AIM

Association for Information and Image Management

1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910
301/587-8202

Centimeter

Inches

MANUFACTURED TO AIM STANDARDS
BY APPLIED IMAGE, INC.

1 of 1

**Automatic TLI Recognition System,
Part 1: System Description**

**J. K. Partin
G. D. Lassahn
J. R. Davidson**

**HQ PROJECT MANAGER - Stephen Herrick
PROJECT NUMBER - ST474E**

Published May 1994

**Idaho National Engineering Laboratory
EG&G Idaho, Inc.
Idaho Falls, Idaho 83415**

**Prepared for the
U.S. Department of Energy
Office of Arms Control
under DOE Idaho Field Office
Contract DE-AC07-76ID01570**

MASTER

ABSTRACT

This report describes an automatic target recognition system for fast screening of large amounts of multi-sensor image data, based on low-cost parallel processors. This system uses image data fusion and gives uncertainty estimates. It is relatively low cost, compact, and transportable. The software is easily enhanced to expand the system's capabilities, and the hardware is easily expandable to increase the system's speed.

This volume gives a general description of the ATR system. The user's manual is a separate volume, EGG-EE-11298, *Automatic TLI Recognition System, Part 2: User's Guide*.

CONTENTS

ABSTRACT	ii
INTRODUCTION	1
PROCESS DESCRIPTION	3
MATHEMATICAL APPROACH	5
Surveillance Process	5
Training Process	5
HARDWARE DESCRIPTION	14
SOFTWARE DESCRIPTION	18
Mask Creation Program G	18
Training Program F	18
Surveillance Program E	20
SCSI Software	20
EXECUTION SPEED	22
CONCLUSIONS	26
APPENDIX A	
EXAMPLE 1: Median Filter Testing	A-1
APPENDIX B	
EXAMPLE 2: Roads and Riverbanks	B-1
APPENDIX C	
EXAMPLE 3: Buried Waste Location	C-1
APPENDIX D	
EXAMPLE 4: Finding Airplanes	D-1

Automatic TLI Recognition System, Part 1: System Description

INTRODUCTION

This report comprises two parts, printed in separate volumes. This is Part 1, which is a general description of the automatic target recognition (ATR) system and gives some indication of its capabilities. Part 2 is for people who do the hands-on data analysis, giving instructions on how to use this ATR system.

The purpose of this task is to develop an Automatic Treaty-Limited Item Recognition system or, in the jargon of the literature, an automatic target recognition system. This task was started as part of the development of the Airborne Multisensor Pod System (AMPS), a reconnaissance system that can be attached to and carried aloft by any of several aircraft. This AMPS system will include several imaging sensors, such as a visible light camera, an infrared imaging system, and a radar imaging system. Each scene of interest will be photographed by each sensor, so we will have several (tentatively three, although this number may change as the AMPS design evolves) different images of each scene. The several images will be analyzed by a computer to determine the presence of previously specified objects of interest in the scene. (In this report, we will refer to these objects of interest as "targets", not to imply that we intend to shoot at them, but because of the established language in the technical community.) The development of this computer system, an *automatic target recognition* (ATR) system, is the present task.

This task includes choosing and assembling the computer hardware, developing the ATR algorithms, and developing software to implement the algorithms on the selected hardware. Special requirements for this ATR system include *image fusion* and *uncertainty estimation*. "Image fusion" means that the ATR system must analyze the three images jointly to produce a single, high-confidence statement of the presence of a target, as opposed to analyzing the three images separately and producing three separate, lower-confidence, possibly contradictory, indications of the presence of targets. "Uncertainty estimation" means that the ATR system must tell the user what confidence the user should have in the ATR system's report of the presence or absence of targets. In addition, there are the common requirements for speed, portability, and low cost of the ATR system.

This ATR system was designed specifically for the AMPS project, but it should be equally applicable to other projects. This ATR system can work with any number of images per scene; it is not restricted to using three different types of image for each scene. In particular, it can work with the most common and simple case of one single image per scene. This ATR system was designed to scan a scene for the presence of traditional objects such as trucks, golf balls, or airport runways. However, it can equally well find "objects" or targets such as a particular type of vegetation, a chemical spill, or some specific type of terrain. This ATR system can be useful in automatically scanning any set of images for any type of target, assuming of course that the target is in some way discernible in the images. The images do not need to

be images in the traditional sense of light intensity versus position in two dimensions; for this ATR system, an image is any two-dimensional array of numeric values. This ATR system informs the user of the importance of keeping each calculated feature used to discriminate between target and background, and thus gives the user the option of improving efficiency by eliminating unnecessary calculations and unnecessary sensors.

PROCESS DESCRIPTION

This ATR system is based on the assumption that targets can be distinguished from background by a quantitative measurement of the presence of *local features*. The term "local" normally implies that the features are small enough to be defined within a region that is small compared to the target, although the formalism does not impose any limit on the size of the features (except of course that the features must be smaller than the whole image). Examples of simple features are edges; lines; intensity averaged over a small region; and, speckle intensity. More complicated features include run length; other descriptors of size or shape of some region; and, local co-occurrence matrices. One could define as a local feature something as complicated as a detailed image of a particular target object, but this would not be a normal or easy use of the present formalism, and this use would not be covered by the usual interpretation of the phrase "local feature".

A *training* process is required by this ATR system. This requires a set of images, one image from each imaging sensor, for each of several training scenes. The system will work with as few as one training scene, but better results might be expected from the use of many, perhaps hundreds, of training scenes. The set of training scenes must include some (at least one) targets and some background. A knowledgeable person must examine each training scene and, for each scene, create a *mask*. The mask is a set of data with the format of an image (one byte per pixel, for example), but with each pixel value being 2, 1, or 0 indicating whether that pixel is part of a target, background, or unspecified region in the training scene. Thus, the entire set of training data comprises several real images and one mask "image" for each of several — or possibly only one — real world training scenes. Each training scene should contain either some target region or some background region or both; a training scene which contains neither target nor background is useless and should be discarded from the set of training data.

In addition to designating target and background regions in the training scenes, a skilled operator must select an initial set of local features that might distinguish between the target and background regions. If this operator-selected initial set of features does not include features that can distinguish target from background, the ATR computer will still function, but the uncertainty in target identification will be large, perhaps so large that the ATR system results will be useless. If, on the other hand, the operator-selected initial set of features includes more features than are necessary to distinguish target from background, no real harm is done. The ATR system will tell the operator the importance of each feature so that the operator can, if he wishes, discard unimportant features and make the final process more efficient. The only disadvantages of an operator selecting too large a feature set initially are some computational inefficiency in the training process (which is probably tolerable) and, in the extreme case, the inability of the computer programs to handle so many features. (The formalism can in principle handle any number of features, but of course any particular computer program will be limited.) Thus, although great skill is not essential in selecting the initial set of features, a clever choice can make the ATR process much more efficient and effective.

One additional task of the operator is to select a value for the relative weighting of two types of error. A *type 1 error* is the incorrect designation of a background region as a target by the ATR system; in a *type 2 error*, a target is wrongly called background. The ATR system could make either of these two error rates as small as desired, at the expense of making the other

error rate larger. We would like to make both error rates small simultaneously, but that may not always be possible. The procedure used here is to form a weighted sum of the two error rates, with weights W_1 and $1.0 - W_1$, where $0 < W_1 < 1$, and to minimize this weighted sum. The W_1 value is the relative weight of type 1 errors; type 1 errors are considered more important if W_1 is larger (closer to 1 than to 0).

The ATR training process takes as input (1) the training images and masks; (2) the definitions of the operator-selected feature set; and, (3) the value of the type 1 error weight W_1 . The training process then calculates values for coefficients, optimized to minimize the total (weighted sum) error rate. These coefficients are used in the surveillance process, to be discussed shortly. The training process also gives indications of the importance of each feature, and an estimate of the total error rate. The training program does this training process for the complete feature set specified by the operator, and also for the subset in which the least important of the original features is removed, and also for the best subset of this subset, and so on for all successive best subsets of features until the smallest subset is too small to allow a calculation (usually, one feature). This allows the operator to select an efficient subset of features with an acceptably low error rate, if such a subset exists. Or, in the worst case, the operator must recognize that this ATR system, with the feature sets that he has tried, cannot distinguish his targets from his background.

One product of the training process is the magnitude of the minimized total error rate. This is used as the estimated *uncertainty* in the ATR system for this particular set of features and coefficients, for this type of image data. This is a prediction of the fraction of pixels that will be misclassified as "target" or "background" in the surveillance process.

After an acceptable feature set is found, the coefficients for that feature set are used in setting up the surveillance program part of the ATR system. Then, a set of three images (no mask) from any scene is input to the surveillance program, and the program tells whether there is a target in the scene. This surveillance process does not require any special skill on the part of the operator. Since this surveillance process takes multiple images as input and yields one single result image as output, it accomplishes *image data fusion*, the simultaneous analysis of several input images to produce a single result.

MATHEMATICAL APPROACH

Surveillance Process

The surveillance process is used to scan large amounts of data with minimal expert interaction. This process accepts as input a set of T images of one scene; T is tentatively 3 in the AMPS application, but the mathematical approach allows T to be any value greater than 0. From the T scene images, we calculate F feature images $F_{s,f}$, where s is the scene number and f is the feature number, $f=1,2,\dots,F$. The user decides the number of feature images F and the nature of each feature image. Each feature image is simply an "image", a two-dimensional array of values in image format, that is calculated from the scene images by any methods that the analyst chooses. Thus, any single feature image might be derived from a single scene image, from another feature image, or from any combination of scene images and feature images. A raw scene image can be used as a feature image. The ATR algorithm always adds one special feature image to the set, the feature image numbered $f=0$, in which every pixel is 1 regardless of the content of the scene images. The user should not include this, or any image in which all the pixels have the same value, as one of his F feature images. Each feature image is multiplied by a coefficient C_f whose value has been determined in a training process, and these scaled feature images are then added to obtain a result image R_s for this scene s :

$$R_s = \sum_{f=0}^F C_f F_{s,f} \quad \text{for each scene } s. \quad [1]$$

This operation requires that all the feature images must be of the same size. If a pixel in the result image R_s has a value greater than the value Q determined from the training process, that pixel is called a target pixel; otherwise, the pixel is called a background pixel. Thus, the result image classifies every pixel in the scene as either target or background. This surveillance process is very simple in principle, although the calculation of some of the feature images from the scene images may be complicated in practice.

Training Process

The values of the coefficients C_f and Q , which are used in the surveillance process, are determined by a training process. This training process uses S training scenes, indexed $s=1,2,\dots,S$. For each scene, there are T training images, indexed $t=1,2,\dots,T$, and one three-level mask image. The mask pixel values are 2 for target regions, 1 for background regions, and 0 for unspecified regions. These regions have been selected manually by some knowledgeable person. The T training images are from T different sensors, so that all T training images contain different information. The T training images are assumed to be in registration. Let $T_{s,t}$ be the training scene images, and let M_s be the mask, for each scene s .

For each scene s , we extract F feature images $F_{s,f}$, $f=1,2,\dots,F$, from the T scene images.

We define a result image R_s for each training scene s :

$$R_s = \sum_{f=0}^F C_f F_{s,f} \quad \text{for each } s=1,2,\dots,S. \quad [2]$$

The feature image with index f equal to 0 is not a real image, but is an imagined image in which each pixel has the value 1. It is mathematically advantageous to include this image in the set of feature images. The coefficients C_f are to be determined in this training process. The C_f are the same for all s . We would like to be able to choose values for the C_f to make R_s equal to M_s in regions 1 and 2 (we don't care about region 0), for every s . We do not expect to be able to accomplish this exactly.

We use a least squares fitting procedure to make a first estimate of the coefficients C_f . We find the C_f values that minimize the sum of the squares of the weighted differences between the result images and the mask images for regions 1 and 2, with the weight for each pixel being inversely proportional to the number of pixels in that pixel's region. That is, we minimize

$$\begin{aligned} SSQ &= \sum_{s=1}^S \sum_{i,j} \text{weight}(s,i,j) [R_s(i,j) - M_s(i,j)]^2 \\ &\quad \text{sum over all region 1 + region 2 pixels} \\ &= \sum_{s=1}^S \sum_{i,j} [R_s(i,j) - M_s(i,j)]^2 / M_{1,0} \\ &\quad \text{sum over all region 1 pixels} \\ &\quad + \sum_{s=1}^S \sum_{i,j} [R_s(i,j) - M_s(i,j)]^2 / M_{2,0} \end{aligned} \quad [3]$$

where $M_{1,0}$ and $M_{2,0}$ are simply the number of pixels in region 1 and region 2 respectively. The indexes i,j denote a particular pixel in the image. The minimization condition is

$$\sum_{f=0}^F C_f \left\{ \sum_s \sum_{i,j} \text{weight}(s,i,j) F_{s,f}(i,j) F_{s,g}(i,j) \right\}$$

region 1 + region 2

$$= \left\{ \sum_s \sum_{i,j} \text{weight}(s,i,j) M_s(i,j) F_{s,g}(i,j) \right\}$$

region 1 + region 2

or

$$\sum_{f=0}^F C_f \left\{ \sum_s \sum_{i,j} F_{s,f}(i,j) F_{s,g}(i,j) / M_{1,0} + \sum_s \sum_{i,j} F_{s,f}(i,j) F_{s,g}(i,j) / M_{2,0} \right\}$$

region 1 region 2

$$= \left\{ \sum_s \sum_{i,j} M_s(i,j) F_{s,g}(i,j) / M_{1,0} + \sum_s \sum_{i,j} M_s(i,j) F_{s,g}(i,j) / M_{2,0} \right\}$$

region 1 region 2

for each $g=0,1,2,\dots,F$.

[4]

This represents $F+1$ inhomogeneous linear equations in $F+1$ unknowns, the C_f for $f=0,1,2,\dots,F$. These can be easily solved unless two (or more) of the feature images contain the same information, in which case one (or more) of the redundant feature images should be deleted from the set.

After these first estimate C_f values are determined, we can calculate the R , and generate two histograms, one for values of R pixels in region 1 and another for values of R pixels in region 2, both summed over all $s=1,2,\dots,S$. The region 1 histogram should have a peak near the pixel value 1, and the region 2 histogram should have a peak near the pixel value 2. We would like these two histograms to appear as well separated peaks, but in practice we expect that the high end tail of the region 1 histogram will overlap the low end tail of the region 2 histogram. We define a parameter Q such that, for any scene, we call a pixel a target pixel if its value in the result image R is greater than Q , and we call the pixel a background pixel if its value in the result image R is less than Q . The value of Q will of course be between 1 and 2, between the positions of the two histogram peaks. We would like to be able to choose a Q value such that, in the training scenes, every region 2 pixel is called a target pixel and every region 1 pixel is called a background pixel. This is clearly impossible if the two histograms overlap, which condition we expect in practice. In this case, for whatever value of Q we choose, there will be some region 1 pixels with R values greater than Q , or there will be some region 2 pixels

with R values less than Q , or both. These pixels represent errors in identifying targets. A region 1 pixel with an R value greater than Q is incorrectly called a target pixel; this will be referred to as a type 1 error. A region 2 pixel with an R value less than Q will be called a type 2 error. Part of the purpose of the training process is to find the value of Q that minimizes the total error probability.

We approximate the two histograms, after normalization, with two continuous distribution functions (probability density functions) D_1 and D_2 . We define two error probabilities E_1 and E_2 , representing type 1 and type 2 errors:

$$E_1 = \int_Q^{\infty} D_1(x) dx \quad [5]$$

and

$$E_2 = \int_{-\infty}^Q D_2(x) dx. \quad [6]$$

We define a total error rate as

$$E = W_1 E_1 + W_2 E_2 \quad [7]$$

with $W_1 + W_2 = 1.0$, where W_1 and W_2 are positive coefficients that have been selected previously to indicate the relative importance of the two types of error. We now want to find the values of Q and all the C_i to minimize the total error E . We have first estimates of the C_i values from the least squares fitting procedure; our first estimate of Q can be 1.5 (although there may be better first estimates of Q). We adjust all these parameter values by an iterative minimization procedure. It is necessary to impose two constraints while minimizing E : the mean values of the two distributions must be constrained to be 1 and 2. We use the method of Lagrange multipliers and minimize the quantity $E + \lambda_1 \mu_1 + \lambda_2 \mu_2$ while $\mu_1 = 1$ and $\mu_2 = 2$, by adjusting the values of C_i and Q . λ_1 and λ_2 are the Lagrange multipliers, and μ_1 and μ_2 are the mean values of D_1 and D_2 .

Let M be the number of moments needed to define D_1 or D_2 . If D_1 is the Gaussian distribution, for example, $M=2$; D_1 is defined by the mean (moment 1) and the variance (moment 2 - [moment 1]²), and thus by the first two moments. (For normalized distributions, the zeroth moment is always 1. We do not count this as one of the defining moments.) Let $M_{1,m}$ and $M_{2,m}$ be the m -th moments (about zero) of the distributions D_1 and D_2 , for $m=1,2,\dots,M$. Note that the above μ_1 and μ_2 are the same as $M_{1,1}$ and $M_{2,1}$. These moments are useful intermediate variables in the minimization of E .

Let the C_t and Q represent the current estimates of the coefficients as defined above, and let $B_t + C_t$ and $P + Q$ represent the optimum values of those coefficients. It is helpful to linearize E by expanding in a Taylor's series about the current values:

$$\begin{aligned}
 E(P+Q, B+C) = & E(Q, C) + P \frac{\partial E(Q, C)}{\partial Q} + \sum_{f=0}^F B_f \frac{\partial E(Q, C)}{\partial C_f} \\
 & + P^2 \frac{\partial^2 E(Q, C)}{\partial Q^2} / 2 + P \sum_f B_f \frac{\partial^2 E(Q, C)}{\partial Q \partial C_f} + \sum_f \sum_g B_f B_g \frac{\partial^2 E(Q, C)}{\partial C_f \partial C_g} / 2 \\
 & + \text{higher order terms to be neglected} \quad [8]
 \end{aligned}$$

or, using p and q to represent the sets of P with B_t and Q with C_t respectively and using the gradient operator ∇_q for simpler notation,

$$E(p+q) = E(q) + p \cdot \nabla_q E(q) + (p \cdot \nabla_q)^2 E(q) / 2 + \text{h.o.t.} \quad [9]$$

Then the minimization condition is

$$\begin{aligned}
 0 = & \nabla_p [E(p+q) + \lambda_1 M_{1,1}(p+q) + \lambda_2 M_{2,1}(p+q)] \\
 = & \nabla_q E(q) + p \cdot \nabla_q \nabla_q E(q) + \lambda_1 \nabla_p M_{1,1}(p+q) + \lambda_2 \nabla_p M_{2,1}(p+q) \quad [10]
 \end{aligned}$$

with the constraints

$$1 = M_{1,1}(p+q) \quad [11]$$

and

$$2 = M_{2,1}(p+q). \quad [12]$$

More explicitly, equations 10-12 can be written as

$$\begin{aligned}
 & B_0 \frac{\partial^2 E}{\partial C_0 \partial C_0} + B_1 \frac{\partial^2 E}{\partial C_1 \partial C_0} + \dots + B_F \frac{\partial^2 E}{\partial C_F \partial C_0} + P \frac{\partial^2 E}{\partial Q \partial C_0} \\
 & + \lambda_1 \frac{\partial M_{1,1}}{\partial C_0} + \lambda_2 \frac{\partial M_{2,1}}{\partial C_0} = - \frac{\partial E}{\partial C_0}
 \end{aligned} \tag{13.0}$$

$$\begin{aligned}
 & B_0 \frac{\partial^2 E}{\partial C_0 \partial C_1} + B_1 \frac{\partial^2 E}{\partial C_1 \partial C_1} + \dots + B_F \frac{\partial^2 E}{\partial C_F \partial C_1} + P \frac{\partial^2 E}{\partial Q \partial C_1} \\
 & + \lambda_1 \frac{\partial M_{1,1}}{\partial C_1} + \lambda_2 \frac{\partial M_{2,1}}{\partial C_1} = - \frac{\partial E}{\partial C_1}
 \end{aligned} \tag{13.1}$$

...

$$\begin{aligned}
 & B_0 \frac{\partial^2 E}{\partial C_0 \partial C_F} + B_1 \frac{\partial^2 E}{\partial C_1 \partial C_F} + \dots + B_F \frac{\partial^2 E}{\partial C_F \partial C_F} + P \frac{\partial^2 E}{\partial Q \partial C_F} \\
 & + \lambda_1 \frac{\partial M_{1,1}}{\partial C_F} + \lambda_2 \frac{\partial M_{2,1}}{\partial C_F} = - \frac{\partial E}{\partial C_F}
 \end{aligned} \tag{13.F}$$

$$\begin{aligned}
 & B_0 \frac{\partial^2 E}{\partial C_0 \partial Q} + B_1 \frac{\partial^2 E}{\partial C_1 \partial Q} + \dots + B_F \frac{\partial^2 E}{\partial C_F \partial Q} + P \frac{\partial^2 E}{\partial Q \partial Q} \\
 & + 0 + 0 = - \frac{\partial E}{\partial Q}
 \end{aligned} \tag{14}$$

$$B_0 \frac{\partial M_{1,1}}{\partial C_0} + B_1 \frac{\partial M_{1,1}}{\partial C_1} + \dots + B_F \frac{\partial M_{1,1}}{\partial C_F} + 0 + 0 + 0 = 1 - M_{1,1} \quad [15.1]$$

$$B_0 \frac{\partial M_{2,1}}{\partial C_0} + B_1 \frac{\partial M_{2,1}}{\partial C_1} + \dots + B_F \frac{\partial M_{2,1}}{\partial C_F} + 0 + 0 + 0 = 2 - M_{2,1} \quad [15.2]$$

Equations 11 and 12 can be written in the form of 15.1 and 15.2 because $M_{1,1}$ and $M_{2,1}$ are linear functions of the C_r . In equations 13.0 through 15.2, E , M_1 , M_2 , and their derivatives are to be evaluated at C_r and Q . Equation 10, or equations 13.0 through 13.F, represents $F+2$ equations in the $F+4$ unknowns P , B_r for $r=0,1,\dots,F$, λ_1 , and λ_2 . Equations 11 and 12, or 15.1 and 15.2, represent two more equations in the same unknowns. This set of linear equations can be solved to get values for the B_r and P , which can then be used as corrections to the values of the C_r and Q . This process can be repeated, presumably with the corrections B_r and P becoming smaller with successive iterations until the values of C_r and Q converge. These converged values represent the values that are optimal in the sense that they minimize the total error E .

It is convenient to use the chain rule with the moments as intermediate variables in evaluating the derivatives in equations 10-15:

$$\frac{\partial E}{\partial Z} = W_1 \frac{\partial E_1}{\partial Z} + W_2 \frac{\partial E_2}{\partial Z} \quad \text{for } Z = \text{any } C_r \text{ or } Q \quad [16]$$

$$\frac{\partial E_1}{\partial C_r} = \int_Q^\infty \sum_{m=1}^M \frac{\partial D_1(x)}{\partial M_{1,m}} \frac{\partial M_{1,m}}{\partial C_r} dx = \sum_m \frac{\partial M_{1,m}}{\partial C_r} \int \frac{\partial D_1(x)}{\partial M_{1,m}} dx \quad [17]$$

$$\begin{aligned} \frac{\partial^2 E_1}{\partial C_r \partial C_g} &= \sum_m \sum_n \frac{\partial M_{1,m}}{\partial C_r} \frac{\partial M_{1,n}}{\partial C_g} \int \frac{\partial^2 D_1(x)}{\partial M_{1,m} \partial M_{1,n}} dx \\ &+ \sum_m \frac{\partial^2 M_{1,m}}{\partial C_r \partial C_g} \int \frac{\partial D_1(x)}{\partial M_{1,m}} dx \end{aligned} \quad [18]$$

$$\frac{\partial E_1}{\partial Q} = -D_1(Q) \quad [19.1]$$

$$\frac{\partial E_2}{\partial Q} = D_2(Q) \quad [19.2]$$

$$\frac{\partial^2 E_1}{\partial Q^2} = - \frac{\partial D_1(Q)}{\partial Q} \quad [20]$$

$$\frac{\partial^2 E_1}{\partial C_r \partial Q} = - \frac{\partial D_1(Q)}{\partial C_r} \quad [21]$$

In this formalism, D_1 and D_2 , their derivatives, and the integrals of these distribution functions and their derivatives, which appear in equations 17 - 21, all depend on the particular functional form chosen for the distribution functions that represent the histograms. The moments and their derivatives are independent of this choice of functional form:

$$M_{1,0} = \left\{ \sum_{s=1}^S \sum_{i,j} 1 \right\} \quad [22]$$

The indexes i and j denote the i -th column and the j -th row in an image. The sum over i,j is to be taken over all the region 1 pixels for $M_{1,m}$ or all region 2 pixels for $M_{2,m}$. $M_{1,0}$ and $M_{2,0}$ are simply the number of pixels in all the training scenes in regions 1 and 2 respectively. The higher moments are normalized using these zeroth moments:

$$\begin{aligned} M_{1,1} &= \frac{1}{M_{1,0}} \sum_s \sum_{i,j} R_s(i,j) \\ &= \sum_{f=0}^F C_f \left\{ \frac{1}{M_{1,0}} \sum_s \sum_{i,j} F_{s,f}(i,j) \right\} \end{aligned} \quad [23]$$

$$\begin{aligned}
M_{1,2} &= \frac{1}{M_{1,0}} \sum_s \sum_{i,j} [R_s(i,j)]^2 \\
&= \sum_f \sum_g C_f C_g \left\{ \frac{1}{M_{1,0}} \sum_s \sum_{i,j} F_{f,g}(i,j) F_{f,g}(i,j) \right\} \quad [24]
\end{aligned}$$

and so on for the higher moments through $m=M$, with the obvious analogs for the region 2 moments. Note that these moments are taken about 0, not about the mean.

In practice, we need to go through all the feature images one time and calculate the sums that appear in braces { } in equations 22-24, and then we no longer need to access the images. It is important to access the images as infrequently and as efficiently as possible, because reading the images represents a significant part of the time consumption in executing the program (at least in the present implementation).

HARDWARE DESCRIPTION

The main part of the hardware selected for this ATR system is a parallel processing system using *transputers*. A transputer is essentially a computer processor designed to work well with other transputers in a parallel processing system. Each transputer includes its own logical and numerical processors, a limited amount of fast (on-chip) memory, and serial links to allow communication with other transputers or other devices. In common systems, each transputer is installed with its own block of external memory, which cannot be accessed directly by other transputers; transputers do not share memory.

A transputer system normally includes a separate host computer to allow user interaction. The host provides capabilities for keyboard input and screen output; for these purposes, any of a wide variety of low-cost small computers serves very well. In simple systems, the host computer also provides access to other peripheral devices, including such mass storage devices as hard disks and floppy disks.

The T805 prototype system is diagrammed in Figure 1. There are 11 general purpose computation transputer nodes, the root node (ROOT) attached to the host and ten other nodes (DAISY) in a daisy chain configuration, each with a 30 MHz T805 processor and 4 Mbytes of memory. There are two more special nodes, one to interface with a high-resolution video display device and another to interface with a SCSI bus to allow access to a variety of peripheral devices. The SCSI devices currently installed are a 1 Gbyte hard disk and an Exabyte 8mm tape drive. The host computer (HOST) is an IBM compatible system with an Intel 80486DX processor operating at 33MHz, an ISA bus, and the usual hard disk (80 Mbyte), diskette drives, keyboard, and monitor.

Figure 2 is a photograph of the system components, including one of the PC boards with transputers which would normally be inside the PC case. This board and another similar board are shown in more detail in Figure 3. As the figures imply, this one PC board holds 10 "daisy" transputer nodes, or CTRAMs (plug-in Computation TRANputer Modules), each with a T805 transputer and 4 Mbytes of memory. The transputers are the large square components, gray around the edges and gold in the center. The 4 Mbytes of memory are in the 8 black components on each CTRAM. The second PC board in Figure 3 includes the interface to the host computer, one CTRAM which serves as the network root node, one TRAM which occupies 4 slots and serves as an interface to the SCSI bus, another 4-slot TRAM which is a high-resolution video interface, and one empty slot.

The replacement cost of this system would be about \$ 20000. Additional daisy nodes could be added to increase the speed of the system, at a cost of about \$ 1000 per node.

Figure 1: Hardware schematic of the T805 prototype ATR system.

INEL Imaging System Hardware

Figure 2: Major hardware components of the T805 prototype ATR system. The system also includes a high-resolution video display, which is not in this picture.

Figure 3: Two personal computer circuit boards with transputer modules from the T805 prototype ATR system.

SOFTWARE DESCRIPTION

The host computer operating system software is MS-DOS version 5.0. The transputer programs were written and compiled with INMOS' ANSI C toolset D7214C.

Mask Creation Program G

For each training scene, a skilled operator must create a mask that indicates which parts of the scene are targets, which parts are background, and which parts are not designated as either target or background regions. The program G is a tool to help the operator do this task. G reads a scene image from a file specified by the operator, and displays the image on the screen. The operator can then use a cursor to draw polygons on the image and can designate the areas inside of each polygon to be whichever region (target, background, or unspecified) he wishes. Regions not marked by the operator are left as "unspecified". G creates a mask, an image that contains the operator's designations, and saves the mask to the file of the operator's choice. The original scene image file is not changed. The training program F later reads the mask file along with the scene image files.

Training Program F

The training processes is implemented in program F. The training program F calculates feature images according to an operator-supplied list of instructions, and finds the optimum values of coefficients to be used with these features in the surveillance program E.

Feature Calculation

Feature calculation typically starts with a single scene image and calculates several feature images. This may involve arithmetic or logical operations on a pixel-by-pixel basis; convolution in two dimensions with a template that is typically much smaller than the scene image; morphological operations; local order sorting operations, such as are done in a median filter; local co-occurrence matrix calculation; and others, limited only by the imagination of the operator. Of course, a given computer program can implement only a finite number of operations. The operations currently in this F program are listed in Part 2 of this report. These built-in operations can of course be combined to calculate very complicated features. The software can be expanded reasonably easily to incorporate additional operations, if that should become necessary.

The early part of program F stores the feature images in disk files, for use by the later part of program F. Of course, it would be much faster to store all the feature images in memory. However, image storage requires a lot of memory, and this in-memory storage approach would require that the number of features be limited according to the amount of memory available. To make the program as general as possible and avoid unnecessary restrictions on the number of features, this approach of writing the feature images to disk was adopted. This disk file storage is the motivation for including a high-capacity disk interfaced directly to the transputer network.

Sums Calculation

This ATR training process in the F program assumes (with some justification) that the distributions of result image pixel values are approximately Gaussian, both for the target pixels and for the background pixels. One result of this is that the error minimization process requires sums of pixel values from every product of two feature images. It would be more convenient to calculate these products and sums if all the feature images could be in memory at one time, but this would limit the number of features depending on memory availability. The approach used here is to read one row of pixels from every feature image into memory, and calculate the products and accumulate the sums from that row before replacing it with the next row from all feature images. This means that the program must repeatedly access a series of disk files, which is somewhat time consuming, but this seems to be the best alternative.

After the sums are accumulated, the images are no longer needed. The sums are all that is required to complete the training procedure. In addition to being used in the last part of the F program, the sums are written to a disk file to save the effort of re-accumulating them if they are needed in later runs of F.

Coefficient Calculation

After the sums for a given set of features are available, program F calculates the optimum values of the coefficients for that original feature set and for certain feature subsets. It will often happen that a substantially smaller subset will give error rates that are essentially the same as the error rate for the full original feature set. It is important to know this, because efficiency of the surveillance process (the analysis of images in the field) is greater for smaller feature sets. Therefore, the F program is arranged to look for optimal feature subsets. For each case, F calculates the optimal coefficient value for each included feature, and F also calculates the importance of keeping each feature included in that case. Then, a new case is formed by deleting the least important feature from the old set. This process of forming successively smaller subsets by deleting one feature at a time does, in effect, yield a graph of total error rate versus number of features, like the example in Figure 4. The operator can then see how many features he must include in the surveillance process to attain the desired error rate, and he can judge the trade-off between increased error rate and increased number of features.

For each case, the optimum coefficient values are determined by minimizing the total error rate. Program F does this minimization by using a linear approximation of the non-linear equations. This is an iterative, successive approximation procedure, and it requires a first guess for the coefficient values. A first guess can be obtained from a simple linear least squares fitting process; or, in the F program, a first guess can sometimes be obtained from the results of a previously calculated case which includes most of the same features as the present case. As is usually the case in this type of calculation, it is possible that the process will not find the desired absolute minimum in the error rate, but will converge to a substantially different relative minimum point. Sometimes this is evidenced by a smaller subset of included features giving a smaller error rate than a larger, previously calculated feature set. When this occurs, program F uses the coefficient values from the smaller subset, smaller error rate case as a first guess and restarts with the full original set of features included. This procedure sometimes leads to the discovery of optimal points with error rate minima that are substantially smaller than those associated with the first-indicated optimal point for a given number of features.

Figure 4: Typical example of total (weighted sum) error rate versus number of features. The optimization calculation tried several times, with slightly different resulting error rates, for some numbers of features; this is apparent from the double points plotted for 36 and 37 features.

Surveillance Program E

The surveillance program E is used in the field to analyze a set of scene images (no mask) and report on whether there is a target present in the scene. Program E uses the values of the coefficients determined by program F. Program E inputs scene images and outputs a result image. If the value of any pixel in the result image is greater than Q, a value supplied by the training program F, that pixel is interpreted as an indication of the presence of a target. The result image can be written to a disk file with the Q value embedded in the file so that the image can be displayed later with obvious indications of any targets that may be present, and of course the result image can be displayed on the monitor immediately when it is calculated.

SCSI Software

The SCSI bus is interfaced directly to the transputer network, not to the host computer. One result of this is that the SCSI devices (disk drive and tape drive) are not controlled by the host DOS operating system, and the familiar DOS disk file system is not used on the SCSI devices. In this application, separate file handling system software is used for the SCSI devices. The SCSI disk may be used by programs E and F very much as if it were a DOS disk with the device name SCSIDISK: (instead of a usual DOS disk name like A: or C:). The SCSI tape drive is not accessible via programs E or F. A separate program, TSCSI, was written to allow more specific access to the SCSI devices. This program allows such operations as copying files to and from the SCSI devices, reading the disk directory, deleting files, and certain diagnostic operations.

EXECUTION SPEED

An important aspect of this ATR system is its speed. There are two parts of the software that dominate the time required for execution: the feature calculations, and the coefficient optimization. The coefficient optimization is very strongly dependent on the number of features, and it also depends significantly on the details of the training data, so it is practically impossible to predict how long this calculation will take for any given case. This calculation only occurs in the training process, in which time would normally not be as important as in the surveillance process. Feature calculation, on the other hand, occurs in the surveillance process as well as the training process, so it is more important. Feature calculation time of course depends strongly on the number and type of features being calculated. We can allow some predictability, and perhaps allow comparison with other ATR systems, by measuring the times required for calculating some typical, common features. The times quoted here were measured with the echo turned off, using images having 512x480 pixels, unless otherwise stated. The tolerance statements of the times are absolute bounds on the possible errors, not statistical uncertainties. Some explanation of these times follows:

REM, remark without echo, at least 0.000178 ± 0.000010 second

This is the time required for the master transputer to read a command from the feature calculation file and do nothing else with the command. This time does include a few overhead operations that the master program executes for each command. This time is of course included in the times for all the other operations as well.

CLEAR, clear memory, 0.00147 ± 0.00010 with no memory deallocation

The master writes values to a small table in memory and sends a short message to all the slaves (daisy chain nodes), so this is an example of minimal one-way master-to-slave communication. Each slave also writes a small table in its own memory and, if images or kernels (operators) are defined, de-allocates memory.

READY, wait until slaves are finished, 0.00186 ± 0.00010 if slaves are not busy

The master sends a short message to all the slaves. Each slave responds separately, and the master waits for all the slaves to respond. Thus, this command involves brief two-way communication between master and slave, with essentially no other operation.

SET_AB, set scale factors, 0.00169 ± 0.00009

The master sets the values of two variables in a table in the master program, without interacting with the slaves. Thus, this represents is the minimum time that could be expected for any command that actually does something.

DEFKERN, define a kernel (operator), 0.00371 ± 0.00020

This operation allocates memory for a local convolution type of kernel (operator), with the whole kernel being stored in each slave's memory. This operation does not set the values of the kernel. The master sends a short message to all the slaves, and writes a few values to a table in memory. Each slave also writes a few values to memory and allocates a memory block

and, if the kernel was already defined, de-allocates the old memory block.

DEFIMG, *define an image*, 0.00371 ± 0.00025

This operation allocates memory for an image, with different parts of the image being stored in different slave's memories. This operation does not set any pixel values in memory. The master sends a short message to all the slaves and writes a few values to a table in memory. Each slave calculates which part of the image it should store in its own memory, and then tells the next slave which part of the image is already taken, to avoid slaves duplicating parts of the image. Thus, there is a small amount of inter-slave communication, done sequentially (each slave begins after the previous slave finishes). Each slave allocates a memory block and, if the image was already defined, de-allocates the old memory block. This time is expected to *increase* as the number of slave nodes (daisy chain nodes) increases.

ZEROIMAGE, *zero an image*,

0.0306 ± 0.0005 with no overlap rows,

0.0435 ± 0.0005 with 10 overlap rows

ADD, *add two images*, 0.0474 ± 0.0010

MUL, *multiply two images*, 0.0444 ± 0.0010

DIV, *divide two images*, 0.2393 ± 0.0010

SQRT, *square root of an image*, 0.2860 ± 0.0020

ABS, *absolute value of an image*, 0.1040 ± 0.0020

MAXCON, *clip low end values of an image*,

0.0550 ± 0.0010 if all pixels clipped

0.0520 ± 0.0010 if no pixels clipped

ADDCON, *add constant to an image*, 0.0390 ± 0.0010

MULCON, *multiply an image by a constant*, 0.0420 ± 0.0010

DIVCON, *divide an image by a constant*, 0.0440 ± 0.0010

For each of these image manipulation operations, the master sends a short message to all the slaves. Each slave does the indicated pixel-by-pixel operation for its part of the specified images. There is no inter-slave communication. The bulk of the time is for the actual image manipulation. These times should be inversely proportional to the number of slave nodes.

CONVOLVE, *correlate an image with a kernel (operator)*,

1.542 ± 0.005 for a 5x5 pixel kernel

The master sends a short message to all the slaves. Each slave does the correlation calculation directly (no Fourier transform) for its part of the image. This operation normally must be preceded by the OVERLAP operation to set the values of the pixels in the overlap rows. This time should be inversely proportional to the number of slave nodes and roughly proportional to the size of the kernel domain.

OVERLAP, *set pixel values in overlap rows*,

0.0148 ± 0.0005 for 2 overlap rows

0.0583 ± 0.0005 for 10 overlap rows

The master sends a short message to all the slaves. The slaves exchange the values of

the pixels in the overlap rows; thus, there is a lot of inter-slave communication. This time is expected to be roughly proportional to the number of overlap rows and independent of the number of slave nodes.

SSQ, local sum of squares, 2.000 ± 0.005 for 5x5 pixel kernel

The master sends a short message to all the slaves. Each slave calculates a weighted sum of squares of values of the pixels in the local region indicated by the domain of the specified kernel, with weights equal to the kernel values. This operation normally must be preceded by the OVERLAP operation to set the values of the pixels in the overlap rows. This time should be inversely proportional to the number of slave nodes and roughly proportional to the size of the kernel domain.

SMTHX, smooth in X direction, 0.680 ± 0.005 for 5 passes

The master does a short calculation and sends a short message to all the slaves. Each slave smooths its own part of the indicated file without communication with other slaves. The smoothing function is more Gaussian for more passes, with 5 passes being a good approximation to a real Gaussian and 1 pass having a sharp cusp in the smoothing function. The time is proportional to the number of passes and inversely proportional to the number of slave nodes.

SMTHY, smooth in Y direction, 1.880 ± 0.005 for 5 passes

This is like SMTHX, X direction smoothing, except that this Y direction smoothing requires significant inter-slave communication. This time should be roughly inversely proportional to the number of slave nodes.

MED1X, median filter in X direction,

0.910 ± 0.010 for 3-pixel window

1.380 ± 0.010 for 5-pixel window

1.860 ± 0.020 for 7-pixel window

2.300 ± 0.020 for 9-pixel window

2.760 ± 0.020 for 11-pixel window

3.220 ± 0.020 for 13-pixel window

The master sends a short message to all the slaves. Each slave does the calculation for its part of the image without inter-slave communication. This time should be inversely proportional to the number of slave nodes.

MED1Y, median filter in Y direction,

0.950 ± 0.010 for 3-pixel window

1.410 ± 0.010 for 5-pixel window

3.260 ± 0.020 for 13-pixel window

The master sends a short message to all the slaves. Each slave does the calculation for its part of the image without inter-slave communication. The Y direction median filter takes longer than the X direction because there is in effect some duplication of effort among the slaves for the Y calculation. This operation normally must be preceded by the OVERLAP operation to set the values of the pixels in the overlap rows. This time should be roughly inversely

proportional to the number of slave nodes.

MEDIAN, two-dimensional median filter, 11.70 ± 0.10 for a 21-pixel window

The master sends a short message to all the slaves. Each slave does the calculation for its part of the image without inter-slave communication. This operation normally must be preceded by the OVERLAP operation to set the values of the pixels in the overlap rows. This time should be inversely proportional to the number of slave nodes.

UNDERSAMPLE, decimate (undersample) an image, 0.0190 ± 0.0010 for decimation by a factor of 2 in each direction with a pre-defined destination image

The master reads two lines from the feature calculation file and sends a short message to all the slaves. If the destination image is not already defined, this operation defines it, which requires some inter-slave communication as in the DEFIMG operation. If the destination image is already defined, which is the more common case, there is no inter-slave communication. This time should be inversely proportional to the number of slave nodes.

SCALE, scale an image, 0.1660 ± 0.0010

The master sends a short message to all the slaves. Each slave scans its part of the specified image for the maximum value, and then the slaves daisy-chain communicate the maximum value to the master. The master then sends a short message to all the slaves, and each slave multiplies its part of the image by a constant. Thus, there is some master-slave and inter-slave communication involved in this scaling operation. This time should be approximately inversely proportional to the number of slave nodes.

READIMAGE, read an image, 2.25 ± 0.05

WRITEIMAGE, write an image, 2.20 ± 0.05

These times are for the SCSI disk. Accessing the system disk via the host computer bus takes roughly the same time. These times should be essentially independent of the number of slave nodes.

QUADUV, fit every local region with a quadratic polynomial, 111.0 ± 1.0 for a 21-pixel kernel

This requires a substantial amount of calculation, including a 6-parameter least squares fit, for every pixel. This time should be inversely proportional to the number of slave nodes.

CONCLUSIONS

This Automatic Target Recognition system is expected to be very effective for rapid screening of large amounts of multi-image data. This system is versatile and should be broadly applicable. It incorporates image data fusion and uncertainty estimation. It is reasonably portable, low cost, fast, and not subject to license restrictions. The software is easily expandable to incorporate new capabilities, and the hardware is easily expandable to increase the speed. This ATR system also informs the user of the importance of each feature used in the analysis, so that the user can make the process more efficient by eliminating unimportant features from the calculations and perhaps eliminating unimportant sensors from the multisensor data acquisition system.

APPENDIX A

EXAMPLE 1: Median Filter Testing

EXAMPLE 1: Median Filter Testing

This example is presented to illustrate very simply what the ATR system does. In this example, we use only one training scene, and we have only one image per scene. The goal in this application is to find medium-sized lettering and reject large lettering, small lettering, and other kinds of texture. This example was actually intended as a test of the one-dimensional median filter calculation subroutines, but it does demonstrate some ATR system capabilities.

The training scene image is shown in Figure A1, color coded to indicate the target and background regions specified by the user with program G. The green regions are designated as targets for the training program. They include only the medium-sized lettering, in the top left and the bottom right corners of the scene. The red regions, designated as background, include the larger lettering, in the top right corner and at the left edge; some of the small lettering, at the bottom of the scene; and, a large amount of the picture engraving. The blue regions are not designated as either target or background, and are ignored by the training process. Note that it is not necessary to mark all of the small lettering, for example, as background; if some of it is marked as background and none of it is marked as target, that should be sufficient.

Two different values of the type 1 error weight coefficient W_1 were tried: 0.5 and 0.9. The latter, larger value gives more weight to errors in which the ATR system incorrectly classifies a background pixel as a target pixel, whereas the $W_1=0.5$ value gives equal weight to this type 1 error and type 2 errors in which a target pixel is incorrectly classified as a background pixel. The two different values of W_1 give significantly different results. Figure A2 shows the error rate versus number of features for both of two training program runs. The error rate values above 0.15 are for the first run, and the values below 0.05 are for the second run. That is, we get a smaller total (weighted sum) error rate if we attach more importance to type 1 errors, for this training data.

In this example, the coefficients chosen for use in the surveillance process are those for the second training calculation, with $W_1=0.9$. These coefficients were installed in the surveillance program data files. The surveillance program was then used to analyze four scenes, looking for medium-sized lettering. The four scenes, and the surveillance program result images obtained for each scene, are shown in Figures A3-A6. One of these surveillance scenes was also used as a training scene. Of course, when the image is used as a surveillance scene, we do not tell the ATR system which parts are target and background; we let the ATR computer tell us. All of these images were originally the same size as the training image of Figure A1, but they were reduced in size for easier presentation here. Unfortunately, the images lose a little quality and detail in the reduction and reproduction for printing. Nevertheless, the results clearly illustrate how the ATR system works.

In the result images, the left half of each of Figures A3-A6, the bright, green to pale green to white, regions indicate what the ATR system classified as targets; the dark, red to black, regions indicate background. The lighter regions are stronger indications of target.

IMAGE FILE: \img\DEMO4.img
MASK FILE: \img\M4D.img

Figure A1: Training scene image with color-coded target (green), background (red), and unspecified (blue) regions.

Figure A2: Total (weighted sum) error rate versus number of features for two training program calculations.

Figure A6 shows that the ATR system did correctly designate the medium-sized lettering as target, and it did correctly reject the larger letters as background. The results are mixed for the small letters at the bottom center of the scene, some being correctly rejected as background and some being incorrectly identified as targets. A few other regions in the Figure A6 scene, as well as a few small regions in the other scenes, are incorrectly identified as targets by this ATR system. These incorrect identifications simply mean that this ATR system, with the particular features which were chosen for this application, cannot distinguish between medium-sized lettering and whatever is shown in the scene images at those regions incorrectly identified as targets. Presumably, the operator could choose a better set of features and reduce the incidence of errors in this ATR application. Note, however, that the relatively small set of relatively simple features used in this simple example did quite well: there are essentially no type 1 errors (incorrect designation of targets as background), and there are not many type 2 errors (incorrect designation of background as target). The absence of type 1 errors is largely due to the large value (0.9) chosen for the type 1 error weight in the F optimization calculation.

Conclusion

As expected, with a limited feature set (one-dimensional median filters only) and limited training (only one training scene), the ATR system is quite effective but not perfect at distinguishing targets from background.

RESULT FILE: \RI\DEMO1.RI

IMAGE FILE: \IMG\DEMO1A.IMG

Figure A3: Result image and scene image for DEMO1.

RESULT FILE: \RI\DEMO2.RI

IMAGE FILE: \IMG\DEMO2A.IMG

Figure A4: Result image and scene image for DEMO2.

Figure A6: Result image and scene image for DEMO4.

Figure A5: Result image and scene image for DEMO3.

APPENDIX B

EXAMPLE 2: Roads and Riverbanks

EXAMPLE 2: Roads and Riverbanks

This simple example illustrates fusion of image data. We have two images of the same scene, one visible light and one infrared image. (These images are part of a set supplied by Karen Steinmaus of Battelle, Pacific Northwest Laboratories, one of the participants in the Department of Energy's Airborne Multisensor Pod System project.) The visible light image (Figure B1) shows roads quite clearly, but it also shows riverbanks and it is difficult to distinguish between the two features in this image. The infrared image (Figure B2) does not show the roads very well, but it clearly indicates where the river is. The two images together should allow us to find roads and reject riverbanks. Note, however, that this cannot be done by looking for roads in each of the two images separately and then simply adding or averaging the two results; a more sophisticated approach to image data fusion, such as that used in this ATR system, is required.

For this illustration, we do a very simple analysis using only 4 features. For the first feature, we do a convolution of the visible light image with a 13x13 pixel kernel in which the pixel values are proportional to X^2 with the mean subtracted out, clip the result to keep only negative values, and take the absolute value. This first feature indicates the presence of both vertical roads and vertical riverbanks.

For the third feature, we do a convolution of the infrared image with a 13x13 pixel kernel in which the pixel values are proportional to X , and square the result. This third feature is insensitive to roads, but shows vertical riverbanks very well.

The second and fourth features are analagous to the first and third, using kernels with dependence on the Y coordinate instead of the X coordinate to detect roads and riverbanks that run horizontally in the images instead of vertically.

Figures B1 and B2 both show the operator-selected target (green) and background (red) regions used in the training process. Note that these regions do not need to be marked on both of the scene images separately; it is sufficient to use either one of the scene images with the program G to mark the target and background regions. The training process was done with these two images (one scene) as the training data. The optimized coefficients were put into the surveillance program file, and the surveillance program was then used to analyze this same scene. The result image is shown in Figure B3. In this result image, the white is the strongest indication of roads, darker green is a weaker indication of roads, black is the strongest indication of background (non-road), and lighter red is a weaker indication of background. This result does distinguish clearly between road and riverbank, thus satisfying the goal for this illustration: the two images analyzed jointly give a clear indication of a result that is not obvious in either image separately. This result indicates horizontal roads more strongly than vertical roads; this is because the horizontal roads appear narrower in these images, and the particular convolution calculations used here are more sensitive to narrower features. This result also indicates as roads some regions that are neither road nor riverbank; this is not surprising, since no significant effort was made to exclude miscellaneous clutter from being identified as target in this very simple example.

Conclusion

This minimal example demonstrates image fusion: neither image alone allows finding roads and distinguishing them from riverbanks, but the two images analyzed jointly, with a very simple algorithm, accomplish the goal very easily.

IMAGE FILE: image1b.img
MASK FILE: PNLmsk03.img

Figure B1: Visible light image, color coded to indicate operator-selected target (green) and background (red) regions.

IMAGE FILE: image6b.img
MASK FILE: PNLmsk03.img

Figure B2: Infrared image, color coded to indicate operator-selected target (green) and background (red) regions.

RESULT FILE: epn12b.ri

Figure B3: Result image.

APPENDIX C

EXAMPLE 3: Buried Waste Location

EXAMPLE 3: Buried Waste Location

This example illustrates the use of the ATR system with data that is not standard image data, and the use of a completely naive but still quite effective analysis method in which the ATR system's capabilities replace operator understanding of the data. The primary purpose of this study was to assess the utility of this ATR system in buried waste recovery operations. This data is from the Department of Energy's Buried Waste Integrated Demonstration project at the Idaho National Engineering Laboratory. The images in the figures in this example should be regarded as maps of areas from which buried waste is being excavated.

In this application, as many as 7 sensors acquired different types of data. This data is not image data in the traditional sense. Rather, each "image" is merely a set of values measured at a two-dimensional array of points on the surface of the ground. Sensor S1A is the vertical component of the earth's magnetic field, and S1B is the gradient (derivative with respect to vertical position) of the vertical component of the magnetic field. S2A, S3A, S4A, and S5A are measurements of the electrical conductivity of the soil, like eddy current measurements, taken with different combinations of field orientation and phase shift. S6A is a volatile chemical sensor. Most scenes include measurements from the first six sensors. Because data from the seventh sensor S6A was available for only 3 of the 35 scenes, and for one of these three scenes no other sensor data was available, this seventh sensor data was not used in this brief study.

There were 5 experiments, referred to as E1, E2, E3, E5, and E6. The different experiments represent data recorded over five different areas with different buried objects. The five rectangles on the left of Figure C1 indicate the *approximate* sizes and locations of buried objects in the five experiments, as seen from above. In these sketches, the solid objects are magnetic, and the objects drawn with outlines only are, at least mostly, not magnetic. In E2, for example, there are a magnetic barrel (object 7) and a wooden box (object 6). Objects 9 and 10 in E3 are two boxes, one above the other, with some magnetic material in the top box. The top box was removed between L3 and L4 (to be described later). Similarly, object 20 (a vertical barrel) in E5 was removed between L2 and L3, and object 27 (a vertical steel pipe) in E6 was removed between L2 and L3.

For each experiment, there are several different levels or vertical positions of the sensors. The number of different levels is not the same for all experiments. The several levels are referred to as L0, L1, L0 is the highest level, and the sensors are 6 inches lower for each successive level. In some cases, layers of soil were removed between successive measurement levels. In these analyses, any one level of any one experiment is treated as a separate *scene*. Different scenes are expected to give different results for any measurement, because they contain different objects or because the objects are at different distances from the sensors. There are a total of 34 usable scenes in this data set.

The scene images in this data have 43 columns (43 x values, spaced 3 inches apart) and 25 rows (25 y values, spaced 6 inches apart). Some of the data sets did not have this many rows or columns, and some were missing a few data points from what was expected to be a regularly-spaced array. In all of these cases, the missing data points were filled in using a linear interpolation or extrapolation procedure so that each scene image used in this study

had a full $43 \times 25 = 1075$ data points.

Perhaps the simplest analysis we can use with this ATR system for a set of data with 6 sensors is a linear combination of the 6 raw data images (augmented with the "constant" image which is always included for mathematical completeness). This simple linear analysis was tried with the buried waste data. The training was done with 3 scenes: E2L7, E5L5, and E6L5. The masks used were M2A, M5B, and M6A, shown in Figure C1. In this figure, the red (actually more brown) areas of the masks are designated as background, the green areas are target, and the black areas are not used in the training process. These masks mark barrels as targets and other regions as background; that is, this analysis is a search for barrels. The results of this training process were used to analyze 30 scenes from experiments E2, E3, E5, and E6; experiment E1 was not included in this analysis because this analysis uses 6 sensors and only 4 sensors were used in experiment E1. The results are indicated in Figure C2. In this figure, as in any of the result images, lighter regions are stronger indications of targets, and darker regions are stronger indications of background. Green and white (that is, very light green) indicate what the ATR system classifies as targets. Red and brown and black (dark red) indicate what the ATR system classifies as background. Figure C2 shows that this analysis did correctly identify the barrel (object 7) in E2 and another barrel (object 17) in E5 as targets when the excavation was deep enough so that the sensors were close to the barrels. This process also incorrectly gave small target indications at the top of E5L5 and in E6L6. These are ferromagnetic objects other than barrels, so it is not too surprising that this simple linear combination algorithm did not distinguish these objects from magnetic barrels. The process failed to detect a barrel at the center of E5, in levels L0-L2 (this barrel was removed between L2 and L3). The reason for this failure is not known. This barrel shows rather weakly in the raw data, suggesting the possibility that it may have been buried too deep to be detected by this algorithm. Although this algorithm for finding barrels did not yield perfect results, it did quite well, especially considering the simplicity of the algorithm and the fact that the algorithm required absolutely no understanding of the measurements or the physical processes involved.

This processes was repeated with an additional training scene, E5L2 with mask M5A, which specifies that the troublesome barrel (object 20) is designated as a target in the training process. The result of this analysis is shown in Figure C3. This process did locate object 20, but it also incorrectly identified a box (object 6) and other objects as targets. This rather poor performance is merely an indication that this simple linear combination algorithm cannot distinguish reliably between the objects designated as targets and those designated as background in this training set. The problem here is that object 20, which we believe to be a barrel, looks to the sensors like some other (background) object more than a target. This is a situation in which we must recognize that the algorithm we are using will not work well with this type of data. The obvious procedure is to try a better algorithm.

The next obvious level of complication in analysis is to add quadratic terms to the linear analysis, adding 21 product terms to the linear and constant terms in the previous analysis. This was tried using the same 4-scene training data set described for the linear case, with the result shown in Figure C4. As would be expected, this quadratic analysis is better than the linear analysis at distinguishing barrels from background, but it is still not perfect.

The same quadratic analysis procedure was used to try to find boxes instead of barrels. In this analysis, the training process used two training scenes, E5L5 with mask M5C and E2L7 with mask M2B. These masks select only boxes without ferrous metal contents as targets. The results of this analysis for 30 scenes are shown in Figure C5. This analysis was repeated using only the first training scene, with results shown in Figure C6. Obviously, the results of this type of analysis can depend significantly on the choice of training data. In both of these two analyses, boxes (objects 6, 9, 10, 21, 22, 23, and 24) were correctly identified as targets, and a few other objects were incorrectly identified as targets, with the single-training-scene result a little better than the two-training-scene result. This result is reasonably good.

An interesting aspect of these linear and quadratic analyses is that they require no understanding of the measurements.

In an effort to more accurately identify barrels, we tried a more sophisticated analysis that made use of some understanding of the measurement processes. In this analysis, we made an effort to find peaks in the magnetic field and its gradient, and to discriminate on the basis of the widths of the peaks, on the assumption that barrels should cause peaks of a certain width in magnetic field measurements. This analysis used only sensors S1B and S1A; the other sensor data was ignored. The training process used the same training scenes as were used for the quadratic function search for barrels. The results of this analysis are shown in Figure C7. Note that experiment 1 is included, since this analysis does not require the sensors that are not included in experiment 1. This analysis was repeated with a different set of training data, using scenes E2L8, E5L2, E5L5, and E6L6 with the same masks as before. This gave slightly better results, shown in Figure C8. These results generally show the barrels as targets, which is good. They also show other ferromagnetic objects as targets, which we had hoped to avoid. However, it is not surprising that we are not able to distinguish well between barrels and other magnetic objects, since the peak width which we tried as the distinguishing feature is not really unique to barrels. Surprisingly, this analysis also shows as targets some presumed non-magnetic objects, such as a box (object 6) in E2. This is interpreted as a fairly strong indication that these boxes do in fact include some ferromagnetic material.

Conclusion

This ATR system is easily applicable to areas other than arms control. Specifically, this example indicates that the ATR system used with this data can do quite well at finding buried objects, and it can give some indication of the nature of the object (magnetic or not) and the depth of the object. Failure to do better at indicating object type or depth is primarily due to limitations in the type of data available. This example also illustrates the importance of having a good training set, knowing exactly what background and target objects the training data represents, and making the training set self-consistent.

RESULT FILE: MASKS3.img
Figure C1: Buried objects, and masks used in training.

RESULT FILE: com1A2.img
Figure C2: Search for barrels, linear combination, three training scenes.

RESULT FILE: com4G2.img

Figure C3: Search for barrels, linear combination, four training scenes.

RESULT FILE: com4I2.img

Figure C4: Search for barrels, quadratic combination.

RESULT FILE: com4M22.img

Figure C5: Search for boxes, quadratic, two training scenes.

RESULT FILE: com4M21.img

Figure C6: Search for boxes, quadratic, one training scene.

RESULT FILE: com5K2.img

Figure C7: Search for barrels, peak width, first training set.

RESULT FILE: com502.img

Figure C8: Search for barrels, peak width, second training set.

APPENDIX D

EXAMPLE 4: Finding Airplanes

EXAMPLE 4: Finding Airplanes

This example illustrates the ability of the ATR system to function with poor data and in spite of errors or ignorance on the part of the operator. The goal is to find airplanes in pictures of airports, like the scene shown in Figure D1. There is an elegant approach to this problem, based on the assumption that the airplanes have the general shape of crosses or + 's. We can define a pair of kernels or operators which, together, find this general cross shape using the convolution operation. After some experimentation, this approach was used along with some very basic operations including looking for the correct size of darker area on a lighter background. For the training process, the scene was marked with two targets indicated in green (the airplane at the bottom center, and the airplane closest to the center of the image) and several background areas indicated in red in Figure D2. The training program gave the surprising result that searching for the cross shape did not significantly improve the ability of the system to find airplanes. The reason for the ATR system's not using the cross feature is obvious from an examination of the details of the image. The human vision system, seeing a "good" image like Figure D1 (perhaps in effect enhanced by the printing process) and knowing what to look for, is very good at idealizing shapes and filling in missing details. However, as Figures D3 - D5 show for the 7 airplanes in the top half of the scene, the airplanes comprise a surprisingly small number of pixels and they do not have much of the assumed cross shape. This image is actually of rather poor quality in terms of representing airplanes. It is no longer surprising that the ATR system did not give much weight to the cross feature in searching for airplanes. The interesting point is that the ATR system automatically discounted the unproductive search for crosses and used only the simpler features, even though this was contrary to the operator's expectations.

The result of using this very simple procedure is shown in Figure D6, in which the computer has drawn circles around each area it interprets as a target. The ATR system with this very simple scheme did correctly find all 9 airplanes, and it mistakenly marked a lot of background areas as targets. This general behavior, finding all the real targets and some false targets, is generally expected in a screening operation in which the system is optimized to not miss any real targets. However, the number of false targets indicated in Figure D6 is probably unacceptably high for most applications. We can improve on this result by placing more stringent requirements on the sizes of the dark spots that are called targets. Using this modified simple approach gives the result shown in Figure D7. This result correctly indicates all the airplanes as targets, and it incorrectly indicates two background areas, one of them so small as to be immediately discounted, as targets. This result is not perfect, but it is regarded as quite good, especially considering the poor detailed definition of the airplanes in the scene image and the simplicity of the search scheme.

Conclusions

This ATR system can correct some operator's mistaken assumptions, and can do a surprisingly good job with poor data and simple analysis schemes.

Figure D1: A scene including 9 airplanes.

IMAGE FILE: INPUT.img
MASK FILE: MASKD.img

Figure D2: The scene marked with two target (green) and seven background (red) regions for training.

IMAGE FILE: detail1A.img

IMAGE FILE: detail1B.img

IMAGE FILE: detail1C.img

IMAGE FILE: detail1D.img

Figure D3: Details of the four airplanes closest to the top of the scene.

IMAGE FILE: detail1E.img

IMAGE FILE: detail1F.img

IMAGE FILE: detail1G.img

IMAGE FILE: detail1.img

Figure D4: Details of the next three airplanes.

Figure D5: Detail image printed with one dot per pixel.

RESULT FILE: E9Xres.ing

Figure D6: Result of the simplest search scheme, showing a number of false target indications.

RESULT FILE: E9Nres.img

Figure D7: Result of a more restrictive search scheme, showing only two false target indications.

DATE

FILMED

8/24/94

END