A fuzzy image algebra neural network for target classification

by

Rashmi Srivastava

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Electrical Engineering and Computer Engineering
Major: Electrical Engineering

Iowa State University
Ames, Iowa

1994

Dedicated to my parents

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis presents a neural network application to target classification using a new type of neural network called the Fuzzy Image Algebra Neural Network (FIANN). Most pattern classification schemes make use of linear convolutional values or morphological structuring elements as pattern features that can be learned by a neural net. The FIANN, however, has an activation function based on the generalized mean. This function theoretically offers a variety of mapping functions, ranging from linear to the fuzzy morphological. The FIANN is used in a heterogenous network structure. The first layer of the net performs linear template operations, while the remaining aggregation layers are used for classification. Generalized image algebra operations are used to obtain fuzzy morphological or linear operations. The parameters for the generalized operations are learned in a fashion similar to standard backpropagation, but with training rules based on a combination of stochastic learning and gradient descent techniques. The type of data used is the range data part of tank LADAR data. The objective is to classify the tanks by type. The range data is first converted to elevation data, which is input to the net for classification. A two tiered approach is used. First, the elevation data is divided into zones, with the expectation that a partially occluded object can still carry forward enough information from fewer zones for object identification. The aggregation layer of the net is trained using gradient descent. The decision-making layers are trained using a combination of stochastic learning and gradient descent. Results obtained show that with proper training and selection of parameters, the network can perform very good classification.

PROGRAM OF STUDY COMMITTEE

Major Professor

J. Davidson

Committee Members

L. Udpa

E. A. Stanley

I. INTRODUCTION

An *artificial neural network* (ANN) is a complex dynamical system that is inherently parallel in nature and can perform a variety of tasks such as pattern recognition, optimization, and function determination. By nature of its methodology, an ANN solution can be more robust in nature than those from conventional methods. ANNs can be used for the solution of problems where the data may be noisy, incomplete, as well as where classical methods perform poorly, such as in *automatic target recognition* (ATR) problems.

ATR requires the extraction of important information from the image of a target or a vehicle. The data is often complex and uncertain, and traditional solutions such as statistical pattern recognition and rule based artificial intelligence techniques often do not provide adequate solutions. Neural network technology provides a number of tools which can be applied successfully to ATR problems. Each target in an ATR system has a signature associated with it. ATR requires a solution system that describe targets and background completely, and yet are robust to signature and environmental variations. The system should be capable of adapting to additional targets and environments. Neural networks are robust by nature and also have the capability to "learn" hence, they form a good medium for solving ATR problems.

This thesis presents a neural network application to target classification using a new type of neural network called the *Fuzzy Image Algebra Neural Network* (FIANN) [1]. Most pattern classification schemes make use of linear convolutional values or morphological structuring elements as pattern features that can be learned by a neural net. The FIANN, however, has an activation function based on the *generalized mean.* This function theoretically offers a variety of mapping functions, ranging from linear to the nonlinear fuzzy morphological.

Theoretically, the network is data independent and can be applied to a wide variety of pattern recognition problems.

The United States Air Force (USAF) is interested in new methodologies that are both versatile and accurate in solving ATR problems. This thesis details the research carried out in applying the FIANN to solve a tank classification problem on *laser radar* (LADAR) data supplied by the USAF. LADAR range images provide a topographic mapping of the visible portion of the target through direct measurement of the range to points on the target. The LADAR data used for this investigation has the added advantage of low expense and high resolution. The objective is to classify the tanks in the image data by type. The range data is first converted to elevation data, which is input to the net for classification. A two tiered approach is used, consisting of a feature extractor followed by a pattern classifier. First, a subimage of the target is extracted and converted into elevation form. That subimage is divided into overlapping rectangular zones, with the expectation that a partially occluded object can carry forward enough information from fewer zones to the neural network for object identification. Linear convolution is carried out on each zone to provide feature information about the object, and the convolution values are passed to the output layer of the network, which performs pattern classification on the features.

Previous approaches to ATR on range images have used model based approaches where the target is matched with 2–dimensional and 3–dimensional target templates [2], or image features are matched against model features based on an object description called attributed relational graph (ARG) [3]. Morphological approaches make use of structuring elements, which are ideally suited for analysis of range images due to their inherently geometric nature [4]. Also, they can handle features of known physical size as shown in [5].

The architecture of the FIANN is heterogenous. Generalized *image algebra* operations are used to obtain *fuzzy morphological* or *linear* operations. The parameters for the generalized operations are learned in a fashion similar to standard *backpropagation*. The lower layer of the net is trained using gradient descent. The upper, decision-making layers are trained using a combination of stochastic learning and gradient descent. These learning rules were developed after investigation of a number of different training schemes, including simulated annealing, gradient descent, and stochastic learning. Results are presented for the alternative learning schemes.

Problem Statement

The FIANN's performance in ATR problems is evaluated. The specific problem is to perform target classification on LADAR range data using one form of the FIANN.
Data: LADAR data was used for this work. It consists of two parts: a spatial location, and the range or distance measurement from the laser radar to the location. The data is converted to an elevation map, so as to obtain shape and surface information on the target. Several target classes include M60 and M47 tanks, M42 missile launchers, M53 artillery, and others, as well as one nontarget class consisting of natural terrain, such as trees, bushes and sand. The elevation map is converted to a standard size 30 x 50 for input to the FIANN.
FIANN: The network activation function is called *weighted fuzzy erosion*, and is based on the generalized mean. Theoretically, this can approximate mapping functions ranging from linear to morphological nonlinear. This allows the net to perform independent of the data it receives as input. The network architecture is based on the standard multilayer perceptron using backpropagation. Training rules, however, are modified to suit the weighted fuzzy erosion parameters. Training is carried out using a combination of stochastic learning,

gradient descent and the backpropagation delta rule. This is described in detail in chapters VIII and IX.

Figure 1 gives an overview of the problem solution. Each of the blocks are discussed in detail in the following chapters.

The thesis is organized as follows: Chapter II gives a background review of the different approaches to ATR, with emphasis on neural network based approaches. The chapter gives information on ongoing research involving range data. Chapter III gives an overview of the image algebra and the next chapter gives a brief overview of fuzzy logic. Chapter V gives a description of the data used in the research and the preprocessing necessary to prepare it for input to the FIANN. Chapter VI introduces neural networks, their methodology and describes in detail the backpropagation net. Chapter VII details the complete theory behind the FIANN and its important properties. Chapter VIII details the architecture and training scheme for the neural network used in this specific target identification problem. Finally, Chapter IX gives details of the implementation and the results obtained. The last chapter suggests directions for possible future research.

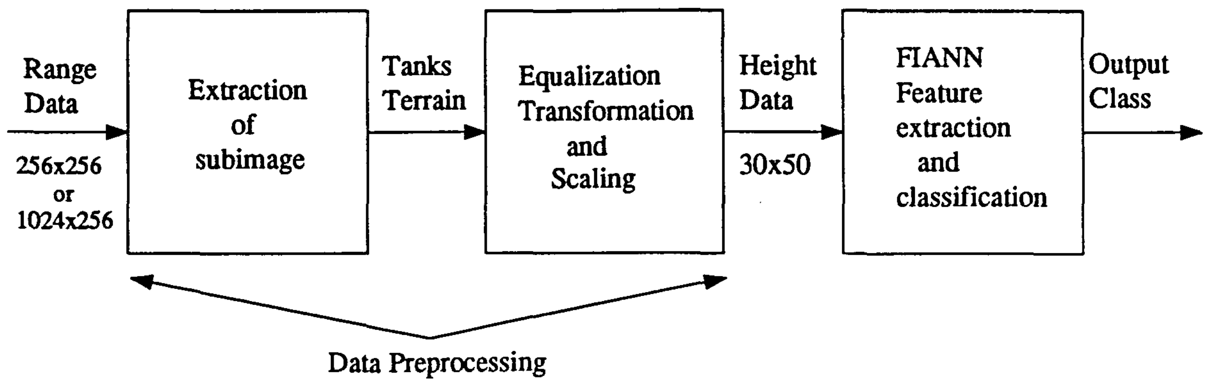


Figure 1. Block diagram for the ATR problem.

II. LITERATURE REVIEW

The target classification solution presented in this thesis has a two tiered approach. First, the available range data is converted to elevation data for better object surface understanding, and, second, this data is classified using the fuzzy image algebra neural network. The literature survey showed that much research using neural nets for Automatic Target Recognition (ATR) is being investigated. However, this is a relatively recent development and not much research using range data in neural networks for ATR is available.

ATR has been an area of research for longer than the last two decades. The field, which originally started with the goal of automatic target cueing to aid helicopter pilots in air-to-ground combat situations, has led to the creation of several generations of automatic targeting systems that have been effective and useful mostly in localized domains. The realization of the ultimate goals of ATR — the detection, tracking, and recognition of targets of interest in all and every scene, scenario, and weather condition — however, has proven to be difficult. This is partly due to the low signal quality used for ATR, which is caused by many elements such as phenomenological parameters, natural and intentional occlusions, and so forth. Much research is being performed to try to improve signal quality for ATR. This includes improving sensor design, more efficient signal retrieval, and data fusion, where multiple sensors and multiple signals are combined to achieve better results. Another approach to ATR uses pattern classification systems which are robust in nature and can adapt themselves to noisy data. These include neural networks and multisensor/multisource information fusion systems.

Most ATR solutions base their approach on the particular type of sensors used for collecting the data, as well as the form in which the data is collected. Algorithm design varies from model-based to texture-based approaches. Object detection using remotely sensed images is a complex process involving a number of distinct function modules, which include: (1) the physical scene, comprising the objects of interest and the background; (2) the sensor system, which generates signals proportional to the reflected or emitted radiation from the scene; and (3) the data processing and interpretation. This last module discriminates measurements associated with objects of interest from those associated with background events. For successful detection of objects and rejection of false alarms, it must be assumed that objects possess distinctive “signatures” in some measurement space. The discrimination task is not a simple one-dimensional thresholding function to separate signal from noise, but involves a series of steps that first convert the sensory signals into meaningful measurements and then analyze these measurements, to assign them to either an object or background class.

One approach to ATR is the model based approach used in [6]. This details an experimental target recognition system (XTRS) capable of detecting and recognizing armored tactical vehicles in range and intensity images provided by ground based and airborne laser radars. XTRS is a model-based recognition system in which the key components are an object representation scheme and a generic matching engine.

Recent neural net approaches to ATR include the Infrared (IR) based SAHTIRN target recognition system [7] being developed at Hughes Aircraft Company. SAHTIRN stands for Self Adaptive Hierarchal Target Identification and Recognition Neural network. This system combines three parts: (1) a vision segmenter, (2) a hierarchal feature extraction and pattern recognition system, and (3) a pattern classifier based on the backpropagation algorithm. This system is undergoing continual development and is being tested on ground vehicular targets using terrain board modeled IR imagery. These are images simulated using scenes filmed with a camera. The image is oriented in several different directions, creating a comprehensive data base. Another neural net based system is MODALS (3–D multiple object detection and location systems) [8], which is being developed by Booz-Allen and Hamilton, Inc., under the visible sensor ANVIL (artificial neural vision learning) program. This system simultaneously detects, locates, segments, and identifies multiple targets. MODALS distinguishes itself from most ATR approaches by performing all these tasks simultaneously, unlike other systems which follow a serial approach. A serial approach can lead to cumulative errors, thereby reducing overall performance of the system.

Another approach to target recognition is the feature based approach. With this approach an attempt is made to discover features that are inherently invariant with respect to size and orientation of the object in the sensor's field of view. A feature based classifier is used to map the extracted set of feature values into various object classes. Two-dimensional moment variants [9] are a good example of this type of classifier. Another approach using template matching approach is based on matching the silhouette of the unknown object with those in a prestored library of templates of likely objects. Chamfer matching [10] is an example of this technique. Both of these approaches have shortcomings. One of the main disadvantages is the dependence on availability of information about the complete object. Performance of the feature-based system deteriorates considerably when only partial information is available. However, a combination of the two methods as proposed in [11] combines the advantages of the feature based method, namely invariance to size and orientation, with the advantages of silhouette based methods, particularly in dealing with partial information caused by occlusion and poor segmentation.

Another area of research in ATR is sensor fusion. Information from a number of different sensors is combined to give better results than by a single sensor. Different approaches include data fusion, feature fusion, and decision fusion, although at present most research focuses on the first two. An ATR system using sensor fusion is described in [12]. Finally, an approach to ATR using texture is detailed in [13]. In this approach, image characteristics, object characteristics, and detection methodology are assumed to be the variables associated with object detection. The development and extensive evaluation of an image clutter measure is studied.

ATR performance is limited both by the capabilities of the ATR algorithms as well as the information content provided by the sensor. In the case of laser radar sensors, these parameters include the range to the target, the aspect angle of the target, the atmospheric extinction rate, the sampling rate, and signal-to-noise ratio. A considerable amount of research has been carried out to find optimal ways of processing sensor information for ATR so as to reduce uncertainty in the signal. One way is to improve sensor design as proposed in [14]. Here, authors propose a technique for evaluating the information content of a sensor and study the intrinsic separability of laser radar data with a view towards its usefulness to "algorithm independent" ATR sensor design.

Much research using range data has focused on segmentation and edge detection of the images [15] [16]. Three-dimensional object recognition currently is not a very mature field. Some schemes handle only single, pre-segmented objects while others can interpret multiple object scenes [17] [18]. Some systems perform only 2–D processing using 3–D information [19]. Some methods require that intermediate data be provided by the person operating the system. Many techniques assume that idealized data is available from sensors and intermediate processors. Others require high contrast or backlit scenes. Most efforts have restricted the class of recognizable objects to spheres, cylinders, cones or a combination of these [20].

Information such as edge maps or planar region descriptions is often available from range data. Surface and 3–dimensional shape information can also be extracted from range images. In 1979, Duda et al. [21] discussed the use of registered range and reflectance images to find planar surfaces in 3–D scenes. A sequential planar region extraction procedure was used on images obtained from an amplitude-modulated imaging laser radar. This was one of the earliest papers to use range data for scene analysis. Milgrim and Bjorklund [22] presented a different approach to planar surface extraction from range images. Their system was planned for vehicle navigation. Spherical coordinate transformations were used to convert range data, azimuth angle, and elevation angle sensor data into Cartesian x, y, z coordinates. They used a more straightforward approach than Duda et al., since no *a priori* assumptions were made. Reeves et al. included range moments from range images to carry out moment based 3–D analysis [23]. Single object, pre-segmented, synthetic range images were classified using silhouette moments, range moments and a combination of the two.

Other approaches to ATR include statistical pattern recognition, syntactical pattern recognition, and expert systems. Table 1 gives a brief description of these approaches, their advantages, and the requirements for each approach.

The fuzzy image algebra neural net has been used previously for the ATR problem of tank classification [24]. A simple version of the FIANN performed classification on tank range data using features intrinsically determined by the neural net. Fairly positive results were achieved.

Table 1 Various approaches to ATR

ATR Approach	Working	Advantages	Disadvantages
Statistical Pattern Recognition	Computes discriminant function parameters to obtain primitive shape features.	Works well on patterns with well defined distributions.	Not a robust scheme. Very comprehensive data set is required.
Syntactic Pattern Recognition	Follows linguistic rules that describe target structure.	Rules can be created from limited data.	Does not work well on noisy data.
Expert Systems (Knowledge Base)	A data base of targets, scenes, etc., is created and used.	Can include "knowledge" not present in pattern recognition systems.	No clear cut rules for acquiring and representing knowledge.

III. IMAGE ALGEBRA

In recent years the use of image processing has increased considerably in the military, industrial, and academic fields. However, research and development of image processing algorithms has not made use of any standardized, mathematically rigorous, algebraic structure that is designed specifically for image manipulation. In response to this need for a common algebraic structure for image processing, the Air Force Armament Laboratory at Eglin Air Force Base, in conjunction with DARPA, called for the development of a highly structured mathematical environment for image processing and image analysis with the intent that the fully developed structure would subsequently form the basis of a common image processing language. The structure developed in response to this is *image algebra.*

Image algebra provides a common mathematical environment for image processing algorithm development as well as methodologies for algorithm optimization, comparison and performance evaluation.

Image algebra is a heterogenous algebra that has six basic types of operands. These are *value sets*, *point sets*, the elements of each of these sets, *images*, and *templates*.

Value Sets

Some common types of value sets are the set of integers $\mathbf{Z}$, the set of real numbers $\mathbf{R}$, the set of complex numbers $\mathbf{C}$, the set of binary numbers of fixed length k, $\mathbf{Z}_{2^k}$, $\mathbf{R}_{+\infty}^{\geqslant 0} \equiv \{+\infty\} \cup \{\mathbf{r} \in \mathbf{R} : \mathbf{r} \geqslant 0\}$, and extended real numbers (which include one of the symbols $-\infty$ or $+\infty$) denoted by $\mathbf{R}_{+\infty} = \mathbf{R} \cup \{+\infty\}$ and $\mathbf{R}_{-\infty} = \mathbf{R} \cup \{-\infty\}$. These sets correspond to values commonly encountered in image processing operations. However, in its most general form, image algebra allows for any semi-group $\mathbf{F}$ to be a value set. We denote an unspecified value set by **F**. The operations on and between elements of a given value set $\mathbf{F} \in \{\mathbf{Z}, \mathbf{R}, \mathbf{C}, \mathbf{Z}_{2^k}, \mathbf{R}_{-\infty}, \mathbf{R}_{+\infty}\}$ are the standard elementary operations associated with **F**. Thus, if **F** = **R**, then the operations could be the arithmetic and logic operations of addition, multiplication, and maximum, and the corresponding operations of subtraction, division and minimum. In addition to these elementary operations, image algebra also includes the operations of union ($\cup$), intersection ($\cap$), set subtraction ($\backslash$), choice function, and cardinality function on subsets of **F**. The choice function when applied to a set returns an arbitrary value from the set, while the cardinality function gives the number of elements in the set.

Point Sets

A point set is a subset of *n*-dimensional Euclidean space $\mathbf{R}^n$. The letters **X**, **Y**, and **W** are reserved to denote point sets. Point sets can be rectangular, hexagonal, or toroidal arrays, as well as infinite subsets of $\mathbf{R}^n$. Due to the wide variety of shapes, sizes and dimensions provided by point sets, image algebra can model and manipulate continuous as well as discrete images on any desired type of point set.

Image algebra operations on point sets are operations on subsets of point sets as well as operations between points. Operations on subsets of point sets are union, intersection, set difference, choice function, and cardinality function. Examples of image algebra operations on or between elements of point sets are vector addition, scalar and vector multiplication, and dot product.

Images

Images are the most fundamental of the image algebra's operands. The most general definition of an image involves both value sets and point sets. If we have a value set **F** and a point set **X**, then an **F** *valued image a* on **X** is the graph of a function a: $\mathbf{X} \rightarrow \mathbf{F}$. Thus, an **F** valued image a on **X** is of the form

$$a = \{(x, a(x)) : x \in \mathbf{X}\},$$

where $a(x) \in \mathbf{F}$.

The set **X** is called the set of *image points* of *a*, and the range of the function is the set of *image values* of *a*. An element (x, $a(x)$) of the image a is called a *picture element* or *pixel*, where x is the pixel location and $a(x)$ the pixel value at location x. The set of all **F** valued images on **X** is denoted by $\mathbf{F}^{\mathbf{X}}$. If the value set $\mathbf{F} = \mathbf{R}$ or $\mathbf{F} = \mathbf{Z}$, then we have real valued or integer valued images, respectively. Similarly, selecting $\mathbf{F} = \mathbf{C}$ or $\mathbf{F} = \mathbf{Z}_{2^k}$ provides for complex or k—bit images.

Operations on and between **F** valued images are the natural induced operations of the algebraic system **F**. Thus, real valued image operations reflect the arithmetic and logic operations on **R**. For instance, the binary operations of addition, multiplication, and maximum on $\mathbf{R}^{\mathbf{X}}$ are defined as follows:

Let $a, b \in \mathbf{R}^{\mathbf{X}}$. Then

$$a + b = \{(x, c(x)) : c(x) = a(x) + b(x), x \in X\}$$

$$a * b = \{(x, c(x)) : c(x) = a(x) * b(x), x \in X\}$$

$$a \vee b = \{(x, c(x)) : c(x) = a(x) \vee b(x), x \in X\}$$

These are basic binary operations for real valued images. Operations on different image types vary with the value set to which they belong.

We define the *Domain* of an image *a* as the point set over which *a* is defined, eg., for $a \in \mathbf{R}^{\mathbf{X}}$, Domain(*a*) = **X**. The *range* of *a* is the set of values assumed by the image. eg., Range(*a*) = $\{a(x) : x \in x\mathbf{X}\} \subset \mathbf{F}$ is the set of all values *a* assumes on **X**. Thus the output of domain or range is not an image array, but a set of points or a set of values, respectively.

Generalized Templates

In terms of image processing applications, templates and template operations are the most powerful tool of the image algebra. The image algebra definition of a template unifies and generalizes the usual concepts of templates, masks, windows, and neighborhood functions into one general mathematical entity. These templates also generalize the notion of "structuring elements" as used in mathematical morphology.

Let **X** and **Y** be point sets and **F** a value set. A *generalized* **F** *valued template t* from **Y** to **X** is a function *t*: $\mathbf{Y} \rightarrow \mathbf{F}^{\mathbf{X}}$. Thus, for each $y \in Y, t(y) \in \mathbf{F}^{\mathbf{X}}$, or, equivalently, *t*(*y*) is an **F** valued image on **X**. The set **Y** is called the *target domain* or simply the domain of *t*, and the set **X** is called the *range space* of *t*. For notational convenience we define $t_y \equiv t(y)$. Thus, $t_y = \{(x, t_y(x)) : x \in X\}$. The point *y* is called the target point of template *t*, and the values $t_y(x)$ are called the *weights* of the template *t* at *y*. The set of all **F** valued templates from **Y** to **X** is denoted by $(\mathbf{F}^{\mathbf{X}})^{\mathbf{Y}}$.

The *k-support* of a template is defined as:

$$S_k(t_y) = \{x \in \mathbf{X} : t_y(x) \neq k\}.$$

The set $S_k(t_y)$ is also referred to as the configuration of *t* at *y*. When *k* = 0, we write $S(t_y) \equiv S_0(t_y)$. Figure 2 illustrates this concept.

If $t \in (\mathbf{F}^{\mathbf{X}})^{\mathbf{X}}$, then t is called *translation invariant* if for each $x, y, z \in \mathbf{X}$ with $y + z$ and $x + z \in \mathbf{X}$, we have that $t_y(x) = t_{y+z}(x+z)$. A template that is not translation invariant is called *translation variant.* Translation invariant templates can be represented pictorially; an example is given in Figure 3.

For example, let $\mathbf{X} = \mathbf{Z}^2$, where $\mathbf{Z}^2 = \mathbf{Z} \times \mathbf{Z} \subset \mathbf{R}^2$ is a point set. Let $\mathbf{y} = (x, y)$ be an arbitrary point of $\mathbf{X}$, $\mathbf{x}_1 = (x, y - 1)$, $\mathbf{x}_2 = (x + 1, y)$, and $\mathbf{x}_3 = (x + 1, y + 1)$. We now define a template $t \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{X}}$. For each $y \in \mathbf{X}$, we define its weights as $t_y(y) = 1,\ t_y(x_1) = 3,\ t_y(x_2) = 2,\ t_y(x_3) = 4,$ and $t_y(x) = 0$ if x is not an element of the set $\{y, x_1, x_2, x_3\}$. In this case $S(t_y) = \{y, x_1, x_2, x_3\}$. Thus t has a configuration and weights as shown in Figure 3. The marked cell represents the target point y.

Operations between Images and Templates

The three basic operations between images and templates are denoted by $\oplus$, $\boxed{\vee}$, and $\textcircled{\vee}$, and called generalized convolution, additive maximum, and multiplicative maximum, respectively. These are defined as follows:

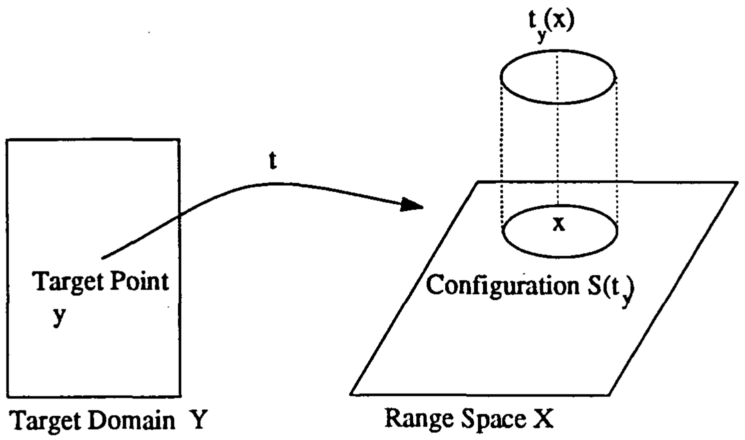


Figure 2. Pictorial example of a template from Y to X.

Let $\mathbf{X} \subset \mathbf{R}^n$ be finite and $\mathbf{Y} \subset \mathbf{R}^m$. Suppose $a \in \mathbf{F}^{\mathbf{X}}$ and $t \in \left(\mathbf{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$, where $\mathbf{F} \in \{\mathbf{C}, \mathbf{R}\}$. Then the *generalized convolution of image a with template t* is defined as

$$a \oplus t \equiv \left\{ (y, b(y)) : b(y) = \sum_{x \in \mathbf{X}} a(x) * t_y(x),\ y \in \mathbf{Y} \right\}.$$

Similarly, if $a \in \mathbf{R}_{-\infty}^{\mathbf{X}}$, and $t \in \left(\mathbf{R}_{-\infty}^{\mathbf{X}}\right)^{\mathbf{Y}}$, then the additive maximum is defined as

$$a \boxtimes t \equiv \left\{ (y, b(y)) : b(y) = \bigvee_{x \in \mathbf{X}} a(x) + t_y(x),\ y \in \mathbf{Y} \right\}.$$

The multiplicative maximum operation is defined for $a \in \left(\mathbf{R}_{+\infty}^{\geq 0}\right)$, and $t \in \left(\left(\mathbf{R}_{+\infty}^{\geq 0}\right)^{\mathbf{X}}\right)^{\mathbf{Y}}$, and is given as follows:

$$a \circledvee t \equiv \left\{ (y, b(y)) : b(y) = \bigvee_{x \in \mathbf{X}} a(x) * t_y(x),\ y \in \mathbf{Y} \right\}.$$

In all these cases, we assume that the support for t is finite. However, these definitions can be extended to the continuous case as well. Thus image algebra can be used for expressing both discrete and continuous image transforms.

One thing to be noted at this point is that while a is an image on X, b, which is produced from the above operations, is an image on domain Y. Thus, since X and Y can be of

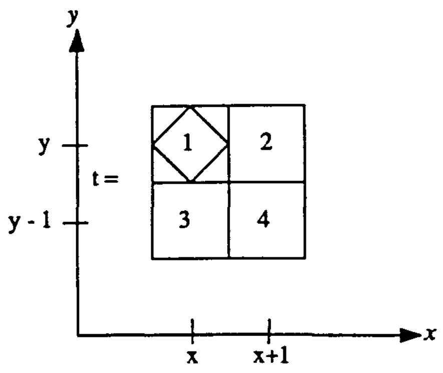


Figure 3. Pictorial representation of a translation invariant template.

different dimension and shape, template operations can be used to change the dimensionality and shape of images. Thus, apart from the usual local or global convolutions that occur in edge enhancement, smoothing, and averaging operations, templates can also be used for image rotation, zooming, image reduction, and matrix multiplication.

The above material gives a very general overview of image algebra. For a more detailed view of image algebra we refer the reader to [25].

IV. FUZZY LOGIC

Introduction

Fuzzy systems is an alternative to the traditional notions of set membership and logic that has applications at the leading edge of the area of artificial intelligence. The notion central to fuzzy systems is that truth values (in fuzzy logic) or membership values (in fuzzy sets) are indicated by a value on the range [0.0, 1.0], with 0.0 representing absolute Falseness and 1.0 representing absolute Truth. For example, let us take the statement:

"Jane is old."

If Jane's age was 75, we might assign the statement the truth value of 0.80. The statement could be translated into set terminology as follows: "

"*Jane is a member of the set of old people.*"

This statement would be rendered symbolically with fuzzy sets as:

mOLD(Jane) = 0.80,

where m is the membership function, operating in this case on the fuzzy set OLD of old people, which returns a value between 0.0 and 1.0.

Here it is important to point out the distinction between fuzzy systems and probability. Both operate over the same numeric range, and both appear to have similar values: 0.0 representing *false* (or *non-membership*), and 1.0 representing *true* (or *membership*). However, there is a distinction to be made between the two statements: The probabilistic approach yields the natural-language statement, "There is an 80% chance that Jane is old," while the fuzzy terminology corresponds to "Jane's degree of membership within the set of old people is 0.80." The semantic difference is significant: the first view supposes that Jane is or is not old; it is just that we only have an 80% chance of knowing which set she is in. By contrast, fuzzy terminology supposes that Jane is "more or less" old, or some other term corresponding to the value of 0.80. Further distinctions arising out of the operations will be noted below.

Definitions

We next give some basic definitions for the fuzzy logic system. These are:

- Definition 1: Let X be some set of objects, with elements noted as x.
- Definition 2: A fuzzy set A in X is characterized by a membership function: mA(x) which maps each point in X onto the real interval [0.0, 1.0]. As mA(x) approaches 1.0, the *grade of membership* of x in A increases.
- Definition 3: A is *EMPTY* iff for all $x \in X$, mA(x) = 0.0.
- Definition 4: A = B iff for all $x \in X$, mA(x) = mB(x) [or, mA = mB].
- Definition 5: mA′ = 1 — mA, where A' denotes the complement of A in X.
- Definition 6: A is *CONTAINED* in B iff mA ≤ mB.
- Definition 7: C = A *UNION* B, where: mC(x) = MAX(mA(x), mB(x)).
- Definition 8: C = A *INTERSECTION* B where: mC(x) = MIN(mA(x), mB(x)).

Applications

Some fuzzy system applications include information retrieval systems, a navigation system for automatic cars, a predicative fuzzy-logic controller for automatic operation of trains, laboratory water level controllers, robot arc-welders, graphics controllers for automated police sketchers, and more.

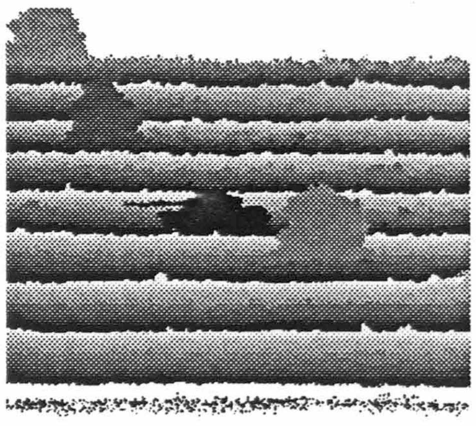
Expert systems have been the most obvious recipients of the benefits of fuzzy logic, since their domain is often inherently fuzzy. Examples of expert systems with fuzzy logic central to their control are decision-support systems, financial planners, diagnostic systems for determining soybean pathology, and a meteorological expert system in China for determining areas in which to establish rubber tree orchards [26]. Another area of application, akin to expert systems, is that of information retrieval [27].

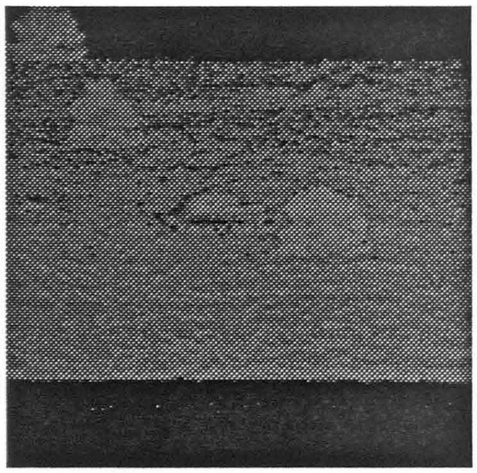
The fuzzy image algebra neural network has an activation function which is based on the generalized mean. This definition of the generalized mean assumes fuzzy memberships for each of the elements of the data set. This relationship is explained in greater detail in chapter VII.

V. DATA

One goal of a machine vision system is to interpret automatically the information provided in a physically sensed signal. In computer vision research, two–dimensional (2D) signals are used to determine the geometric structure of three–dimensional (3D) scenes. Each data point in a digitized 2D signal has three values associated with it: the first two are the spatial x, y coordinates representing the location, and the third is the magnitude z of the sensed physical quantity. These 2D signals are represented as a matrix of integers. Each matrix element or *pixel* has a (row, column) location, and a digitized value representing the magnitude. These matrices are referred to as *digital images*. Some common images are intensity, range, tactile, X-ray, infrared, radar, and ultrasound images.

The type of data used for this research is laser radar (LADAR) data collected by LTV. Range is measured by using the time difference between a start pulse derived from the transmitted pulse, and the received signal from the target to the signal detector. LADAR data is visually very good and has very high resolution, of the order of 0.1 milliradians in both vertical and horizontal directions. This data consists of two types of information at every pixel location. The first is the distance, from the sensor to the sensed object in the real world, and is called *range* data. It has 16 bits with values ranging from 1 to about 2000 units. The second value is the *intensity* of the returned signal which is 8 bit data with a range of values from 1 to 255. See Figure 4 for a sample of range and intensity data. One approach to ATR uses both range and intensity data [28], but for our classification problem, we used only of the range data. Even though the intensity data has greater visual contrast and hence appears to carry more information, it is actually the range data which carries more information. Range images are unique in a mathematical sense among 2D signals for scene analysis, because they directly approximate the physical surfaces of objects in scenes. With other types of data, such surface information has to be inferred *indirectly* from the sensor data, regardless of the resolution or quality. From the range data an elevation image can be constructed using simple geometric and trigonometric rules. Thus, there is 3–dimensional geometrical shape information in the range data that is not available in the intensity data. The range resolution along the line of vision is very good and is of the order of 0.15 meters.





Range Image Intensity Image

Figure 4. 256 x 256 Range and intensity images of the same scene.

Data Classes in the Range Data

The range images from the LADAR data are of size 256 x 1024 pixels, or 256 x 256 pixels, and contain one or more vehicles within a one kilometer range from the sensor. The vehicles are tanks, trucks, and jeeps. Some of the vehicles present in the data are M60, M114, and M47 tanks, M53 artillery tanks, and M42 missile launchers. One set of images, consisting of natural terrain such as trees, bushes, and sandhills, provides the nontank class.

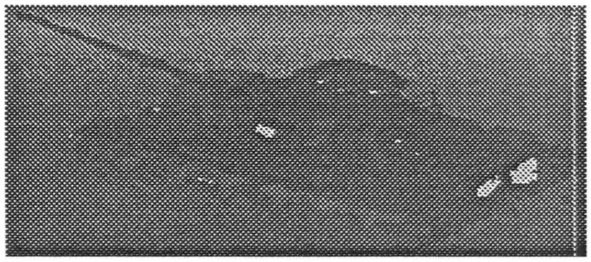
See Figure 5 for sample images of some of the data classes. We remark that the five images in Figure 5 are of different sizes. The desired subimage consisting of a tank or nontank image is extracted, then histogram equalized. The extraction process is as follows:

1. The coordinates of the top lefthand corner of the subimage are determined by visual inspection. For the vehicular data this information was available in the header files for each image. This file gives information on the number of targets present in each image, and the size and location of the targets.
2. The size of the subimage (rows x columns) to be extracted is determined.
3. The subimage is extracted starting from the top lefthand corner point. The *Image Algebra Fortran* (IAF) preprocessor was used for this purpose.

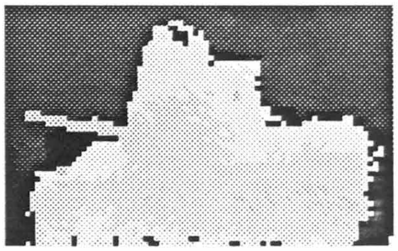
An image processing software called Khoros can also be used for extracting subimages. Khoros is a public-domain software that allows, among other things, windowing of images, and also gives exact coordinates at any point in the image. Further research could be carried out to perform automatic extraction of the vehicle subimage. After extraction, the subimage is converted to elevation data using either the Cartesian method or the locus method. We used the Cartesian method [29]. The locus method is described in [29].

Converting Range Data to Elevation Data: The Cartesian Method

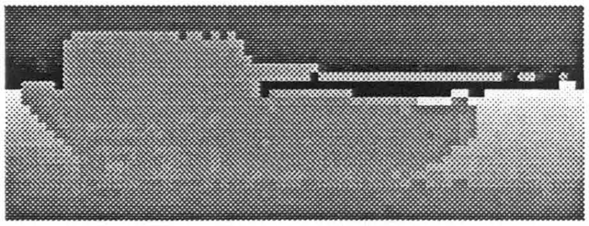
The position of a point in the Cartesian coordinate system can be derived from the measured range value and the direction of the sensor beam at that point [29], both of which are available from the data. Figure 7 shows the range measurement (ρ) and the direction of the sensor beam which is specified by its vertical (ϕ) and horizontal (θ) scanning angles. The



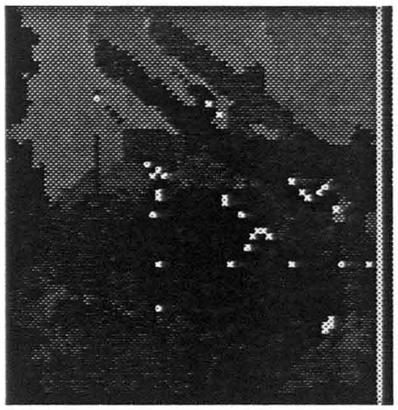
M60 Range Image



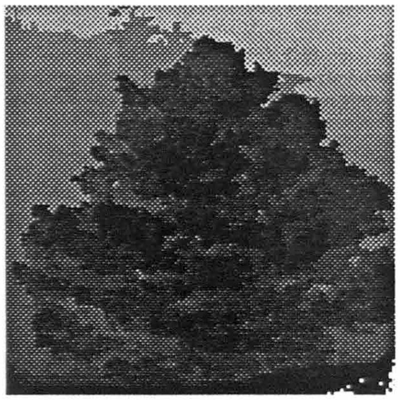
M47 Range Image



M53 Range Image



M42 Range Image



Nontank Image (Tree)

Figure 5. Range images of the five input classes.

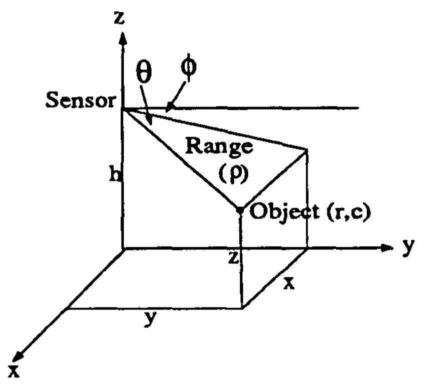


Figure 7. Range sensor geometry.

value of the horizontal and vertical scanning angles for each point in the image is derived from the row and column position, (r, c) using the equations

$$\theta = \theta_0 + c * \Delta\theta$$

$$\phi = \phi_0 + r * \Delta\phi.$$

Here, ϕ_o, θ_o are the initial values, and $\Delta\phi$ and $\Delta\theta$ are the increments for the vertical and horizontal scanning angles, respectively.

The variables ϕ, θ, and (r, c) in Figure 7 have the following definitions:

ρ : range measurement.

ϕ : vertical scanning angle (initial value ϕ_0, increment $\Delta\phi$).

θ : horizontal scanning angle (initial value θ_0, increment $\Delta\theta$).

(r, c): range image row and column coordinates.

The x, y, and z coordinates are then obtained using the following equations based on simple trigonometry and geometry. The header file for the image provides the information on sensor angle, vertical and horizontal resolution, as well as range resolution and range offset.

$$x = \rho * sin\theta$$

$$y = \rho * cos\phi * cos\theta \tag{1}$$

$$z = \rho * sin\phi * cos\theta$$

The resulting transformed image is called the *elevation image.*

Converting Range Data to Elevation Data: The Locus Method

The traditional Cartesian approach has several problems associated with it. First, it introduces nonuniform samples in Cartesian space. Even though the sensor scans with equal angle intervals, the Cartesian map is progressively more sparse at points further from the sensor. Secondly, the environment may cast range shadows due to rugged terrain, such as the shaded area in Figure 8, due to rugged terrain. Without explicit information about the shadow areas, the surface would be smoothly interpolated, possibly incorrectly. Third, the uncertainty of range measurement gets distributed across a region in the elevation map, i.e., uncertainty in the elevation is dependent not on a single range measurement but on many neighboring sensor measurements. The locus method suggested in [29], uses a model of the sensor and works in image space. In the locus method, the elevation *z* at a point (*x*, *y*)

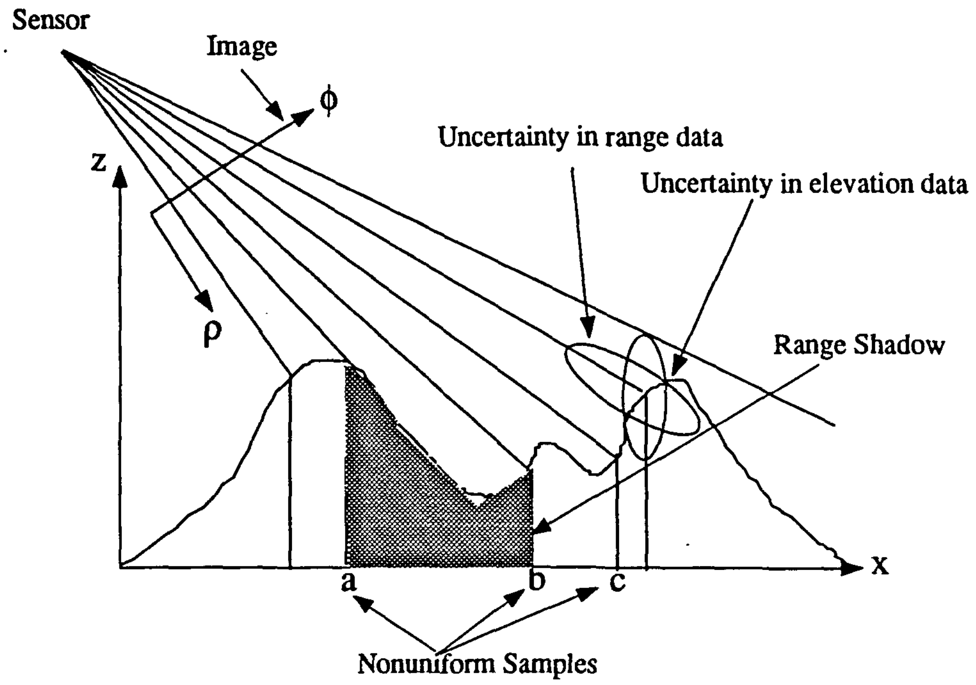


Figure 8. Problems associated with the Cartesian method.

on the reference plane is found by computing the intersection of the terrain with a vertical line at (*x*, *y*). By computing the distance from the sensor to the scene points on this vertical line, we can create a locus of this vertical line in image space. We derive the equation of the locus as a function of ϕ from Equation 1, assuming x and y constant:

$$\begin{aligned} \rho = \rho_l(\phi) &= \sqrt{\frac{y^2}{cos^2(\phi)} + x^2} = \sqrt{\frac{y^2 + x^2 * cos^2(\phi)}{cos^2(\phi)}} \\ \theta = \theta_l &= arctan\frac{x * cos(\phi)}{y}. \end{aligned} \tag{2}$$

We now describe the locus method algorithm for computing the elevation z at a grid point (x, y):

1. Compute the locus $\rho_l(\phi)$ using Equation 2.
2. Compute the corresponding $\theta_l(\phi_j)$ using Equation 2.
3. Obtain a sample data at (ϕ_j, θ_l), $\rho_m(\phi_j)$, from the range image.
4. Compute the difference between the locus and the sample of data, $\Delta(\phi_j) = \rho_l(\phi_j) - \rho_m(\phi_j)$.
5. Find the two scanlines of the range image ϕ_1 and ϕ_2 between which the intersection is located. This will be the point where sgn $\Delta(\phi_1)$ is not equal to sgn $\Delta(\phi_2)$, where

$$sgn(x) = \begin{cases} +1 & if \quad x > 0 \\ -1 & if \quad x < 0 \end{cases} .$$

6. Apply a binary search between ϕ_1 and ϕ_2, and find the intersection point ϕ_n.
7. Compute ρ and θ using Equation 2, and then compute an elevation value using Equation 1.
8. Repeating steps 1–7 for vertical lines at every desired (x, y) point results in a dense elevation map at the desired resolution.

The locus method avoids some of the problems of the Cartesian method. However, it is computationally very intensive, and thus for this thesis, the Cartesian method was employed.

Interpolation

Since the Cartesian method introduces nonuniform samples in image space several data values are found to be missing. To compensate for the nonuniform samples produced, interpolation between data points is necessary. The following steps are carried out for interpolating within the elevation image.

1. The x, y, and z values are mapped into an image whose dimensions extend from 0 to the maximum value of x calculated for the columns, and 0 to maximum value of y calculated for the rows. An even grid is created and all missing z values are left empty.
2. A variation of the linear interpolation scheme is used to fill in the data.
    a. Let $a \in R^X$ be the incomplete elevation image and $b \in R^X$ be the filled image. X is of size Rows x Columns, where Rows = maximum x dimension and Columns = maximum y dimension.
    b. We check for a string of zeros in a row in between two nonzero z values. Let the two nonzero values be at locations (i,j) and $(i+l,j)$. Then the values in between are given by:

$$for \quad r = j, j+k$$

$$a(i,r) = (a(i,j) + a(i,j+k))/(j-r)$$

    c. If a boundary location has a zero value, we look for the nearest nonzero values along the row and along the column and average these. A point with this value is assumed outside the boundary point and interpolation carried out as in (b). See Figure 9 for a pictorial description of the interpolation scheme.

The interpolated elevation images vary in size from 300 x 300 to 100 x 100, depending on the size of the image extracted. It is necessary, therefore, to scale them to a standard size to input to the neural net.

Scaling

The standard image size used for input to the neural net is 30 x 50. The different sized elevation image is scaled to size 30 x 50 by convolving it with a shrinking, averaging template. The template used is a translation variant template and is obtained using the following steps.

1. If X is an M x N rectangular array and $a \in R^X$. Let Y be a P x Q rectangular array and $b \in R^Y$ where $P < M, Q < N$. We write

$$M = U * P + R \qquad U, V, R, S \text{ are integers}$$

$$N = V * Q + S \qquad 0 \leqslant R \leqslant M, 0 \leqslant S \leqslant N$$

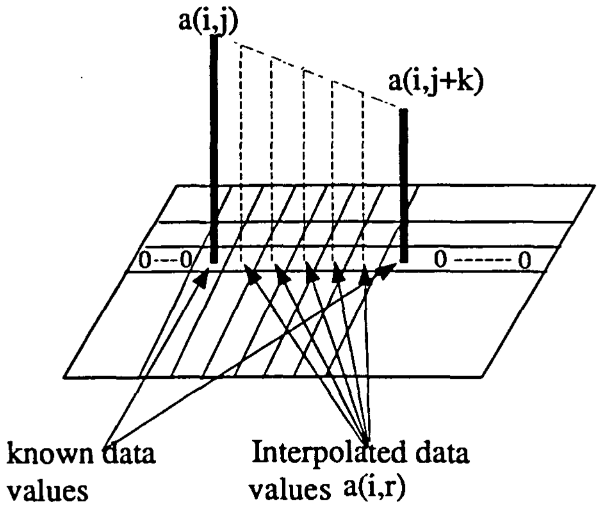


Figure 9. Linear interpolation scheme.

Example: $X = 3 \times 3$, so a is an element of $R^{3 \times 3}$. Then $Y = 2 \times 1$, and $b \in R^{2 \times 1}$.

2. If $(R, S) \neq (0, 0)$ then define $t \in (R^X)^Y$ as follows:

   if $1 \leq i \leq P\text{-}R$, and $1 \leq j \leq Q\text{-}S$, then;

$$t_{(i,j)}(h,k) = \begin{cases} \frac{1}{(u*v)} & \text{if } ((i-1)*u+1) \leqslant h \leqslant i*u, \text{ and,} \\ & \quad ((j-1)*v+1) \leqslant k \leqslant j*v, \text{ and is} \\ 0 & \text{otherwise.} \end{cases}$$

   If $P\text{-}R+1 \leq i \leq P$, and $1 \leq j \leq Q\text{-}S$, then;

$$t_{(i,j)}(h,k) = \begin{cases} \frac{1}{(u+1)*v} & \text{if } ((i-1)*(u+1)+1) \leqslant h \leqslant i*u, \text{ and} \\ & ((j-1)*v+1) \leqslant k \leqslant j*v, \text{ and is} \\ 0 & \text{otherwise.} \end{cases}$$

   If $1 \leq i \leq P\text{-}R$, and $Q—S+1 \leq j \leq Q$, then;

$$t_{(i,j)}(h,k) = \begin{cases} \frac{1}{(v+1)*u} & \text{if } ((i-1)*u+1) \leqslant h \leqslant i*u, \text{ and} \\ & ((j-1)*(v+1)+1) \leqslant k \leqslant j*(v+1), \text{ and is} \\ 0 & \text{otherwise.} \end{cases}$$

   If $P-R+1 \leq i \leq P$, and $Q-S+1 \leq j \leq Q$, then;

$$t_{(i,j)}(h,k) = \begin{cases} \frac{1}{(v+1)*(u+1)} & \text{if } ((i-1)*(u+1)+1) \leqslant h \leqslant i*(u+1), \text{ and} \\ & ((j-1)*(v+1)+1) \leqslant k \leqslant j*(v+1), \text{ and is} \\ 0 & \text{otherwise.} \end{cases}$$

Then the scaled image is $b = a \oplus t$.. We now work out a small example to illustrate how the shrinking, averaging template works.

Example: Let $X = M \times N = 3 \times 3$ and $Y = P \times Q = 2 \times 2$. Here, X indicates the input image size, and Y the output image size. Then we write

1.
$$3 = U * 2 + R$$

$$3 = V * 2 + S$$

This gives us $U = 1$, $V = 1$, $R = 1$, and, $S = 1$.

2. Since $R \neq 0$, and $S \neq 0$, we get $t \in (\mathbb{R}^X)^Y$ as follows:

3. Then on convolving a with t we get the reduced, averaged image b, given in Figure 12.

   Thus, the image a is averaged and shrunk using the above translation variant template.

$$a = \begin{array}{|c|c|c|}\hline 3 & 2 & 6 \\ \hline 3 & 7 & 1 \\ \hline 7 & 11 & 5 \\ \hline \end{array} \qquad t = \begin{array}{|c|c|}\hline (1,1) & (1,2) \\ \hline (2,1) & (2,2) \\ \hline \end{array}$$

Figure 10. Input image a and template t.

$$t_{(1,1)} = \begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad t_{(2,1)} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 1/2 & 0 & 0 \\ \hline 1/2 & 0 & 0 \\ \hline \end{array}$$

$$t_{(1,2)} = \begin{array}{|c|c|c|}\hline 0 & 1/2 & 1/2 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad t_{(2,2)} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1/4 & 1/4 \\ \hline 0 & 1/4 & 1/4 \\ \hline \end{array}$$

Figure 11. Template t.

$$b = \begin{array}{|c|c|}\hline 3 & 4 \\ \hline 5 & 6 \\ \hline \end{array}$$

Figure 12. Result of convolving a with t.

A pictorial representation of this is given in Figure 13. The data is now in a form where it can be input to the neural net for classification. Figure 14 gives the original range image (an M60 tank), the corresponding interpolated elevation image and the scaled 30x 50 image. The image resolutions are different in each case.

A clearer view of the elevation data is obtained from the 3–dimensional graphs as shown in Figures 15 and 16.

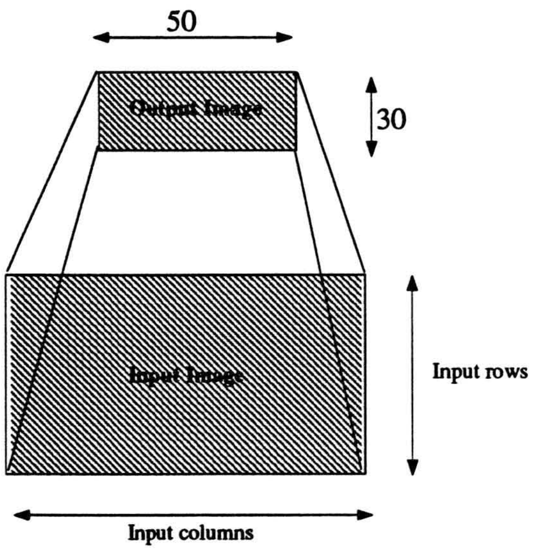
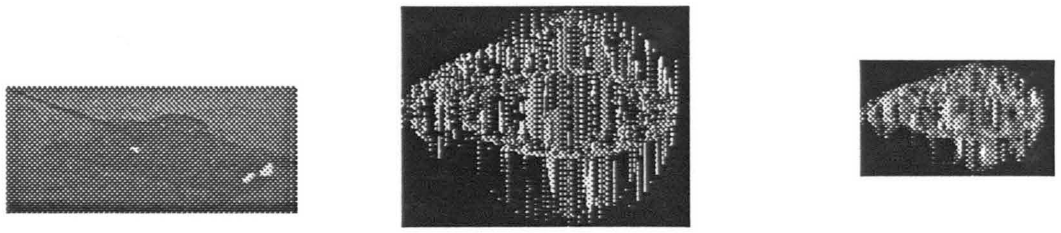


Figure 13. Scaling original image to standard size 30 x 50.

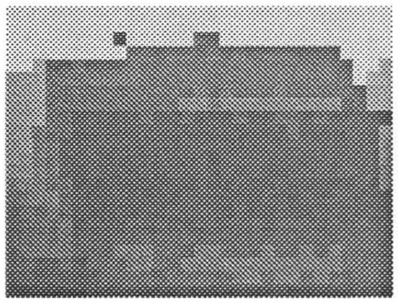


Original range image Interpolated elevation image Scaled 30 x 50 image

Figure 14. Image processing stages.

Range Image

Elevation Image



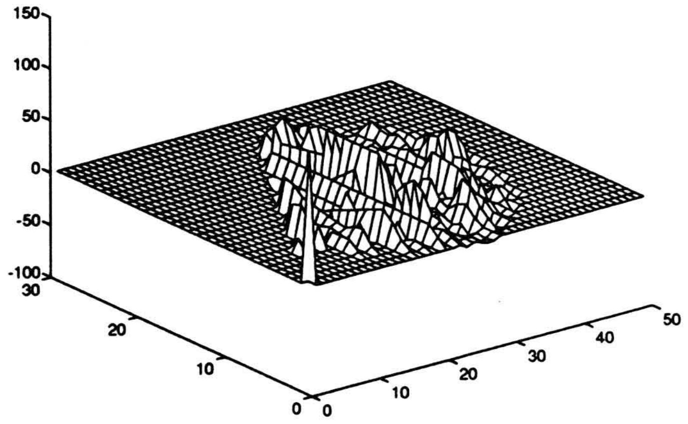
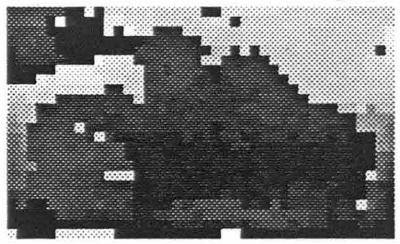


Figure 15. Range image and elevation graph of the M114 tank.

Range Image

Elevation Image



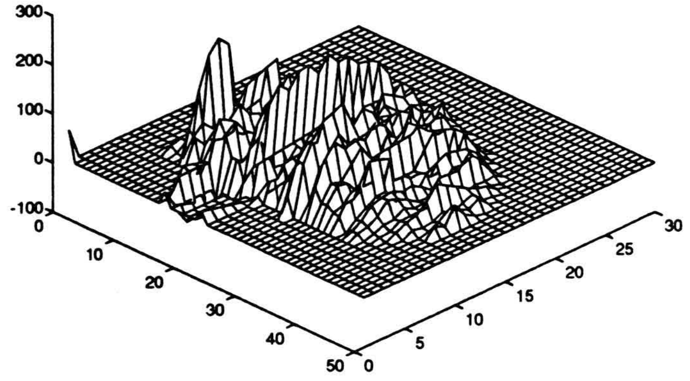


Figure 16. Range image and elevation graph of the M60 tank.

VI. NEURAL NETWORKS

History

Two fundamentally different approaches to information processing are *Neurocomputing* and *programmed computing.* Neurocomputing is the technological discipline concerned with information processing systems that autonomously develop operational capabilities in adaptive response to an information environment. Programmed computing, on the other hand is based on algorithms and set rules. Neural networks come under the field of Neurocomputing. The area of Neurocomputing started when researchers attempted to emulate the human brain. The models developed were called *neural networks* for the biological neurons from which they were modeled. The field then diverged into two primary areas of research: (1) neurobiological network models, which are computational models of biological nervous systems, and, (2) artificial neural networks, which are biologically inspired mathematical models having technical applications. *Artificial neural networks* (ANN's) are also called *parallel distributed processing systems* or *connectionist models.* This research is concerned with ANNs.

The beginning of Neurocomputing dates back to 1943, when McCulloch and Pitts [30] showed that even simple types of neural networks could in principle compute any arithmetic or logical function. Other researchers, principally Weiner and Neumann [31], suggested that research on computers inspired by the functioning of the brain would be fruitful. In 1949 Hebb wrote a book entitled "The Organization of Behavior" [32], which carried further the idea that classical psychological conditioning in animals is a property of individual neurons. The first successful neurocomputer, the Mark I perceptron, was developed during 1957 and

1958 by Rosenblatt, Wightman, and others [33]. Due to his many contributions in this field, Rosenblatt is often considered the founder of Neurocomputing as known today. Shortly after Rosenblatt, Widrow developed a different type of neural network processing element, called ADALINE [34], with a new learning law, which is still in widespread use. However, these early successes in Neurocomputing were followed by a period of stagnation, partially as a backlash against the hype associated with them. During the period from 1967 to 1982 very little research was carried out in artificial neural networks. Associated research in the fields of adaptive signal processing, pattern recognition, and biological modelling, however, remained active during this time.

In the 1980's there was a revival of interest in artificial neural networks, and extensive research since then has been investigated in this area. One person responsible for this resurgence was Hopfield, an established physicist who wrote some very interesting papers on neural networks [35], and lectured widely. In 1986, "Parallel Distributed Processing, Volumes I and II", edited by Rumelhart and McClelland were published [36], and in 1987 the first open conference on neural networks, the IEEE International Conference on Neural Networks, was held in San Diego, marking the coming of age of this field.

Introduction

Artificial neural nets are most often used as pattern classifiers. They perform a mapping from an input object space to an output classification space. The input-output relation is learned by an iterative optimization procedure, where the error between the input and a corresponding desired output is minimized. This is called the "training" phase. The *data* used for training should be statistically representative of the data to be classified, to obtain good results. Neural nets can be fairly good at determining complex relationships between input data and classification representations. Although neural networks do not have a very rigorous or broad mathematical foundation, they have been used to solve a wide variety of problems and have produced very acceptable results in many applications. They have been used for pattern recognition, speech and image processing, data forecasting, and many other tasks. Lippmann [37] gives a tutorial on a wide variety of networks for pattern classification.

A neural network is a parallel distributed information processing structure, consisting of processing elements or nodes, each of which can possess local memory and carry out localized information processing operation. A network can have one or more layers, each layer consisting of one or more nodes depending upon the *architecture* of the network. Nodes in different layers are interconnected through *connections* or *weights*. The nodes in a layer may be connected to nodes in other layers, or to each other, or both, based upon the connection scheme used for that particular neural net. Information processing at a node can be defined arbitrarily with the restriction that it must be completely local, i.e., it must depend only on the current values of the input signal arriving at the node via impinging connections, and on values stored in the node's local memory. Each node has a single output that may be passed onto as many other nodes as is desired.

The basic design principles of neural networks are based on the following three important properties:

- Adaptivity — This is the ability to encode information in a meaningful way by means of an adaptation rule.
- Fault Tolerance — In pattern recognition, terms this implies the ability to recognize noisy patterns.
- Parallel Processing — This is associated with neural networks because of their ability

to operate in a purely distributed fashion.

While fault tolerance and parallel processing are features commonly shared by a wide variety of computing devices, adaptivity is a feature unique to neural networks.

Connection Schemes in Neural Networks

There are four basic types of connection schemes between nodes in a network that indicate the flow of data within the net. These are:

- Feed forward — The network accepts data from the input nodes and passes it on to the output or next layer of nodes. Data flows only in the forward direction and only once.
- Recurrent — The network allows data to flow from the node to itself.
- Feedback — The information between nodes can flow forward or backward, that is, from the output nodes to the input nodes.
- Recursive — The network recursively iterates in the feed forward or feedback process.

Learning in Neural Networks

This is a mathematically defined process by which the weights of a network are changed. Learning occurs in networks that are recurrent and/or have feed forward or feedback connection schemes. There are two types of learning a net can undergo. These are:

- Supervised Learning — The network is provided information from outside to help make adjustments in weights. This information includes:

  a. The desired output and an error criterion.

  b. When to terminate the training process, eg., after how many iterations, or after what error level.

c. The number of times the data must be presented to the net during training.

Examples of this are error correction learning, backpropagation, and stochastic learning.

- Unsupervised Learning — This is a process that relies only on information available internal to the net. This allows natural groupings of the data to form by themselves using predefined similarity measures. One example is the Kohonen net.

Methodology of Operation

To illustrate the nature of neural networks we describe briefly a classical neural net architecture known as the *perceptron.* This is mainly of historical importance now as it has since been superceded by other, more powerful networks.

Perceptron: This is a neural network that consists of one or more of the processing elements or nodes shown in Figure 17. For simplicity, we next describe the operation of a single perceptron processing element. Figure 18 shows the classification problem we are trying to solve using the perceptron. We have two *classes* of patterns, class 0 and class 1. A *pattern* is

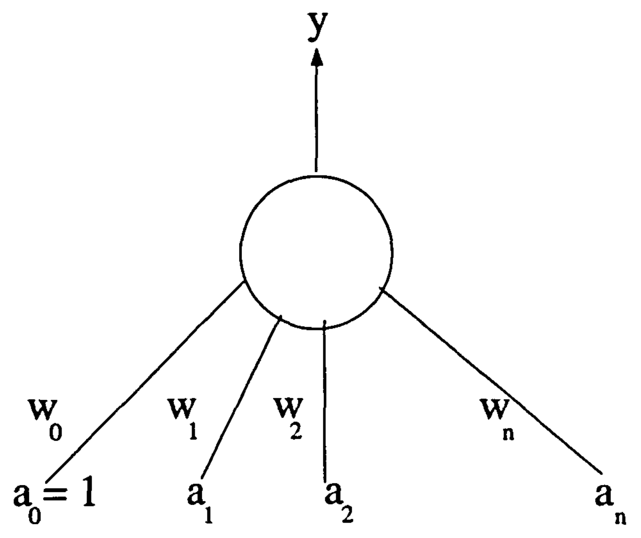


Figure 17. Single processing element of a perceptron.

simply a point in n-dimensional space, where the coordinates of the points represent attributes or features of the object to be classified, such as height, weight, density, or frequency. In order to be classified correctly by the perceptron, the two classes must be separated from each other by a simple linear hyperplane. In 2–dimensional space, a hyperplane is a line, in 3–dimensional space it is an ordinary plane, and in n-dimensional space it is called a hyperplane residing in (n-1) dimensional space. A set of classes having this property is termed *linearly separable*. The goal is to find a set of weights or coefficients $w_0, w_1, w_2, \ldots, w_n$ which determine a unique hyperplane such that the output of the perceptron is 1 if the input pattern vector $(x_0, x_1, x_2, \ldots, x_n)$ belongs to class 1, and 0 if the pattern vector belongs to class 0. The value y at the output is calculated as shown in Equation 3.

$$y = \begin{cases} 1 \quad if \ \sum_{i=0}^{n} w_i * x_i \geqslant 0 \\ 0 \quad if \ \sum_{i=0}^{n} w_i * x_i < 0 \end{cases} \tag{3}$$

The weights are stored in the node's local memory and are automatically modified by the node itself using the perceptron learning rule. We represent the weight w_i as the strength

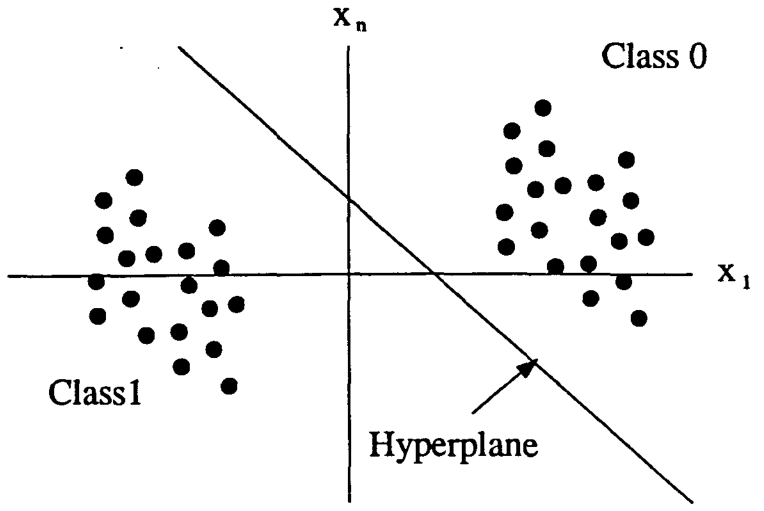


Figure 18. Pattern classification problem in n-dimensional space.

of the connection from input node i to the single output node. The *learning* process for the network is simply the methodology by which the weights are changed to produce the correct classification results. During *training*, the perceptron is given a sequence of randomly selected patterns, one at a time. A pattern is represented by a vector a. Each time a training pattern is presented to the perceptron, the system is also told to which class, 0 or 1, it belongs. With each input vector, the weights are modified using the equation

$$w^{new} = w^{old} + \left(y^d - y\right)a,$$

where y^d is the correct class number for input a, and y is the output of the perceptron. The idea behind the learning rule is that if the perceptron makes an error in its output, that is, if $y^d - y \neq 0$, then this indicates that the solution hyperplane needs to be reoriented so as to not make the error in the future. If the error is zero, the weights will not change. Once training is completed, the network weights will not change anymore. The neural net can then be tested on patterns which were not present in the training set.

Most neural networks are more complex in nature then the perceptron, and can have a variety of architectures. The architecture of a neural net is based upon:

- The number of information processing layers.
- The number of nodes in each layer.
- The connections between the layers. The nodes in two consecutive layers can be fully or partially interconnected.
- The equation determining how the new node value is calculated at each layer.
- The learning rules for all the layers in the net.

Due to the flexibility provided by these factors, neural networks can be applied to a variety of tasks such as business analysis, forecasting, signal and image processing, pattern recognition, and medical diagnostics, to name just a few. In the next section, one of the more popular neural nets called the *backpropagation neural net* is described in detail. This particular net forms the basis for the architecture of the FIANN.

Backpropagation Neural Networks

Backpropagation nets are capable of implementing a broad variety of tasks, including pattern association, pattern classification, image compression, data compression, robot control, and function approximation tasks. It is a powerful network that has been applied successfully to solving many different types of problems.

There are two phases in the operation of the backpropagation net. The first phase is the training phase, during which the weights are iteratively updated, and the second phase is the recall phase, where the weights are held constant while the net simply calculates output values for classification purposes. The backpropagation network follows the feedback method of connection and uses supervised learning methods. It uses a layered hierarchial architecture of simple nodes with a high degree of connectivity between layers. Connections are allowed between two adjacent layers, but connections within a layer are not allowed. A schematic diagram showing a two layered feed-forward network with full connectivity is shown in Figure 19. The backpropagation algorithm does not require full connectivity of adjacent layers, so partial connections between layers are allowed. Further, weights or connections that skip one layer are permitted. The major restriction is that information in the net can only flow forward in the layer hierarchy, not backward, laterally or recursively. The network architecture consists of one input layer, one or more hidden layers, and an output layer.

The input layer performs no processing on its inputs, but merely serves to distribute them to the first processing layer. Following the input layer are the hidden layers, so called

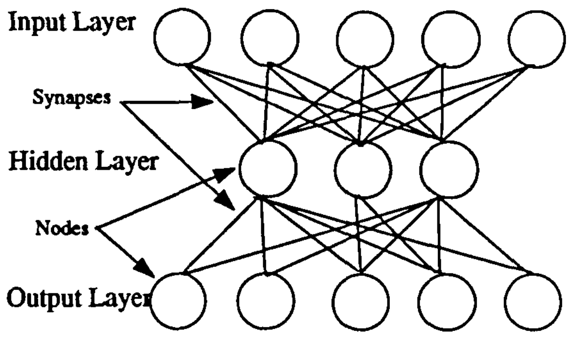


Figure 19. Fully connected backpropagation network.

because they receive no input from, and produce no output to, the outside world. Finally, the output layer produces the output results of the network for the user. In designing the net architecture, the number of input nodes are fixed by the number of input variables needed for the task, and the number of output nodes are fixed by the number of values or classes that are desired. Each node in the network receives one or more inputs from the outside world or from preceding layers, and produces a single output value, which is broadcast to other node inputs in succeeding layers as determined by the connection scheme. Each node i in the first hidden layer calculates its state value via Equation 4. This is simply the dot product of the node's weight vector with its input vector, together with an additive bias θ:

$$b_i = \sum_{j=1}^{n} w_{ji} * a_j + \theta. \tag{4}$$

Here, a_j represent the value at node j in the input layer; w_{ji} represents the corresponding weight applied to that input; θ is the offset or bias term for the node, and n is the number of connections to node i. Next, node i's output is obtained by passing the value b_i through a non linear function called an *activation function*. Typical functions include the sigmoidal function given in Equation 5. For the backpropagation network, the activation function must be continuously differentiable and monotonically increasing.

$$c_i = \frac{1.0}{1.0 + e^{-b_i}} \tag{5}$$

Here, c_i is node i's output, and b_i is as in Equation 4.

Learning in Backpropagation Nets

These nets are used to implement supervised learning tasks for which a training data set consisting of input and desired output pairs is available. All backpropagation nets learn by approximating a unidirectional mapping from an n-dimensional input space R^n, where n is the number of input variables, to an m—dimensional output space R^m, where m is the number of output variables or classes. The network updates the weights w_{ji} using error feedback from the training examples. One common measure of error is the mean squared error, defined as

$$E = \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{j=1}^{m}\left(c_{ij}^{d} - c_{ij}\right)\right], \tag{6}$$

where $c^d{}_{ij}$ represents the correct output value for pattern i; c_{ij} is the value calculated by the net with its current set of weights; m is the number of output nodes; and n is the number of training patterns.

To minimize the error in Equation 6, we first obtain the partial derivative of the error with respect to each of the weights as follows:

$$\Delta w_i = k * \frac{\partial E}{\partial w_i}.$$

The backpropagation learning law, also called the *generalized delta rule*, is given by:

$$\Delta w_{ji} = \eta * \delta_i * b_{ji},$$

where b_{ji} represents the jth input to the ith node on that pattern presentation; η is a stepsize parameter that controls the rate at which the change in weights takes place; and δ_i is the "delta" term representing the error for the ith node. The weights are then updated following the gradient as follows:

$$w_{ji}^{new} = w_{ji}^{old} + \Delta w_{ji},$$

where δ_i for an output node is given by

$$\delta_i = \left(c_i^d - c_i\right)[c_i(1 - c_i)]. \tag{7}$$

Here, the second term $[c_i(1 - c_i)]$ represents the derivative of the sigmoidal activation function with respect to the net input.

The first term in Equation 7 can be specified directly for an output node but not for a hidden layer node. For hidden nodes, it is defined recursively in terms of the delta values of the nodes above it. The δ_i values are passed back through the synapses, or "backpropagated" so that the error value for a hidden layer node i is taken as a weighted sum of the errors of the nodes in the layers above to which it is connected. This is given by

$$\delta_i = [c_i(1 - c_i)] \sum_{j=1}^{n} w_{ji} * \delta_j, \tag{8}$$

where w_{ji} is the weight on the output line of node j to node i; δ_j is the delta value for node j in the layer above; and n is the number of synapses for node i.

The architecture for the FIANN is similar to the multilayer perceptron with backpropagation. The next chapter describes the FIANN and the theory associated with it.

VII. FUZZY IMAGE ALGEBRA NEURAL NETWORK (FIANN)

Classical neural networks combine the information at a particular node as follows:

$$c_i = f\left(\sum_{j=0}^{N} w_{ji} a_j\right),$$

where c_i is the output node value at upper level node i, a_j is the node value at a lower node j, w_{ji} is the connecting weight between i and j, and f is the activation function.

The activation function of the FIANN is motivated by the *generalized mean*, and is useful for defining *generalized template operations*. This is because the generalized mean can represent both *linear* and *morphological* operations. The generalized mean for a set of numbers $\{x_i\}_{i=1}^{n}$ is given by:

$$g(x_1, x_2 \ldots\ldots x_n) = \left[\sum_{i=1}^{n} w_i x_i^P\right]^{1/P}$$

Here, $x_i \in [0.0, 1.0]$, represents the *input fuzzy class memberships*, and w_i can be viewed as relative importance factors for different criterion. Further, we must have $x_1 + x_2 + \ldots\ldots\ldots + x_n = 1$ [38].

One of the properties of the generalized mean that makes it attractive is that the mean value always increases with an increase in P, that is, if $-\infty < P < +\infty$, then we can obtain all values from the minimum to the maximum of the data set. Thus, the whole range of fuzzy operations from intersection to union and anything in between can be approximated. An interesting feature is that $P = -1$ gives the harmonic mean, $P = 0$ gives the geometric mean, and $P = 1$ gives the arithmetic mean.

Governing Equation for the FIANN

We first define the operands involved in the fuzzy image algebra neural net. Let $X = Z \times Z$ be the coordinate set, where Z is the set of integers. An image on X with values in the closed interval [0,1] is a function a: $X \rightarrow [0,1]$. A template on X is a function t: $X \rightarrow [0,1]^X$. The maximum support, S_0, of the template t, and its minimum support S_1 are defined by

$$S_i(t_{j,i}) = \{(i,j) \epsilon X : t_{(j,i)} \neq i\} \qquad i \in \{0,1\} \tag{9}$$

In this thesis we use the subscript ji to designate (j,i) for templates, to ease the notation.

The governing equation for the fuzzy image algebra neural net is given by: $b = a \boxed{\triangle} g$, where

$$b_j = \left[\sum_{i \epsilon S_0(t_j) \cap S_1(t_j)} w_{ji} * r[1 - t_{ji} + a_i]^P \right]^{1/P} \tag{10}$$

Here, b represents the *weighted fuzzy erosion* of image a with a *parametrized template* group g given by : $g = (t, w; P)$, where w and t are image algebra templates, and P a parameter. Here, a is an image with values in [0,1]. The supports S_0 and S_1 are as defined in Equation 9.

The generalized mean has two important properties which make it a very flexible operator and very useful in pattern recognition problems. These two properties when applied to the operation of weighted fuzzy erosion give the following two important results. These are as follows:

1. **Fuzzy erosion property**: The first of these properties is as follows. Let $N_j = \{S_0(t_{ji}) \cap S_1(t_{ji})\}$, that is, all the points where t_{ji} is not equal to either 0 or 1. Then

$|N_j|$ denotes the cardinality of the set N_j. Then if

$$w_{ji} = \frac{1}{|N_j|} \qquad for\ all\ i, j$$

$$and\ \ P \rightarrow\ \ -\infty$$

$$then\ \ c \rightarrow a \boxtimes_f t.$$

Here $d = a \boxtimes_f t$ represents the fuzzy erosion of image a with template t and is defined by:

$$d_j = \bigwedge_{i \in S_1(t_j)} r(1 - t_{ji} + a_i).$$

The ramp function $r(x)$ is defined by:

$$r(x) = \begin{cases} 0 & x < 0 \\ x & 0 < x < 1. \\ 1 & 1 < x \end{cases}$$

See Figure 20 for a graph of r. In effect, the weighted fuzzy erosion represented by b should approximate the fuzzy erosion c of image a with template t when the above conditions are met.

2. **Convolution property**: The second property is if P =1, and $t_{ji} = 1$ for all i, j, then

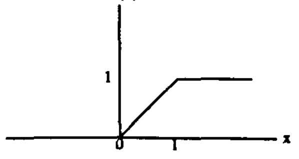
$$c = a \bigoplus w,$$

which is given by:

$$c_j = \sum_{i \in S_0(w_{ji})} a_i * w_{ji}.$$

If w is translation invariant then $a \bigoplus w$ represents linear convolution.

Figure 20. Ramp function r.



These two properties of the generalized mean make the activation function of the FIANN very interesting. They show the wide range of operations that can be obtained from this function, from linear convolution to fuzzy erosion. However, the key point in using this activation function is that it is differentiable. This allows us to derive a *gradient descent learning algorithm* using the chain rule, to train the network.

The weighted fuzzy erosion between a and g is carried out over an image area which represents the intersection of the maximum and minimum supports of t.

The input to the network is an image. We train on N pairs of data, each pair consisting of an image a_k and a desired output d_k ; $k = 1,...N$. We wish to find a template group g that minimizes the sum of the squares of the difference between the actual and desired output:

$$\begin{aligned} E &= \sum_{k=1}^{N} \sum_{(i,j) \epsilon X} E_k^2(i,j) \\ &\text{where} \\ E_k(i,j) &= [(a_k \boxed{\circledwedge} g - d_k)(i,j)] \\ d_k &= a_k \boxed{\wedge}_f \tilde{t}, \end{aligned} \tag{11}$$

and where $\tilde{t}$ is the ideal template for fuzzy erosion.

Learning Schemes

A number of different algorithms were implemented to train the FIANN. Initially the network was tested for its convolution and erosion properties. Three learning schemes were implemented. Gader suggested gradient descent as a learning scheme [1]. Two other learning schemes used were stochastic learning [39] and simulated annealing. These three schemes are now described in detail.

Gradient Descent

1. In this method the cost function is minimized by following the gradient of E.
2. Algorithm:
    1. The templates *t*, *w* are randomly initialized to values in [0,1]. Parameter *P* is randomly initialized to some large negative number.
    2. Training pairs of input image a_k and desired output image d_k are presented to the net.
    3. Error *E* is calculated and generalized template *g* modified using the gradient descent learning rules.
    4. Steps 2–3 are repeated until error falls below acceptable level.
3. Training Rules:
    - For Template *w*: The weights w_{ji} are updated following the gradient of E as follows:

$$w_{ji}^{new} = w_{ji}^{old} - \eta_1 \frac{\partial E}{\partial w_{ji}}, \tag{12}$$

where

$$\frac{\partial E}{\partial t_{ji}} = \sum_{k=1}^{N} \left(f_j^k - d_j^k\right) \frac{\partial f^k}{\partial t_{ji}},$$

and

$$\frac{\partial f^k}{\partial w_{ji}} = \frac{2w_{ji}}{p\|w\|^2} \left(f^k\right)^{1-P} \left(r[1 - t_{ji} + a_i]^P - \left(f^k\right)^P\right).$$

Further, the value of $\frac{\partial f^k}{\partial w_{ji}}$ is nonzero only when $r(1 - t_{ji} + a_i) \epsilon [0,1]$, and this holds true for all the partial derivatives. The computation of $\frac{\partial f^k}{\partial t_{ji}}$ and $\frac{\partial f^k}{\partial P}$ is the same as for w_{ji}.

- For Template *t*: Learning equations for template *t* are as follows:

$$t_{ji}^{new} = t_{ji}^{old} - \eta_2 \frac{\partial E}{\partial t_{ji}} \tag{13}$$

$$\frac{\partial f^k}{\partial t_{ji}} = \frac{-2w_{ji}}{\|w\|^2}\left(f^k\right)^{(1-P)/P}\left(r[1 - t_{ji} + a_i]^{P-1}\right).$$

The value of the parameter η can be determined by the training set size [40]. Thus, if we have three classes of data and the number of training images in each class is a, b, c respectively, then the value of η is determined by:

$$\eta = \frac{1.5}{\sqrt{a^2 + b^2 + c^2}}.$$

- For Parameter P: The equations for updating P are as follows:

$$P^{new} = P^{old} - \eta_3 \frac{\partial E}{\partial P} \qquad (14)$$

where

$$\frac{\partial f^k}{\partial P} = \frac{\left(f^k\right)^{1-P}}{P^2} \sum \frac{w_{ji}^2}{\|w\|^2}\left(m_{ji}^k\right)^P ln(m_{ji}^s)^P - \left(f^k\right)^P ln\left(\left(f^k\right)^P\right),$$

and where

$$m_{ji}^k = r[1 - t_{ji} + a_i].$$

Stochastic Learning

1. In this method we randomly perturb one of the weight vectors, and if the resultant cost function is lower we select this vector and start gradient descent at this point. This technique allows us to move about in the error space and jump out of a local minima [39].
2. Algorithm:

   1. The templates t, w are randomly initialized to values in [0,1]. Parameter P is randomly initialized to some large negative number.

2. The initial error E^{old} is calculated.
3. Small random changes are made to one of the templates t, w or to parameter P, using the following equations:

$$t_{ji}^{new} = t_{ji}^{old} \pm \delta_t$$

$$w_{ji}^{new} = w_{ji}^{old} \pm \delta_w$$

$$P^{new} = P^{old} \pm \delta_P$$

   Here δ represents the small incremental change in the weight vector for the templates or in the parameter P. The value of δ can be the same for both the templates and P or can be different. However within each template the incremental value is constant, i.e., the entire weight vector is perturbed by the same amount. The decision to increase or decrease the value is taken randomly. The new error E^{new} is calculated based on the new weights.
4. If $E^{new} < E^{old}$ then goto step 5. Else repeat step 3.
5. Keep the new weight set and discard the old.
6. Gradient descent is carried out for a fixed number of iterations.
7. Once error falls below some predetermined acceptable cost, we end training, else return to step 3.
8. Steps 3–7 are repeated until error falls below the acceptable level.

Simulated Annealing

1. Simulated Annealing is a combinatorial optimization method that minimizes "cost function" or error.
2. Algorithm goal: To minimize cost function E

   a. Initialize control parameter C_k, and values t, w and P.

b. Compute initial cost E^{old}.

c. Randomly change P, or one element of t, or one element of w. Compute E^{new} with this new weight vector.

d. Let $\Delta E = E^{new} - E^{old}$. If $\Delta E \leqslant 0$ then accept new values of t, w, P otherwise accept them with probability,

$$p = e^{\frac{-\Delta E}{C_k}} \text{ Metropolis Criterion.}$$

(Generate $\beta \in$ U[0,1], uniformly and randomly. If p>β, accept the new state, otherwise retain old state.)

e. Repeat steps 2–4 a fixed number of times.

f. Decrease C_k and repeat until $C_k = 0$ and error falls below an acceptable minimum.

Simulated annealing will in theory converge to the optimal solution, but in practice this does not always happen. This method produces much smaller jumps in error space than in the case of stochastic learning, but it also allows the net to jump out of a local minima when it accepts a weight vector with a higher cost.

All three methods had different degrees of success in testing the properties of the FIANN. However, for the target classification problem, a combination of gradient descent and stochastic learning was used and produced the best results.

VIII. NETWORK ARCHITECTURE AND TRAINING

In the design of an FIANN, an infinite number of architectural models are possible. For a particular classification problem, the net's input and output layers will be constant. The number of input nodes will depend on the data and the number of parameters we wish to train on, while the number of output nodes will be dictated by the number of classes or the type of output desired. Thus, the greatest variability exists in the backpropagation nets hidden layers. The net can have one or more hidden layers and as many nodes as is desired in these layers. For the ATR problem, several different architectures were implemented before the final architecture was determined.

There were two stages to the research. In the first stage the nets properties were tested, while in the second stage the FIANN was used for target classification. We now describe the particular network architectures used for the two stages.

Stage I: Testing the FIANN's Erosion and Convolution Properties

Architecture

In this case, the net is trained on a pair of images, an input image and a desired output image. To test the erosion property the desired image used is an image eroded with an ideal template. Similarly for testing the convolution property the desired output image consists of the training image convolved with an ideal template. The network architecture remains the same for testing both properties. A network architecture with a single input-output layer is used.

The number of nodes in the layer is equal to the number of pixels in the image. For instance, if the image is of size 16 x 16, then the number of nodes in the layer will be 256. This architecture is shown in Figure 21.

Training

1. Convolution property: To test the FIANN's convolution property, the above architecture was used and the net was trained using gradient descent rules as given in Chapter VII.

2. Erosion property: All three training schemes given in Chapter VII are used to test this property. Comparative results are given in Chapter IX.

Stage II: Using the FIANN for Target Classification

The second stage of the research, after verifying the nets erosion and convolution properties was to test the nets classification abilities. The FIANN was used for classifying the tanks in LADAR tank data. A network architecture somewhat similar to the backpropagation

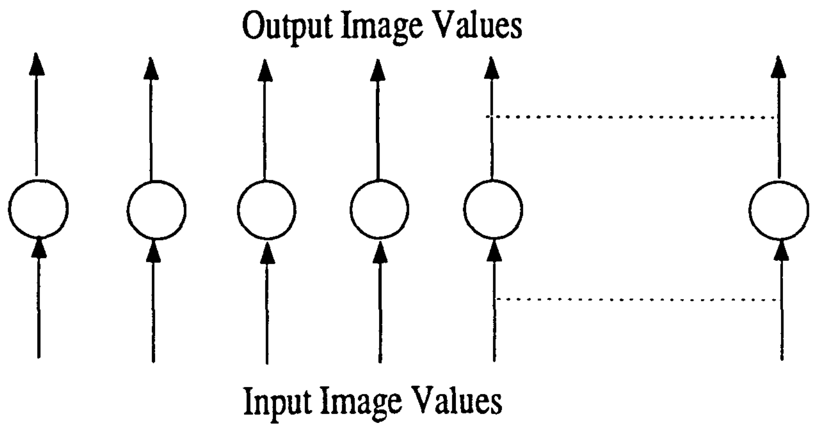


Figure 21. Network architecture for the FIANN in Stage I.

net was used, but with different training rules. We now describe the architecture and training in detail.

Architecture

This net has an input layer, one or two hidden layers and an output layer. The architecture for each layer is determined based on several factors which are now listed.

1. Input Layer: The number of nodes in the input layer corresponds to the number of pixels in a 30 x 50 elevation image, taking values in the closed interval [0,1]. Thus, the input layer has 1500 nodes arranged in a 30 x 50 size grid. To facilitate feature discrimination, the image is divided into a number of overlapping subimages of size 6 x 8. All the subimages overlap by two rows and two columns, except at the edges. The row-column overlap for a subimage in the center of the main image is shown in Figure 22. Other subimage sizes that we experimented with were 5 x 7, and 10 x 12.

2. Hidden Layers: Either one, or two hidden layers are used.

   a. Hidden layer 1: The number of nodes in the first hidden layer is determined by the number of subsections of the input image. All the pixels from one subsection of the input image are connected to one node in the first hidden layer. For instance, when the subimage size is 6 x 8, then the 30 x 50 image gets divided into 56 subimages. Thus, in this case the first hidden layer has 56 nodes. Further, a limited interconnection scheme is applied between the input layer and the first hidden layer. Each node in the first hidden layer is connected to the 48 image pixels from one subsection, and to no other pixels. This hidden layer approximates the operation of linear convolution in order to learn the features of different sections of the input image.

b. Hidden Layer 2: This is implemented in two different ways.

- Partially connected: Four nodes from the first hidden layer are connected to one node in the second hidden layer. This is used so that the net can learn input image features more efficiently. The four nodes from the first hidden layer are selected so that they represent four adjacent subsections of the input image. This is shown in Figure 23. Thus, for an image size of 30 X 50, and a subsection size of 6 x 8, the second hidden layer will have a total of fourteen nodes.
- Fully connected: Eight nodes are used in the second hidden layer, and these are

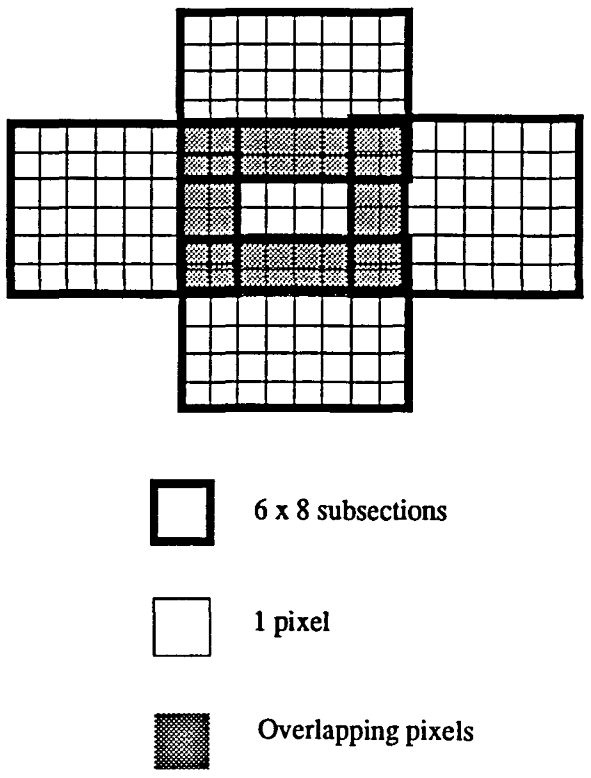


Figure 22. 6 x 8 subsections of 30 x 50 input image.

fully connected with all the first hidden layer nodes. This second layer acts as an aggregation layer where the feature information from the first layer is collected and learned as one cohesive unit by the net.

3. Output Layer: This is the final decision making layer and the number of nodes are determined by the number of output classes desired. For instance, for four classes, three tank and one nontank, two different architectures can be used.
   a. The output layer can have one node for each class, that is, a total of four nodes.
   b. Use a binary scheme, whereas four classes can be represented by just two nodes as shown in Table 2.

Due to the variations possible in each layer, several different architectures can be implemented. Some possible schemes are given in Table 3.

Of all the schemes possible, two are now explained in detail. The first of these uses two hidden layers, while the second uses only one hidden layer.

Training

Using two hidden layers

The architecture for the net is shown in Figure 24. The net is trained as follows:

Table 2 Classification values at the output layer.

Class	Output node 1	Output node 2
Tank 1	Low	Low
Tank 2	Low	High
Tank 3	High	Low
Nontank	High	High

1. An elevation image of size 30 x 50 with values in [0,1] is input to the first layer of the net. The image is divided into subimages of size 6x8 which overlap with adjacent images by 2 rows and 2 columns.

   a. Each of these subimages is connected to one hidden node in the first hidden layer.

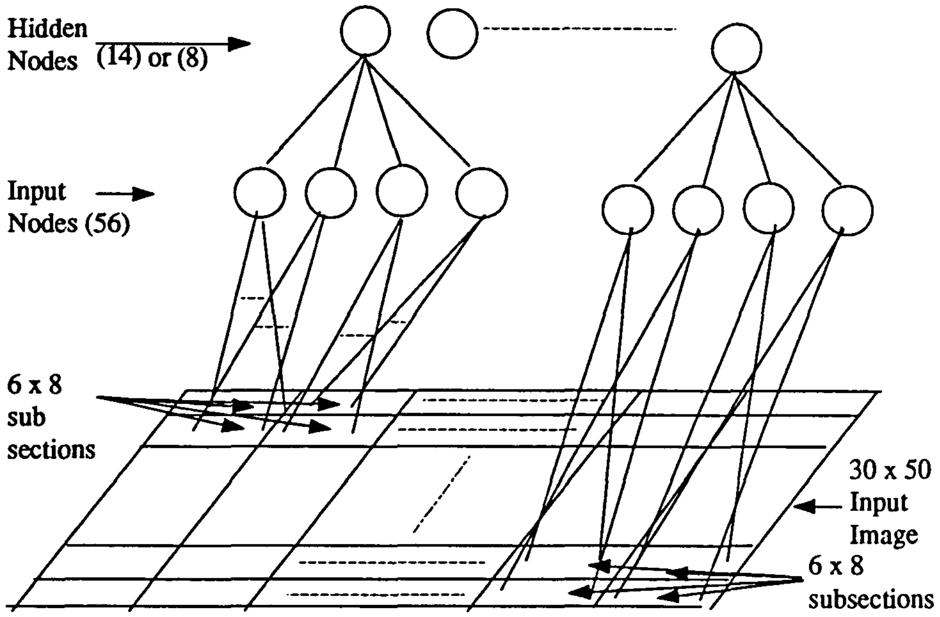


Figure 23. Connection scheme for hidden layers.

Table 3 Architecture schemes for target classification.

Input Layer	Hidden Layer 1	Hidden Layer 2	Output Layer
1500	56	-	2
1500	56	14 (partially connected)	2
1500	56	8 (partially connected)	2
1500	56	8 (fully connected)	2

This layer has a total of 56 nodes. A simple linear convolution is performed between the subimage and the node to combine the information from all the data points in the subimage. Multiple template groups are used for the image, one for each subimage. Let these be represented by $g1 = (t1, w1, P1)$. This layer extracts the image features.

b.
- Partially connected: Four successive first hidden layer nodes are connected to one second hidden layer node. The multiple templates used here are represented by $g2 = (t2, w2, P2)$.
- Fully connected: For fully connected nodes only one template is used.

c. Finally, the second hidden layer nodes are completely connected to the output layer

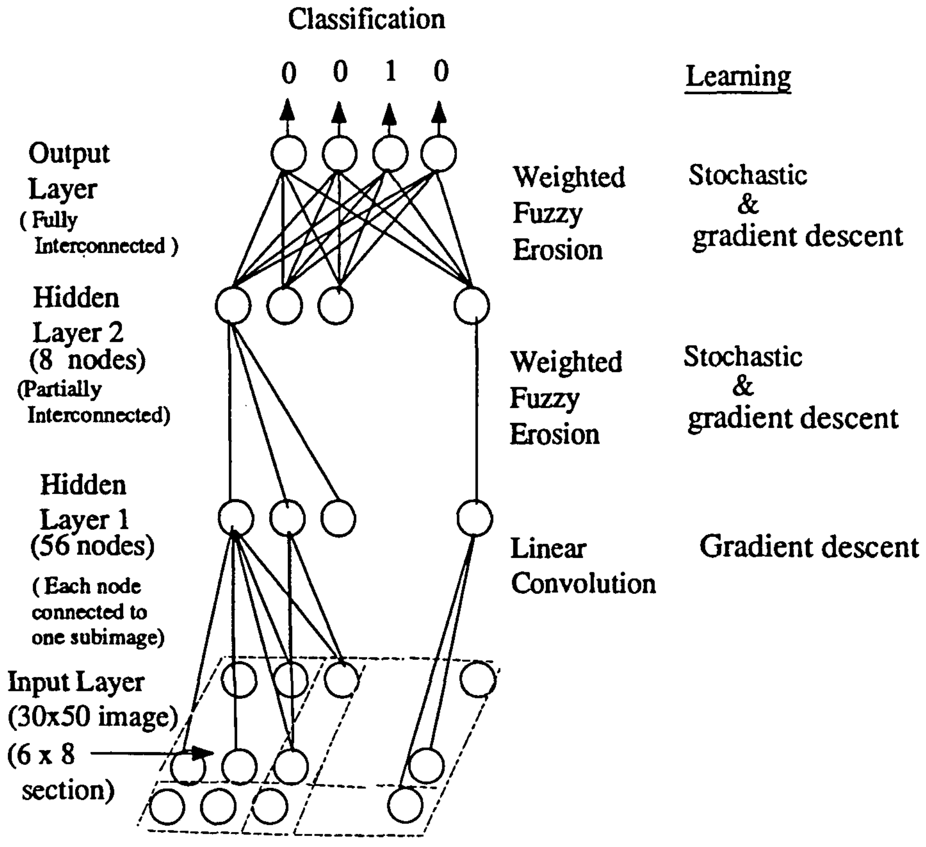

Classification
0 0 1 0
Learning
Output Layer (Fully Interconnected)
Weighted Fuzzy Erosion
Stochastic & gradient descent
Hidden Layer 2 (8 nodes) (Partially Interconnected)
Weighted Fuzzy Erosion
Stochastic & gradient descent
Hidden Layer 1 (56 nodes) (Each node connected to one subimage)
Linear Convolution
Gradient descent
Input Layer (30x50 image)
(6 x 8 section)

Figure 24. Network architecture I for the FIANN

nodes. This generalized template for classification in this layer is represented by *g3* = (*t3*, *w3*, *P3*).

2. The *t* and *w* templates associated with the generalized templates are initialized randomly to values in [0,1]. The parameter *P* is initialized to some negative number. In *g1*, *P1* = 1, since we are trying to simulate the linear convolution property of the generalized mean. All the templates are translation invariant.
3. The output at all the nodes in the hidden and output layers is calculated successively using weighted fuzzy erosion, as follows:

$$b = a \boxed{\bigcirc\!\!\!\!\wedge}\, g$$

$$b_j = \left[\sum_{i \epsilon S_0(t_j) \cap S_1(t_j)} w_{ji} * r[1 - t_{ji} + a_i]^P \right]^{1/P}$$

4. Training the weights: A modification of the standard backpropagation network is used. A training image is input to the net. The node outputs are calculated for the successive layers up to and including the output layer. Knowing the desired values for the output nodes, error is calculated for that image. The weights are trained using a combination of stochastic training and gradient descent techniques. The template weights are updated based on the error surface. At each layer stochastic training is carried out. This allows the template weights to move around in error space and search for a global minima. The backpropagation law then adjusts these weights along the error gradient. The weight training for each layer is now described in detail.

   a. Output Layer: For each presentation of the training data the following two types of training are carried out:

      - As each successive input is presented to the net, the output at the final layer and the error associated with the input are calculated. The weight vectors for *g3* are perturbed and the output calculated using Equation 14. Error e_j for the input image is calculated by taking the sum of the squared difference between actual output b_j and desired output $b^d{}_j$, over all output nodes as follows:

$$e_j = \sum_{j=1}^{n} \left(b_j^d - b_j\right)^2$$

where n is the number of output nodes. This process is repeated iteratively recalculated until we obtain weight vectors for *g3* which gives us a lower error for that image. This is done until we get a lower error or for 25 iterations, whichever comes first.

- Using the set of weights for *g3* obtained after stochastic training the partial derivatives are calculated as per the gradient descent rule. A combination of stochastic learning and gradient descent gives the best results for this net. Batch mode training is carried out using the rules given in Chapter 3. All the training images are passed through once, before the template weights are changed as per equations 12, 13, 14.

Stochastic training is carried out for each of the training images. After all the training inputs have been presented, weights are updated based on the total error of the system. The total error is calculated using Equation 11.

b. Hidden Layers: The same training procedure is carried out for both hidden layers. This is similar to the training for the output layer weights. The chain rule is used to derive the learning rules. The training is now described in brief.

- Initially, stochastic training is carried out. The weight vectors for *g2* are perturbed, and node values calculated at the hidden and output nodes using Equation 14. Error is calculated for the output nodes using Equation 14 and the process repeated until a lower error is obtained or 25 iterations are done, whichever comes first.

- Using the new weight vector for *g2*, the partial derivatives for *E* with respect to *t*, *w*, and *P* are calculated. Batch training is performed.

c. The training images are presented in iterative fashion, with each complete presentation counting as one iteration. The weights are trained until the error at the output nodes is very small, of the order of 10^{-3} as compared to an initial error of above 20.

d. The network is now trained and ready for classification.

5. Testing the network: The net is now tested on tank elevation images, that were not represented in the training set.

Using one hidden layer

The net has an input layer, one hidden layer and an output layer. This net is identical in all respects to the previous one except that it uses just a single hidden layer.

1. Input Layer: This has 1500 nodes corresponding to the pixels of the 30 x 50 input elevation image. This is identical in all respects to the input layer in the previous net.
2. Hidden Layer: This is identical to the first hidden layer of the previous net. It has 56 nodes, each node corresponding to a 6 x 8 subsection of the input image. There is only partial connectivity between the input and hidden layers.
3. Output Layer: Number of nodes correspond to the desired number of classes. This layer is completely connected to the hidden layer, and performs classification based on the image features passed to it.

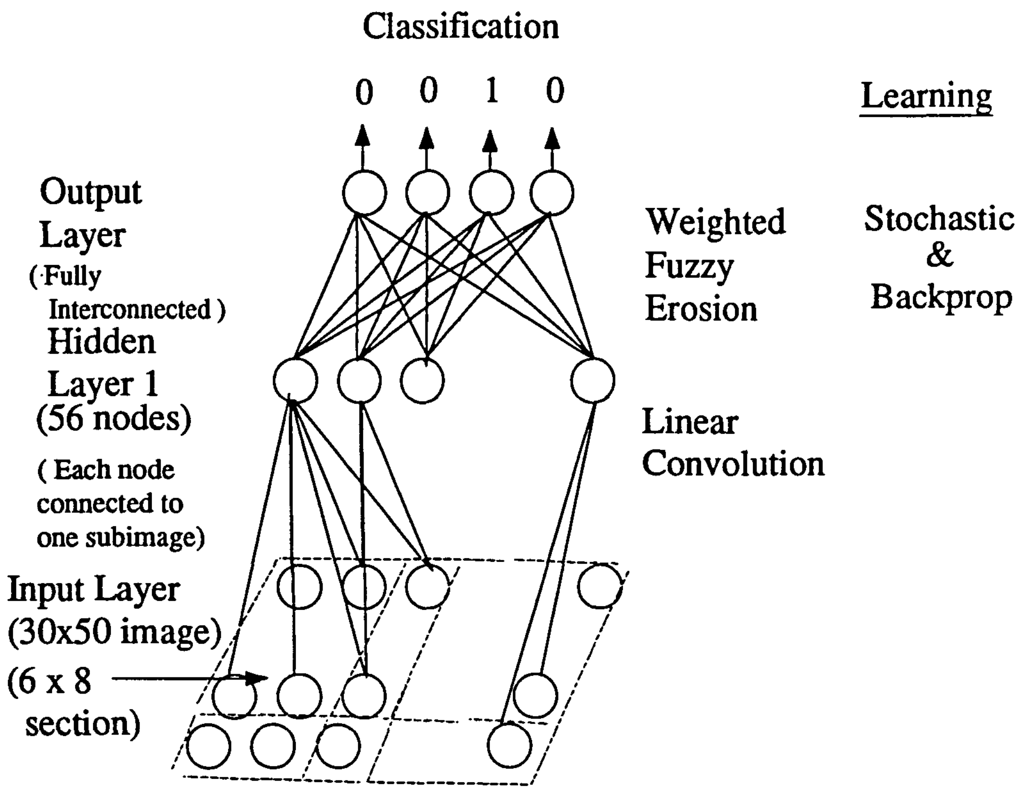


Figure 25. Network architecture II for the FIANN

4. Training the net: Network training for the output and hidden layers is the same as for the previous net.

The results obtained are presented in the next chapter.

IX. IMPLEMENTATION AND RESULTS

The net implementation was carried out in two stages. In the first stage a one layer FIANN was designed whose training data consisted of N pairs of an input image and a corresponding desired output image. The FIANN's erosion and convolution properties were tested and optimal training rules obtained. In the second stage, a multilayer FIANN was implemented to carry out target classification. A more detailed description of each stage now follows.

Testing the FIANN's Erosion and Convolution Properties

Erosion property

A one layer net was implemented in which the number of nodes equalled the number of pixels in the input image. The network architecture and training are described in the previous section. The input was an image of size 16 x 16 or 64 x 64, and the desired output was its corresponding fuzzy eroded image c obtained as follows:

$$c_j = \bigwedge_{i \in S_1(t_j)} r(1 - t_{ji} + a_i).$$

Figure 26 shows one input and corresponding desired output pair. The generalized template g was initialized such that $w_{ji} = 1/|N_j|$ for all i, j, where $N_j = \{S_0(t) \cap S_1(t)\}$; template t and parameter P are initialized randomly. The net was trained to "learn" a template g such that weighted fuzzy erosion of input a with this template would produce the same output image as would the fuzzy erosion of a with $\bar{t}$, where $\bar{t}$ is some template commonly used for fuzzy erosion. Figure 27 shows the template used for creating the desired output, as well as the templates generated by the net. Several different training rules had to be applied before an optimal training rule was obtained. The net was trained using three different learning schemes. These were:

1. Stochastic Learning
2. Simulated Annealing
3. Gradient Descent

Figure 27 shows the template values obtained for the fuzzy erosion case with the different learning schemes. As can be seen, none of the templates match the original template exactly. However, it should be noted that the operations of fuzzy erosion and weighted fuzzy erosion are algebraically different operations, hence two different templates could produce the same output with these two operations. The lowest error, calculated using Equation (11), was obtained using the stochastic learning algorithm. Simulated annealing and gradient descent

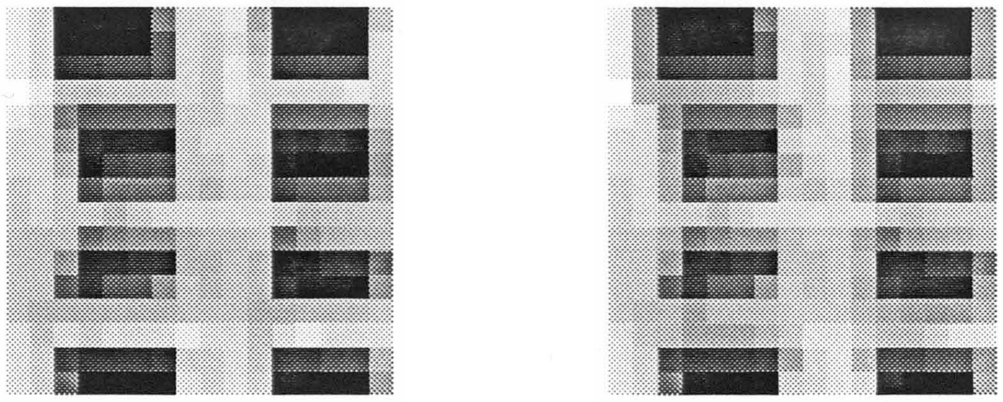


Figure 26. Input and corresponding fuzzy eroded image.

both did not give good results for this experiment. The output images for stochastic training are given in Figure 28.

Convolution Property

This was tested similarly on a one layer net where number of nodes equalled the number of pixels. Generalized template g was initialized such that $P = 1$ and $t_{ji} = 1$ for all i, j. The net was trained to "learn" a template g such that weighted fuzzy erosion of input a with g would be equivalent to convolution of a with template w given by:

$$c_j = \sum_{i \in S_0(t_j)} a_i * w_{ji}$$

Figure 29 gives an input image and its desired output image. This property was tested using the gradient descent scheme. This learning method gave fairly good results. As can

0.333	0.083	0.333
0.416	1.0	0.416
0.333	0.083	0.333

Original Template

-0.56	-0.06	0.41
0.75	1.12	-3.7
0.0	-1.01	0.35

Template from gradient descent training

0.9	-.484	1.227
.484	1.13	.444
.603	.516	.066

Template from simulated annealing

.401	.102	.388
.544	1.1	.391
.333	.09	.54

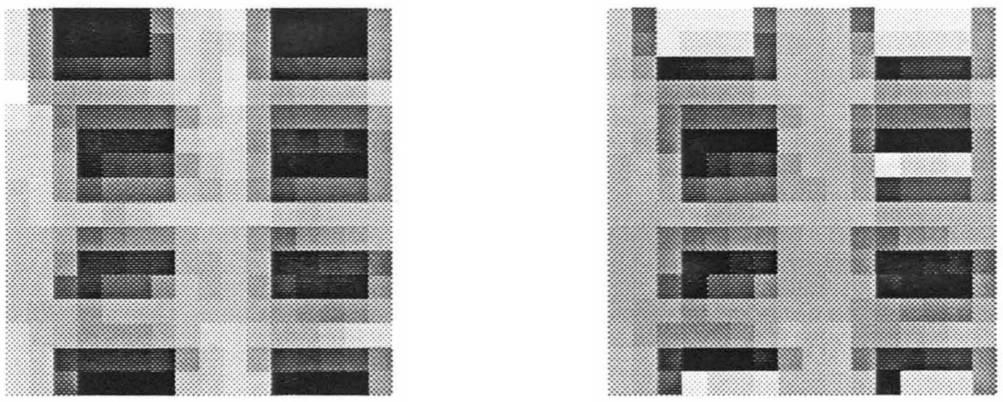
Template from stochastic learning

Figure 27. Templates obtained for different training schemes.

be seen in Figure 30, the template values do not correspond to one another but the template shown below produced an output image closest to the desired image. This is displayed in Figure 31.

Target Classification using a Multilayer FIANN

The network architecture and learning rules are explained in the previous chapter. Three different classification problems were addressed: one class, two class, and a three class problem. These are now described in detail.



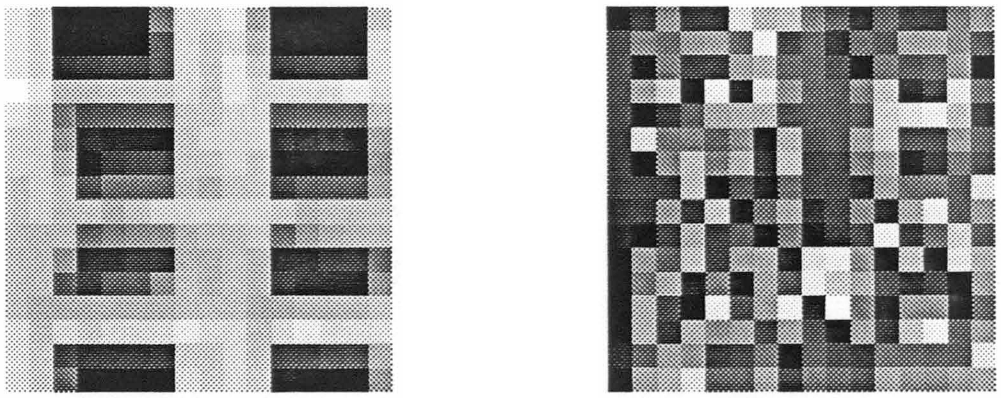
Desired output image

Image obtained from weights trained by the stochastic learning scheme.

Figure 28. Desired output and actual output produced by the FIANN.

One class problem

The first test that was done was to train the net on a single data class. The tank used was the M60. The FIANN was trained on an M60 of one particular orientation and tested on an M60 of a different orientation. The results for this are given in Table 4.



Original Image Image convolved with ideal template

Figure 29. Input and desired output image pair.

.333	.083	.333
.416	1.0	.416
.333	.083	.333

Original Template

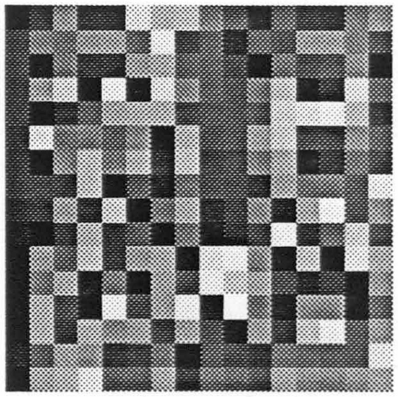
4.06	3.08	5.6
4.51	2.7	5.89
3.85	3.54	6.89

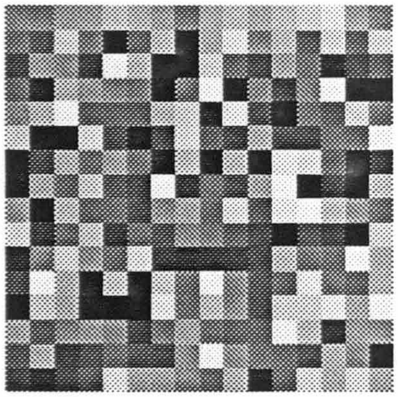
Template from gradient descent learning

Figure 30. Results for the convolution property of the FIANN.

Two class problem

The two classes consisted of one tank and one nontank class. The tank used was an M60 and the nontank data consisted mainly of terrain data. The results are given in Table 5.





Desired output image Image obtained from trained weights

Figure 31. Desired and actual output for convolution property.

Table 4 Results for one class problem using the FIANN.

Training Data		Recall Data	% Classified as the correct tank	% Classified as the wrong tank
M60	20	20	78%	22%

Three class problem

The net was trained on three different sets of training data consisting of the following sets of classes:

1. Data Set I contained the following classes:

   a. Tank of type M60.

   b. Artillery of type M53.

   c. Nontank.

2. Data Set II contained the following classes:

   a. Tank of type M47.

   b. Missile launcher of type M42.

   c. Nontank.

3. Data Set III was:

   a. M114 artillery tank

   b. 2.5 ton truck

   c. Nontank.

Table 5 Results for two class problem using the FIANN.

Training Data		Recall Data	% Classified as the correct tank	% Classified as the wrong tank	False alarm rate
M60	20	10	90%	8%	-
Nontank	20	10	70%		30%

The results obtained are given in Tables 6, 8, and 10. "False alarm" signifies that a nontank object was identified as a tank. The recall data was a combination of training data and new inputs. Sample range images for the various classes can be seen in Figures32, 33, 34 and 35. As can be seen in this figure, the data is varied. The tanks are at several different orientations and the images are of different sizes. We compared the FIANN's performance on this data with that of a regular backpropagation net. The backpropagation net performed better than the FIANN, but the results were still not very good. These results are given in Tables 7, 9, and 11.

Considering the complexity of the network, the FIANN gave a reasonably good performance on the range data. Further, the data used for training purposes was real data. This data was extracted from several different range images, each having a different range offset

Table 6 Results for M60, M53, and nontank using FIANN.

Training Data		Recall Data	% Classified as the correct tank	% Classified as the wrong tank	False Alarm
M60	32	16	83%	17%	-
M53	32	16	76%	24%	-
Non-tanks	32	8	80%	-	20%

Table 7 Results for M60, M53, and nontank with backpropagation.

Training Data		Recall Data	% Classified as the correct tank	% Classified as the wrong tank	False Alarm
M60	32	16	88%	12%	-
M53	32	16	91%	9%	-
Non-tanks	32	8	84%	-	16%

and sensor angle. The horizontal and vertical resolutions also varied in these images. The elevation data obtained was very sparse and had to be filled in using linear interpolation. Thus, the data used was fairly noisy. This could be a reason for the inconsistent results.

Table 8 Results for M42, M47, and nontank using FIANN.

Training Data		Recall Data	% Classified as the correct tank	% Classified as the wrong tank	False Alarm
M42	20	10	74%	26%	-
M47	20	10	68%	32%	-
Non-tanks	20	10	84%	-	16%

Table 9 Results for M42, M47, and nontank with backpropagation.

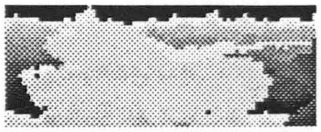
Training Data		Recall Data	% Classified as the correct tank	% Classified as the wrong tank	False Alarm
M42	20	10	78%	22%	-
M47	20	10	75%	25%	-
Non-tanks	20	10	86%	-	14%

Table 10 Results for M114, 2.5T truck, and nontank using FIANN.

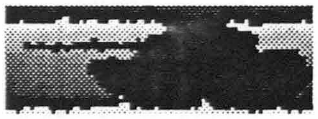
Training Data		Recall Data	% Classified as the correct tank	% Classified as the wrong tank	False Alarm
M114	20	10	80%	20%	-
2.5T	20	10	74%	26%	-
Non-tanks	20	10	82%	-	18%

Table 11 Results for M114, 2.5T truck, and nontank with backpropagation.

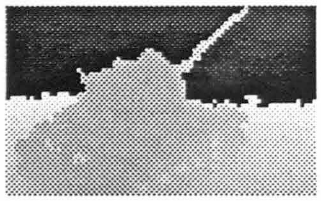
Training Data		Recall Data	% Classified as the correct tank	% Classified as the wrong tank	False Alarm
M114	20	10	86%	14%	-
2.5T	20	10	82%	18%	-
Non-tanks	20	10	84%	-	16%



M60 Tanks (30 x 76)

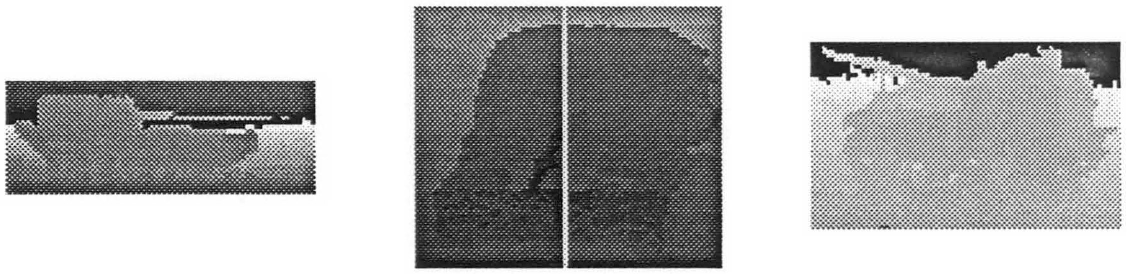


M60 Tanks (30 x 86)



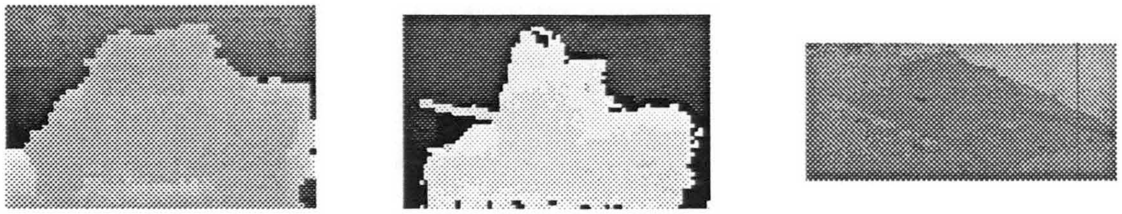
M60 Tanks (48 x 78)

Figure 32. Sample range images of M60 tanks.



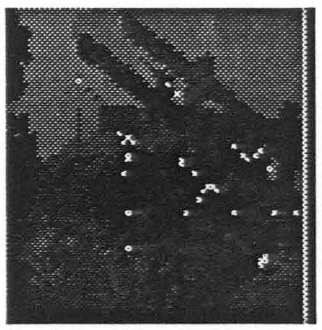
M53 Tanks (26 x 70) M53 Tanks (62 x 72) M53 Tanks (48 x 78)

Figure 33. Sample range images of M53 tanks.

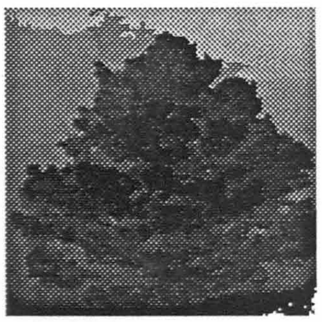


M47 Tanks (34 x 52) M47 Tanks (48 x 76) M47 Tanks (98 x 218)

Figure 34. Sample range images of M47 tanks.



M42 Tank (63 x 62)



Nontank (Tree)

Figure 35. Sample range images of an M42 tank and a tree.

X. CONCLUSIONS

Various experiments were carried out to improve the performance of the net. The starting point was to try various learning algorithms to improve weight training. The three methods used were gradient descent, stochastic learning and simulated annealing. Gradient descent gave fairly promising results for the FIANNs convolution property, while stochastic learning performed well for its erosion property. Simulated annealing did not perform well in training the net. Stochastic learning allowed the weights to jump to a lower error space and then used a traditional method like gradient descent from that point. This facilitated fast training and helped avoid the problem ofgetting stuck in local minima, which gradient descent techniques can run into.

The next stage of research was to apply the net for target classification. The first step here was designing the network architecture. Once again, we experimented with a number of different architectures. The basis for the design was a standard backpropagation network. We used two main architectures, one with a single hidden layer, and another with two hidden layers. The one layer net did not perform well. Since we were using an input image of size 30 x 50, this single layer had to aggregate information from 1500 pixels and pass them to the output layer for classification. Further, we allowed only partial connection between the input layer and this hidden layer. Thus this layer learned the features of the input image, but the net had no means of aggregating this information. Using a second hidden layer to combine the information helped improve classification results. Thus the final net used had two hidden layers.

Another point for experimentation was to change the image subset size, thus changing the features learned by the first hidden layer. We experimented with different sizes such as

5 x 7, 10 x 12 and finally settled on 6 x 8. The reason for not taking a smaller size was purely practical. Training time increased considerably, plus the program compiler could not handle such large amounts of data at a time. Larger subimage sizes led to fewer features for the net to learn which reduced the classification percentages. So we optimized by taking a midsize subimage.

Once subimage size and network architecture had been decided on, data had to be run on the net to obtain classification results. The first set of experiments was carried out for two classes, an M60 tank class and one nontank class. We obtained a classification rate of 90% in this case. However the false alarm rate was a high 30%.

To test the nets performance on variant data, we tried a one class problem in which the net was trained on a tank of one orientation and then tested on a similar tank, but of a different orientation. The FIANN did not perform well here, giving just 78% classification.

The final step was to test the net's classification abilities when more than one type of tank was used. Three different cases were tried and the net gave average results in all three.

Several factors came into the picture and contributed to these results. The first factor is the data used in the research. We use range data from LADAR tank data. This data has a high level of resolution. It is first converted to height data using the cartesian method, and then classified. However, the height data obtained through the cartesian method was noisy and did not give distinct features for each type of tank. It had several problems associated with it, such as irregular elevation points and range shadow. Hence classification results were not very good. A different approach for converting range data to height data would help in this case. Secondly the net was very sensitive to small variations in weights and depending on the initial weights, classification results could vary considerably. Third, one of the basic conditions for the generalized mean was that the sum of the numbers whose mean is being taken should be zero. No such precondition was applied to this data. All these factors added together to produce only fair results.

The FIANN shows great potential as a pattern classifier [24]. The network's differentiable activation function, based on the generalized mean, requires careful selection of initial values for the parameters. The training schedule for the FIANN, as formulated in our research, is a complex combination of several different training schemes. Hence, parameter selection is a critical factor in network training. The results obtained so far are quite promising, and further research into the FIANN appears viable. Selection of initial starting points for the weight vectors, finding optimum values for the various parameters used in training and weight update, as well as different training schemes for the net, are all areas requiring further investigation. Further, the network architecture proposed in the report, which is a variation of the standard backpropagation net, is just one type of architecture possible. For example, this net could be modified to have different numbers of hidden layers and nodes. A completely different architecture, one that is not based on the backpropagation algorithm, could also provide more experimentation. The network's performance on data other than elevation data would be an indicator of its pattern recognition capabilities. A more efficient conversion from range to elevation data, such as the locus method proposed in [29], could be explored to improve performance of the system.

The results from this thesis will be presented at the Fifth Annual Image Algebra and Morphological Image Processing Conference in San Diego, California, July 24 — 29, 1994.

REFERENCES

[1] P. D. Gader, "Template generation for pattern classification," *Proc. of the 1992 SPIE Image Algebra and Morph. Image Proc. III*, vol. 1769, pp. 72–81, 1992.

[2] J. Bevington, R. Johnston, and J. Lee, "A modular target recognition algorithm for ladar," *Second Automatic Target Recognizer Systems and Technology Conference*, pp. 91–104, 1992.

[3] S. Z. Li, "Towards 3d vision from range images: An optimization framework and parallel networks," *Computer Vision, Graphics and Image Processing*, vol. 55, pp. 231–260, May 1992.

[4] R. Krishnapuram and S. Gupta, "Morphological methods for detection and classification of edges in range images," *Journal of Mathematical Imaging and Vision*, vol. 2, pp. 351–374, 1992.

[5] J. G. Verly and R. L. Delanoy, "Adaptive math morphology for range imagery," *IEEE Transactions on Image Processing*, vol. 2, pp. 272–275, Apr. 1993.

[6] J. G. Verly, R. Delanoy, and D. E. Dudgeon, "Model-based system for automatic target recognition from forward-looking laser-radar imagery," *Optical Engineering*, vol. 31, pp. 2540–2552, Dec. 1992.

[7] C. E. Daniell, D. H. Kemsley, W. P. Lincoln, W. A. Tackett, and G. A. Baraghimian, "Artificial neural networks for automatic target recognition," *Optical Engineering*, vol. 31, pp. 2521–2531, Dec. 1992.

[8] W. A. Thoet, T. G. Rainey, D. W. Brettle, L. A. Slutz, and F. S. Weingard, "Anvil neural network program for three-dimensional automatic target recognition," *Optical Engineering*, vol. 31, pp. 2532–2539, Dec. 1992.

[9] F. Sadjadi and E. Hall, "Numerical computations of moment invariants for scene analysis," *Proc. of the IEEE Conference on Image Processing and Pattern Recognition*, 1978.

[10] H. Barrow, "Parametric correspondence and chamfer matching: two new techniques for image matching," *Proc. of the DARPA Image Understanding Workshop*, 1977.

[11] F. Sadjadi, "Object recognition using coding schemes," *Optical Engineering*, vol. 31, pp. 2580–2583, Dec. 1992.

[12] S. R. F. Sims and B. V. Dasarathy, "Automatic target recognition using a passive multisensor suite," *Optical Engineering*, vol. 31, pp. 2584–2593, Dec. 1992.

[13] M. V. Shirvalkar and M. M. Trivedi, "Developing texture-based image clutter measures for object detection," *Optical Engineering*, vol. 31, pp. 2628–2639, Dec. 1992.

[14] J. D. Leonard and E. G. Zelnio, "Intrinsic seperability of laser radar imagery," *2nd Automatic Target Recognizer Systems and Technology Conference*, pp. 85–99, Mar. 1992.

[15] O. D. Faugeras, M. Herbert, and E. Pauchon, "Segmentation of range data into planar and quadric patches," *Proceedings of the 3rd Computer Vision and Pattern Recognition Conference*, pp. 8–13, 1983.

[16] k. S. Fu and J. K. Mui, "A survey on image segmentation," *Pattern Recognition*, vol. 13, pp. 3–16, 1981.

[17] R. Ohlander, *Analysis of natural scenes,Ph.D. Dissertation.* Pittsburgh, PA: Dept. of Comp. Sc., Carnegie-Mellon University, 1975.

[18] R. Hoffman and A. K. Jain, "Segmentation and classification of range images," *IEEE Trans. on Pattern Analysis and Machine Analysis. PAMI-9*, vol. 5, pp. 608–620, Sep.

1987.

[19] B. K. P. Horn, "Extended gaussian images," *Proceedings of IEEE*, vol. 72, pp. 1656–1678, Dec. 1984.

[20] G. J. Agin and T. O. Binford, "Computer description of curved objects," *Proc. of the 3rd International Joint Conference on Artificial Intelligence*, pp. 629–640, August 1973.

[21] R. O. Duda and P. E. Hart, "The use of hough transform to detect lines and curves in pictures," *Comm. ACM*, vol. 15, pp. 11–15, 1979.

[22] D. L. Milgrim and C. M. Bjorklund, "Range image processing: planar surface extraction," *Proceedings of the 5th International Conference on Pattern Recognition*, pp. 912–919, Dec. 1980.

[23] A. P. Reeves, R. J. Prokop, and R. W. Taylor, "Shape analysis of 3-d objects using range information," *Proceedings of Computer Vision and Pattern Recognition Conference*, pp. 452–457, June 1985.

[24] J. L. Davidson, "Performing target classification using a fuzzy morphology neural net," *Final Report for: Faculty Summer Research Program*, 1992.

[25] G. X. Ritter, J. N. Wilson, and J. L. Davidson, "Image algebra: an overview," *Computer Vision, Graphics and Image Processing*, vol. 49, pp. 297–331, 1990.

[26] L. A. Zadeh, "Making computers think like people," *I.E.E.E. Spectrum*, pp. 26–32, Aug. 1984.

[27] T. Radecki, "An evaluation of the fuzzy set theory approach to information retrieval," *Progress in Cybernetics and System research, Proc.*, vol. 11, 1982.

[28] K. Hackett and M. Shah, "Segmentation using intensity and range data," *Optical Engineering*, vol. 28, pp. 667–674, June 1989.

[29] I. S. Kweon and T. Kanade, "High resolution terrain map from multiple sensor data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 278–292, Feb. 1992.

[30] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology*, vol. 5, pp. 115–133, 1943.

[31] J. V. Neumann, *The general and logical theory of automata*, pp. 1–41. New York: Wiley, 1951.

[32] D. Hebb, *The Organization of Behavior*. New York: Wiley, 1949.

[33] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychology Review*, vol. 65, pp. 386–408, 1958.

[34] B. Widrow, *Generalization and information storage in networks of ADALINE neurons.* Washington DC: Spartan Books, 1962.

[35] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554–2558, 1982.

[36] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, I and II.* Cambridge MA: MIT Press, 1986.

[37] R. Lippmann, "An introduction to computing with neural nets," *ASSP Magazine*, vol. 4, pp. 4–22, Nov. 1987.

[38] R. Krishnapuram and J. Lee, "Fuzzy-set-based hierarchical networks for information fusion in computer vision," *Neural Networks*, vol. 5, pp. 335–350, 1992.

[39] E. B. Bartlett, "A stochastic training algorithm for artificial neural networks," *Neurocomputing*, vol. 6, no. 1, pp. 1–13, 1994.

[40] H. A. C. Eaton and T. L. Olivier, "Learning coefficient dependence on training size," *Neural Networks*, vol. 5, pp. 283–288, 1992.

ACKNOWLEDGMENTS

I thank my major professor, Dr. Jennifer Davidson, for her constant encouragement and support during the course of my M.S. My interactions with her, and the other members of the research group, during my stay at Iowa State was a stimulating experience.