Hybrid Fuzzy-Neural Classifier for Feature Level Data Fusion in LADAR Autonomous Target Recognition*

Stephen W. Soliday[a] and Melissa T. Perona[b] and Danial G. McCauley[c] and

Raytheon Company
[a]Tactical Mobile Robotics / FCS, C3I & Information Systems
[b]Advanced Programs, Missile Systems
[c]Processor Technology Product Center, Electronic Systems

ABSTRACT

This paper will discuss the design of a hybrid fuzzy-neural classifier for fusion of range and intensity channels coming from a LADAR sensor. Fusion was performed on a feature rather than pixel level. Results will be compared between ATR performance with and with out fusion. Also, discussed in this paper is the use of genetic algorithms for the training and optimization of the ATR system with a limited set of ground truth.

Keywords: Fuzzy, Neural, Genetic, Fusion, ATR, LADAR

1. INTRODUCTION

1.1. Overview

Raytheon has long experimented with various *Automatic Target Recognition* (ATR) techniques using LADAR data of *Critical Mobile Targets* (CMT). The ability to test these concepts against moving targets in a real-time, moving-platform environment, however, has been limited. This environment poses system-level restrictions on the memory and throughput that the ATR algorithm may utilize, and hence limits the algorithm capability. Additionally, system-level effects frequently alter the sensed data significantly from stationary-platform or, to a greater extent, synthetic data. These restrictions have led to increased interest in machine intelligence as a means to perform target identification. The ability to apply a hybrid fuzzy-neural algorithm to the classification problem has allowed increased throughput and required less memory than other popular techniques.

2. LADAR ATR

The LADAR sensor returns two data channels, *range* and *intensity.* The *range* data must be motion compensated and transformed to a coordinate system relative to a fixed point. At this point the data is segmented. In segmentation the *regions of interest* ROI are tagged and listed. Geometric features are extracted from the *range* image and fused at a feature level with reflectance features extracted from the *intensity* image. The features are fused using a hybrid fuzzy–neural classifier that matches measured values against a tabulated target database.

2.1. Rapid Prototyping

The ATR system was built using a rapid prototyping system called Rippen[†]. This allowed the development of individual modules written in ANSI C. The modules were laid out in a flow-graph, see Section 2.2. Each module is a self–contained object, and these are linked together to form an executable. The flow of data is determined at run–time and may be saved for future runs. Rippen has the advantage of being multi–platform allowing the development to be done on a host processor. In this case the host processor was a Sparc Station.

Further author information: (Send correspondence to S.W.S)
S.W.S.: Falls Church, Virgina 22042 USA, soliday@raytheon.com
M.T.P.: Tucson, Arizona 85734 USA, mtperona@west.raytheon.com
D.G.M.: Plano, Texas 75023 USA, E-mail: danmccauley@raytheon.com

[†]Rippen is a registered trademark of Orincon

2.2. Flow-graph

2.2.1. Acquisition

The LADAR sensor returned two channels of data. The range data consists of a pixel image where each pixel has a scaled value representative to the range between the detector and the scene. The intensity image is the strength of the return signal. Both of these images are generated at the same resolution and at the same time, therefore they may be assumed to be registered images.

2.2.2. Transformations

The range data required motion compensation to counteract the distance the sensor traveled between the beginning and the end of a scan. The navigation data from the guidance system was used for performing these transformations.

The data was sheared and rotated to compensate for the movement of the sensor. This data combined with a camera model and the navigation data was then used to generate three more images. The X, Y, & Z images represent a coordinate system of *north*, *east*, and *down* (NED). For each pixel in the range image an X, Y, & Z coordinate with the origin at the sensor was determined.

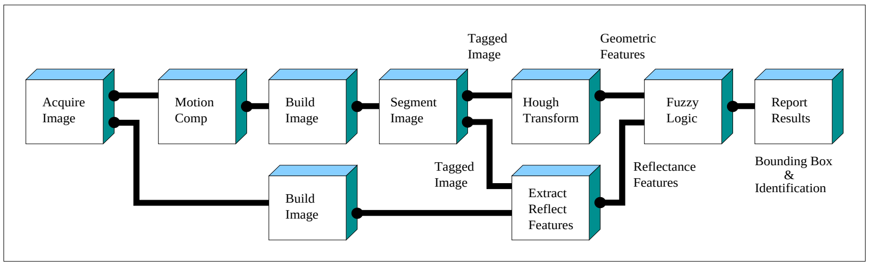

Figure 1. CMT LADAR ATR Algorithm

2.2.3. Segmentation

The detection algorithm segments the scene into labeled regions of interest using motion compensated X, Y, and Z images and an estimate of seeker position in the same coordinate system. The outputs of this algorithm are the reconstructed range image, a tag list used by the classification and other tools to quickly index to sub-image regions, and the tagged region image itself. All of these steps are done in a sparse X, Y array in NED space, which is effectively a nadir view.

The tagging process groups pixels within a pre-specified ground grid size into bins using the X, Y, and Z pixel values. Eliminating pixels that vary significantly from other members of the bin discards noise and dropout pixels. The bins are classified as object types, such as “shadow”, or “tall” based on their neighbors. Adjacent bins are then combined based on their relation in height and direction. Estimation of the ground plane is performed, and that segment of the image is removed. Any remaining objects are examined for potential similar measures and grouped accordingly. At this point, objects that are too large or too small are rejected, resulting in only target-sized objects left for the classifier to analyze.

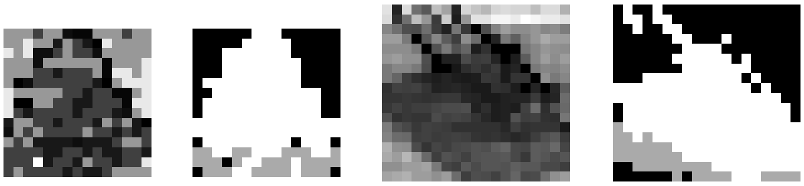

Figure 2. Clutter: (a) intensity (b) tagged Target:(c) intensity (d) tagged

The segmentor produced a list of *regions of interest* (ROI). These ROIs contained window coordinates into the X, Y, and Z images. Additionally the segmentor produced a “tagged” image. This was an integer image that represented the ROIs. Each segmented ROI, as well as the background, was labeled with a unique identifier. Additionally, any area adjacent to the target that is indeterminate from ground pixels receives a label that is unique, but correlated to the target label, such that it would be analyzed either separately from or in conjunction with the target if desired.

This tagged image was used to mask the intensity image. This hierarchy gave us a simple way to determine the foreground and background pixels in the intensity image, See Figure 2. The example *clutter* image contained 131 object pixels, 29 shadow pixels and 65 background pixels. The example *target* image contained 178 object pixels, 34 shadow pixels and 130 background pixels.

2.2.4. Geometric Features

A Hough transform is used to generate the geometric features and to estimate target pose. The Hough returns both dimensional and positional information. Due to the possibility of occlusion it is not possible to simple label the longer of the two horizontal length measurement the long axis of the vehicle. Therefore two different aspects approximately 90 degrees apart are extracted from the histogram. Likewise, two more measurements are made approximately 180 degrees separated from the first two. All four of these measurements are presented to the classifier The extracted features were as follows:

- **Range** — average distance in meters between the detector and the object pixels in a given ROI
- **Elevation** — look-down angle of the ROI
- **Aspect angle** — two long axis and two short axis measurements. Zero is looking down the estimated long axis. Each of the four aspects has a measurement for dimension:
  - **Length** — long axis length
  - **Height** — length perpendicular to the long axis
  - **Width** — height above the estimated ground plane

These geometric features are used to classify ROI against tabulated geometries of known vehicles.

2.2.5. Reflectance Features

Two principle reflectance features were extracted from the intensity image. Mean reflectance was determined by examining only the intensity of the object pixels for an individual ROI. The object pixels were determined by the tagged image received from the segmentor, see Section 2.2.3. The second feature was contrast. This feature measured the relative reflectance between a target-like object and that of a "typical" background. Natural objects tended to have low contrast against the natural background. Variance of the object pixels was also examined. This feature did not have a significant effect on identification, within the limited training set. The improvement in identification was not deemed to offset the computational cost to include the additional feature, both to feature extraction and to the classifier itself.

2.2.6. Target Database

Table 1. TargetDB.dat - Target database

4 7 mods	path	length	width	height
SA13	/target	6.18	2.87	1.96
SA13 UP	/target	6.18	2.87	3.24
:	:	:	:	:
SA8	@clutter	8.50	3.38	4.13
ZIL	@clutter	6.63	3.00	2.85

The target database consisted of one vehicle entry per line, See Table 1. The first column provided the name of the target. The next column had multiple purposes. It could act as a pathname to a classifier data base, or simply as a target type tag. By using the tag *@clutter* a vehicle that matches that entry could be rejected. It was determined that to positively identify a vehicle and then call it a non-target was superior in performance than to reject a vehicle because it did not match one in the database. It is important to note that the purpose of incorporating reflectance features was to help distinguish natural from man made. By placing a non-target vehicle in this database it reduced the classifier's confusion to be told that reflectance features characteristic to a man-made object should be rejected as clutter. The final columns contained the tabulated *length, width,* and *height* of the target vehicle. Originally this was just the tabulated dimensions of the vehicle.[1] There was a near linear scaling factor noticed between the tabulated values and the actual values produced by the Hough.

2.2.7. Model Matching

The target classifier is not a classifier in the traditional sense. The classifier did not group targets based on their *height, length, and width.* Instead, for each tabulated vehicle in the database an error between the measured value and the tabulated value was determined. These errors were presented to the classifier in order to receive a score that determined how well the measured values "matched" the tabulated values. The *range, elevation and aspect* modified the allowable error in *height, length, and width.* The advantage to using fuzzy-logic was that often two really small errors out-weighed three average errors. This helped make matches when one of the three dimensions was partially occluded.

In this sense the classifier rules reflected less knowledge about the sizes of particular vehicles and more knowledge about the sensor's and the Hough's ability to correctly measure geometric features. This allowed the addition of a new target vehicle not contained in the training runs, see Section 6.3. Using only geometric features it was a straight forward task to design a rule base for transforming the geometric measurements into scores.

The principle problem with using only geometry for ATR is that a 6.6 x 3.0 x 2.8 meter bush could easily be classified as a 5-ton truck. Table 3, in Section 6.1, illustrated the limited performance of the ATR with the ground truthed data available.

Material information about the ROI was necessary. This data was available from the intensity channel of data. By fusing the geometric features with reflectance features extracted from the intensity image the clutter was successfully separated from the targets, see Section 6.2.

3. FUZZY-LOGIC

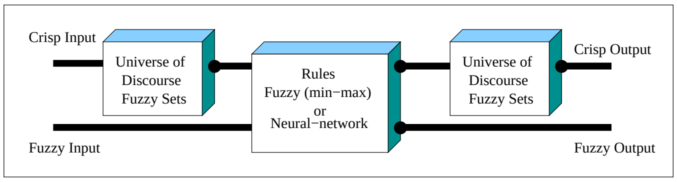

Figure 3. Fuzzy-logic allows mixed mode Input/Output

Fuzzy-logic is proven classifier technology. It has the ability to operate in environments of incomplete data. The model matching requirement of the CMT ATR required that the system be able to make partial matches. If one dimension of the target is obscured behind clutter, the ATR system must be capable of making a match based on the remaining two dimensions.

A typical fuzzy rule based systems has three major components, see Figure 3. First, crisp numbers are assigned fuzzy memberships. Second, these memberships are applied to the rule base. Rules are traditionally expressed in Min–max pairs. That is; If condition (A) and condition (B) then perform response (C). The three basic fuzzy rules correspond with their boolean counter parts. Correspondence occurs when fuzzy membership is set to the extreme values of (0) for no membership, or (1) for complete membership. When this condition is met the fuzzy operator must provide the same response as its boolean counter part:

$$\begin{aligned} \mathrm{AND}(a,b) &= \min(a,b) \\ \mathrm{OR}(a,b) &= \max(a,b) \\ \mathrm{NOT}(a) &= 1-a \end{aligned} \tag{1}$$

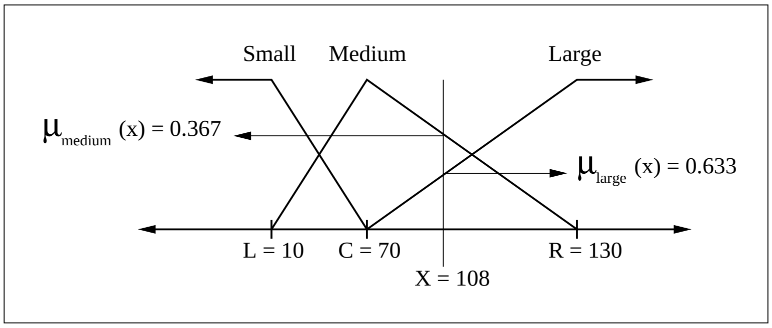

Figure 4. Determining Fuzzy Memberships

3.1. Fuzzy Memberships

Figure 4 is a collection of fuzzy sets known as a *universe of discourse* (UOD). A crisp numerical value has a degree of membership between zero and one in each of the fuzzy–sets within a UOD. To say that 108 has a degree of membership equal to 0.633 in the set of large is not the same as saying that 108 is 63 percent large. Memberships are degrees of belonging not probabilities. In boolean logic there would be a threshold at 100. A value of 108 would be labeled *true* for large and *false* for medium and small. The rules would then only operate on the boolean values, producing a *true* or *false* result for each rule. Fuzzy logic allows rules to operate on the memberships resulting in varying memberships of output. These outputs are gathered using a centroid operator to produce a crisp output.

3.2. Replacing Rules with Neural Network

Table 2. Require Number of Rules

sets / UOD	# of inputs	# of rules
3	3	27
3	4	81
3	5	243
3	6	729

There are three reasons for replacing a fuzzy–rules–table with a learning network. The first was a practical reason. As the number of inputs to a classifier increase the number of rules required to describe the mapping increases exponentially, see Table 2. The second was related to the fist as it involves the speed of processing. A feed forward network consisting of only 28 neurons is far less expensive, computationally, then processing 729 fuzzy rules, see Section 4.1. The third reason was a requirement that the classifier learn. The process of mapping errors in geometric measurements was well understood. The mapping of reflectance characteristics to dividing man–made objects from natural clutter was not as easily understood. This required a classifier that could "learn" this process when exposed to ground truthed data. Feed forward neural–networks have been shown to learn relationships in very non–linear systems.

3.3. Neural-network

Each layer of a neural–network is made up of *processing elements* (PE) or neurons. Each PE is connected to the outputs of all of the PEs from the previous layer. Each connection is weighted. The weights are multiplied by the output of the connected PE, these products are summed and added to a bias.

$$I_{\text{layer,node}} = W_{\text{layer-1,node}} + \sum_{j=0}^{N_{\text{layer-1}}-1} W_{\text{layer,node,j}} \cdot X_{\text{layer-1,j}} \tag{2}$$

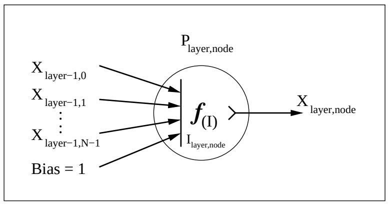

Figure 5. Neural–network Neuron

This sum is then "squashed" by a transfer function. The transfer function is designed to produce a bounded output from an unbounded input.

$$X_{\text{layer,node}} = f_{\text{layer}}\left(I_{\text{layer,node}}\right) \tag{3}$$

In this hybrid classifier, each layer had a unique transfer function, see Figure 6.

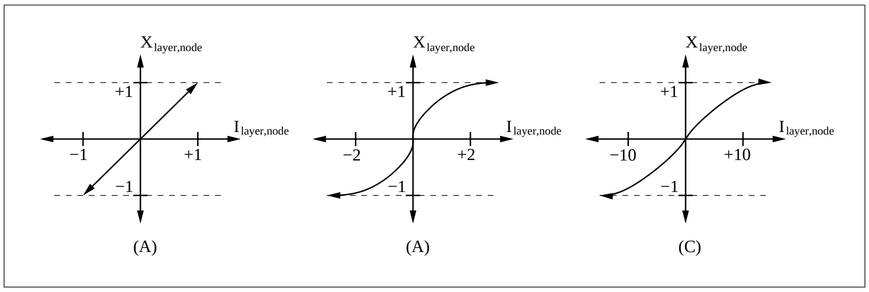

Figure 6. Transfer Functions

The input layer received unweighted numbers that were bound between *zero* and *one*. Therefore, the transfer function was linear with no scaling. This had the effect of making the input layer a layer of distribution nodes. The hidden layer or the first processing layer used a sigmoidal transfer function with a steep of thresholding characteristic. The output layer, or second processing layer, had a central slope to ensure a gradual spread from negative–one to positive–one assuming eight connections and one bias with weight bounded between negative–one and positive–one.

4. FEATURE LEVEL DATA FUSION

Fusion was performed at the feature level rather than the pixel or data level. The advantage to this is that the source data need not be registered. By this it is meant that resolution, and special orientation are not necessary. When data pixels are combined it is to create a new image with data that is derived in some mathematical mapping between the source images. When features are independently extracted from differing types of images that quantitative association between pixels in not necessary. The only thing that is required is that the two ROIs cover the same target.

In this case, the data came from the same sensor so it was accepted that a segmented ROI was the same in both the range and the intensity images. Geometric features from the range data were combined with the reflectance features extracted from the intensity image. These features made up the inputs to the fuzzy-neural classifier.

4.1. Fuzzy-neural Classifier

The heart of the LADAR ATR system is the target classifier. This classifier attempts to match measured values to those in a database and provide the system with a degree of match (*or a score.*) The crisp inputs are presented to fuzzy UODs to produce degrees of membership in fuzzy sets, see Section 3.1.

The first input is the aspect angle of the target in the ROI. This input is applied to a cyclic fuzzy UOD. The next three inputs are the errors between measured dimensions of *length*, *width*, and *height* and the tabulated values

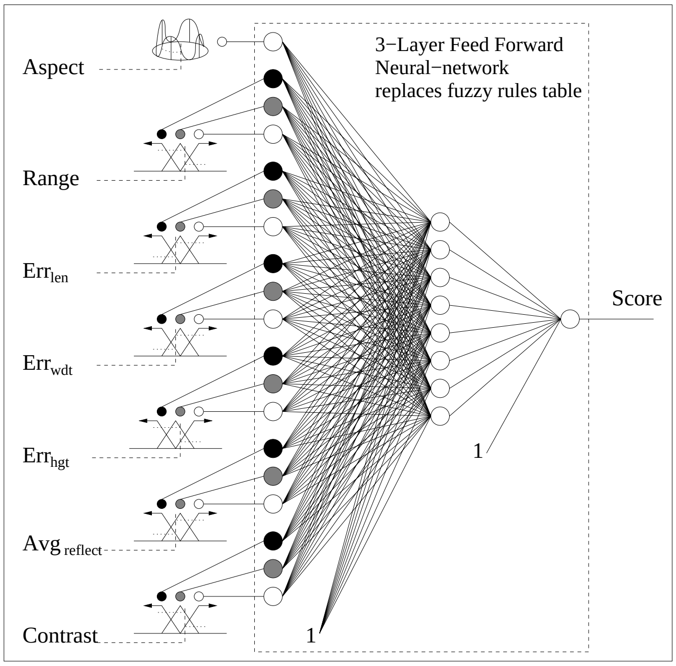

Figure 7. Hybrid Fuzzy-neural Classifier

in the target database. The next input is the average range of the pixels in the segmented ROI. The last two inputs are the extracted reflectance features, reflectance, and contrast.

The fuzzy UODs transform the seven crisp values into nineteen fuzzy memberships. These memberships are processed by the rules. In this case the rules are implemented by a three layer fully interconnected feed forward neural network. The number of input layers is governed by the number of features being classified. The output is a single score. The number of hidden layers began by using the rule of thumb that states the number of hidden layer neurons should be $\sqrt{n \cdot m}$. Where (n) is the number of inputs and (m) is the number of outputs.[2] The number was then manually adjusted up and down to produce the desired classification. Eight was determined to be the optimal number. Seven significantly decreased the performance, and nine did not show an improvement worth the computational expense.

4.2. Adjustable Parameters

The classifier is defined by a weights file that contains the adjustable parameters. This file contains thresholds for *length width* and *height* measurement errors, see Section 2.2.7. If a measured dimension exceeds any one of these numbers the ROI is declared clutter and no further processing is made. There also Any classifier score below this number is identified as clutter. This number may be manually adjusted to allow more or less clutter through the system. In addition this file contains the fuzzy set parameters. The low and high fuzzy sets for each geometric and reflectance feature were defined statistically from the test data. The middle fuzzy set for each feature was an adjustable parameter. Finally, the file contained the weights for the neural-network. The topology of the network was fixed but the weights were adjustable. In all there are 179 adjustable floating point numbers that define the behavior of the hybrid fuzzy–neural classifier for the LADAR ATR system.

5. TRAINING

5.1. Tower Data

The training set data was derived from the manual ground truth of data collected for a similar LADAR sensor. Data was collected from a prototype LADAR. The LADAR was mounted on a tower. The targets were stationary in a field with trees and other clutter, at various articulations and ranges. From the images that were acquired a manual ground truth was performed. Approximately 195 examples of clutter were manually verified. Approximately 62 targets were manually identified and truthed. This constituted the entire training set.

5.2. Problems for BPN with small Ground Truth

Back-propagation requires, like most gradient decent models, a great deal of training set data. Typically, there needs to be on the order of thousands exemplars per input of data to get proper convergence. Additionally to support generalization and prevent building an interpolative network it is required to have approximately half again as many exemplars for testing purposes.[2]

Initial training sessions showed very poor convergence. There was not enough data to split off a separate set of data to be used only for testing. There was also the problem of generating a derivative for the various inputs. A non gradient method needed to be employed. In a previous project by this author a genetic algorithm was used to train the weights of a neural-network classifier for line quality assessment in autonomous image registration. The genetic algorithm proved superior to back propagation due to the limited training set and also the non uniformity of the exemplars to cover the input space. The ground truthed data used here had a similar bias. There was more than three times the amount of clutter then targets. It was decided that a Genetic Algorithm should be considered.

5.3. GA Solution

Genetic Algorithm (GA) is a search algorithm that borrows its operators from the biological model of natural selection and evolution.[3] GA have been very successful in combinatorially–explosive problems such as; Job Shop Scheduling,[4–7] The Non-linear Transportation Model,[8–10] and the Traveling Salesman Problem[11–13].[14–17]

GA differ from traditional optimization and search algorithms[3]:

Genetic Algorithm	**Traditional Optimization and Search**
work with coding of the parameters	manipulate parameters directly
search from a population of points	use a single point to search
use payoff (objective function) information	use derivatives or auxiliary knowledge
use probabilistic transition rules	use deterministic rules

Genetic algorithms are very different form traditional calculus based approaches to optimization. While most calculus based algorithms operate directly on the parameters to be optimized, genetic algorithms may operate on representations of those parameters. Instead of a single point search as in conventional algorithms, genetic algorithms perform population searches. This invites the use of massively parallel processing. Each member of the population may be evaluated simultaneously. Genetic algorithms may be guided by more "qualitative" evaluation than "quantitative." This means that metrics are generated as a function of final performance. Conventional algorithms often need knowledge of the derivative affects of the individual parameters on the final metric for performance. Genetic algorithms may optimize a set of initial parameters for a model whose internal characteristics are completely unknown. Very non–linear systems may thus be optimized with relative ease. Another advantage is that calculus base optimization almost always requires the score surface to be continuous. With GAs it is only necessary to be able to "hold up" two candidate solutions and "say" which of the two is better.

5.3.1. Operators

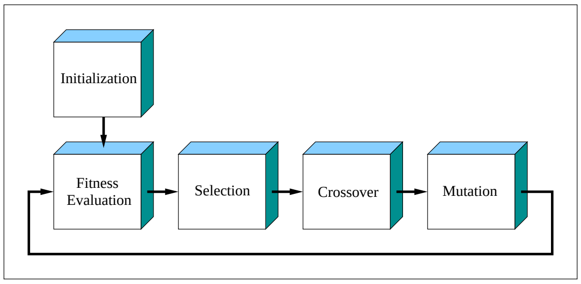

Figure 8. Genetic Algorithm

Similar to simulated annealing, GA uses a population of possible solutions to search the solution space. GA uses biologically inspired operators to improve the search, see Figure 8. These operators alter the population and allow information about possible solutions to be shared amongst the individuals. The basic GA has the following operators that affect the population:

- **Initialization** – The initial generation should be uniformly distributed across the parameter space.
- **Fitness Evaluation** – Each member of the population is a potential solution. Fitness is assigned to each trail solution.
- **Selection** – Allows individuals with higher fitness to have an exponentially greater chance of participating in the next generation.
- **Cross–over** – Individual that have been selected may share information about the solution space.
- **Mutation** – Individuals in the next generation make small moves about the parameter space.

Each of the following sections will begin with a statement describing the function of the particular operator as it pertains to genetic algorithms in general. This is followed by a discussion of how the operator has been implemented to handle this particular problem, thus making this tool a hybrid genetic algorithm.

Representation: Each member of the population is expressed as a set of adjustable parameters for a model. Each element of this set is referred to as an allele. The adjustable parameters in this ATR classifier were all floating point numbers, see Section 4.2. A variant of the simple genetic algorithm was used. The floating point genetic algorithm uses an array of floating–point numbers as its allele representation. The subsequent operators are all modified to accommodate the floating–point math as opposed to boolean operations performed on the alleles of a simple genetic algorithm.

Selection Individuals from the current population must be selected to generate the population that represents the next generation. Selection schemes must be designed to insure that highly fit individuals are selected at an exponentially greater rate than lower fit individuals. Low fit individuals must have a finite chance of being selected. While their overall fitness scores may be low, they may contain "pieces" of the optimal solution. This is the key premise in Holland's *Schema Theorem*.[18]

Tournament selection was chosen as the selection scheme. Tournament selection is superior to other stochastic selection schemes, because it readily maximizes as well as minimizes fitness scores. Individuals do not need to be ranked against the population as a whole. This allows the use of more "qualitative" and less "quantitative" means of determining fitness, see Section 5.3.2.

Selection is made uniformly and randomly with replacement. A fixed number of individuals are selected from the population and their relative fitness is examined. A functioned named *isLeftbetter* was created to assist the programmer in building the selection process. When presented with two selected parameter lists the function returns a boolean value indicating whether the left argument to the function call was the population index for the better of the two parameter lists when used to define the ATR classifier. The winning parameter set is then held out as parent number one. The process is repeated for parent number two, with that added test to insure that the two parents were not the same parameter list. These two parents were then used in the cross–over function to generate the next generation, see above.

Crossover: Members of the population may exchange information and explore new regions of the solution space. Members of the population are selected based on their relative fitness, see Section 5.3.1 (Selection). There is a finite probability that these members will simply be cloned into the next generation, thus preserving their representations. Otherwise they exchange information about their parameter lists to generate wholly new population members. The probability that the selected parents would undergo cross–over, thereby producing children, was set at 95 percent for this application. This number showed particularly fast convergence. An analysis of adjusting probabilities and the affect on convergence rates is not within the scope of this paper.

The adjustable parameters in this ATR classifier were all floating point numbers, see Section 4.2. Therefore the cross–over operator was implemented using floating–point parametric crossover.

$$\begin{array}{lllll} P_1 = \bar{X} = \{x_1, x_2, x_3, ..., x_n\} & C_1 & = & (1-t)\,\bar{X} + (t)\,\bar{Y} \\ P_2 = \bar{Y} = \{y_1, y_2, y_3, ..., x_n\} & C_2 & = & (1-t)\,\bar{Y} + (t)\,\bar{X} \end{array} \qquad (4)$$

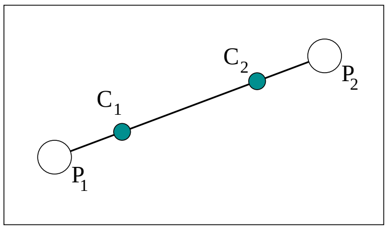

Figure 9. Floating point parametric crossover

Figure 9 graphically illustrates the results of the above equations. The two parents represent two points in n–dimensional parameter space. A uniform random number between *zero* and *one* was generated. This became the parametric index (t). The first child is then found on a line between the two parents. The second child is symmetrically found on the same line.

Mutation: This the operator introduces change into the population. This allows the genetic algorithm to explore new regions of the solution space that might not be accessible through the cross–over operator. The adjustable parameters in this ATR classifier were all floating point numbers, see Section 4.2. The mutation operator was implemented using the addition of Gaussian noise. The probability of mutation was applied to each allele of each representation of parameter set. For this reason the probability of mutation was held very low. Less than five percent of all adjustable parameters were mutated. Mutation was a very small amount of Gaussian noise as defined by the following equation.

$$x_{i,t+1} = \mathrm{rndGauss}\left(\mathrm{low}_i, \mathrm{high}_i, x_{i,t}, \sigma_t\right) \tag{5}$$

The current value for an allele was the mean. The standard deviation was reduced uniformly from 0.1 to 0.0 as a function of the number of generations into the evolution cycle. The low and high boundaries were set individually for each adjustable parameter.

The mutation operator was also used in the initialization of the first population. Half of the initial population had its adjustable parameters set completely at random and uniformly within the low and high bounds of each parameter. The second half of the population was initialized by adding Gaussian noise to a single provided parameter list. This list may have been a human's best guess or it may have been the best parameter list that was found in a previous training run. This illustrates one of the features that makes genetic algorithms superior to other optimization algorithms. The training may begin from a previous known solution or even a human's "best guess."

5.3.2. Metrics

Metrics are the most important part of using a genetic algorithm to optimize a set of parameters. It is absolutely necessary to be able to "hold up" two models with different adjustable parameter values and "say" which of the two is better.

Tournament selection allowed the use of a "heuristic sequence" rather than an equation to determine fitness. Three key points were examined in sequence to determine which parameter set performed better as a target classifier:

1. How many objects passed as targets were correctly identified?
2. How many targets were identified as clutter?
3. How many pieces of clutter were accepted as targets?

Determination of which which parameter set was better began with looking at the reporting tables, see Tables 3 & 4. The chief determining factor was how many targets where correctly identified. If one classifier correctly identified targets more than the other, that classifier was declared better. If both classifiers identified that same number of targets correctly then the next metric was examined. Determination of the better classifier was then made by minimizing the number of targets that were declared as clutter. If both of these metrics were equal for the tested classifiers then the determining metric became how much clutter was being accepted by the classifiers and identified as targets. Finally, as an arbitrary decision for uniformity, if all metrics were equal the "left" classifier was declared the winner, see Section 5.3.1 (Selection).

6. RESULTS

6.1. Geometric Features Only

Table 3. Geometric Features Only

ID = 35	Clutter	Target	
Reject	130	10	140
pass	65	52	117
	195	62	

The performance of the original classifier with only geometric features was less than desirable. The system passed 65 out of 195 tagged regions as targets when they were ground truthed to be trees or bushes. Also, 10 of the ground truthed targets where rejected as being clutter. After, analyzing the rejected targets it was found that most had large errors in at least two of their dimensions. The clutter that was passed was found to have at least two very good matches in each dimension. it was clear that other features would be needed to reject the natural clutter from the man made objects.

6.2. Fusion with Reflectance Features

Table 4. Geometric + Reflectance Features

ID = 58	Clutter	Target	
Reject	195	0	195
pass	0	62	62
	195	62	

After the reflectance features were added to the classifier and the classifier was trained on the ground truthed data the performance was drastically improved. All clutter items were rejected as clutter. All of the "man made" objects were passed as targets. 58 of the 62 targets were then correctly identified. Three of the SA-6 radars were identified as SA-6 TELs and one of the "Specials" was reported in the wrong articulation.

6.3. Conclusion

The analysis and experiments performed during the CMT project have effectively demonstrated the use of a hybrid fuzzy-neural network for critical mobile target identification for LADAR sensors. Additionally, we have illustrated the ability to train such a classifier with additional features available from that sensor to improve performance.

This classifier has since been tested on other target types in various conditions, and has illustrated relatively robust performance in clutter rejection, identification, and articulation discrimination within classes. The fact that features can be easily added and the system may be relatively quickly re-trained will provide ample opportunity for further feature extraction for classification, particularly if improvements in detection allow increased throughput or data integrity.

Section 2.2.7 mentioned that the classifier "learned" the capabilities of the sensor and segmentor to correctly measure a vehicles dimensions. This created a robustness that was demonstrated during field tests. The dimensions of the SA8 were removed from the target database. A new entry was made for SA13s. SA13s were not part of the limited training set, see Section 5.1. The ATR system proved to be able to identify the SA13 during subsequent field tests.

ACKNOWLEDGMENTS

The authors would like to thank the supporting staff for the ATR CMT team, which included members from various states, as well as functional and program organizations. From the Raytheon Engineering Directorate in Tucson, AZ, Dave Bujak, Brian Lacy, Lee McDonald, Sandra Morris, and David Vega provided invaluable software, and algorithm support. Thanks are also due to the Advanced Programs management teams, who provided funding and moral support during this effort.

REFERENCES

1. C. F. Foss, *Jane's Tank & Combat Vehicle Recognition Guide*, Harper Collins, Glasgow, 1996. ISBN: 0–00–470995–0.
2. R. Hecht-Nielsen, *Neurocomputing*, Addison Wesley, Reading, MA, 1990.
3. D. E. Goldberg, *Genetic Algorithms, in Search, Optimization & Machine Learning*, Addison–Wesley Publishing Company, Inc., Massachusetts, 1989.
4. G. E. Liepins, M. R. Hilliard, and M. M., "Greedy genetics," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 231–235, Lawrence Erlbaum Associates, 1987.
5. D. Smith and L. Davis, "Adaptive design for layout synthesis," internal report, Texas Instruments, Dallas, 1985.
6. D. Whitley and T. Starkweather, "Genitor ii: A distributed genetic algorithm," *J. Expt. Thoer. Artif. Intell.* **2**, pp. 189–214, 1989.
7. G. A. Cleveland and S. F. Smith, "Using genetic algorithms to schedule flow shop releases," in *Proceedings of the Third International Conference for Genetic Algorithms*, Schafer, ed., pp. 160–169, Morgan Kaufmann Publishers, Inc., 1989.
8. Z. Michalewicz and C. Z. Janikow, "Genetic algorithms for numerical optimization," *Statistics and Computers* **1**(2), 19xx.
9. Z. Michalewicz, C. Z. Janikow, and K. J. B., "A modified genetic algorithm for optimal control problems," *29th CDC* , 1990.
10. Z. Michalewicz, G. A. Vignaux, and M. Hobbs, "A non-standard genetic algorithm for the nonlinear transportation problem," *Accepted for Publication in the ORSA Journal of Computing* , 19xx.
11. D. E. Goldberg and L. R., "Alleles, loci and the traveling salesman problem," in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Grefebstette, ed., Lawrence Erlbaum Associates, 1985.
12. I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference of Genetic Algorithms*, Grefenstette, ed., pp. 224–230, Lawerence Erlbaum Associates, 1987.
13. J. J. Grefenstette, R. Gopal, R. Rosmaita, and G. D. V., "Genetic algorithms for the traveling salesman problem," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Grefebstette, ed., pp. 154–159, Texas Instrument and U.S. Navy Center for Applied Research and Artificial Intelligence, 1985.
14. J. P., J. Y. Suh, and V. D., "The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem," in *Proceedings of the Third International Conference for Genetic Algorithms*, S. J. D., ed., pp. 110–115, Morgan Kauffmann Publishers, Inc., 1989.
15. G. E. Liepins, M. R. Hilliard, J. Richrdson, and M. Palmer, "Genetic algorithm applications to set covering and traveling salesman problems," in *Operations Research and Artificial Intelligence: the Integration of Problem Solving Strategies*, Brown and White, eds., pp. 29–57, Kluwer Academic Press, 1990.
16. M. L. Lid, "Traveling salesman problem domain application of a fundamentally new approach to utilizing genetic algorithms," tech. rep., Air Force Office of Scientific Research and Office of Naval Research, 1991. Research sponsored by Contract F4920–90–G–0033.
17. D. Whitley, T. Starkweather, and F. D., "Scheduling problems and traveling salesman: the genetic edge recombination operator," in *Proceedings of the Third International Conference for Genetic Algorithms*, Schafer, ed., pp. 133–140, Morgan Kaufmann Publishers, Inc., 1989.
18. J. H. Holland, *Hidden Order, How Adaptation Builds Complexity*, Addison–Wesley Publishing Company, 1995.