Draft Final Report: Robust Recognition of Interesting Objects in Images
For the Army Research Lab
Version 2.11

Lawrence O. Hall and Steven Eschrich
Department of Computer Science and Engineering, ENB 118
University of South Florida
4202 E. Fowler Ave.
Tampa, Fl 33620
hall@csee.usf.edu
Ph. (813) 974-4195, FAX: (813) 974-5456

November 30, 2000

1 Introduction

This is the final report on the grant Robust Recognition of Interesting Objects in Images. The goal of this research project was to recognize targets in synthetic infrared images. Our approach was use a relatively simple target finder to initially identify potential targets. In the next stage, a fast fuzzy clustering approach, partially developed under this grant, was used to find regions of trees and regions of grass. Given this information it was possible to eliminate a significant number of false positives. That is, it was possible to eliminate many regions that were not actually targets. Almost no targets were lost in this process.

In the proceeding, we discuss two simple target finders. The latter one finds almost all of the targets. Our knowledge guided fuzzy clustering approach is able to greatly reduce the number of false positive targets. The goal is to recognize every target and only recognize targets as objects of interest. This goal is only approximately met. However, our work clearly shows that the integration of knowledge allows for a significant reduction in false positive targets.

In the process of this research, we have further developed a very fast, very accurate fuzzy clustering algorithm. A complete paper for the IFSA/NAFIPS'01 conference is under preparation.

1.1 Infrared Dataset

The dataset consists of 172 synthetic infrared images of natural outdoor scenes with 0, 1 or 2 targets within the scene. The targets were of 9 different types and were either partially obscured (hidden) or placed in an open area. Natural features of the scene include trees, shrubs and grass. Several scenes also had sections of sky. Various camera angles were used, and the relative temperatures of objects varied.

The infrared images provide gray-level values from 0-255 at each location in the image. Each image is 830 x 480 pixels. The gray-level value at each pixel indicates the heat at that pixel.

From the dataset of 172 images, 128 had targets and 44 did not. Of these 128 target images, 10 images had 2 targets and the remaining 118 had one. Target ground truth was provided for the images in the form of the center pixel of the target, together with range information and aspect. The results presented here were produced using the rule that a pixel is a target pixel if the Euclidean distance to the center pixel is less than a given threshold (20 pixels in our experiments). Visual inspection of the images indicates that 20 pixels is generally a reasonable value. It is possible to use more sophisticated shape-based metrics in future experiments.

A subset of the infrared images demonstrate an interesting phenomenon, the "negative" effect. Normal images have dark trees and shrubs, and lighter shades of grass. This shading corresponds to the ground being warmer than the trees. However, "reversed" images consist of dark grass and light trees, corresponding to the ground being colder than the atmosphere. This phenomenon makes identification of grass vs. tree pixels (and clusters) extremely difficult.

1.2 Area Reduction

The images consist of many "uninteresting" regions of natural scenery and a few regions of interest (ROIs). The algorithms used here focus on identifying areas of images as ROI's. Any algorithm which generates potential targets in or as regions of interest can be used as the first step in our approach. Later sections will describe the two simple algorithms which were used here.

Our strategy is to generate many different ROIs based on a simple heuristic, then use knowledge of the environment (in the form of fuzzy rules) to eliminate as many false positives, or non-target regions, as possible. The process should quickly generate targets and reduce the suspect regions to only target regions, which can then be identified by later-stage processing.

1.3 Report Contents

The report proceeds with a description of our target finders, followed by a description of the clustering algorithm used. Then a description of knowledge guided analysis using clustering is given. Experiments with the simple target finders are described and results provided. Next, a discussion of the results of applying clustering and knowledge to reduce false positives is given and supported by experimental results. Finally, a summary of results is given with future directions described.

2 Target Finding

2.1 Overview

Simple methods of detecting targets within infrared images are an important component of our overall strategy. Computationally expensive methods of analyzing the image are to be applied only after a large portion of the image has been rejected as background. Our algorithm does not attempt to classify the type of target. Rather it identifies the existence of a target within a natural scene.

Our approach to analyzing a scene involves first generating a large number of possible targets based on human-observed characteristics of the images. The algorithms at this stage should be computationally simple, yet effective enough to capture a very large percentage of the targets. We then apply expert knowledge of the potential locations of targets and their characteristics to further reduce the number of potential targets.

Once the image has been reduced to a manageable number of "interesting" regions, more complex image processing algorithms could be employed in an attempt to classify the target according to type (e.g. M60, HVT).

2.1.1 Approach

Both approaches to target finding are motivated by the observation that targets are typically very bright or very dark in intensity. This observation is discussed by other ATR researchers, including [3], [5], [7]. Recall that the images being considered are infrared images, therefore the intensity corresponds with the temperature of the area. The physical explanation of extreme bright/dark targets is that these "hot" regions generally correspond to high-temperature machinery (e.g. wheels and engines). It was also found that many "cold" regions correspond to targets, likely in the form of metallic surfaces that lose heat at a different rate than the ambient environment.

The two approaches to target finding are described first, then experiments and results based on these approaches are presented. Each algorithm has various parameters that must be empirically tuned, in order to achieve the best observed performance. The first approach was used to tune many of the basic parameters used throughout the experiments.

As described earlier, the set of images was separated into a training set and a test set. The parameters tuned within these and later experiments were generally taken from training set performance. Ten percent of the full

image set was chosen randomly and set aside as a test set to evaluate the final performance of the algorithm. The bias inherent in tuning an algorithm should not exist within the test set results. However, random choices of images do create the possibility that the set of test images will be a particularly difficult one. This appears to be the case in our experiments, as both target finding algorithms demonstrate. We note that it is certainly the case that for most other test subsets the target finding algorithms (due to the global nature of their tuning) would do much better.

2.2 Simple Intensity Thresholds

2.2.1 Overview

The first target finding algorithm used was a simple intensity threshold method. This method was used as a baseline at the outset of the project. Although the algorithm generates many non-target regions of interest, it is a simple approach and suitable for a baseline. In particular, we wish to quickly generate ROIs and then use expert knowledge to reduce the set of ROIs to (ideally) target regions.

The simple approach used searches for "hot" regions in the infrared images. It was observed that many of the targets had maximum intensity (white) or minimum intensity (black) pixels. In addition, most of the images appear to contain very few extreme-intensity pixels in other regions. "Regions" of extreme intensity were identified in the image. Several approaches were tried using this strategy, and the final results were reasonable.

2.2.2 Parameters

Several parameters exist within the simple targetfinder. They are listed below with a brief description. See Appendix A for a description of the experiments (and results) used to determine reasonable values for these parameters.

`TGT_DISTANCE` Target ground truth was provided for each of the images, as a pixel location representing the center of the target. Since the approach is simple and the ground truth does not directly indicate the size (in pixels) of the target, a simple Euclidean distance threshold was applied from the center of the target to the nearest pixel in the discovered region. This distance threshold is known as (`TGT_DISTANCE`).

The simple use of `TGT_DISTANCE` may not be completely appropriate, since aspects and distances-to-target vary across images. The value of 40 for `TGT_DISTANCE` was chosen based on experiments (see Appendix A).

IntensityDecision Many of the targets contain both min and max intensity pixels (i.e. Black (0) and White (255)), however it was not clear whether looking for one or the other extreme intensity was appropriate. If only one of min/max was chosen for consideration, the choice had to be made as to which one would provide the most information. From examining the images, some of the images were "negative" images, in the sense that the trees were of a lighter color than the surrounding grass (e.g. the trees were warmer than the ground).

In these "negative" images, black pixels were thought to indicate the target more often than the white pixels. An attempt was made to determine whether the image was "negative" or not, using the following heuristic:

> *Grass can be recognized via long runs of similar intensity. If the majority of long runs in the image are less than the average intensity, the image is "reversed."*

Once the image was determined to be "negative" (or not) the black (white) pixels were considered as targets. Experiments determined that looking for either minimum or maximum intensity provided the best results. Therefore IntensityDecision was set to `IMAGE_EITHER` as opposed to looking for just "hot" or "cold" regions.

`EPSILON` Instead of simply considering pixels of intensity 0 and 255, a threshold was used to consider pixels of intensity 0/255 +/- `EPSILON` . A value of 20 is used. Testing indicates increasing this value beyond 20 only slightly improves the true positive rate while significantly increasing the false positive rate. Setting it to a percentage of the average intensity might be more effective, although this was not explored.

Noise It was observed that a number of images had a single, black pixel in the upper right hand corner of the image (row 0, column 829). The pixel did not appear to correspond to the scene, therefore this pixel was automatically excluded from consideration.

AdjacentPixel In order to determine a "region" of bright/dark pixels, a threshold distance was set within which pixels were grouped together as a region. This value was chosen as 20 to prevent large "regions" from being created consisting of most features in the image.

Table 1 summarizes the parameters used throughout the remainder of the project.

Variable	*Value*	*Description*
`EPSILON`	10	Amount by which a pixel may vary from the extremes (0 and 255) and still be considered anomalous.
AdjacentPixel	20	Maximum Euclidean distance between two pixels considered to be adjacent.
IntensityDecision	`IMAGE_EITHER`	Use both minimum (0) and maximum (255) pixel values for detection.
`TGT_DISTANCE`	40	Allowable distance from center of target to flagged pixel to be considered a successful identification.

Table 1: Simple TargetFinder Parameters

Using the parameters in Table 1, the results in Table 2 were generated for the two different groups of images. The simple targetfinder does unusually poorly on the test set. This appears to be due to an "unlucky" random choice of images for this set.

Also of note is that a large number of false-positives were generated for the non-target images relative to the target images. Several types of images create problems for both our simple targetfinders and will be discussed now. The "problem" images represent a much larger percentage of the non-target dataset than the target dataset. The two key characteristics of these images that create problems for the simple target finders are embodied in the high contrast and high-intensity images. First, eight of these 44 images contained very low intensity tree pixels. These trees pixels are in the target intensity range (0-10) and are generally distributed across the entire image. The region-growing method used in both targetfinding approaches is limited to 20 pixels in order to capture the size of the target. This limit generates an extraordinarily large number of false positives, corresponding to all or most of the tree tops in the images.

A second problem in the non-target image set is that eight of the 44 total images had an extremely high average intensity level. In many of these cases the image was almost completely "washed out". Therefore most of the image was considered target regions and the darker spots in the image broke the image into the target regions.

Both these problems could be addressed by utilizing knowledge (i.e. that they occur and the characteristics which make them unique). However, we have not done this because our focus was mostly on finding targets and we were not fully aware of this phenomenon in time.

Dataset	*Total Targets Found*	*Total Targets*	*Total Target Percent*	*Total False Positives*	*False Positive per Image*
Train_TGT	111	126	88.1%	312	2.69
Test_TGT	6	12	50.0%	85	7.08
Total_TGT	117	138	85.0%	397	3.10
Total_NTGT				1273	28.93

Table 2: Simple TargetFinder Full Results, where TGT stands for target ground truth and NTGT stands for no target ground truth image.

2.3 Edge-like Detection

2.3.1 Overview

The second target finder implemented used slightly more information to identify targets. From the images, it was observed that the targets generally corresponded to large changes in intensity for most of the edges of the target. While this often was seen directly as a minimum or maximum intensity value, the average intensity of some images could be quite high. This high average intensity will generate large numbers of false positives for the simple intensity target finder. The distinction in these images was thought to be the sudden change in intensity.

For each pixel in the image, the absolute value of change in intensity to each neighbor (in an 8-neighborhood about the pixel [10]) was calculated. The maximum absolute change in intensity was recorded. Once the entire image was processed, the top 1% of intensity changes were grouped into "regions" using the methods of the simple-intensity targetfinder.

2.3.2 Results

The edge-like target finder was observed to locate many of the pixels within the targets. However, in high-contrast images the tree edges were also flagged. Therefore, the false-positive rate was considerably higher than in the simple intensity targetfinder. However, the knowledge-guided analysis of the image will be used to eliminate approximately half of the false positives.

Dataset	*Total Targets Found*	*Total Targets*	*Total Target Percent*	*Total False Positives*	*False Positive per Image*
Train_TGT	124	126	98%	3201	27.59
Test_TGT	9	12	75.0%	400	33.30
Total_TGT	133	138	96%	3601	28.13
Total_NTGT				2272	51.64

Table 3: Edge-like TargetFinder Results, where TGT stands for target ground truth and NTGT stands for no target ground truth image

3 Clustering

One of the major goals of this research is to reduce the size of the problem by quickly (and simply) identifying many possible targets and using a knowledge-based method to eliminate some of the generated false positives. As a result, the targetfinder was not extensively developed. Instead, the knowledge-based approach of eliminating possible options was pursued. A key element of this approach is the identification of grass and trees within the images, which are the major physical components of most of the images. Clustering was used to distinguish the different types of objects within the image.

Clustering is an unsupervised learning technique for grouping like data [6]. Extensive effort in this project involved clustering the images in order to provide higher-level information for reasoning. While large images are often difficult to cluster effectively in reasonable computing time, we used a bit-reduction fuzzy clustering algorithm with impressive speedup capabilities [8].

3.1 Laws' Texture Features

Simple features were considered for the clustering algorithm. In addition to the image intensity, Laws' texture features were investigated. Laws texture features determine texture properties through several key characteristics: edges, spots, ripples waves and average gray-level [9], [10]. The measure are computed from three initial vectors:

Label	*Vector*	*Description*
L_3	(1,2,1)	grey-level averaging
E_3	(-1,0,1)	edges
S_3	(-1,2,-1)	spots

Table 4: Laws' vectors

These vectors are convolved together and result in the following five vectors [4]:

Label	*Vector*	*Description*
L_5	(1,4,6,4,1)	local averaging
S_5	(-1,0,2,0,-1)	spot detector
R_5	(1,-4,6,-4,1)	"ripple" detector
E_5	(-1,-2,0,2,1)	edge detector
W_5	(-1,2,0,-2,1)	"wave" detector

Table 5: Laws' Energy Vectors

Finally, these length 5 vectors are multiplied together (e.g. $R_5^T E_5 = R_5 E_5$) to form a series of 5x5 image masks that can be applied to the image. We generated all 25 texture features for each image, then chose the feature with the largest mean variance across images. This was hoped to be a reasonable method of selecting a feature with a large discriminative ability. The texture feature chosen and used throughout the experiments was $S_5 W_5$ (see Table 6). It should be noted that experimentally, the addition of other Laws' texture features did not significantly improve clustering and therefore the chosen feature may be adequate.

1	-2	0	2	-1
0	0	0	0	0
-2	4	0	-4	2
0	0	0	0	0
1	-2	0	2	-1

Table 6: Laws' $S_5 W_5$ Mask

3.2 Fuzzy Clustering

The clustering algorithm used in this work is a modified fuzzy c-means algorithm. A c-means clustering algorithm is an iterative optimization of a given criterion function, where the value of c (the number of clusters to be found in the data) is predetermined [2].

A hard c-means clustering algorithm finds a partition of c clusters of the dataset that generally minimizes the sum of squared distances from a subset of the dataset to the cluster center of that subset [2].

The hard c-means can be made more flexible through the use of fuzziness in the cluster assignment. In other words, an example from the dataset need not completely belong to one cluster or another, but rather it can belong to several clusters to some degree.

Consider the set X of n feature vectors of a dataset and a given c. A fuzzy membership matrix U can be defined as a c x n matrix such that $\{u_{ik} 1 \leq k \leq n, 1 \leq i \leq c\}$ represents the membership of vector k in cluster i. u_{ik} is subject to the following conditions [2]:

$$0 < u_{ik} < 1 \ , \ for \ all \ i, k; \tag{1}$$

$$\sum u_{ik} = 1 \ , \ for \ all \ k; \tag{2}$$

$$0 < \sum u_{ik} < n \ , \ for \ all \ i. \tag{3}$$

Fuzzy c-means clustering involves the minimization of the following family of functionals:

$$J_m(U, v; X) = \sum_i \sum_k u_{ik}^m (\|x_k - v_i\|_A)^2 \tag{4}$$

where $1 < m \leq \infty$ corresponds to the degree of fuzziness in the partition, U is the membership matrix and v is the set of cluster centers. In practice, the fuzzy c-means clustering algorithm computes U and V alternately via:

$$u_{ik} = \left[\sum_{j=1}^{c} \left(\frac{\|X_k - V_i\|}{\|X_k - V_j\|} \right)^{\frac{2}{m-1}} \right]^{-1} \ , \ 1 \leq i \leq c \ ; \ 1 \leq k \leq n; \tag{5}$$

and

$$V_i = \frac{\sum_{k=1}^{n} (u_{ik})^m X_k}{\sum_{k=1}^{n} (u_{ik})^m} \ , \ 1 \leq i \leq c \ ; \tag{6}$$

The alternating computations are calculated until the observed change in U, the membership matrix, is small. This is determined by calculating the squared difference in original and updated U values during the computation of u_{ik} from Equation 5, and stopping when this squared difference is below a threshold, epsilon.

3.3 2rfcm

A variation of the fuzzy c-means clustering algorithm, known as 2rfcm [8], was used for fast, accurate clustering of the images. The algorithm consists of two stages, a preliminary bit-reduction stage and a modified fuzzy c-means clustering stage. The performance of the algorithm was significantly enhanced, providing an even faster method of fuzzy clustering.

3.3.1 Algorithm

The key concept introduced with 2rfcm is the bit-reduction stage of the algorithm. Recall that a dataset consists of n examples and each example is a vector of dimension s. In the simple case, consider $s = 1$. In an infrared image, the examples consist of the grey-level intensity of each pixel in the image. The 2rfcm algorithm divides the grey-level intensity space into equal bins of size 2^k, where k is the number of bits dropped. The intensity values are then distributed into the appropriate bins. A bin centroid is calculated and the number of members is recorded.

Once the bins have been constructed, the bin centroids are presented to a modified fuzzy c-means algorithm as the examples. There are at most $L/2^k$ bins in the dataset (where L is the possible number of intensity levels,

Bit Reduction	Average Collisions
3	1.63
4	0.18

Table 7: Average Collisions in Hashing Function

256 in infrared images). These examples are then clustered. Each example is weighed by the number of members in the bin (known as w) [8].

$$v_i = \frac{\sum_{j=1}^{n_0} w_j u_{ij}^m x_j}{\sum_{j=1}^{n_0} w_j u_{ij}^m} \tag{7}$$

The implementation of 2rfcm capitalizes on the fact that the feature values used are integer valued and are represented as bit strings. The choice of bin size 2^k was used so that an intensity value can be easily placed into the appropriate bin by dropping the k least-significant bits. Thus a value of 5, which in binary is represented as (0 0 0 0 0 1 0 1) becomes (0 0 0 0 0 1) for $k = 2$. The bit-reduction can also be seen as a reduction in the precision of the features [8]. This method, when combined with the efficiency improvements implemented as part of this project, can quickly reduce the size of the dataset.

When examples of more than one dimension are used, the s-dimensional space is separated into hypercubes, with each dimension of size 2^k. The number of possible non-empty bins increases, but as described below the characteristics of the infrared images limit this number.

3.3.2 Performance Enhancements

In [8], the authors introduced the bit-reduction of the example space, however the implementation of the algorithm required significant time overhead in bin creation. A more sophisticated bin creation method was implemented and careful attention was paid to the implementation of the summation formulas.

Creation of the bins was implemented via a simple forward search through existing bins [8]. Even for moderate sized bins, the creation overhead was large. Therefore, a hashing scheme was introduced to quickly find the appropriate bin. A hashing function is necessarily particular to the data used; the following function

$$h(x) = \left(\sum_{i=1}^{s} (i + 51) * X[i] \right) mod\ hn \tag{8}$$

was used and proved to be efficient. A hash table of size $hn = 1000$ was used and must be set based on the expected number of unique vectors for a clustering application. See Table 7 for details on the average collision rate for the infrared image dataset.

The next performance enhancement was from the implementation of the membership and centroid calculations within 2rfcm. Careful consideration was made as to when values should be calculated. Consider the calculation of the cluster centroids,

$$V_i = \frac{\sum_{k=1}^{n_0} w_k \left(u_{ik}\right)^m X_k}{\sum_{k=1}^{n_0} w_k \left(u_{ik}\right)^m}\ ,\ 1 \le i \le c\ ; \tag{9}$$

In the literal implementation, the value of u_{ik}^m would be calculated both for the numerator and denominator. By calculating the value once and temporarily storing it, we save n calculations. In addition, recall the X_k is a s-dimensional vector – the literal implementation calculates u_{ik}^m s times for each example. Thus by utilizing the cached value, there are $n_0 * s$ calculations saved per cluster centroid, or $c * n_0 * s$ operations during each iteration.

Next, consider the calculation of the membership values u_{ik}

$$u_{ik} = \left[\sum_{j=1}^{c} \left(\frac{\|X_k - V_i\|}{\|X_k - V_j\|} \right)^{\frac{2}{m-1}} \right]^{-1} \quad , 1 \le i \le c\ ;\ 1 \le k \le n; \tag{10}$$

In this equation, when $i = j$, the fraction evaluates to 1. By adding a special case check for this condition, we save $c * n$ divisions and $c * n$ exponentiations. Additionally, by arranging for the outer loop to be over the index

Bit Reduction	Average	Compression
3	1621	99.59%
4	480	99.88%

Table 8: Average Number Of Reduced Vectors (from original 398400)

Bit Reduction	Speedup
3	6.45
4	2.75

Table 9: Speedup Results from Performance Enhancements vs. Original 2rfcm

k (representing the particular example in the dataset), we can cache the value of $X_k - V_j$ for all j values. Each distance is used c times, therefore a savings of $c * (c-1)$ operations can be realized. Thus a total of $2cn + c^2 - c$ calculations can be avoided in the membership function updates during each iteration.

3.3.3 Results and Comments

Simple grey-level intensity images provide tremendous opportunities for the bit-reduction stage of 2rfcm. Consider a simple grey-level image consisting of 256 distinct values. Clustering in one dimension, our dataset would be reduced from 398400 distinct 1 dimensional examples to 256 examples, assuming a 0-bit reduction. This is simply representing duplicates within the dataset in a compact way.

Introducing another dimension, in the form of Laws' texture feature will reduce the amount of compression, however images often have similar characteristics and therefore repeated examples are common. See Table 8 for the remarkable compression rates in two-dimensional reductions using the infrared image dataset.

As mentioned earlier, performance enhancements were made to the 2rfcm algorithm both in bin creation and formula implementation. Several results are included in Table 9 that clearly demonstrate the success of the improvements over the first version of 2rfcm. As described by [8] the algorithm itself is generally very accurate relative to fuzzy c-means . Except for the pathological case of a set of vectors which are each unique after bit reduction, 2rfcm will always be faster than fuzzy c-means . The performance enhancements make it an even more attractive alternative.

Clustering using 2rfcm on the provided images was generally very successful. Images were normalized, then the Laws' texture feature was computed. These 2-D feature vectors were used by 2rfcm to cluster into 5 clusters. These clusters were then mapped back to the physical location within the image and graphically displayed. With some notable exceptions (see 3.4.3), the clustering algorithm successfully creates 2 clusters of grass, 2 clusters of trees and often a cluster of edges. Due to the nature of Laws' texture features, the pairs of clusters consist of almost identical centroids, with the Laws' feature both positive and negative.

Several problems were seen on several of the images. Although the images were normalized to provide consistent ranges of intensity, some images had a very low contrast level. These images typically did not cluster well. Image processing techniques for enhancing contrast could be used to address this problem.

3.4 Identifying Clusters

The 2rfcm algorithm was often successful at distinguishing two types of natural features: "tree-like" and "grass-like". Knowledge of the cluster type could be utilized later in the detection process, in order to reason about potential target regions. Two approaches were attempted: a machine learning approach and a shape-based approach. Neither method was entirely successful, although it is believed that the characteristics of these infrared images make this decision process extremely difficult given the feature set used.

3.4.1 Machine Learning

Decision trees are learning algorithms that accept examples in the form of attribute/value pairs stored in an example vector with a class decision and output decision trees [12]. C4.5 [11] , a decision tree program, was used in an attempt to classify clusters as "grass", "tree" or "other." Training examples used were initially cluster centers from each of the images, however vectors with membership of 0.9 or greater in a cluster were later added. This approach was generally unsuccessful - the decision tree was only able to classify clusters correctly 75% of the time.

Successful classification of trees vs. grass in the decision tree could be improved to approximately 86% if the algorithm could detect the existence of a "reversed" image. C4.5 was again used, this time in an effort to train a decision tree to recognize "reversed" images. However this method was also unsuccessful and could only recognize a "reversed" image 72% of the time.

3.4.2 Shape-Based Recognition

An attempt was also made to identify the cluster type using a simple image-processing approach. Experiments focused on simple shape characteristics of grass vs. trees, including the more contiguous nature of grass. Cluster memberships for each pixel in the image were noted, and the number times cluster membership changed within each row of the image were considered. It was believed that fewer cluster changes indicated large runs of grass.

The outcome of the experiments was also disappointing, not even achieving the success rate of the decision tree approach. The image set contained examples of open, tree-less regions as well as dense forest regions, therefore this simple approach to recognition was not adequate.

3.4.3 Discussion

The clustering method was able to separate the two classes of physical features into distinct clusters, corresponding to "grass" and "trees." But simple recognition efforts could not consistently identify the clusters. Both the existence of "reversed" images and a large variety of scenes caused significant problems to the classification algorithms used. It is believed that it is the characteristics of the images and the problem domain that create these difficulties.

Pixel intensity in the image is an important feature in the classification of images. However, the infrared images provide scenes in which there are no invariant rules regarding intensity in the relationship between physical objects. In addition, no generalizations of the scene can be made due to the diversity of the image set. We cannot state heuristics such as "trees are darker in intensity than grass" nor even "large, similar regions are expanses of grass."

Lack of descriptiveness in pixel intensity indicates that even more weight should be placed on derived features of the image, such as Laws' texture measures. More sophisticated image-processing techniques could be used to generate additional features, although this approach was not pursued. As described later in this report, the existence of unique clusters that consist of either "grass" or "trees" was sufficient to introduce some knowledge-based reasoning to the algorithm.

3.5 Splitting/Reducing Clusters

During the analysis of the clustering results, it was observed that some images do not cluster well using 5 clusters. The question of finding a more appropriate number of clusters for these images was considered. A method of splitting "bad" clusters was implemented in an attempt to find a useful number of clusters. In addition, a method of removing sections of the dataset during the clustering process was considered.

3.5.1 Cluster Validity Metrics

In order to implement a split/reduction of the clusters, metrics to describe the "goodness" of a cluster are needed. Two different metrics are considered here: a partition validity index and a cluster validity metric.

Recall that a fuzzy partition of a dataset consists of c cluster centers and a membership matrix U representing the fuzzy degree of membership of each element X to each of the c clusters. A partition validity measure, introduced in [13] and referred to as S, attempts to numerically describe the "appropriateness" of the partition.

The validity measure is given by

$$S = \frac{\sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} \left\| x_k - v_i \right\|^2}{n * min_{i,t} \left\| v_i - v_t \right\|^2} \tag{11}$$

The numerator of S describes the compactness of each cluster, summed across all clusters. The denominator describes the separation between clusters, as the minimum cluster separation in the partition.

Although the validity of a particular cluster within a partition can be derived directly from the Xie-Beni metric S, a modified version was used. Introduced in [1], the cluster validity metric SC is calculated for each cluster i. The equation is

$$SC_i = \frac{\sum_{k=1}^{n} u_{ik}^{m} \left\| x_k - v_i \right\|_A^2}{n_i \sum_{t=1}^{c} \left\| v_i - v_t \right\|_A^2} \tag{12}$$

where A is any positive definite matrix. This index is similar to S, with the exception that separation of clusters, as defined by the denominator, consists of the distances between every cluster not just the minimum inter-cluster distance. This provides a more accurate description of the cluster's position relative to all other clusters within the partition.

It is important to note that the Xie-Beni metric S is used in our algorithm to measure the validity of the entire partition, while the Bensaid metric is used to measure the validity of each individual cluster within the partition.

3.5.2 Cluster Splitting

The cluster-splitting algorithm begins with $c = 2$ and runs the bit-reduction fuzzy c-means algorithm until it stops. The Bensaid cluster validity metric SC is then calculated for each cluster and the "worst" cluster is identified. This cluster is split into two distinct clusters. The 2 cluster centers for the split cluster are initialized as x_{i_f}, representing the feature vector furthest in Euclidean distance from the original cluster center; and $x_{i_{ff}}$, representing the feature vector furthest in Euclidean distance from x_{i_f} [1]. The algorithm then repeats with $c = c + 1$ clusters.

Iteration of the cluster-splitting algorithm continues until either a maximum number of clusters have been generated or the partition validity index, as defined by the Xie-Beni metric S, increases (this corresponds to a "worse" partition than in previous iterations). In this way, the algorithm attempts to find a "good" number of clusters algorithmically.

3.5.3 Example Reduction

The fuzzy c-means algorithm tends to generate relatively balanced cluster sizes. An attempt was made to mitigate this problem by removing "good" clusters from the dataset at each iteration of the cluster-splitting algorithm. In order to maintain consistency in the validity metrics, the 2rfcm algorithm was run after removing examples from the dataset and prior to splitting a cluster.

The Bensaid cluster validity metric, SC, is used to determine the clusters to remove. If SC_i, representing the "goodness" of the ith cluster, is below a fixed threshold, all members of the cluster are removed from the dataset and permanently assigned membership in the given cluster. The clustering algorithm continues with a reduced number of examples until 10% of the original dataset remains (or one of the cluster-splitting criteria is reached). The remaining vectors are assigned cluster membership and the algorithm completes.

3.5.4 Discussion

The reduction-based, cluster-splitting 2rfcm algorithm appears to hold the promise of more precisely partitioning or segmenting the image. It should be noted that although the 2rfcm algorithm is run twice for each iteration of the full algorithm, the performance enhancements and data reduction techniques implemented (see section 3.3) allow this option to be used.

The algorithm contains various thresholds and variables that must be empirically tuned in order to fully realize the success of this method. However, in the available time these parameters could not be investigated. This method, combined with more descriptive features, could possibly generate more useful scene primitives without significantly increasing the computational requirements.

4 Knowledge-Guided Analysis Using Clustering

4.1 Overview

The ability to cluster, or segment, an image into "tree" and "grass" regions allows us to use some basic knowledge of the scene to eliminate "false" targets, or false positives. Our approach uses a simple targetfinder to generate many possible targets, then introduces knowledge to eliminate natural features and other phenomena while retaining the true targets. Such knowledge which can be utlized includes such fuzzy rules as

- Targets are often near tree/grass edges in an attempt to be hidden from view.
- Targets are generally too small to be clustered individually, and often target pixels are assigned to both tree and grass clusters.

So, our strategy is to call ROI's targets when they are not a good fit with any cluster but are spread across clusters.

Given a set of potential targets, as provided by either targetfinder described in Section 2, and cluster membership information, as provided by the 2rfcm algorithm (3), we would like to intelligently eliminate noise or extraneous regions from the list of ROIs. It should be again noted that this method will not increase the number of detected targets, since the simple targetfinder generates the ROIs. Our knowledge-based approach attempts only to eliminate the number of false positives generated. Thus, a simple (and fast) target finding algorithm can be used to generate a moderate number of ROIs, which can be passed to the knowledge-based stage for false-positive reduction.

The results of both targetfinders were analyzed and the characteristics of the ROIs were generalized. This knowledge was combined with the knowledge obtained regarding clusters, and a method of eliminating regions was developed. Both knowledge-based approaches concentrated on the fact that targets are generally too small to be individually clustered and are therefore mixed throughout the different clusters of the image. Generally, the targets have characteristics of both types of clusters and therefore exist in both types.

4.2 Simple Intensity False Positive Reduction

The simple intensity targetfinder generated targets based solely on extreme intensity value. It was observed that this often corresponded to bright regions within grass (e.g. a dirt road or other feature). These naturally occurring bright spots were often large and located within a grass cluster. The approach was to eliminate potential targets that did not appear to be "near" a boundary between trees and grass. Translating this rule with the available cluster information, we examine a NxN neighborhood around a potential target pixel. If there were not a majority of clusters represented in this neighborhood, the target pixel was discarded. The algorithm is described below.

The simple intensity targetfinder program was used to generate ROIs. The bit-reducing fuzzy c-means clustering algorithm (*2rfcm*) was used to produce cluster memberships for each pixel in the image. The algorithm is as follows:

- Use the simple targetfinder to generate ROIs
- Cluster the image using 2rfcm
- For each ROI, R:
  - For each pixel of interest, P, within R:
    - Consider M, the NxN pixel neighborhood about P
    - Determine the cluster membership for each pixel $Q \in M$.
    - If there are a majority of clusters represented in this neighborhood (from the total number of clusters), this pixel represents a target. If this is true, keep this ROI and continue with next.
  - If there are no more pixels of interest in R, remove this ROI from consideration.
- End

The NxN neighborhood used was 10x10, which experimentally appears to be about half the shortest dimension of the target. In our experiments, 2rfcm was used to cluster the image into 5 clusters. Therefore, there had to be at least 3 different clusters represented within the 10x10 neighborhood around a pixel of interest for the region to be kept as "interesting."

4.3 Edge-like False Positive Reduction

The edge-like targetfinder was observed to produce a sample of edges, between the tree and grass boundaries. This suggested that although a group of target pixels may be near a cluster boundary they are generally all of the same type. Therefore, the algorithm looks only at the cluster membership of the target pixels (not the surrounding neighborhood). The pixels of a tree border should generally belong to a tree cluster. And again, the target pixels are mixed between the clusters. Therefore we can discard any regions not representing a majority of the clusters as not indicating a true target.

The algorithm is similar to the simple intensity:

- Use the edge-like targetfinder to generate ROIs
- Cluster the image using 2rfcm
- For each ROI, R:
  - For each pixel of interest, P, within R:
    - Determine the cluster membership of P.
    - If there are a majority of clusters represented in this ROI (from the total number of clusters), this ROI represents a target.
  - If there are no more pixels of interest in R, remove this ROI from consideration.
- End

4.4 Experiments

Both of the above algorithms were implemented, and the results are presented. It should be observed that in the simple targetfinder experiment, the false-positive rate was significantly reduced without dropping a single true target. And in the case of the edge-like targetfinder, only three targets were dropped. The results of both experiments are shown in Table 10 and 11.

Dataset	*Algorithm*	*Total Targets Found*	*Total Targets*	*Total Target Percent*	*Total False Positives*	*False Positives per Image*
Train_TGT	Simple	111	126	88.1%	312	2.69
Train_TGT	Clustering	111	126	88.1%	168	1.45
Test_TGT	Simple	6	12	50.0%	85	7.08
Test_TGT	Clustering	6	12	50.0%	23	1.92
Total_TGT	Simple	117	138	85.0%	397	3.10
Total_TGT	Clustering	117	138	85.0%	191	1.49
Total_NTGT	Simple				1273	28.93
Total_NTGT	Clustering				293	6.66

Table 10: Knowledge-guided Simple Intensity TargetFinder Results.

Dataset	*Algorithm*	*Total Targets Found*	*Total Targets*	*Total Target Percent*	*Total False Positives*	*False* Positives *per Image*
Train_TGT	Simple	124	126	98.4%	3201	27.59
Train_TGT	Clustering	121	126	96.0%	1555	13.4
Test_TGT	Simple	9	12	75.0%	400	33.3
Test_TGT	Clustering	9	12	75.0%	171	14.25
Total_TGT	Simple	133	138	96.0%	3601	28.13
Total_TGT	Clustering	130	138	94.0%	1762	13.77
Total_NTGT	Simple				2272	51.64
Total_NTGT	Clustering				1088	24.73

Table 11: Knowledge-guided Edge-like TargetFinder Results.

4.4.1 Discussion

From the results of these experiments, it is clear the introduction of knowledge greatly reduced the false-positive rate. Recall that the targetfinder generates the maximum number of true positives – the additional algorithms seek only to reduce the number of false positives through use of extra knowledge of the image. Therefore, the true positive rate of the simple targetfinder is lower than the edge-like targetfinder, however both knowledge-guided reduction methods reduce the false positive rate. This process could be used for more sophisticated targetfinders.

Note that the false-positive reduction algorithms can remove targets from consideration. However the first algorithm using simple intensity dropped no targets, yet achieved a 53% reduction in false positives. The second algorithm using the edge-like target finder achieved a 48% reduction with only 3 lost targets. These results do indicate that the false-positive reduction algorithms are utilizing knowledge, or we would expect approximately 1/2 of the targets to have been removed.

5 Summary

We have developed a method to significantly reduce the number of false positive targets found in an infrared image. In the process of doing this, significant enhancements were made to a fast fuzzy clustering algorithm. Our approach utilizes domain knowledge to reduce the number of targets provided by a target finder. We implemented only simple target finders. It is certain that our approach would be more effective if it was combined with better (both more specific and more sensitive) target finders.

Future work, in addition to the integration of better target finders and perhaps knowledge into the target acquisition process, would involve the recognition of image types which tend to cause many false positives targets to be found. These images would be treated separately in the false positive elimination process.

The general approach of utilization of knowledge embodied in approximate or fuzzy rules to recognize interesting objects appears to show promise for target recognition.

A Simple Targetfinder Parameter Experiments

Several parameters are used within the target finding algorithms. These were experimentally determined to be reasonable. The experiments performed and the conclusions reached are included below.

A.1 Experiments

Recall that the `TGT_DISTANCE` is a simple Euclidean distance threshold applied from the center of the target to the nearest pixel in the discovered region. The effect of `TGT_DISTANCE` on detection rates was examined via experimentation (see Table 12 and Figure 1). The results indicate that with `TGT_DISTANCE` below 30 pixels, detection decreased significantly. Empirically, the target size is generally larger than 30 pixels indicating that

"edge" pixels would be misclassified as false positives. Examining the results we see that the improvement in success rate slows at 30, therefore the minimum TGT_DISTANCE slightly above that success rate agrees with the visual observation that a TGT_DISTANCE of 40 is reasonable.

TARGET_DISTANCE	*Total Targets Found*	*Total Target Percent*	*False Positives per Image*
10	84	66.7%	2.99
20	104	82.5%	2.78
30	109	86.5%	2.72
40	111	88.1%	2.69
50	111	88.1%	2.67
60	111	88.1%	2.67
70	111	88.1%	2.65
80	111	88.1%	2.59
90	111	88.1%	2.56
100	113	89.7%	2.48

Table 12: Simple TargetFinder Results

A.2 IntensityDecision

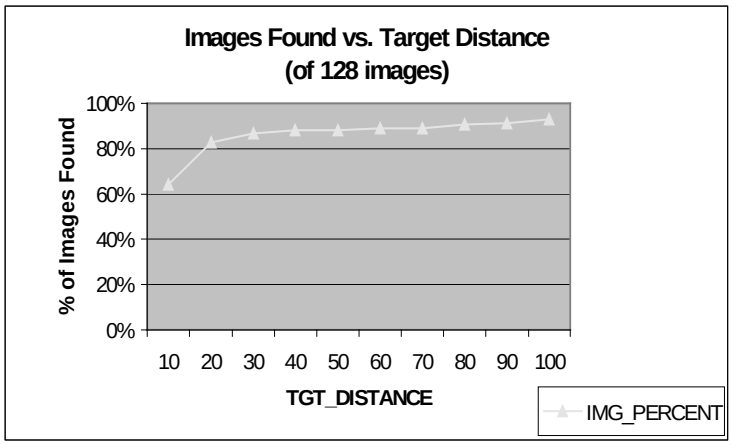
The value of IntensityDecision was also determined experimentally. From Table 13, it is clear that the IMAGE_EITHER decision is more accurate.

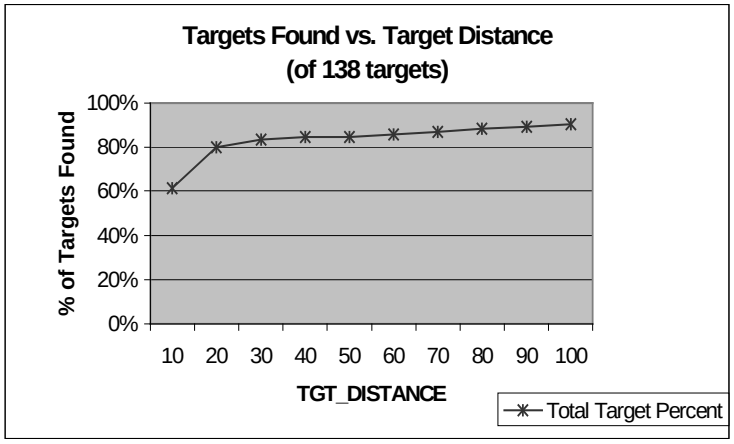
Intensity Decision	*IMG_COUNT*	*IMG_PERCENT*	*Total Targets Found*	*Total Target Percent*	*False Positives per Image*
IMG_EITHER	106	83%	110	80%	3.21
IMG_HEURISTIC	92	72%	95	69%	1.92

Table 13: Experiment 2 Results for Images with Targets.

Intensity Decision	*False Positives*	*Average false positives/image*
IMG_EITHER	706	16.04
IMG_HEURISTIC	365	8.3

Table 14: Experiment 2 Results for Images without Targets.


Images Found vs. Target Distance
(of 128 images)
% of Images Found
100%
80%
60%
40%
20%
0%
10 20 30 40 50 60 70 80 90 100
TGT_DISTANCE
IMG_PERCENT


Targets Found vs. Target Distance
(of 138 targets)
% of Targets Found
100%
80%
60%
40%
20%
0%
10 20 30 40 50 60 70 80 90 100
TGT_DISTANCE
Total Target Percent

References

[1] Lawrence O. Hall Amine M. Bensaid et al. Validity-guided (re)clustering with applications to image segmentation. *IEEE Transactions on Fuzzy Systems*, 4(2):112–123, May 1996.

[2] James C. Bezdek and Sankar K. Pal, editors. *Fuzzy Models For Pattern Recognition.* IEEE Press, New Jersey, 1991.

[3] W.M. Brown and C.W. Swonger. A prospectus for automatic target recognition. *IEEE Transactions on Aerospace and Electronic Systems*, 25(3):401–410, May 1989.

[4] Yung-Chang Chen Chung-Ming Wu and Kai-Sheng Hsieh. Texture features for classification of ultrasonic liver images. *IEEE Transactions on Medical Imaging*, 11(2):141–152, June 1992.

[5] Sandor Z. Der and Rama Chellappa. Probe-based automatic target recognition in infrared imagery. *IEEE Transactions on Image Processing*, 6(1):92–102, January 1997.

[6] A. Jain and R. Dubes. *Algorithms That Cluster Data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[7] C.P. Walters James A. Ratches et al. Aided and automatic target recognition based upon sensory inputs from image forming systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1004–1019, September 1997.

[8] J. Ke, L.O. Hall, and D.B. Goldgof. Fast accurate fuzzy clustering through reduced precision. *IEEE Trans. on Fuzzy Systems*, 2001. Submitted.

[9] K.I. Laws. Texture energy measures. In *DARPA Image Understanding Workshop*, pages 47–51. DARPA, Los Altos, CA, 1979.

[10] Vaclav Hlavac Milan Sonka and Roger Boyle. *Image Processing, Analysis, and Machine Vision 2nd Edition.* Brooks/Cole Publishing Company, CA, 1999.

[11] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1992.

[12] Stuart Russell and Peter Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall, New Jersey, 1995.

[13] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, August 1991.