. Some of the general principles had carry-over to this project, though a lot of the specifics researched with respect to color photographs needed to be adapted to work with infrared images.

2. Collected both CAD model data (in Integrated Flight Simulation (IFS) and Alias Wavefront file formats) and image data (in an Army binary format and in SEQB format). The data were received from the U.S. Army Aviation and Missile Command in Redstone Arsenal, Alabama. The Army desired that we concentrate our efforts on using the IFS CAD models, and we received seven CAD models in this format. Only one, however, contained temperature information and was therefore the most useful for many of our efforts. The SEQB image data we received was one file that contained eight infrared images that were not in sequence.

3. Adapted existing CyberCAD software to be able to read all of these formats in a limited fashion. This provided a testing ground to be sure we could read the data and to understand the quality of the data we had received.

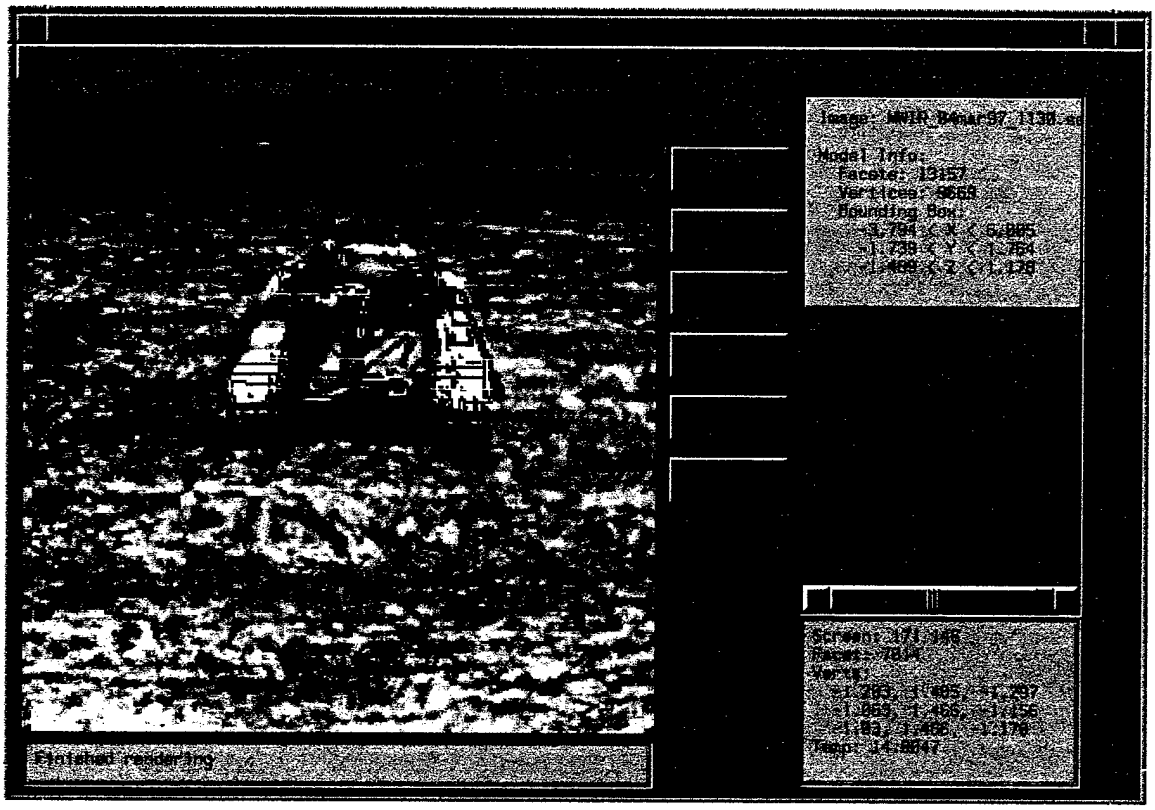


Figure 1: Main window from Ctr, showing a CAD model overlaid on top of an infrared image.

4. Created a new software application, Ctr, that could read all these file formats fully. The software could also automatically detect what type of file was being opened, and open the proper type (CAD or image) and in the proper format. Figure 1 shows the main screen of Ctr with a line image predicted from a CAD model overlaid on top of an infrared image.

5. Created a way to view the image files in either gray shades or in color, where the different hues represent different temperatures. Created ways to zoom in or out. Created user-friendly ways to step through the sequence of images contained within SEQB files.

6. Created edge detection algorithms to use on images. We were instructed to concentrate on the images in the SEQB format, as that was the format that the Army would be using for most of their work. We worked on image processing techniques, such as contrasting, normalizing, eroding, and dilating to improve the edge detection. We researched and used some more advanced edge detection techniques, especially those invented

by Prewitt and Sobel. We also created an extension to normal edge detection techniques that not only looked at the rate of change of the color of a pixel, but also looked at the rate of rate of change to combat curved surfaces, that may be changing in shade quickly, but constantly.

7. Created line detection techniques that attempted to find lines in the patterns of dots created from edge detection runs. These lines need not be straight, but could not vary too randomly. This procedure proved very successful.

8. Implemented an interactive way for the software user to select the intended target, by clicking on it with a mouse. All of the target's pixels were then automatically selected.

9. Created a method of rendering the CAD models using both normal illumination technique, and by using the temperature data imbedded within the CAD model. The temperature rendering used a Gouraud interpolation method to find the temperature value at any point within a facet, based on the temperatures at each vertex of the facet. The rendering was done by creating a z-buffer rendering engine, using only software techniques (some hardware 3D graphics functions do not preserve dimensions properly, leading to photo-inaccurate renderings). We created a way to display these renderings, allowing the user to zoom in or out on the rendering. Implemented a way for the user to click the mouse on the CAD model and obtain information about the model in a text window, such as location and temperature at the point clicked upon. We created a way for the user to select any view location, view direction, magnification, or view twist to view the CAD model from.

10. Created a method of rendering the CAD models in the overlay plane so that the model could be seen in comparison with the image target that it was supposed to represent. A hidden line algorithm was implemented, but this just cluttered the screen and the image could not be seen underneath the CAD model. Next, an algorithm that only showed the lines separating physical features of the CAD model was implemented. This proved effective in allowing the

user to see both the CAD model and the image simultaneously. However, since the image was an infrared image, we then created an algorithm that rendered the CAD model by showing lines that separated areas of significantly different temperature. This allowed direct comparison of the image and the CAD model. For further investigation, and algorithm that rendered the model with isothermal lines was created and implemented.

11. Investigated a method of automatic alignment of the CAD model with the image's target. The algorithm took the model and adjusted the viewer's direction of view, viewer's position relative to the CAD model, the magnification, and the view's twist angle, and perturbed them to bring the model and the image into alignment with each other. This technique required that the software user only approximated the position and view direction of the CAD model observer, and the alignment algorithm would do the rest. This was successful part of the time, but was often fooled by differences in the image's target and the CAD model, such as a different gun position or different fuel barrels loaded onto the target.

12. Researched the effects of running the image through high-pass and low-pass filters in order to automatically find the intended target within an image. Low-pass filters proved effective in discovering where most of the energy in an image was coming from, which in the cases of our data, was the intended target. From there, lines were only selected if they were within a small neighborhood of the focused energy found from the low-pass filter. This technique was successful, but further improvements would help focus on the correct lines in the future.

13. Researched ways of using high-pass filters to improve the edge detection. Though seemingly promising at first, a satisfactory implementation was not found. Further research may yield positive results from this idea.

14. Observed the effects of creating a collection of lines from the image and compared them to the lines created from the overlay rendering algorithms. The lines match closely with the temperature-rendering

algorithm. Researched methods of using the lines to automatically align the image with the CAD model, possibly combining with earlier methods of automatic alignment.

15. Investigated techniques for image match quality investigation. Created a conceptual algorithm that looks at each image as a set of line features, looks for matching line features in another image, and finds the optimal alignment vector based on the best alignment of each of the line features alignment with its counterpart in the other image.

Review of Existing Object Detection

Existing software that Cyberdynamics, Inc. has created for its own purposes lent some experience to the effort of object detection. The software was used to run an edge detection algorithm, and then to use operator interaction to manually edit the edge detection to make sure the entire object was enclosed within lines, and then manually select which enclosed regions belonged to the desired object. This existing software had been developed for use with high-resolution color photography, and many of the specific algorithms were found to be unsatisfactory and incompatible for accurately finding edges within infrared images.

Data Collection and Display

Data was obtained from the U.S. Army Aviation and Missile Command. We obtained both CAD model data in IFS format and infrared images in Seqb Image Format. The CAD model data was then rendered using different methods explained later in this document, namely Gouraud temperature shading, headlight illumination, hidden lines, visible lines, temperature lines, and isothermal lines. The infrared images were displayed by reading the pixel intensity levels from the files and displaying them in a window on a computer display. The formats for the two different types of files are found in the appendices.

Use and Adaptation of Existing Object Detection Algorithms

Algorithms from the existing object detection software were used on the data we received from the U.S. Army Aviation and Missile Command. It was quickly determined that the edge detection algorithms used previously, on high-resolution color photos, were not sufficient to detect edges in the infrared images used in this project. In comparison, the infrared images were of very low resolution, and of limited contrast between the target and its surroundings. The existing algorithm used a single 3x3 mask that was convoluted over the entire image. Switching to a directional mask, such as those created by Prewitt and Sobel[4], improved edge detection significantly, but still left many wholes in the edges. A new mask was added to the algorithm that instead of examining the rate of change of pixel intensity values in all directions (like these other masks do), this new mask examined the rate of rate of change of pixel intensity values. This second derivative was found by following horizontal, vertical, and diagonal lines through the image and finding the change in pixel intensity from pixel to pixel along each line, and then determining how much this changed from the previous pixel's change. For example, the second derivative along a horizontal line in the image would be determined as:

$$\partial^2 I / \partial x^2 = (I(x+2,y) - I(x+1,y)) - (I(x+1,y) - I(x,y))$$

where I is the intensity value at each pixel, and x and y are Cartesian coordinates within the image. All of the changes in pixel intensities from one pixel to another are determined for the entire image, building a histogram, and those values above a certain percentage of the histogram are considered an edge. The percentage was determined by experimentation. The optimal percentage for these infrared images included in our data set was found to be 30%. This addition proved enough to find edges extremely well. Figure 2 shows the edges found with this method from an infrared image and from a routine non-directional mask without using a second derivative edge detector.

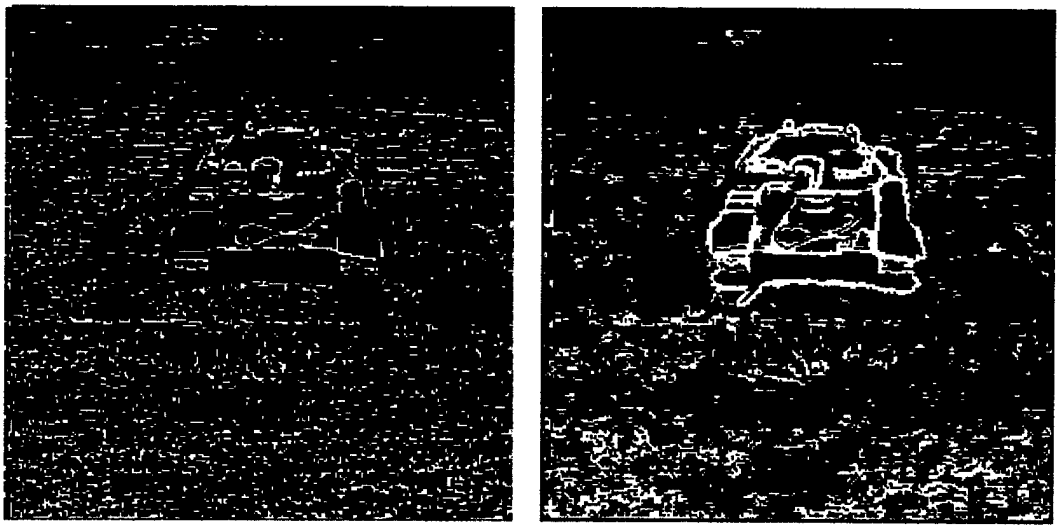


Figure 2: Edges found from an infrared image of a tank using a standard non-directional mask (left), and a Prewitt base edge detector with a supplementary second derivative edge detector (at right).

Masks

The most common form of edge detection looks for the rate of change of pixel intensities, moving along straight lines in eight directions (north, northeast, east, southeast, south, southwest, west, and northwest, where north is considered in the positive y direction on the image, and east is considered in the positive x direction). How do you calculate the derivative of an image in all directions? Convolution of the image with masks is the most common method. The idea is to take a 3 x 3 array of numbers and multiply it point by point with a 3 x 3 section of the image. Sum the products and place the result in the center point of the 3 x 3 section. This is done in the following manner for a point x,y in the image:

$$Sum = \sum_{j=0}^{j=2} \sum_{i=0}^{i=2} pixel(x-i-1, y-j-1) \times array(i, j)$$

After progressing over the entire image, points with values greater than a certain threshold are shown in white (an edge), and points less than that threshold are shown in black (not an edge). The threshold is determined by taking a histogram of all the intensities of the image and setting the threshold to a percentage of the different values in the histogram. The percentage was determined by experimentation, and was determined to be optimal at 30% for these infrared images.

Choosing the proper 3 x 3 mask is a task that is subject to the qualities of the image being processed. A common mask to use is one developed by Phillips. The mask is:

$$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

A far more effective approach for the infrared images used in this project was to use a directional mask, one that rotated the 3 x 3 array for each direction. The mask that worked best with the infrared data, the Prewitt mask, used the following 3 x 3 arrays:

$$\uparrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \nwarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

$$\leftarrow \begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix} \qquad \swarrow \begin{bmatrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\downarrow \begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \searrow \begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \qquad \nearrow \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$$

These eight 3 x 3 arrays are just the same array rotated around the center. The first rotation represents checking the derivative in the north direction, the second in the northwest direction, the third in the west direction, and so on.

Target Detection

To determine which edges belong with the intended target (and not other noise on the image), we first manually clicked the mouse inside of all sections belonging to the target, and those regions would be colored blue on the computer screen. This method determined the object in what

is called volume identification. Volume identification determines all of the pixels belonging to an object.

This type of object detection lacked several abilities needed for the final goal of this project, including automatic detection, an obvious method to compare with CAD models, and a link to moving in on a target from far away. Therefore, a new method of object detection was pursued. This method was to first locate the target. This could be done early on in a series of images as the observer starts out far away from the target and moves closer. This can be done by finding where all of the energy in the infrared image is located. Passing the image through a low-pass filter allowed the location of most of the energy in the image. Figure 3 shows the result of an infrared image passed through a low-pass filter. Notice that the energy is concentrated at the location of the target. This also

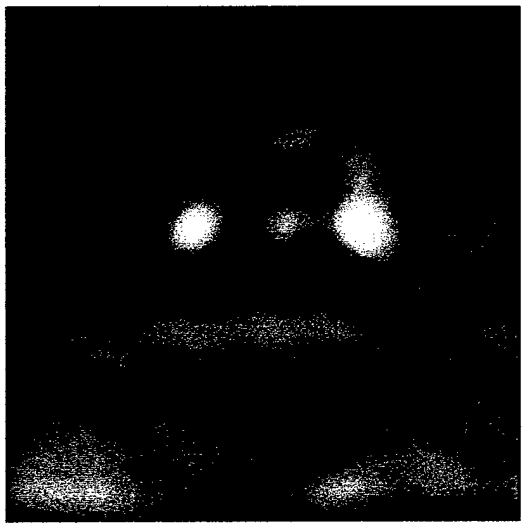


Figure 3: An infrared image after being passed through a low-pass filter.

happens to simulate the blurring effect of moving far away from an object (but does not simulate the reduction in scaling of the object). The original image was then passed through an edge detector. The result was then scanned to locate lines, both straight and curved, from within the edges found. Only the lines that were within the neighborhood of the target(s), selected by the low-pass filter, were kept. This proved successful in outlining the target and its areas of temperature change. Figure 4 shows the lines that were detected from an infrared image.

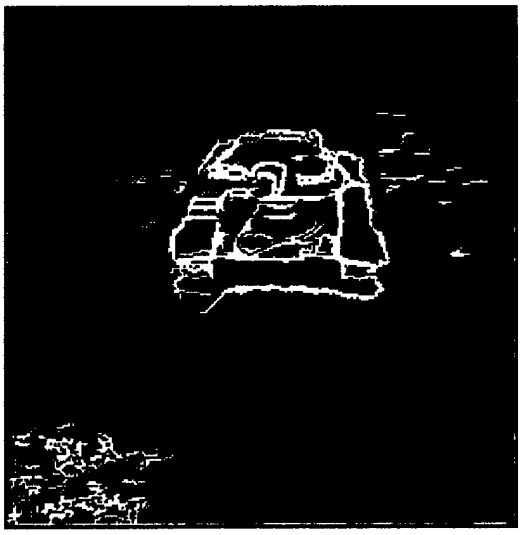


Figure 4: Lines found among the detected edges from an infrared image of a tank.

Use of both the volume detection and the line detection, in conjunction with each other, may offer the best results for alignment with CAD models and ultimately, target recognition.

CAD Model Equivalence

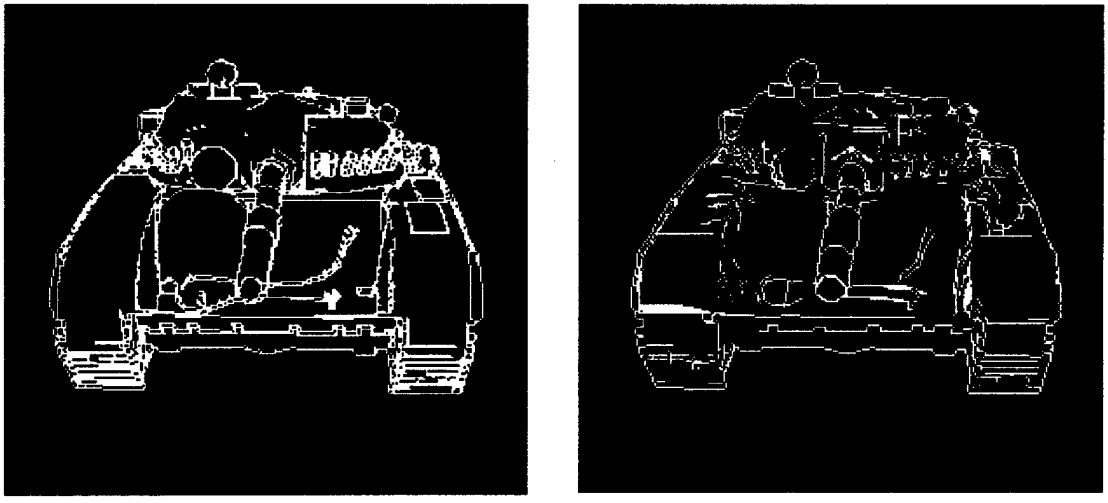
A method to compare the CAD model with the infrared image is needed to determine whether the CAD model is one that represents the target in the image. Our original method of volume identification simply required finding the borders of the CAD model, which was not a difficult task. Any software based model rendering engine should be able to find these borders easily (some hardware implementations of 3-dimensional algorithms may distort models slightly, not preserving photo-accuracy). Previous rendering engines developed at Cyberdynamics were adapted to fit this task..

When the plan of attack changed to compare lines of the image with lines of the CAD model, a new rendering method was needed. The new rendering method developed to supply lines to compare with the image was analogous to the line detection used on the image. Lines were drawn to divide areas of interest on the CAD model. First, an algorithm that drew lines between physical components on the CAD model was developed, as shown in figure 5a. The image, being in infrared, needed compare with temperature lines, so an algorithm was written to draw lines between areas where temperature changes significantly on the CAD model.

To do this, the model is projected into the 2D space of the computer screen. Since the CAD models have temperature given at each vertex of each facet of the model, each point visible to the viewer can have its temperature determined by doing a Gouraud interpolation between the vertices of the facet that the point in question belongs to. To illustrate this, in the software application Ctr, the user can click the mouse on any point on the CAD model and be told, among other information, the temperature at that point on the model. Figure 5b shows an example of a temperature line rendering.

Image/Model Comparison

Comparison of the image and CAD model would be very difficult without a visual way to assist in comparison, especially early in the process of comparison. To provide a visual comparison algorithms were developed to render the CAD model in an overlay bit plane, as described briefly in the previous section. To avoid blocking the view of the image, only important features of the CAD model should be outlined in the overlay bit plane. Important features could be thought of as physical features, which are outlined in one rendering method, or areas of similar temperature, which are outlined in another rendering



Figures 5: Figure 5a, on the left is a rendering of a tank showing the lines between physical features. Figure 5b, on the right, is a rendering showing the lines at areas of large temperature gradients.

method. To aid in further investigation of the comparison, a rendering method drawing isotherms was also developed and included in the Ctr software.

The first step in comparing the target within an image and a CAD model is to bring the CAD model into alignment with the target in the image. The first object detection procedure of volume identification led to an attempt at volume alignment of the model and the image target. Volume alignment is the process of trying to match as many of the pixels found within the image's target to the same pixels in the CAD model. This involved mostly looking at the local extremes (corners, etc.) of both the target and the model and seeing how closely the two aligned. There are seven variables that need to be determined to align an image target with a CAD model. The image is set, so the variables must be determined in the CAD model environment. These seven variables are: The x, y, and z components of the observer's position, the angles, usually azimuth and elevation, of the observer's view direction, the head-twist angle of the observer, and the magnification of the model (which is not the same thing as the distance from the model). The first step was to roughly approximate the variables relative to the CAD model. Approximating the observer's position was a simple task, knowing the CAD model's basic geometry (which is displayed in a text window in Ctr). Deciding on the observer's distance from the model and the magnification of the model is determined by looking at the perspective of the near parts of the CAD model and the far parts of the CAD model, and comparing these to the image target. From there, the direction of view must be determined. The center in the image should be approximately the center of the image, assuming the image had not been cropped unevenly. The head twist angle is simply approximated by looking at how much of an angle the target in the image is at compared with the CAD model that is rendered.

To fine-tune this alignment, further refinement was automated. To improve the observer's view direction, the CAD model was rendered in memory, the center pixel of the image was compared with the center pixel of the CAD model rendering, and then slightly adjusted. This was done repetitively until the two centers were aligned. Next, the observer's position was adjusted until the perspective of the CAD model rendering corresponded to the perspective of the target in the image. This was determined by investigating the ratio of the distance between the farthest points to the distance between the nearest points on the model, and comparing them with the corresponding

points on the target in the image. The observer position was perturbed and the model re-rendered until the perspective came into alignment with the target. The magnification was then adjusted to match the exact distances of the same points used for observer position setting, rather than just the ratios. The head twist angle was then determined by the angles that these same points created when comparing the lines from the CAD model with those from the target.

Many difficulties arose when this method of alignment was implemented. First, the different configurations of the target, such as a tank having its gun at different angles or having different fuel barrels attached to the back of the tank, caused difficulties for the algorithm to select the proper corresponding points to compare between the target and the model. This can probably be improved with statistical sampling in the future. Another problem is determining when to decided that the alignment is as close as it is going to get.

Using both the lines found from the line detection on the image and the rendering methods of lines separating temperature areas on the CAD model can solve these problems. As many lines as possible will be aligned in much the same way as they were in the volume methods. Finding the maximum number of lines that align between the target and the model will give a measurement as to the best alignment possible, and a metric to measure the resemblance of the CAD model with the target. At the time of this report, this method was visually inspected using the Ctr software, but only the beginnings of an automated algorithm had been created.

It is likely that a combination of the two methods would yield the best results. Combining the two offers more metrics to align the models and more metrics to determine the similarity of the CAD model and the targets.

Match Quality Evaluation

A final important area of work during Phase I was an investigation of techniques for image match quality evaluation. This investigation led to a conceptual alignment algorithm which is analogous to the matrix alignment method we use in 1-D radar signature matching. We believe that this algorithm, based on lines and line intersections, provides a direct way to calculate the optimal (in a least-squares distance sense) alignment parameters of two complex 2-D images (the infrared and the rendered CAD model). In this technique, the lines found in the images are first segmented into line features, which may be intersections, corners, near-circles, etc. These line features are then compared pairwise and the least-squares best-fit match parameters of magnification, x and y translation, and rotation are determined. This will result in a large set of parameter vectors representing the optimal matching of pieces of one image with pieces of another. These data are then analyzed in a 4-dimensional histogram to determine the most likely match parameters for the whole images. In other words, this is a divide-and-conquer scheme which determines the match parameters which are maximally self-consistent for all pieces of the image.

A conceptual basis for such a system was developed in our Phase I work. Figure 6 is a block diagram showing the main features of the proposed system. The main steps in the operation of this system are:

1. Input of image data from the infrared sensor,
2. Processing of the infrared image data to produce a database of found lines,
3. Input of CAD model data from the model database,
4. The production of a database of predicted lines from the CAD model,
5. Line feature selection in both the found line and predicted line databases,
6. Line feature matching,
7. Determining best overall image alignment, computing overall match quality metric, and evaluating line feature match quality in optimally aligned images. Compute estimate of

error in viewing aspect. Return to step 4 and iterate until the viewing aspect error is within acceptable bounds, and

8. Correcting the CAD model in areas of poor line feature match. Return to step 3 and iterate until image match quality is within acceptable bounds.

The image data from the infrared sensor will be obtained in the SEQB image format. These data will be interfaced to the operational data sources in a manner defined by specifications provided by the Army.

Determining the found lines in the infrared image data will begin with edge detection. The edges will be detected by convolution of the image with directional masks such as those developed by Prewitt and Sobel. The detected edges are then filtered using a smart fill algorithm to eliminate small gaps, and then processed to collapse the lines to the smallest possible width, and to remove isolated points and lines that are below a specified size.

The CAD model data will be input to this application in the Integrated Flight Simulation (IFS) format. An initial estimate of the target viewing azimuth and elevation viewing aspect will be used to produce the initial predicted image from the CAD model. This image will be the best possible prediction of the infrared surveillance image. Two types of predicted images will be available, one being an infrared image prediction based on an assumed surface emissivity and temperature profile as contained in the CAD model, and another, being the visible line prediction derived solely from the geometry in the CAD model. Both models can be tried and compared to the infrared image in subsequent processing. If the image based on temperature is used then the lines are found in this image in the same manner as that used in the reconnaissance image.

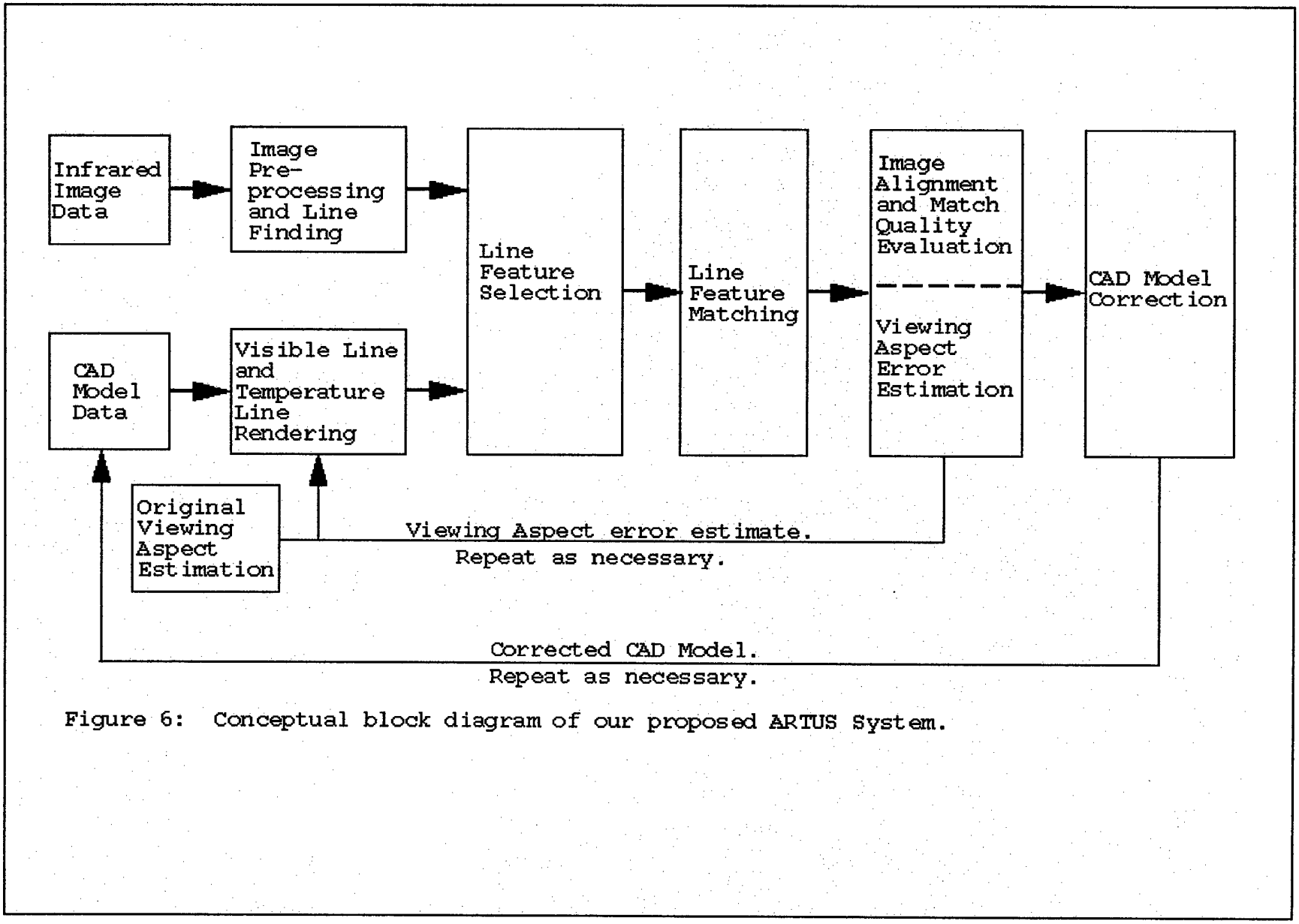


Figure 6: Conceptual block diagram of our proposed ARTUS System.

The two found-line data sets are then input to the line feature selection block. In this block the lines in the two data sets are broken down into line features of different types, such as closed contours, straight lines, line intersections, line intersection pairs, corners, etc. The line features are simply subsets of the found image lines. They are thus collections of x-y pairs of pixel coordinates. Some of the same pixels in the found-line set may occur in different line features, thus the line feature selection process is not a mere breaking apart of the found lines.

The different types of line features can contain different information regarding image alignment and are to be compared individually in the line feature matching block with line features of the same type derived from the other image.

The line features are then input into the line feature matching block. In this block, line features of a similar type are compared between the two images. This comparison consists of determining the alignment parameters (magnification, twist angle, x translation, and y translation) which give the least mean squared distance between the line features. Thus, for each possible line feature comparison, the line feature matching block will produce estimates of up to 4 different alignment parameters as well as a value of the normalized least squares distance between the features (the feature match quality metric).

At this point we now have knowledge of how well the individual line features match one another, and the image alignment parameter values required to achieve these matches. These data are then input to the image alignment block which analyzes individual match results and determines a single set of image alignment parameter values which aligns the greatest number of the line features in one image with corresponding line features in the other image.

An important property of this process is that it separates the line features into two groups; those who match features in the other image, and those who do not. This allows one to determine the optimal image matching parameters between two images which may represent the same object with high accuracy over part of the image, but with significant differences in other parts of the image. The non-matching

parts will contribute nothing to the estimates of the alignment parameters.

After the optimal image matching parameters are determined, they are then used to recompute the match quality metric for all the individual line features, this time keeping separate the x and y directional components of this metric. By an analysis of the x and y directional components of the match quality metric of the line features as a function of the location and type of line feature, one can form an estimate the error in the viewing aspect used to produce the image from the CAD model. If the viewing aspect were accurate, there will be no systematic x and y directional components of the line feature match quality metric. This estimate of viewing aspect error is used to correct the viewing aspect used to produce the image from the CAD model, and another iteration the image matching analysis is made. Such iterations continue until the viewing aspect error falls within tolerable bounds, or it is apparent that no satisfactory match can be found.

The separation of the line features into matching and non-matching groups allows the computation of an overall image match quality metric value for only matching features, which indicates the quality of the match which might be attained if all parts of the CAD model were accurate, and it allows computing an overall image match quality metric including the mismatched features. This quantity will indicate the matching performance to be expected from the CAD model in the operational system.

The final block in this system consists of analyzing the non-matching line features to determine the location and type of CAD model corrections they indicate, if any. Each CAD model will have an associated list of optional geometric models which might be added to the model. These might include geometrical objects such as cylinders, boxes, etc. The line features of these objects can be compared to the non-matching line features using the techniques described above. If a good match is found for a particular object at a particular viewing angle, this object is then added to the CAD model at the location and orientation defined by the alignment parameters of the match, and the revised CAD model reevaluated. This iteration continues until acceptable overall match quality is produced or it becomes apparent that no further improvement in the CAD model is possible. In that case the locations in the CAD

model where significant errors exist can be output for use by the missile matching algorithm to deweight such image locations to prevent degradations in missile performance due to these known CAD model errors.

The remaining unmatched line features will also contain any markings painted on the object which are visible in the infrared image and not present in the CAD model. An analysis of the unmatched line features will be made to identify such markings so that they can be added to the CAD model as a surface texture.

Some of the remaining unmatched line features may include lines caused by shadowing of solar illumination, or by obscuration by other objects. Such features will be ignored by the system.

Conclusion

We have studied various methods of object detection, CAD model rendering, image alignment, and match quality measurement. The goal of automatically detecting and identifying targets within an infrared image, and updating the CAD models to reflect the differences between the model and the image, is a feasible one. We have determined a method that we believe to be both efficient and robust that will attain these goals. The procedure involves using some of the methods we have devised to detect objects and render CAD models. We have determined techniques that can be used to align a rendered image with an infrared image based on work we have pursued to align synthetic and real one-dimensional radar signatures. The method involves dividing the images into line features and determining the best fit between the two images based on least-square differences between each pair of features. The least-square difference of matching features will determine the match quality of the two images. The presence or absence of line features in one image or the other will determine the differences between the infrared image and the CAD models, and will determine whether parts need to be added or subtracted from the CAD models. After editing the models, the steps taken to render, determine features, align, determine match quality, and determine differences between the rendered model and the infrared image will be repeated as necessary.

References

1. Advanced Animation and Rendering Techniques, Alan Watt, Addison-Wesley, 1994.

2. Computer Graphics: Principles and Practice, James D. Foley, Addison-Wesley, 1993.

3. Digital Image Processing, Kenneth R. Castleman, Prentice-Hall, 1979.

4. "Image Processing Part 6: Advanced Edge Detection," Dwayne Phillips, The C Users Journal, Vol. 10, No. 1, January 1992, pp 47-63.

5. Vision in Man and Machine, Martin D. Levine, McGraw-Hill, 1985.

Appendix A

IFS File Format

The IFS File Format is an ASCII format for CAD models. The format divides the model into parts, where the entire model is referred to as an object, and the parts are referred to as sub-objects.

Each line of the file starts with a descriptor of what is on that line. Possibilities for descriptors are: OBJECT, VOLUME, DATE, SUBOBJ, POINT, and FACET. OBJECT is on the first line of the file, and is followed by the model's name and then by the model's reference temperature (Tref) in degrees Celsius. DATE is an optional descriptor and may follow on a line after OBJECT. Following date is an ASCII description of the date. VOLUME is another optional descriptor, and if present, is followed by the x, y, and z coordinates of the geometric center of the model.

The SUBOBJ descriptor defines the start of a part of the model. On the SUBOBJ descriptor line, following the word SUBOBJ, are seven fields separated by white space. The first three are the x, y, and z components of the vector that each point in this part needs to be translated by. The next three are the amount in degrees that every point in this part needs to be rotated about the x, y, and z axes. The transformations are done in the order: rotate about x, then y, then z, then translate. The seventh field is the differential temperature of the part relative to the model, Tsubobj.

The POINT descriptor is followed by five fields. The first field is the vertex index within this part, starting from 1. The next three fields are the x, y, and z position of this point. The fifth field is the vertex temperature relative to the rest of the part, Tv. To get the absolute temperature at each vertex, add Tref plus Tsubobj plus Tv.

The FACET descriptor is followed by three integers. The integers are the indices of the three vertices that define the corners of the facet's triangle. To compute the absolute temperature at any point within the facet, one must bi-linearly interpolate between the vertices of the facet, using the temperature at each vertex as references.

Any lines starting with an exclamation point ("!") are considered comments. Lines starting with and exclamation point and then an asterisk are considered comments that are descriptive about the data used to derive the thermal signature.

Appendix B

Seqb File Format

The Seqb file format is an image file format that can contain multiple images, all of which are in grayscale format. Each image has associated with it a header, which among other things contains the width and height of the image, and a block of data with two-byte pixel values for each pixel in the image.

The header is a 512-byte block, and is defined in the following C-language header:

```
typedef struct{
     u_short   whole;
     u_short   fraction;
} IntFrac;

typedef struct{
     u_short   images_remaining;
     u_short   width;
     u_short   height;
     u_short   bytes_per_pixel;
     u_short   bits_per_pixel;
     u_short   year;
     u_short   day;            /* julian */
     u_short   hour;
     u_short   minute;
     u_short   second;
     u_short   milisecond;     /* millisecond */
     u_short   frame_rate;
     IntFrac   integration_time;    /* msec */
     IntFrac   electronic_gain;
     IntFrac   electronic_offset;
     IntFrac   digital_gain;
     IntFrac   digital_offset;
     IntFrac   horizontal_ifov;
     IntFrac   vertical_ifov;
     IntFrac   cuton_wavelength;
     IntFrac   cutoff_wavelength;
     IntFrac   max_transmittance;
     IntFrac   targ_slantrange;
     IntFrac   targ_aspect;
     IntFrac   targ_elev;
     IntFrac   atmos_temp;
```

```
    IntFrac   dewpoint_temp;
    IntFrac   atmos_visibility;
    IntFrac   atmos_relative_humidity;
    IntFrac   atmos_transmittance;
    IntFrac   atmos_pressure;
    IntFrac   reflected_ambient_temp;
    IntFrac   NUC_high_temp;
    IntFrac   NUC_low_temp;
    u_char    camera_info[16];
    u_char    reserved[128];
    u_char    target_type[16];
    u_char    engine_type[16];
    u_char    engine_state[16];
    IntFrac   calibration_offset;
    IntFrac   calibration_slope;
    u_char    reserved2[72];
    u_char    comment[128];
} Header;

/* the following structure is the data contained within
each image frame in the Seqb Image file. */
typedef struct{
    Header    header;
    u_short   image[256][256];
} SeqbImgFrame;
```