ASPHALT PAVEMENT CRACK CLASSIFICATION: A COMPARATIVE STUDY OF THREE AI APPROACHES: MULTILAYER PERCEPTRON, GENETIC ALGORITHMS, AND SELF-ORGANIZING MAPS

Haroun Rababaah

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Master of Sciences
in the Department of Computer and Information Sciences,
Indiana University South Bend
May 2005

Accepted by the Graduate Faculty, Indiana University South Bend, in partial fulfillment of the requirements for the degree of Master of Science.

Chairperson, James Wolfer, Ph.D.

Dana Vrajitoru, Ph.D.

Master Thesis Committee

Jerry H. Mohajeri P.E.

Morteza Shafii-Mousavi, Ph.D.

DEDICATION

In memory of my father Rasheed A. Rababaah, to my mother, and to all of my family.

H.R.

ACKNOWLEDGMENTS

I would like to acknowledge all people contributions to this thesis through their supervision, efforts, ideas, and support. Dr. James Wolfer, my adviser, for providing me with resources I have needed, and his generous and great support and guidance throughout the study. Dr. Dana Vrajitoru for her kind contribution in the genetic algorithm approach; the classification model, and the GAD software. Mr. Jerry H. Mohajeri P.E. for being the inspiration of this study, for his generous support and vital guidance in data collection and verification, and for being the domain expert for the study. Mr. Jerry Dekeyser, the systems administrator at the department of computer and information sciences for making the sunone IDE studio available for our study on the department network. Ibrahim Chaaban, for his great suggestion to improve the efficiency of our version of nearest-neighbor class resolution, in the GA and SOM classification model.

Haroun Rababaah

ASPHALT PAVEMENT CRACK CLASSIFICATION: A COMPARATIVE STUDY OF THREE AI APPROACHES: MULTILAYER PERCEPTRON, GENETIC ALGORITHMS, AND SELF-ORGANIZING MAPS

ABSTRACT

This study presents a comparison of three Artificial Intelligence (AI) approaches: multilayer perceptron (MLP), genetic algorithms (GA) and self-organizing maps (SOM) to improve automated asphalt pavement crack classification using computer vision. Our system consists of three stages: 1. Image preprocessing, and crack detection 2. Feature extraction, and image representation 3. Image classification. The first stage was done by; thresholding, median filtering, and tiling and local linear regression. The second stage was done by two methods; the first method was Hough transform, which produces feature vectors of the form {average Hough angle, number of crack hits in the image}, and the second method was crack hit projection, which produces a feature vector of the form {number of hits projected on the X-axis, number of hits projected on the Y-axis}, a new approach we introduce, which we believe enhances the classification capability of the system. In the final stage, the three AI approaches were trained, and tested using real pavement crack images. The MLP model was designed as a (2-node input layer, 3-node hidden layer, and 4-node output layer) network. The GA model was used to evolve a two dimensional matrix, containing the labels of the different crack classes. In the SOM model a matrix similar to the one resulted in the GA approach was produced from the best maps we trained. For both models (GA & SOM), the class of the image was found by looking-up the label at the coordinates of the classifier matrix represented by the feature vector. We implemented the nearest neighbor algorithm in case of a NULL cell in the classifier matrix. We chose this third approach (SOM) as a typical non-supervised learning method, to compare it with the first two (MLP & GA) which are considered as supervised learning methods. The scope of the study considered the main four types of asphalt cracks: Alligator, Block, Longitudinal, and Transverse cracks. An extensive testing was conducted following a systematic procedure of training and testing for all classifiers developed. The best classifiers gave the following accuracy: Hough/MLP at 93.6%, Projection/MLP at 98.6%, Hough/GA at 89.2%, Projection/GA at 98.2%, Hough/SOM at 86%, and Projection/SOM at 98.4%.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDIXES

1. INTRODUCTION TO ASPHALT DISTRESSES

Large structures are usually constructed with materials that exhibit distresses after construction because of loading, environmental conditions, and aging. The large structures include pavement, chimneys of nuclear power plants, skyscrapers, pipelines, and others. The distresses are presented in the form of surface cracking in most situations [1]. One of these structures, asphalt pavement, is the focus of the thesis.

1.1. Different Types of Pavement Distresses

In this section, various types of asphalt pavement distress classes are briefly discussed and a subset of interest is defined. These definitions conform to those found in US department of transportation distress identification manual [7].

Patching and Potholes

- Patch/Patch deterioration: Portion of pavement surface, greater than 0.1 m^2, that has been removed and replaced or additional material applied to the pavement after original construction.
- Pothole: Bowl-shaped holes of various sizes in the pavement surface.

Surface Deformations

- Rutting: A rut is a longitudinal surface depression in the wheel path. It may have associated transverse displacement.
- Shoving: Shoving is a longitudinal displacement of a localized area of the pavement surface.

Surface Defects

- Bleeding: Excess bituminous binder occurring on the pavement surface, usually found in the wheel paths.

- Polished Aggregate: Surface binder worn away to expose coarse aggregate.
- Raveling: Wearing away of the pavement surface caused by the dislodging of aggregate particles and loss of asphalt binder.

Miscellaneous Distresses

- Lane-to-Shoulder Drop-off: Difference in elevation between the traveled surface and the outside shoulder.
- Water Bleeding and Pumping: Seeping or ejection of water from beneath the pavement through cracks.

Cracking

- Alligator cracking as shown in Figure 1.1: Occurs in areas subjected to repeated traffic loadings (wheel paths). It can be a series of interconnected cracks in early stages of development. Develops into many-sided, sharp-angled pieces, usually less than 0.3 m^2 on the longest side, characteristically with a chicken wire/alligator pattern, in later stages.

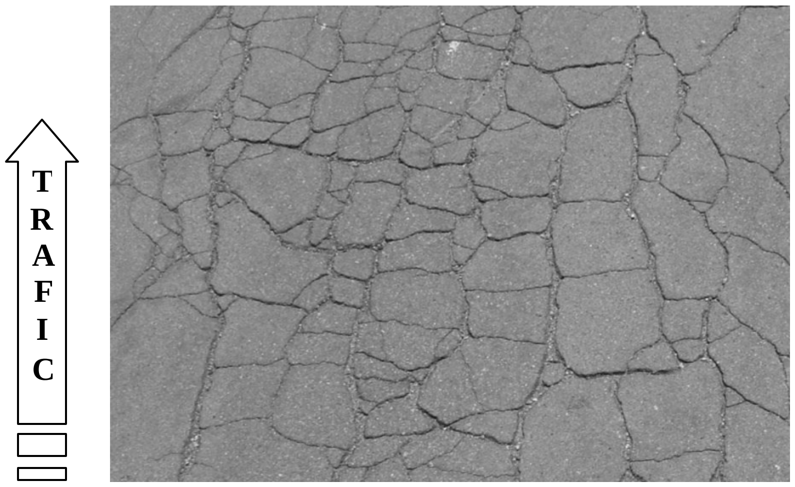


Figure 1.1 Alligator cracking

- Block Cracking, shown in Figure 1.2: A pattern of cracks that divides the pavement into approximately rectangular pieces. Rectangular blocks range in size from approximately 0.1 m^2 to 10 m^2.

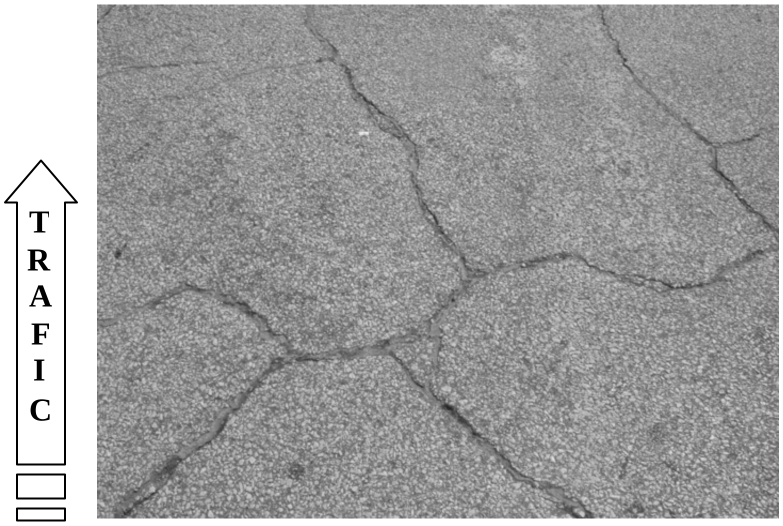


Figure 1.2 Block cracking

- Longitudinal Cracking, shown in Figure 1.3: Cracks predominantly parallel to pavement centerline.

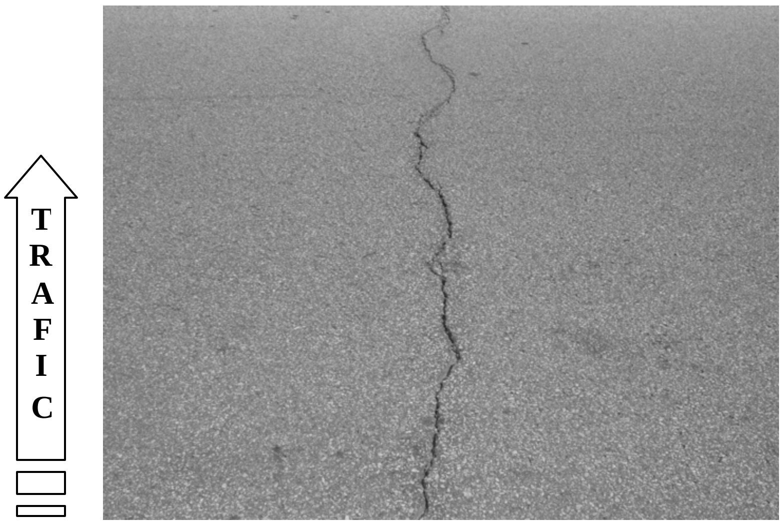


Figure 1.3 Longitudinal cracking

- Transverse Cracking, shown in Figure 1.4: Cracks those are predominantly perpendicular to pavement centerline.

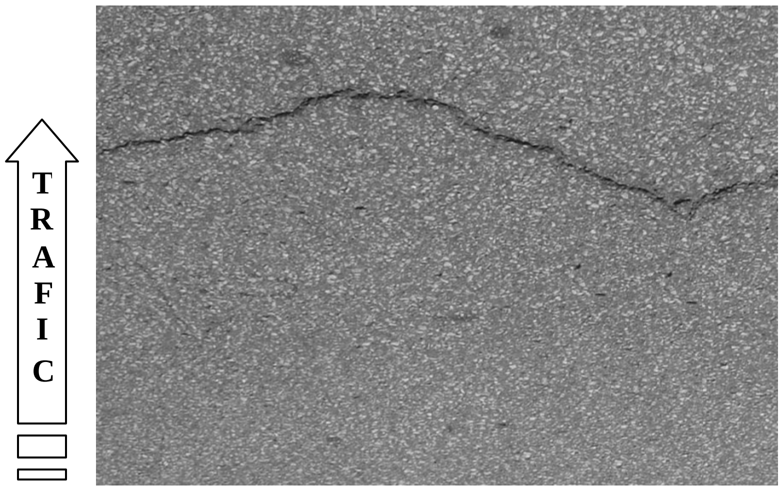


Figure 1.4 Transverse cracking

1.2. Distress Data Collection

A typical pavement management system consists of data collection, data verification and data analysis. A pavement condition survey and analysis provides much of the information necessary for pavement management, and is vital in order to maintain a quantified condition of network, more accurate and accessible information, track performance of treatments, forecast pavement performance, anticipate maintenance and rehabilitation needs, establish maintenance and rehabilitation priorities, and allocate funding. Therefore, it is critical to collect accurate pavement condition data in an efficient and safe manner and to design a reliable analysis system [1, 2, 3].

1.2.1. Manual methods

There are two basic methods for conducting manual pavement condition surveys, walking and windshield surveys. Walking and windshield surveys are also commonly combined to provide a more complete pavement network survey.

Walking surveys are completed by an expert who is trained to rate pavement distresses according to the agency's distress identification specifications. The expert walks down the side of the pavement and fills out a pavement condition form that describes the amount, extent, and severity of each distress present or a randomly selected sample of the roadway.

A windshield survey is completed by driving along the road or on the shoulder of the road. The pavement is visually rated through the windshield of the vehicle. This method allows for greater coverage in less time; however, the quality of the pavement distress data is compromised.

1.2.2. Automatic methods

Automated surveys use technologically complex vehicles traveling at highway speeds to collect and store data (Figure 1.5). There are several types of automated pavement survey vehicles available differing in data collection technology; however, these all share the same goal of collecting accurate pavement condition data. Examples of theses technologies include:

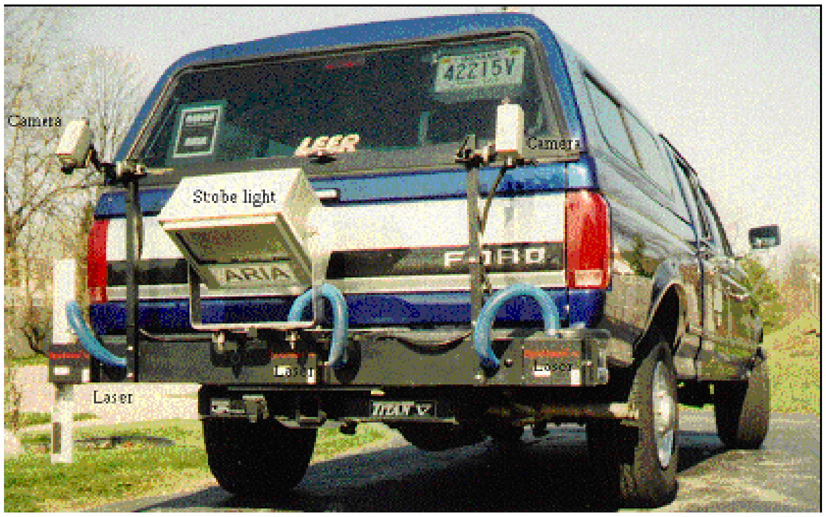


Figure 1.5 Pavement survey vehicles [3]

- Analog and digital cameras are generally used for pavement surface distresses.
- Ultrasound or Laser technology to capture the rutting (the transverse profile of the road).
- High frequency laser to collect the texture of the pavement.

1.2.3. Manual vs. Automatic data collection

The manual collection system is self-validating in the sense that all data are collected by an expert. In automatic collection, data needs to be verified by randomly sampling a percentage of the data and validating it against the actual corresponding pavement. Requirements specifying this percentage vary from state to state.

In any engineering problem it is critical to go through the alternatives evaluation phase which leads to decision making. Table1.1. contains a comparison between the two methods of pavement data collection, manual and automatic [1, 8].

Table 1.1 Manual vs. automatic pavement surveys

Manual	**Automatic**
Expensive and time consuming.	Less expensive and fast.
Labor intensive	Very minimal labor is needed
Hazardous	Safe
Data sampling.	100% survey.
Subjective.	Objective
Difficult to manage	Integratable with management system
Repeatability is low	Proven to be much better [4]

After collecting the pavement data using the techniques described above, the data is analyzed manually or automatically based upon standard formulas and indices which also vary from state to state. Examples of these parameters include:

- Mean slope variance, the variance of slopes measured over a 6-inch wheel.
- Mean rut depth.
- Pavement cracking in ft/1000 ft^2 of pavement surface.
- Patching in ft^2/1000 ft^2 of pavement surface.
- Roughness or present serviceability index.
- Severe localized raveling (Code: 0 = none, 1 = present).

A pavement distress index is then formulated from combinations of these parameters, and used to indicate pavement quality.

2. MOTIVATION OF THE STUDY

Observing methods employed by experts in the pavement assessment industry [3] we noted the following limitations in current practice:

- They involve computing exact features of crack patterns, which can be computationally expensive.
- A hard coded criterion is not likely to perform well in the harsh crack-pavement environment which includes a significant amount of noise and stochastic distribution and geometry of cracks.
- The processing speed is important, since the project of pavement assessment involves surveying 100% of target pavement. Processing time per image plays an important factor in motivating us to design more efficient techniques, especially if the analysis application required real-time processing.

With this context, we believed that this may be a good test bench to assess the real world potential of several AI approaches, specifically speaking, multilayer perceptron, genetic algorithms and self-organizing maps. We chose the three approaches because they represent two main learning categories: supervised (MLP and GA), and the unsupervised learning (SOM). Further more we believed that with these variety of methods and using different computer vision techniques, we can have a rich experimental input, which in turn should produce a rich output that should give us the ability to compare and contrast among the different methods.

It appeared that from the data comparison between results of a study of one of the commercial software, and results from manual survey, that the automated system has no difficulty of finding cracks. The problem lies in the classification and quantification of the cracks. This problem is not vendor specific and has been a research topic for years [4]. The current systems do not have much of problem detecting the existence of cracks, but distinguishing between crack types is a challenge [11]. These two studies significantly enhanced the motivation to investigate other techniques that we hoped will perform better in the classification problem.

3. PREVIOUS WORK

Chou et al. [12]. Cheng introduced a novel approach of moment invariants and neural networks. There approach is summarized as following:

Rotation and image enhancement:

In order to generate enough image samples needed to train the neural network module and to test moment invariant as well, they used image rotation. The images then are enhanced by using the Z-function algorithm described in their paper. For further image enhancement smoothing technique was applied by using an averaging filter (mean filter).

The resulting images then were thresholded using a fuzzy technique, which depends on maximizing of fuzzy entropy; the method is described in their paper, which we might consider if we decide later to automate our thresholding phase.

Seven types of cracks were considered namely: longitudinal, transverse, combined, right and left diagonal (edge cracks), alligator and no crack. Three different moment-based features were used: Hu moments, Bamieh moments, and Zemike moments. Each one has several moments that are computed by processing the resulting binary images.

Eighteen nodes were designed to accept the vector formed by the different moments computed previously and seven nodes for the output were used in a neural network classifier system with back propagation. The results were 100% accurate.

Mohajeri and Manning [3] presented a new system, which was an attempt to fully automate a pavement (asphalt & concrete) management. It uses an analog video camera setup in the front of a vehicle to video tape the road surface. The video signal then is sent to an image processing unit to convert the video signal into digital frames by using a frame grabber unit. Then the images are filtered using a multiplication filter to reduce the noise and to enhance the crack objects in the digital image.

The images are then converted into gray-scale images and then to binary images using dynamic thresholding which improves the segmentation. Since each image will have a different histogram distribution, it is very important to choose the threshold that will produce the best segmentation of cracks from the background.

The classification was done by relying on the characteristics of each individual crack pattern and using the elimination technique. The system did not only classify cracks but also was able to quantify and qualify different types of cracks: longitudinal, transverse, block, and alligator, by computing the length, width, and area of the cracks, which are used to determine the severity of the cracks.

Lee and Cheng [5] presented an investigation about the performance of ANN using different techniques for image representation. Three different techniques were investigated and compared based on the classification capability. In all of the three techniques the tiling method was used to reduce the image data into a matrix of 0s and 1s, each cell represents a tile in the original image and a statistical approach is used to determine if that particular tile contains a crack object or not, and based on that the cell is set to 1 or 0. For training purposes they generated synthesized images using a simulator they introduced. They test the system with real pavement images.

This image-based technique operates on the entire tiles of the original image for example if the size of the image is: 512x512 and the size of the tile was 32x32, then the reduced image will have a vector of (512/32)(512/32) entries = 256 entry, and so the matrix representing the image will have a dimension of (16x16) filled of 1s and 0s. The resulting matrix then is fed into the MLP module to be trained. And so the ANN has to see every single tile in the image in order to learn the pattern.

In the histogram-based technique, two histograms were produced, horizontal and vertical. They simply accumulate the number of 1s in the resulting matrix in both directions to produce the vertical and the horizontal histograms. The training here is done by providing only the two vectors into the ANN, which will be able to recognize the different types of cracks by the distribution of number of crack objects accumulated in the entries of the vertical and horizontal histogram vectors. Taking the same example above we only need 16 vertical and 16 horizontal = 32 entries to plug into the ANN.

The proximity was extremely simplified technique which computes one proximity value for both the horizontal and the vertical histograms described above. The proximity value is computed as the total of the differences of the adjacent cells in the histogram. These two values in addition to the number of crack tiles in the image are presented in to the ANN module as a 3-node input layer only.

The results were surprising because the most accurate technique was the proximity-based one which was not expected since it uses only three values to represent any image. The classification results were: for image-based was 70.2%, for histogram-based was 75% and for proximity-based was 95.2%.

They used an interesting approach for image representation, which we partially adapted for our own. Their system was able to classify the four types of cracks mentioned in the scope with a remarkable accuracy about an average of 95%. But they failed to classify both alligator and block cracks at the same accuracy; for those types it was 88%. After studying their technique of image representation we think we can enhance the accuracy of the system by presenting a different technique, described in the methodology, that will ensure a better image representation and better contrast in the resulting vectors. This in turn will make the job easier and more accurate for the classifier systems to learn how to classify all of the cracks at a higher accuracy.

Wang et al. [6] presented a new automated system capable of collecting and analyzing Pavement cracks in real-time and with high-resolution digital images. Real-time processing is defined as processing data at the same data throughput as the vehicle is collecting images at highway-speed normally between 80 to 100 KPH.

The approach in this paper: the first step in the image processing process is to distinguish any cracks from other non distress noises. The primary method in this step relies on analytical descriptions of distresses' characteristics. The second step is to connect and vectorize the detected cracks, and establish a distress database related to location, orientation, and size of each crack. Based on the geometric information obtained in the second step, cracks can be classified using any pre-defined distress categorization protocols. The result of the analysis is contained in a database regarding the location and geometry of individual cracks. Several distress categorization protocols are incorporated into the system to generate surface crack indices.

An interesting point was addressed in this paper which is the importance aspect of detecting distress is that highway pavement surface may contain numerous foreign objects, such as oil residue, dirt, lane markings, vehicle's tire mark, tree limbs, and other non-distress related items. It is important to develop algorithms to correctly distinguish the distresses or cracks from these non-distress items. In addition, certain images collected in the field may also possess a quality level that may not be sufficient enough, therefore resulting in additional difficulty in processing. In this study we are not going to consider this problem but address it in the recommendation and future work.

Another important point was addressed in this paper is the repeatability of the system which is very important for any classification to be considered robust and reliable. This test was conducted on the introduced system and the results in the entire test showed similar patterns with a standard deviation of 15% of the average CI (crack indicator).

Wang and Elliot [4] investigated the commercial system WiseCrax (hardware and software) the following aspects were noticed:

Pavement surface images are collected with two continuous video cameras, covering the survey lane of about 4 meters. The cameras are black and white Charge-Coupled Device (CCD) cameras. Both cameras are supported by two stretched-out beams in the back of the vehicle and face perpendicularly to the pavement surface. Video images are recorded into S-VHS format. Images from each camera are stored sequentially in one tape. This storage technique is also called multiplexing. The images are demultiplexed when being processed.

A speed-encoding algorithm is applied so that simultaneous images from the two cameras and sequential images from one camera form a uniform pavement surface covering the entire lane. RoadWare indicates that the speed-encoding algorithm allows the cameras to capture images at 80 km/h. Camera shutters are synchronized with strobe lights to provide artificial lighting to ensure that (1) the cameras can get enough visual information in a very short period of time when the vehicle is traveling, (2) collected images are without shadows.

In WiseCrax, the crack identification process begins with the digitizing of the pavement video collected with the two cameras. The video is in analog format which must be converted into digital images for computer processing. 8-bit gray scale images are obtained from the digitization process. The identification process tried to identify each crack. The location of the beginning and end of each crack is referenced using an x-y coordinate system. The crack length, width, and orientation are also computed and saved. The process of digitizing to gathering statistics on individual cracks is similar to the process of "vectorizing" a raster image used in Geographical Information Systems. Once the crack "vectors" have been identified, the system plots them, creating a crack map of the pavement surface. A statistic report is also created during the crack identification phase. Each crack is represented in a single entry in the table, showing the location, start and end points, length, width, and orientation of individual cracks.

Since the definitions of distress categories vary from agency to agency, WiseCrax compares the location, length, and width of cracks against criteria for various crack distress categories. For instance, if cracks in a block pattern are more than 300 mm apart, they may be classified as a block cracking. If they are closer together, they may be classified as fatigue cracking. WiseCrax has the flexibility to process data as new classification definitions are developed.

WiseCrax operates in two modes: automated and interactive. In automated mode, all processing is done without human intervention, once the initialization parameters on pavement type, camera and light settings, etc. are set. Interactive mode allows the user to review, validate, and edit the WiseCrax results. For instance, the automated mode can be run first, the display shows the pavement image with overlaid color lines indicating the presence of cracks. The user can then point-and-click to add, delete, or modify the results. For quality control purposes, the interactive mode is normally used to perform statistical validation of automated results using random samples of data.

Lee [11] presented a survey of the various techniques and algorithms being used in this field. Some of the interesting topics presented in this paper include:

- *Limitation of computer vision*
    1. Noisy pavement surfaces.
    2. Lane markings, skid markings, etc.
    3. Lighting conditions changes with different times of day, cloud covers, shadow, etc.
    4. Reflectivity of paving materials complicates cracks detection.

5. Interreflected light illuminates the crack more than the pavement. This causes a big problem in crack objects segmentation.
6. Complete pavement survey requires huge amount of storage. As a solution for that the real-time parallel processing was introduced in some systems.

- *Crack image processing algorithms*

The two classes of algorithms used in most pavement distress analysis are:

1. Local operators: which are focused on determining if the local sub-area belongs to crack object or not.
2. Global operators: takes the result of local operators and determines the amount and orientation of the detected crack objects.

The author concluded that, the systems did not have much of a problem detecting the existence of cracks, but distinguishing between crack types was a challenge.

Hsu et al. [27] described a moment invariant technique for feature extraction and a neural network for crack classification. Distresses that were considered in there study are: Longitudinal, transverse, netted (alligator & block), and cavities (potholes).

The moment invariant technique reduces a two dimensional image pattern into feature vectors that characterize the image such as: translation, scale, rotation of an object in an image. After these features are extracted they are provided as input to the neural network module in order to classify the different types of pavement cracks. They used the back propagation technique to provide better fitness against noise.

The images were first captured as colored pictures, then they were converted into gray-scale images, and because in concrete pavements the cracks are always darker than the background pavement, the segmentation was fairly easy job which was done using thresholding to produce binary images. After that a filter operation was performed on them to gain some image enhancement by filtering some of the noise.

The overall results of this study were satisfactory and the classification accuracy of the introduced system was 85%.

4. METHODOLOGY

In section 3, we provided an overview of Pavement Management. Now we wish to define the scope of the thesis and its place in the pavement assessment process.

In this thesis we addressed a subset of Asphalt Crack Classification. Specifically, the four types of cracks as they are defined in the standard mentioned above: longitudinal, transverse, alligator and block cracks. In this section, we will present all methods we implemented in the different system phases to approach the problem addressed by our study. The system we developed consists of the following phases: image collection, preprocessing, crack detection, feature extraction & image representation, classification, and results comparison.

4.1. Image collection

This process was done by using a digital camera, by going to arbitrary streets and parking lots and collecting colored pictures of different crack types. Some precautions were taken to maintain consistent, reliable, and systematic image collection:

- Illumination: the illumination is important to the quality of the captured images which affect the contrast between the crack objects and the pavement background, so we always made sure that we take images during a bright day and at similar timing for consistency.
- Distance from the camera to the pavement: we maintained the same pavement-to-camera distance as much as possible for the entire collection process, since it affects the size of the crack and the relative gabs between crack objects in the acquired image, which is important for the classification process.
- Expert validation: the data collection technique was consulted and approved by a domain expert [3].

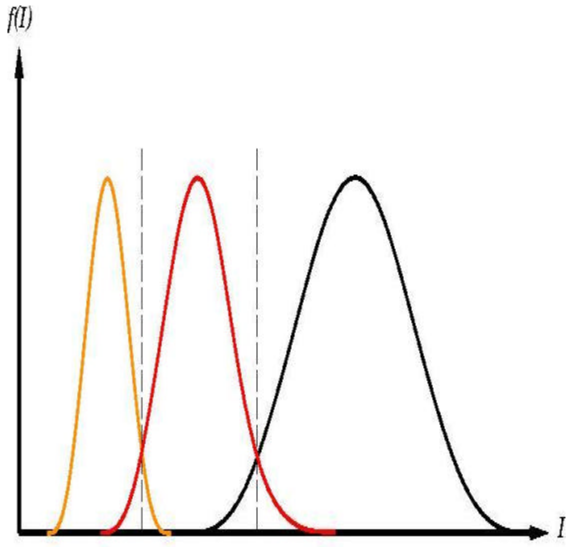
4.2. Image Preprocessing & Crack Detection

For the purpose of investigating various image analysis techniques, we designed a customized application framework to conduct a preliminary investigation. This phase included thresholding, noise filtering, image tiling and local linear regression.

4.2.1 Thresholding

There are several methods we studied in an attempt to find a technique that can do automatic thresholding. These methods include: P-Tile, Mode method, and Iterative thresholding [13]. All these methods require a bimodal histogram of the input image as shown in Figure 4.1. The preliminary investigation of the collected images showed that most histograms were not bimodal, as shown in Figure 4.2. This renders automatic thresholding, described above, ineffective for the study.

The results obtained by an edge detection technique in the segmentation of crack objects from the pavement background were unsatisfactory because this method rendered the noise bulkier which made it to be classified as small edges that would be difficult to distinguish from the cracks, and to be eliminated. A sample image is illustrated in Figure 4.3 therefore; we elected to proceed with manual thresholding, delaying the investigation of automatic thresholding for a future study. To demonstrate the difference between the two techniques (Edge detection & Thresholding), the result of thresholding a sample image is illustrated in Figure 4.4.



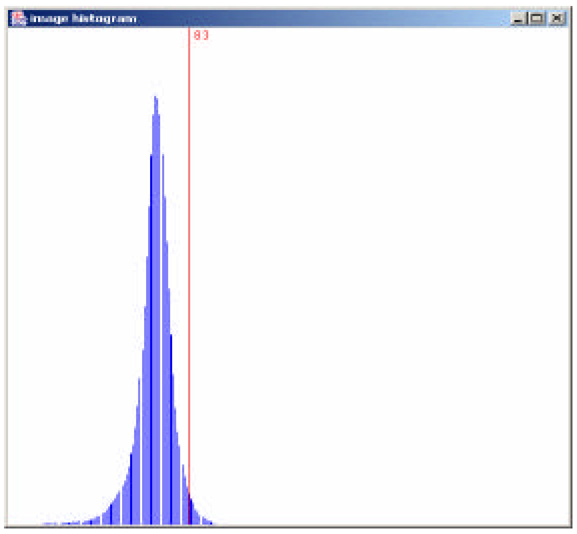


Figure 4.1 Object/background histogram [28]   **Figure 4.2** Histogram of a sample image

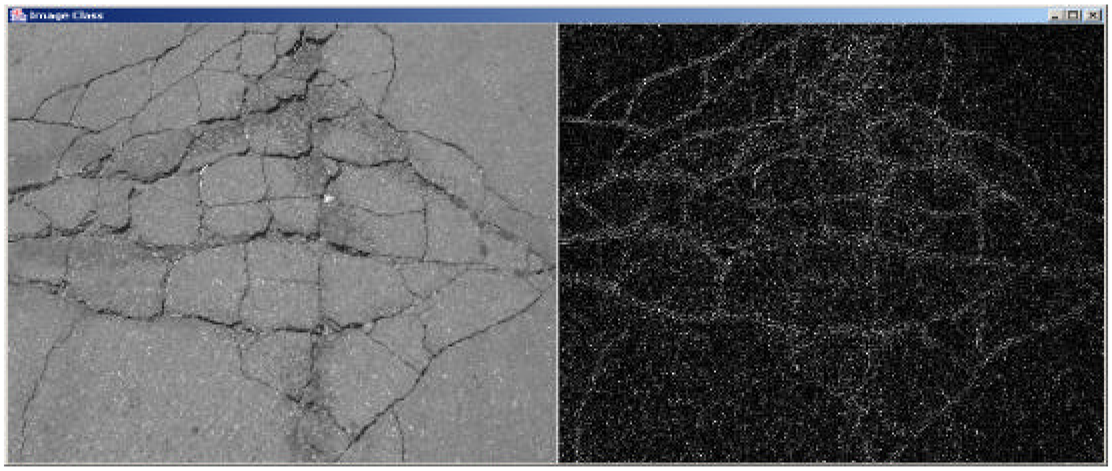


Figure 4.3 Edge-detection

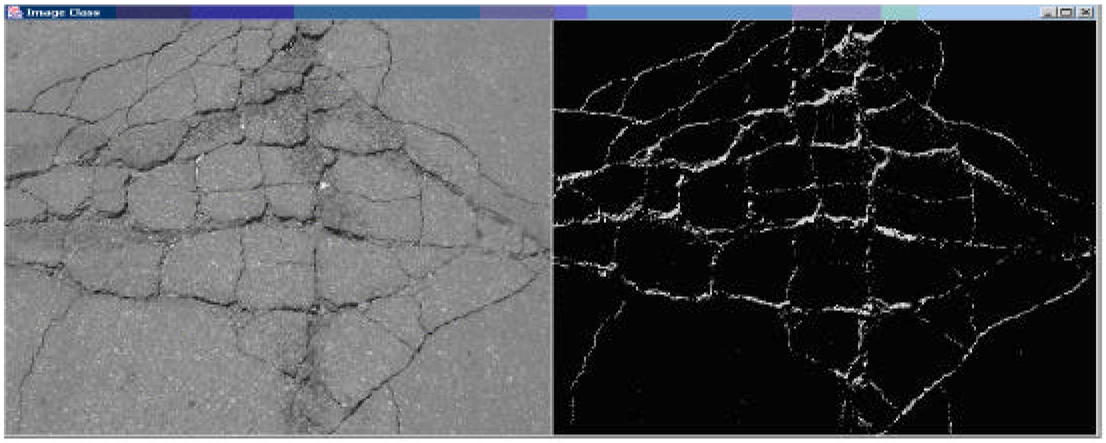


Figure 4.4 Thresholding.

4.2.2. Noise Filtering

The goal of this phase was to perform filtering on the binary images produced in the previous process to eliminate noise as much as possible. We compared several image enhancement techniques, smoothing, low-pass filtering, sharpening, etc. and the best result in the preliminary experiments was using the Median filter [9] after thresholding. Figure 4.5 illustrates a sample of image thresholding, and Figure 4.6 shows the result of applying the Median filtering to the same image after thresholding[1]. We applied this filter to a large number of images representing all four types of cracks, and we found it satisfactory. A sample of each crack is processed by the median filter and shown in Figure 7. The Median filter is described in appendix B-1.

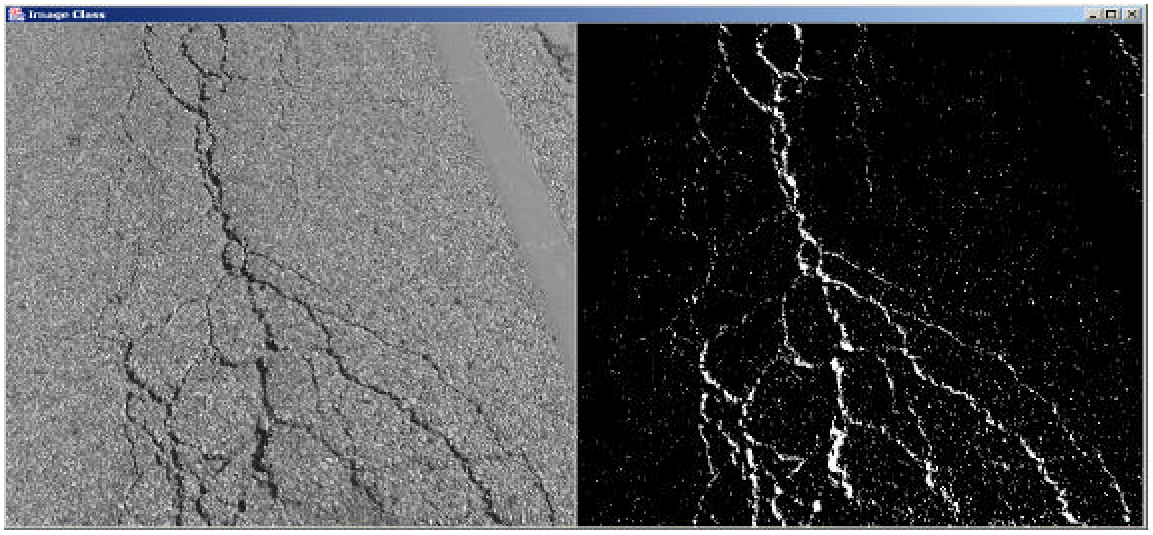


Figure 4.5 Noise after thresholding

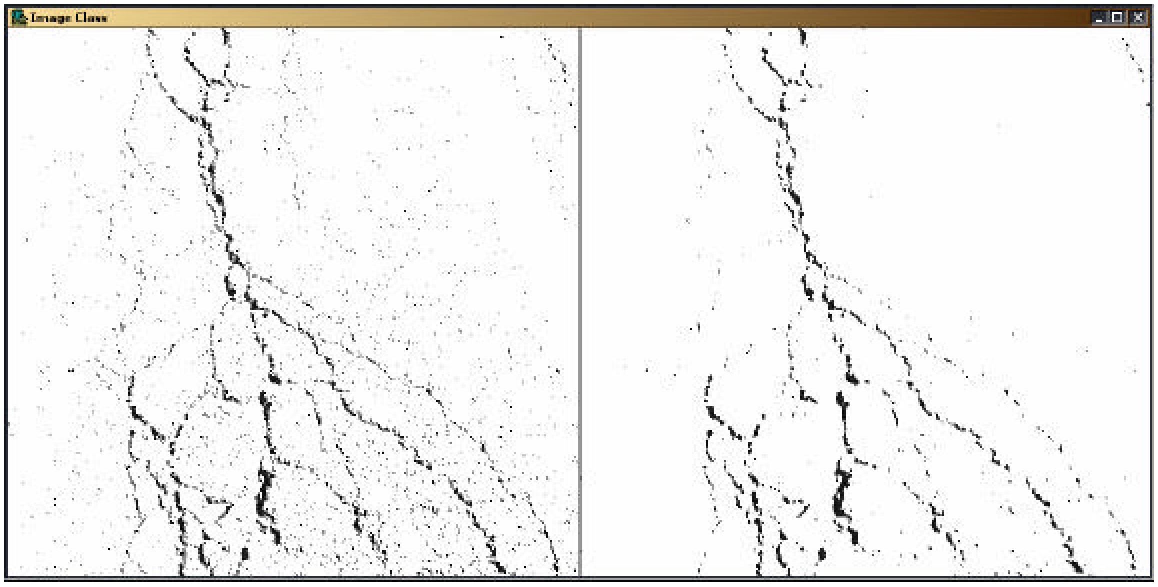


Figure 4.6 Median filtering after thresholding in Figure 4.5

[1] All images other than images of unprocessed pavement have inverted grayscale for printing.

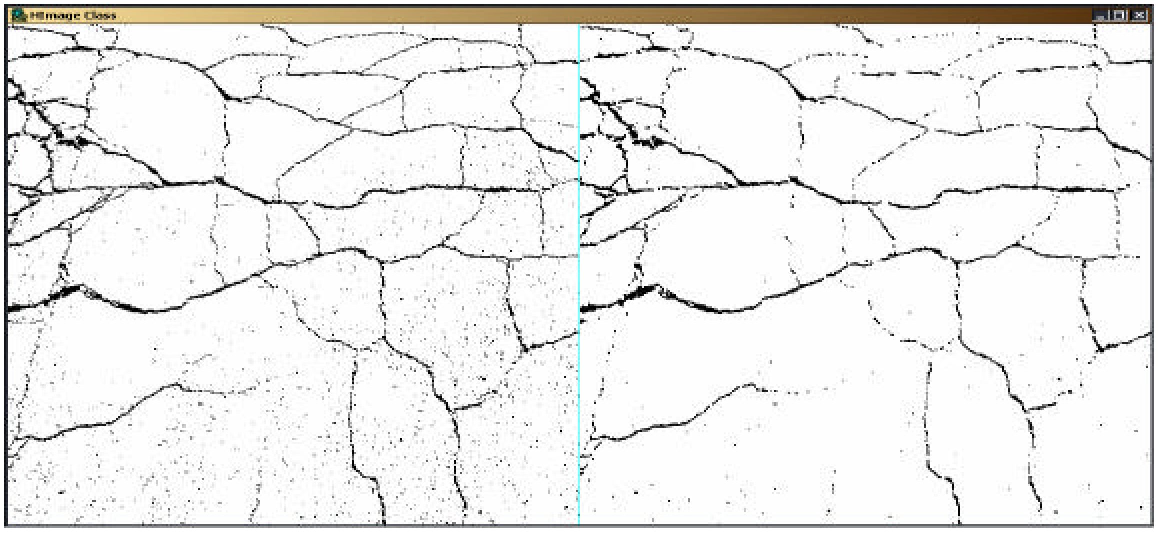


Figure 4.7-a Median Filtering (Alligator)

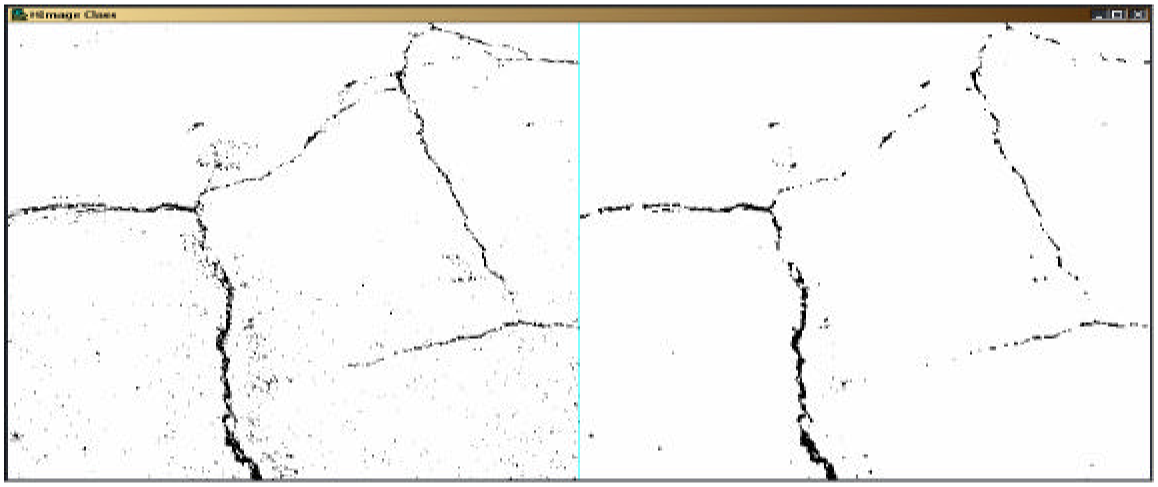


Figure 4.7-b Median Filtering (Block)

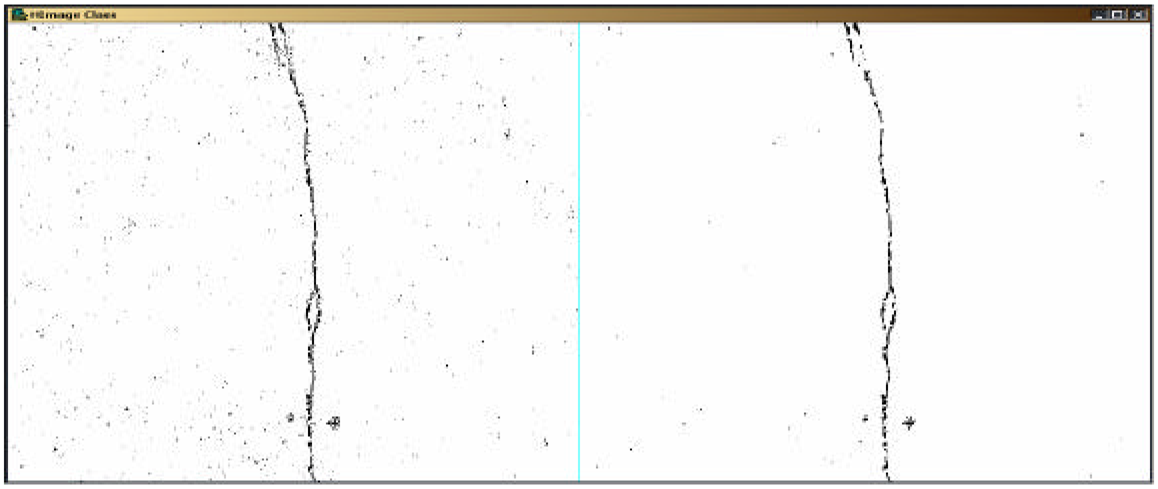


Figure 4.7-c Median Filtering (Longitudinal)

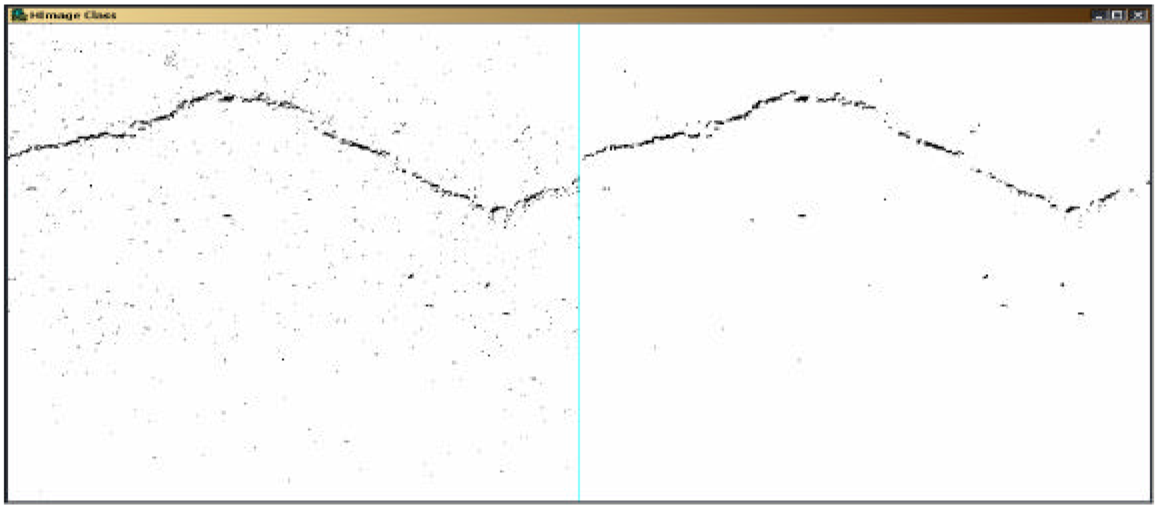


Figure 4.7-d Median Filtering (Transverse)

4.3. Feature Extraction and Image Representation

For image representation, two approaches were considered: image tiling combined with local linear regression and crack hits projection (The projection method), and the Hough transform. The algorithm of the Hough transform is illustrated in appendix B-2.

4.3.1 The Projection Method

This technique quantizes the thresholded image space into blocks, and then performs local linear regression per block to approximate the crack line segments and eliminate noise. The algorithm is summarized as following:

1. The binary image is divided into tiles. Each tile contains 400 pixels in a 20-pixel by 20-pixel square.
2. For each tile perform linear regression on the entire points (pixels) in the space of this particular tile.

3. If the correlation factor r in Equation (5) resulting from the regression operation is greater than a predefined threshold and number of points in the tile exceeds a predefined threshold (10), then we mark the line in the resulting image using the regression equation described in Equation(2), and we discard every thing else in that tile space. A minimum number of points per tile were considered here to eliminate the cases where few noise points could be the only existing points in a tile satisfying the correlation factor threshold.
4. This procedure resulted in pure lines without any single point of noise. Any block that satisfy the criteria above, is considered as a hit of crack object.

$$y = a + bx \tag{2}$$

$$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2} \tag{3}$$

$$a = \frac{\sum y - b\sum x}{n} \tag{4}$$

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{n\sum x^2 - (\sum x)^2}\sqrt{n\sum y^2 - (\sum y)^2}} \tag{5}$$

We did some preliminary experiments with this technique and a sample result is illustrated in Figure 4.8. As a result of this technique, we converted the crack objects into a well defined space of line segments, which were used for feature extraction in the next step.

For this technique we had three parameters that needed to be optimized through experimenting with different types of crack images, since they are important for the effectiveness of the technique. The three parameters were found empirically to be:

- r : the correlation factor; was found best at 0.3 for our images
- Minimum number of points (pixels) per tile; was found best at 10 pixels.
- The size of the tile; was found best at 20x20pixels

The lines are marked on the output images only to visualize the output of this technique. The actual output for each tile was a single point (one pixel/hit) representing a crack object hit in that tile. A sample of this result is illustrated in Figure 4.9, using the same image in the previous step shown in Figure 4.8.

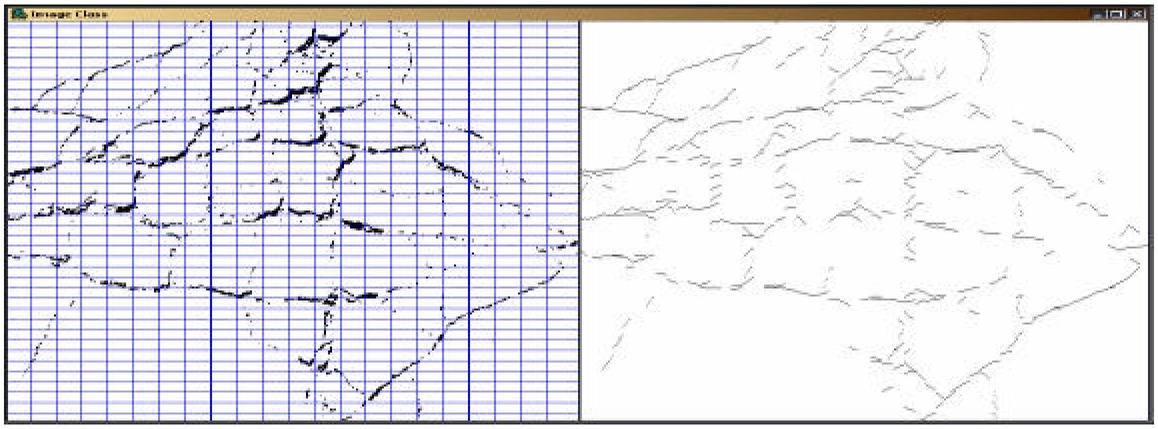


Figure 4.8 Tiling & local regression

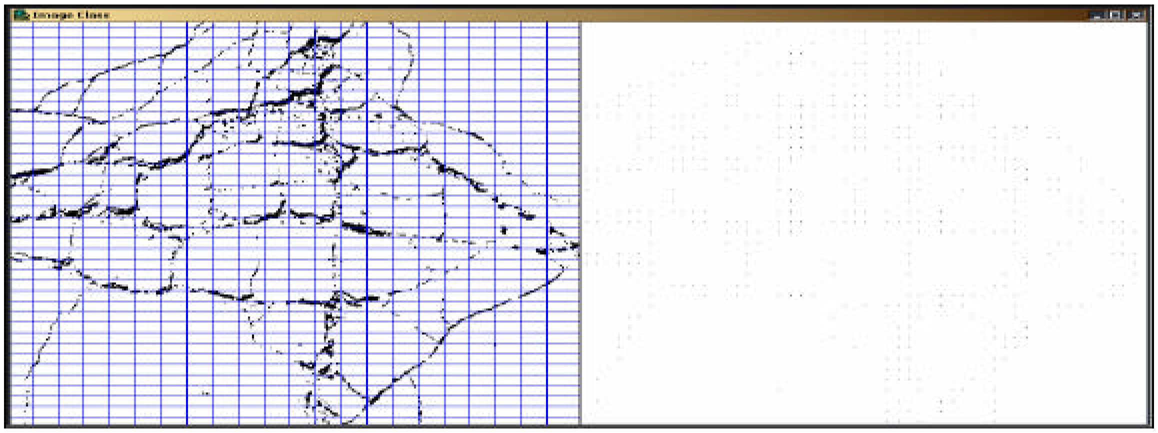
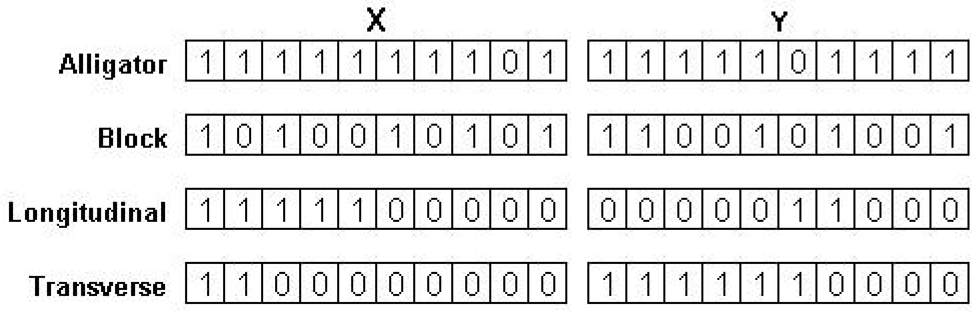


Figure 4.9 Crack hits after local linear regression

4. The final step was image representation, by projecting the crack hits on the X and Y axis. In this operation we considered only the endpoints of the crack objects represented by hit points in the resulting image. The result of the projection was two binary vectors X and Y as shown in Figure 4.10. Combining the two vectors in a single vector to represent the crack pattern produced the final feature vector, which defines the characteristics of the crack patterns as following:

- Alligator cracks: have projection points on both X and Y axis with higher frequency than block cracks.
- Block cracks: have projection points on both X and Y axis.
- Longitudinal cracks: have projection points mainly on the X-axis
- Transverse cracks: have projection points mainly on the Y-axis.



	X										Y									
Alligator	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1
Block	1	0	1	0	0	1	0	1	0	1	1	1	0	0	1	0	1	0	0	1
Longitudinal	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
Transverse	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0

Figure 4.10 Possible projection vectors of the four crack patterns

The theoretical vectors in Figure 4.10 when converted into two dimensional input vectors, they become as follows:

Alligator	[09:09]
Block	[05:05]
Longitudinal	[05:02]
Transverse	[02:05]

An illustration of actual projection is given in Figure 4.11. These sub-vectors [X:Y] were reduced into two components as follows:

- Total number of 1-components in the sub-vector X (X-component).
- Total number of 1-components in the sub-vector Y (Y-component).

The outputs for the four different crack classes, shown in Figure 4.11 are:

Alligator	[47:30]
Block	[16:17]
Longitudinal	[04:00]
Transverse	[00:04]

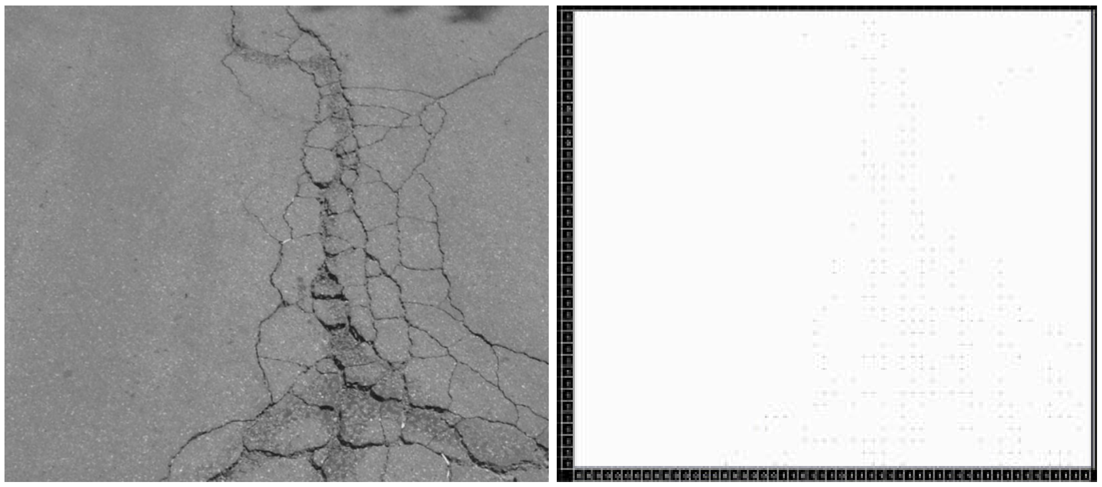


Figure 4.11-a The projection method (Alligator crack)

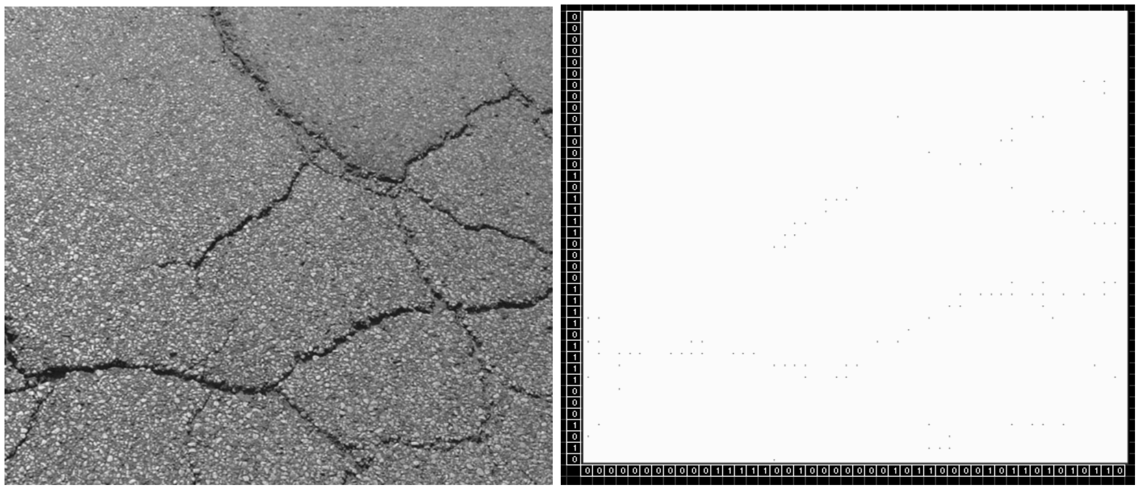


Figure 4.11-c The projection method (Block cracks)

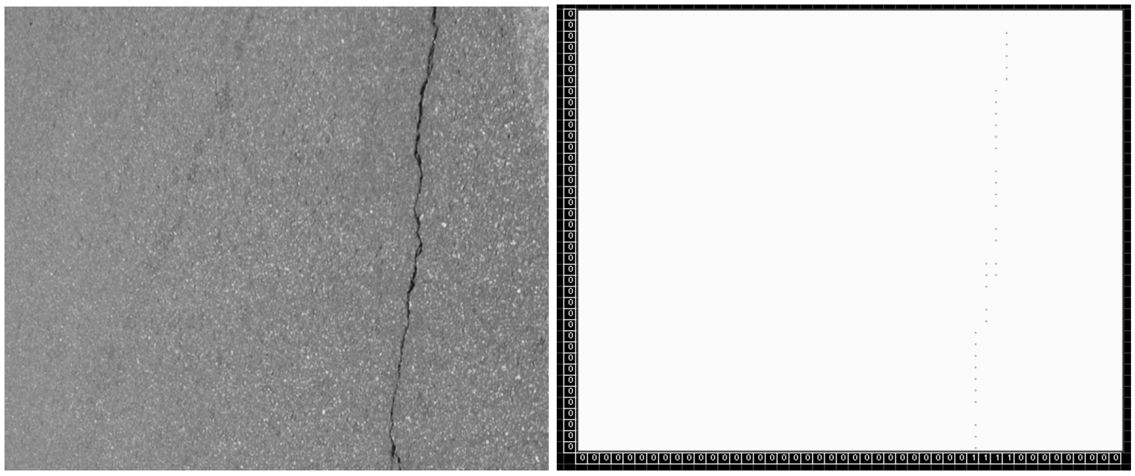


Figure 4.11-c The projection method (Longitudinal cracks)

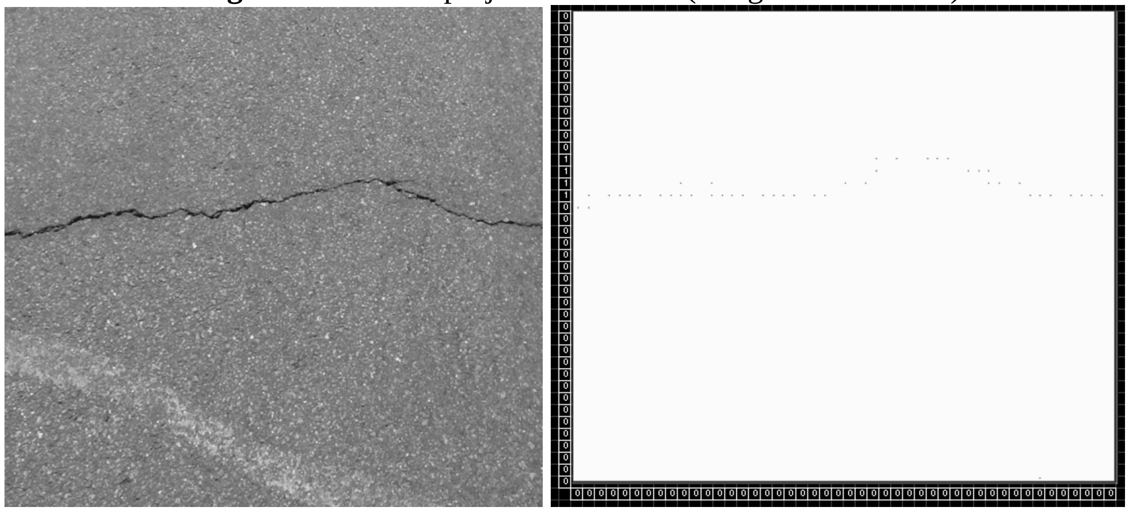


Figure 4.11-d The projection method (Transverse cracks)

4.3.1. Hough Transform

We applied the Hough transform on a subset of the images after thresholding. Our objective in applying Hough transform is to reduce the images to a representative form, from which we can construct the feature vectors that we believe should characterize the crack patterns. Figure 4.12 shows a typical result of the Hough transform applied to a sample image after thresholding and median filtering, with the following parameters:

$$\Delta J = 1,\ \Delta r = 2,\ \ Houghthreshold = 20 .$$

The parameters for the Hough transform were recommended in [9]. We found the Hough transform inadequate for our purpose, specifically; our data had the following characteristics:

- The crack patterns in general do not have well defined lines that can be detected by the Hough transform.
- The remaining noise after the median filtering adversely affected the output image.

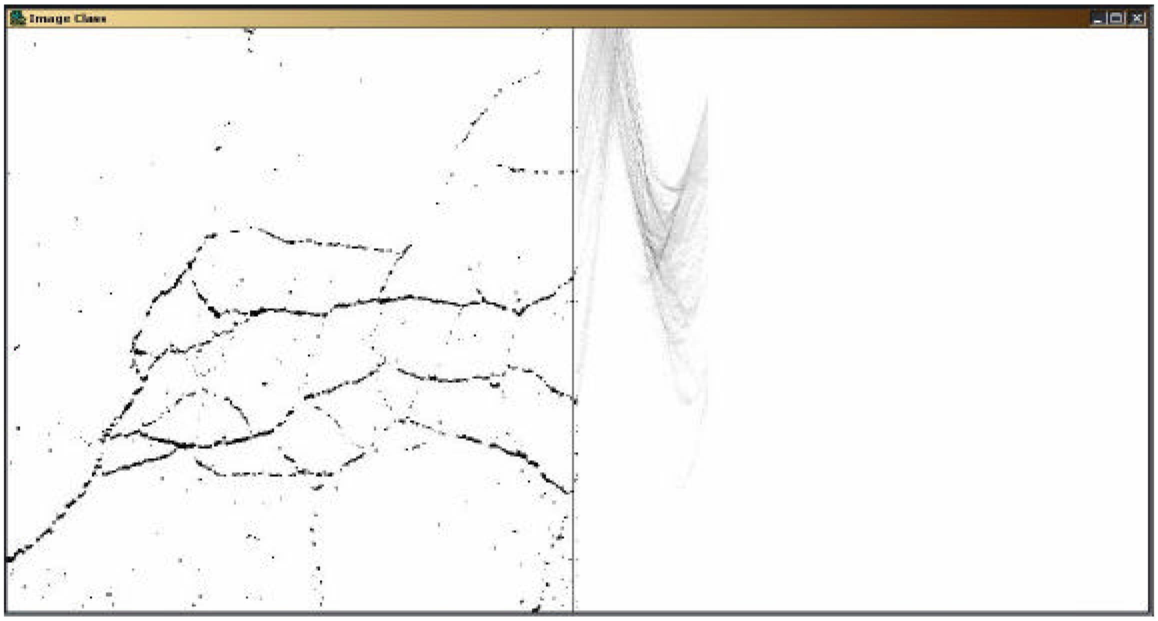


Figure 4.12 Hough transform after thresholding & median filtering

After this trial we considered the effect of applying the Hough transform after tiling and generating the hit-patterns, because we believed that it could solve the previous two problems described above. Since in tiling and using linear regression we are approximating the entire pattern by using the two conditions, minimum correlation factor and minimum number of points per tile, we expected to have better results if we applied the Hough transform after this operation. Results are shown in Figure 4.13 for each crack type.

The feature vectors using Hough transform were constructed as follows:

The parameters for Hough transform were determined by experimenting with all types of cracks using the recommended ones in [9]. For Hough-threshold we used the following empirical formula:

Hough-threshold = 100/((100/ number of hits in the crack pattern)+1) (19)

The threshold was chosen to be dynamically changing based on the number of hits in the pattern because the different patterns produce different numbers of hits. A base hit of 100 was chosen after the preliminary trials with all types of cracks. We needed a base threshold because sometimes the (longitudinal and transverse) cracks do not produce enough hits that can be used as a threshold, therefore this base will prevent the week Hough-lines from appearing in the resulting Hough-image.

The features were represented by 2D vectors: $\{\overline{J}$,N$\}$ where:

$\overline{J}$ = The average angle of the entire quantized Hough space.

N = Total number of pattern-hits produced after tiling and linear regression.

The results of this approach were:

- For a-type cracks: $\{45^o \pm \Delta$, N = larger number than b-type crack$\}$

- For b-type cracks: $\{45^o \pm \Delta$, N = smaller number than a-type crack}
- For t-type cracks: $\{0^o + \Delta$, N = much smaller number than a-type crack}
- For l-type cracks: $\{90^o - \Delta$, N = much smaller number than a-type crack}

Δ was observed in (a & b) types to range from -10^o to $+10^o$. In t-type from 0^o to $+20^o$. In l-type from 0^o to -30^o. All observed values of delta conform to the standards we refereed to in [7] and [5] .

The following vectors are the actual feature vectors extracted from the images illustrated in Figure 4.13 respectively.

Figure 4.13-a →	{43, 316}
Figure 4.13-b →	{42, 140}
Figure 4.13-c →	{0 , 44}
Figure 4.13-d →	{86, 43}

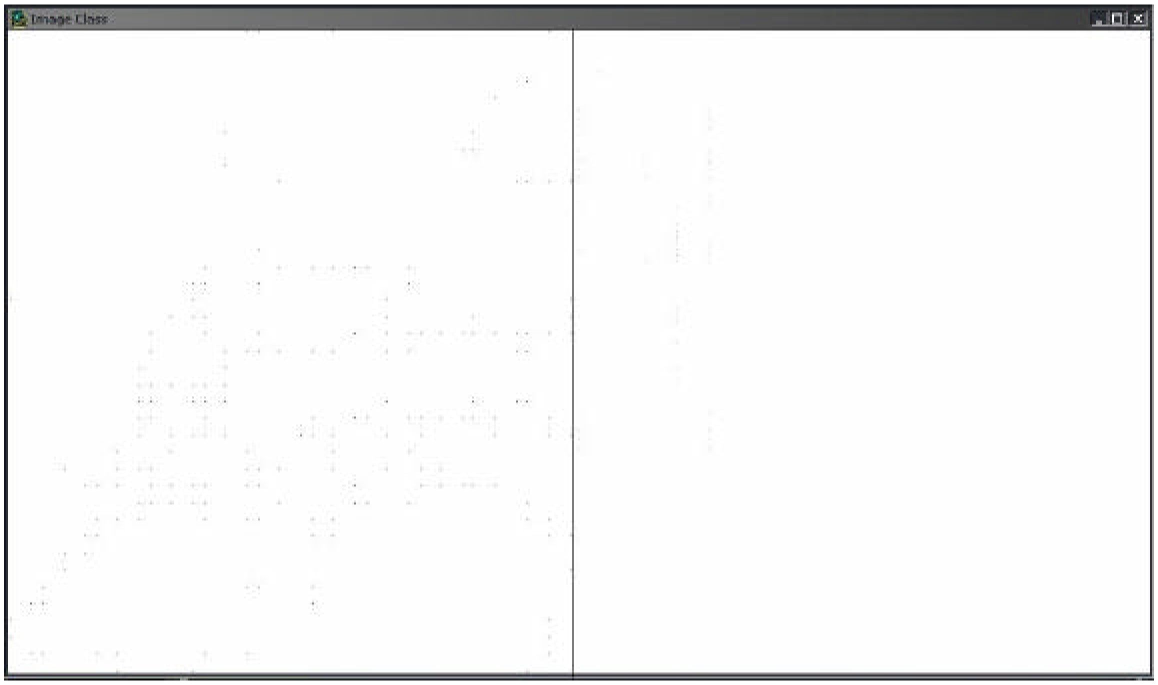


Figure 4.13-a Hough transform after hit-patterns (Alligator crack)

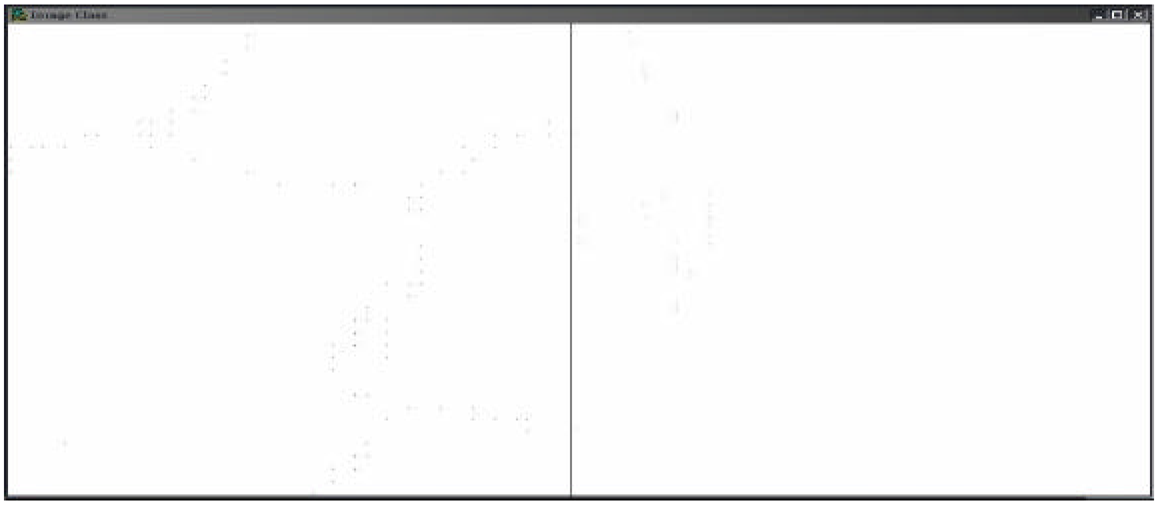


Figure 4.13-b Hough transform after hit-patterns (Block crack)

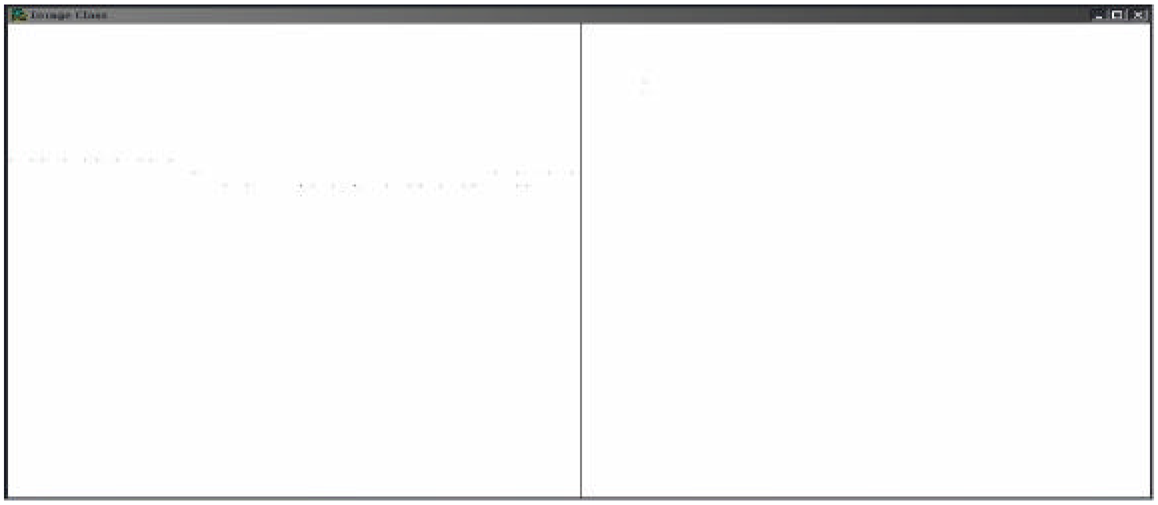


Figure 4.13-c Hough transform after hit-patterns (Longitudinal crack)

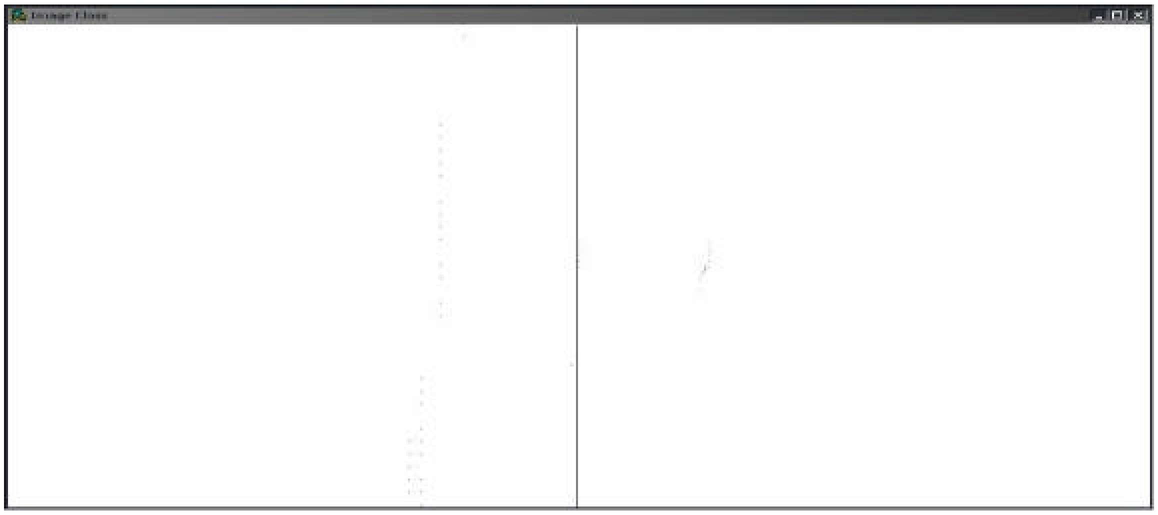


Figure 4.13-d Hough transform after hit-patterns (Transverse crack)

4.3.3. A Special Case

Although (longitudinal or transverse) cracks are found in most cases at angles 0-30 degrees from the center line of the pavement (vertical for longitudinal, horizontal for transverse) [5], but in rare cases they can be found at a common angle of 45^o. From observations while collecting crack images along with expert consulting, it was found to be a rare case.

The engineering stress analysis for this case can be found in any typical reference that deals with engineering materials and stress analysis [21]. We can very briefly describe it as follows:

Engineering materials are classified based on their behavior when subjected to a loading (stress, cloud be mechanical or thermal) system into several types. These types include: ductile materials like metals in general, and brittle materials like glass, concrete, asphalt. The types of loading systems can be described as:

- Tension: the material is subjected to a stretching stress system.
- Compression: the material is subjected to a compressing stress system.
- Torsion: the material is subjected to a twisting stress system.

In the first two cases (tension & compression) the brittle materials fail with cracks that forms $90^o \pm \Delta^0$ angles with the principal axis (the center line of the object in the direction of loading, whereas in the third one (torsion) the brittle materials fail with cracks that forms $45^o \pm \Delta^0$ angles. In order for this system of loading (thermal or mechanical) to occur rare conditions need to exist. So we decided to try to consider this rare case hoping to improve the overall performance of our proposed system as follows:

The number of detected crack objects (N) can be considered in the feature vector to handle the case of a crack pattern of the type (longitudinal or transverse) at 45^o. Because this type of crack pattern will be projected evenly on the X-axis and the Y-axis, so the classifier system will be confused and it might be classified as a block or alligator crack. By adding the (N) feature, we believe the problem can be solved, since a crack (longitudinal or transverse) at 45^o should have much less crack objects (hits) than block or alligator type cracks.

4.3.4. Imaging Constraints

- In the vast majority of the images, the crack objects were darker than the pavement background, so crack segmentation using thresholding gave satisfactory results that satisfied the needed output from the original images.
- The original images are inherently noisy, so thresholding alone was not enough to produce reliable binary images for the next phase (image vectorizing). So the median filter was used to filter out the scattered noise. The median filter enhanced the continuity of the crack objects and filtered out the random noise in the background. One drawback was noticed in the filtration operation: the weak crack objects which have low continuity were affected by the median filter, where some small segments of the crack objects were filtered out. But the overall performance of this operation was satisfying since the loss in segments of cracks objects is insignificant and it is unrealistic to expect any filtering method to produce perfect results. This is illustrated by Figure 4.14.

- Some noise was strong enough to remain untouched by the median filter. Local linear regression, was effective to eliminate the majority of the remaining noise from the image, where the correlation factor and the minimum number of pixels were both combined to form a threshold criteria for a tile pattern to be accepted as a crack object.
- Our method doesn't take into consideration the spatial variations of the crack patterns. We are relying only on reducing the dimensionality of the projected vectors into two dimensional vectors, which we think was sufficient to characterize crack patterns. We believe spatial variations are less important since we are not distinguishing (at this stage) between a crack occurring at the edge of the pavement or in the middle or any where else. So the extracted features here are two: number of projected hits on the X-axis, and Number of projected hits on the Y-axis. Therefore, since we are using a fixed frame size, then the two dimensional projected vector [X:Y] should have the needed information to be recognized by the classifier system.

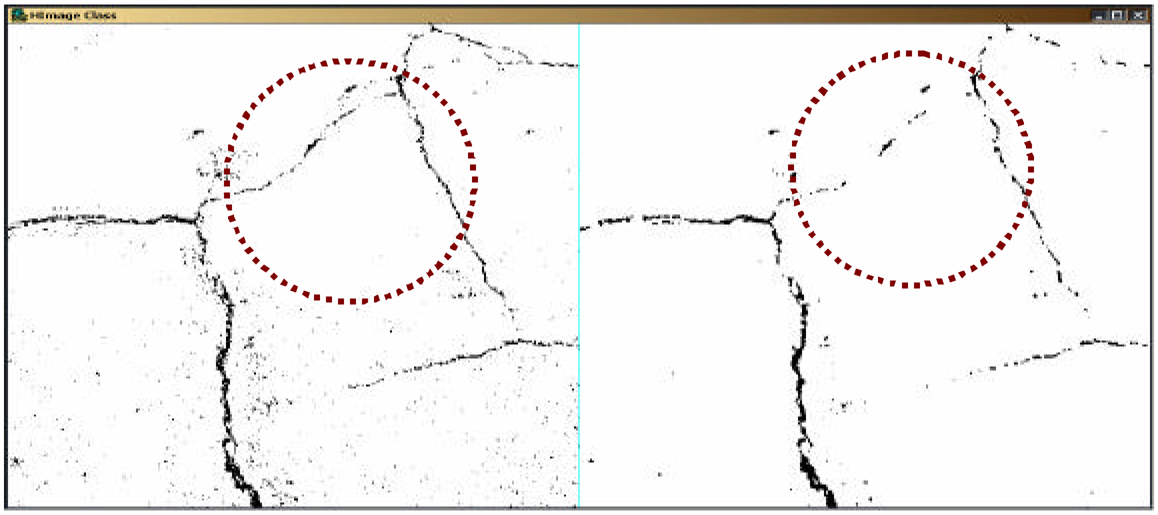


Figure 3.14 Lost weak crack objects

- The special case of cracks at 45^o presented in image representation section could be solved by the Hough-transform based feature vectors since the angle is explicitly presented in the vectors, so we believe that the classifier system can easily learn to recognize this kind of pattern. But for this study we did not consider addressing this specific case, since it is not an issue in pavement crack classification. The crack patterns close to 45^o are either classified as transverse or longitudinal.
- Although there were several other ways to consider for image representation, such as leaving the hit patterns the same way they are projected without compressing them into two numbers representing the count on each vector, and using them directly to train and test our system. Dealing with the pavement as a texture and extracting some features to represent the crack patterns. We elected to keep our system computational less expensive, since the classification can require real time processing.

From the above discussion we believe that the chosen methods in segmentation, feature extraction, and image representation performed in an integrated way created a more robust system.

4.4. Crack Classification

For this stage we used the three approaches MLP, GA, and SOM. We designed a systematic training and testing procedure as following:

1. An electronic copy of collected images were industry expert [3], to provide ground truth for benchmarking the automatic classification. Each image file

was assigned a systematic name to reflect the class and the serial number of the image: a### for alligator-type images, b### for block-type images, l### for longitudinal-type images, and t### for transverse-type images.

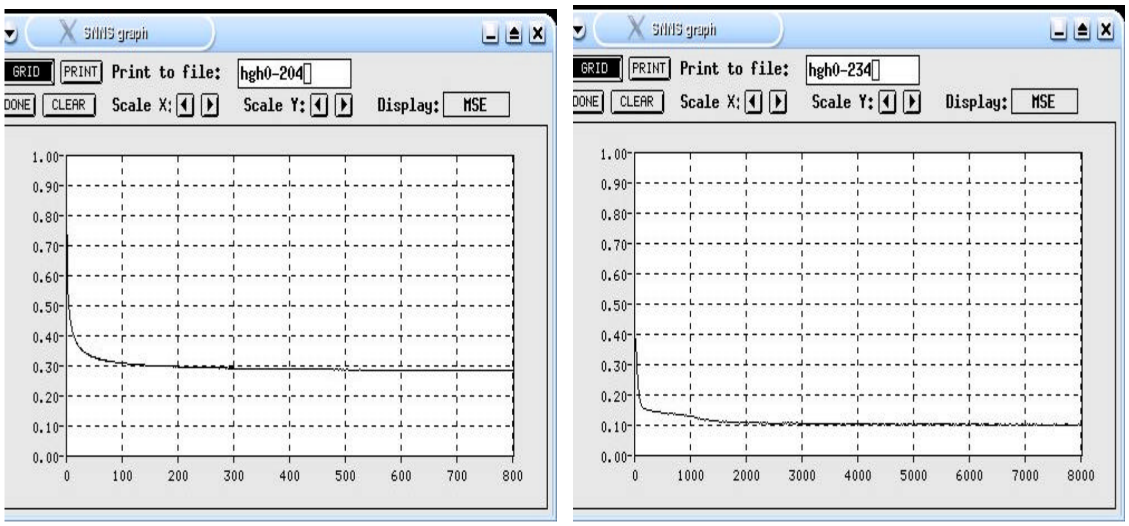
2. Images then were preprocessed by the preprocessing module. This stage produced two master files of vectors: one for projection technique, and the other one for the Hough transform.
3. We implemented the cross validation training method [22] as follows. For each master file, five different files for training (400 vectors each) and five different files (100 vectors each) for testing were generated. The first set of training/testing files were constructed by taking the 1st 100 vectors in the master file as the 1st test file, and the rest of the master file was taken as the 1st training file. The 2nd 100 vectors were then taken as the 2nd testing vectors, and the rest of the master file was taken as the 2nd training file, and so on.

4.4.1. Multilayer Perceptron (MLP)

The model we chose to implement is the MLP (Multilayer Perceptron), because it is said to be the most widely used in pattern classification problems which is a functional capability of MLPs [14]. MLP is illustrated in appendix B-3.

To Apply MLP to our problem we used the Stuttgart Neural Network Simulator (SNNS-4.2) [23]. As described in the feature extraction section, the input space for all classifier systems is a set of 2D vectors. For MLP classifier system for the input patterns we needed 2-node input layer, a 4-node output layer representing the four different cracks, and for the hidden layer we systematically experimented both input spaces (Hough &

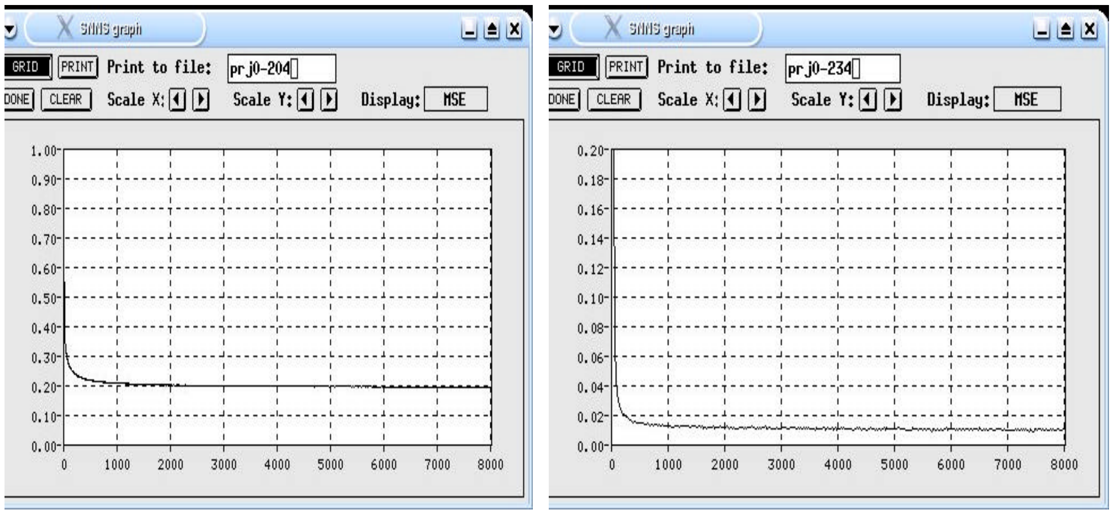
Projection) to find the optimal number of hidden nodes. We present four charts (Figures 4.15), two for each input space. The complete set of charts for both experiments is given in Appendix C-1. As a result of these experiments, we found that three hidden nodes formed an effective classifier. Figures 4.15a, b plot the Mean Square Err (MSE, Y-axis). Vs. number of training cycles (X-axis).



Print to file: hgh0-204 Display: MSE

Print to file: hgh0-234 Display: MSE

Figure 4.15-a Hough data experiments to find optimal number of hidden nodes



Print to file: prj0-204 Display: MSE

Print to file: prj0-234 Display: MSE

Figure 4.15-b Projection data experiments to find optimal number of hidden nodes

After finding an effective network as described above, we followed the following procedure:

1. Each of the 10 training files (5 for Hough and 5 for Projection) was used to train the MLP (2 input nodes:3 hidden nodes:4 output nodes) using parameters recommended by [23], the parameters were:
   - Training algorithm: standard back propagation.
   - Learning rate: 0.2
   - Layers connections: fully connected.
   - Weights initialization: randomly initialized.
   - Patterns input mode: shuffle.
   - Pattern vectors were normalized.

   The trained networks were saved to be used later for testing.

2. Each of the 10 test files was tested against the corresponding trained network. The output result files were stored for further analysis and presentation.

4.4.2 Genetic Algorithms (GA)

Given some problem, a genetic algorithm (GA) seeks to find solutions to it, as follows. Start with an initial generation of candidate solutions to the problem, and then subject this generation to evolutionary forces such as survival of the fittest, mating followed by crossover, and mutation. The hope is that, over time, better and better candidate solutions will surface in the generation. Use the best solution as the working solution The standard GA algorithm is described in appendix B-4.

To Apply the GA to our classification problem we used the software GAD [25]. We applied the first approach of GA as described in the algorithm appendix B-4, which is applying the GA as direct classifier. In GA problems, solution representation and encoding is important and can significantly affect the output. So after studying our image representation method, and the resulted vectors, we had an intuition of a good candidate method for solution representation explained as following:

The implementation Projection Method in GA

In the Projection Method, the target input images were characterized using a two dimensional vector [X:Y] as described in section 4.3.2. The input vectors were defined as a two dimensional space, which was represented by using a 2D matrix, which was expected to have the following properties:

- Has the same dimensions of tiled images (height=600/20, width=800/20); where 600/800 is the height/width of the image respectively, and 15 is the parameter of the square tile.
- Has four distinct regions, representing the four different crack types. To define the regions, we made the following approximations and assumptions:
    - Since the alligator cracks, were expected to have high number of projected hits on both X and Y axis, we represented their region by a quarter of a circle centered at the right bottom corner of the classifier matrix. The radius of this circle is variable (rA).
    - A similar reasoning for longitudinal cracks was made. We represented this region by a triangle along the width of the classifier matrix with two degrees of freedom: the height (lH), and the width (lW).

- ▫ The transverse cracks were represented by a triangle along the height of the classifier of the classifier matrix with two degrees of freedom: the height (tH), and the width (tW).
- ▫ The remaining region is assigned to block cracks with three dgrees of freedom: an offset from the region of alligator cracks (bA), an offset from the region of longitudinal cracks (bL), and an offset from the region of transverse cracks (bT).

- ▪ Each crack-type region was filled with the proper crack-type label: label (*1*) for alligator cracks, label (*2*) for block cracks, label (*3*) for longitudinal cracks, and label (*4*) for transverse cracks.
- ▪ The gap among regions was left for the nearest-neighbor algorithm to resolve any possible hits in that null region. A typical classifier matrix is illustrated in Figure 4.16.

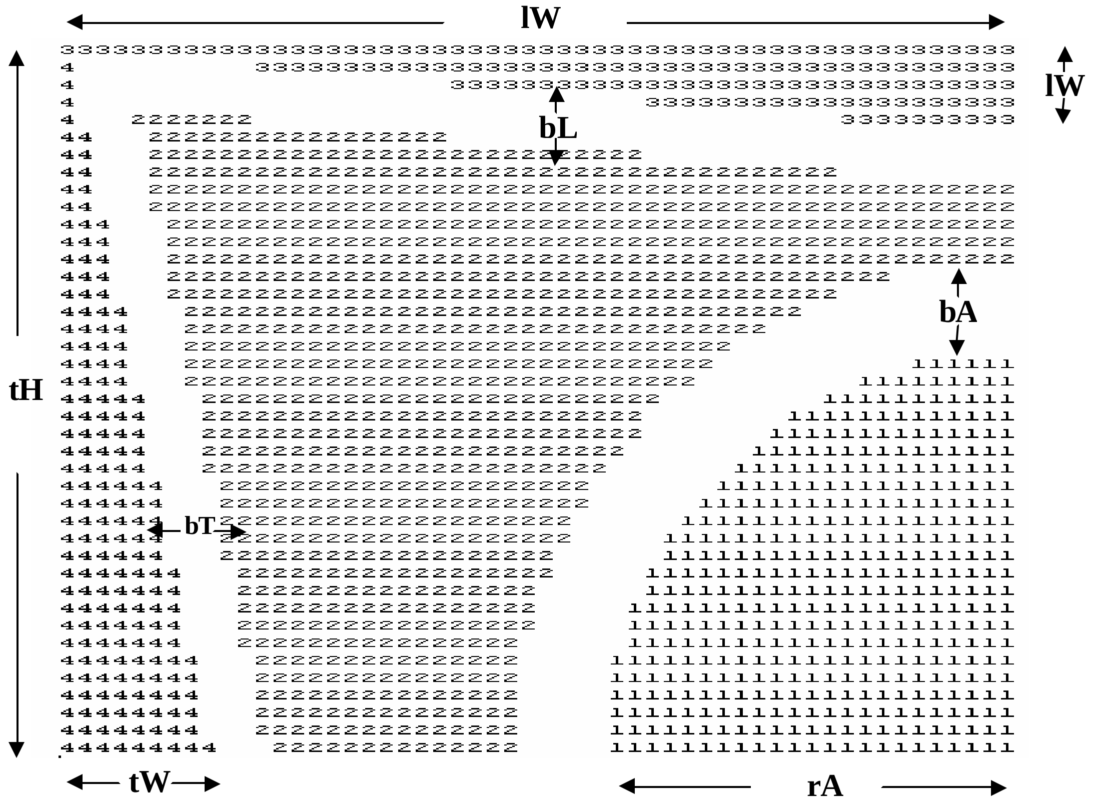


Figure 4.16 Typical GA classifier matrix

Then after this classifier matrix was evolved, each preprocessed input image was classified. The class of the image was determined by using the [X:Y] vector of the image as coordinates of the classifier matrix. At first the given coordinates [X:Y] was probed for a none-null hit. If there was one, then the label is return. If the probed cell was a null, then the nearest-neighbor algorithm (Figure 4.21) was followed.

Mapping GA operators to our work involved the implementation of the following operators: chromosome encoding, crossover, mutation, fitness, and selection. The following part of this section describes the mapping of these operators.

Chromosomes encoding: the chromosome (a string of bits) encoded the set of parameters of the classifier matrix which represents a possible solution. The number of bits needed per parameter was: rA = 5 bits, bA = bL = bT = 3 bits (assuming similar offsets from all directions), lH = tW = 3 bits. Total number of bits needed to encode the parameters of the classifier matrix was = 20 bit. An example of a typical chromosome is shown in Figure 4.17.

rA					**bA**			**bL**			**bT**			**lH**			**tW**		
1	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**

Figure 4.17 Typical chromosome for Projection method

Choosing rA to be larger than the offsets was based on the assumption that regions in the evolved matrices should be larger than the null gaps among these regions.

The crossover operator: crossover operates on selected genes from parent chromosomes and creates new offspring. The simplest way to do that is to choose randomly some crossover point and copy everything before this point from the first parent and then copy everything after the crossover point from the other parent. This is done to the first child and the opposite to the second child. After the GA was done evolving the encoded string, the 2D equivalent matrix was constructed. The recommended crossover rate is 80% - 95% [18]. An example of crossover is illustrated in Figure 4.18.

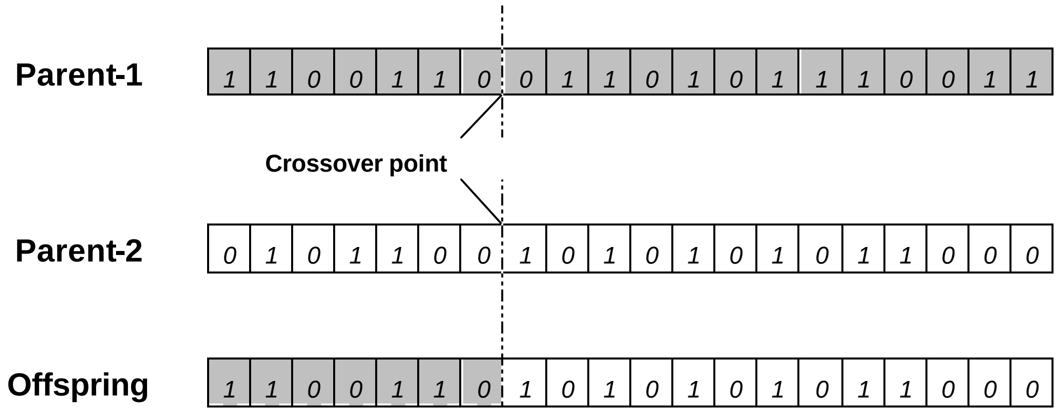


Figure 4.18 Crossover operation

Mutation: the purpose of mutation is to prevent GA from falling into local minima and to explore new regions of the search space. Mutation is done by randomly choosing one or more genes in the offspring resulting from the crossover operation, and alter their value. In our case we are going to change the chosen gene by replacing the label residing on that gene by one of the other three labels chosen randomly. Mutation is illustrated in Figure 4.19. The recommended mutation rate is 1% - 5% [25].

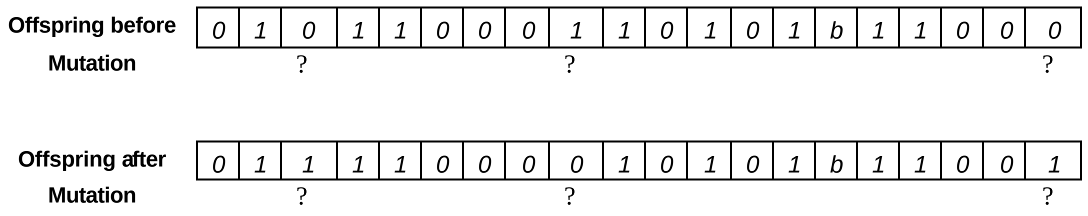

Offspring before Mutation
0 1 0 1 1 0 0 0 1 1 0 1 0 1 b 1 1 0 0 0
? ? ?
Offspring after Mutation
0 1 1 1 1 0 0 0 0 1 0 1 0 1 b 1 1 0 0 1
? ? ?

Figure 4.19 Mutation operation

Fitness: each individual chromosome inherits from its parents the characteristics that suggest a solution for the problem. To evaluate the fitness for chromosomes we need a fitness function that is capable of assigning a value proportional to how well is a chromosome in solving the problem. Our fitness was evaluated base on how well is the current chromosome can classify the input space. This was quantified by assigning the percentage of correctly classified images = (number of correctly classified images/ total number of images).

Selection: in order to produce the next generation, parents have to be selected for cross-over operation to produce the new offspring. Selection is based on the fitness of the chromosomes, the bigger the fitness is, the bigger the chance to be selected. The selection model we are going to implement is the roulette wheel selection and works as follows:

We can imagine a roulette wheel with a section associated with each chromosome in the generation. The size of the section in the roulette wheel is proportional to the value of the fitness function of the chromosome - the bigger the value is, the larger the section is as illustrated in Figure 4.20.

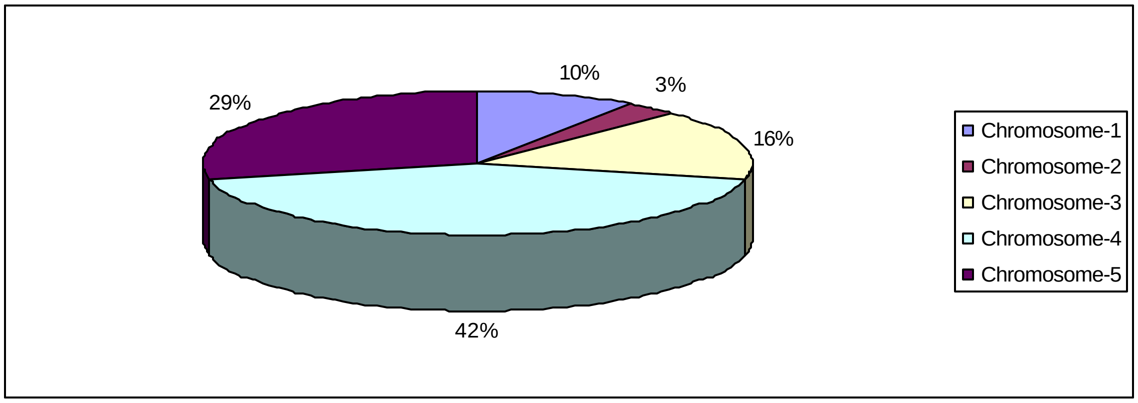


Figure 4.20 Roulette wheel selection model

A marble is thrown in the roulette wheel and the chromosome where it stops is selected. Clearly, the chromosomes with bigger fitness value will be selected more times.

This process can be described by the following algorithm.

1. *Sum:* Calculate the sum of all chromosome finesses in generation - sum *S*. (performed only once for each generation)
2. *Select:* Generate random number *r* in the interval *(0, S).*
3. *Loop:* Go through the generation and sum the fitnesses from *0* - sum *s*. When the sum *s* is greater then *r,* stop and return the chromosome at the current position.

Population size: experience suggested that very big populations usually don’t improve the performance. In general, the recommended size is proportional to the size of the encoded chromosomes. For our application we used a population size of 50 and a number of generations of 100 as recommended by [25].

GA classification model: as described earlier, the design of the parameters of the matrix allowed a null region along the borders of the regions, since the are no clear cuts between adjacent regions. The following algorithm was implemented for the GA classification model:

1. Each time a new image vector is presented to the system to be classified; its coordinates are used to probe the classifier matrix. If a none-null label was found at that location the, the label is returned as the type of the target image. Otherwise the nearest-neighbor algorithm is used to find the closest match.
2. Nearest-neighbor Algorithm
    - The algorithm starts by scanning all neighbor cells at the radius of one unit length (r1 in Figure 4.21). The labels stored in these cells are checked, and if a none-null label is found, the votes are accumulated. The votes per label are checked for a winner every iteration, if there is a winner by the

majority voting, then the winner label is returned as the correct class of the target image, otherwise do next step.

- The radius of the circle is incremented by one, and the same procedure is followed in the first step. The algorithm runs for a maximum number of iterations and then returns the last winner. The winning label with the highest votes is taken as the correct classification.

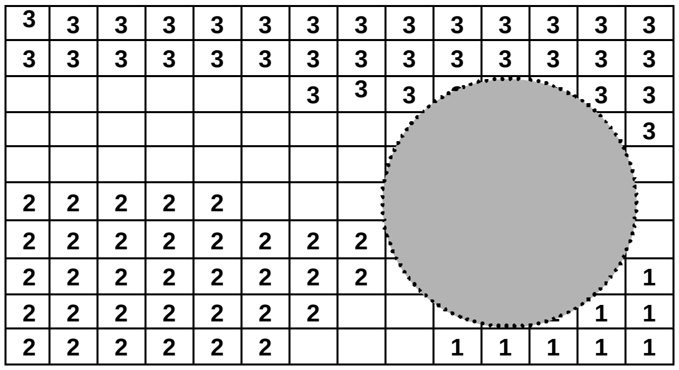


Figure 4.21 The nearest-neighbor algorithm

The implementation of Hough transform method in GA

A similar approach to the one followed in the projection method was applied in the Hough transform, but since we used different image representation technique, there some differences between the two methods, and they were as following:

- The two components of the feature vectors were: [the average angle of the lines detected in the hit pattern, and the total number of hits in the pattern].
- The classifier matrix was represented as shown in Figure 4.22.

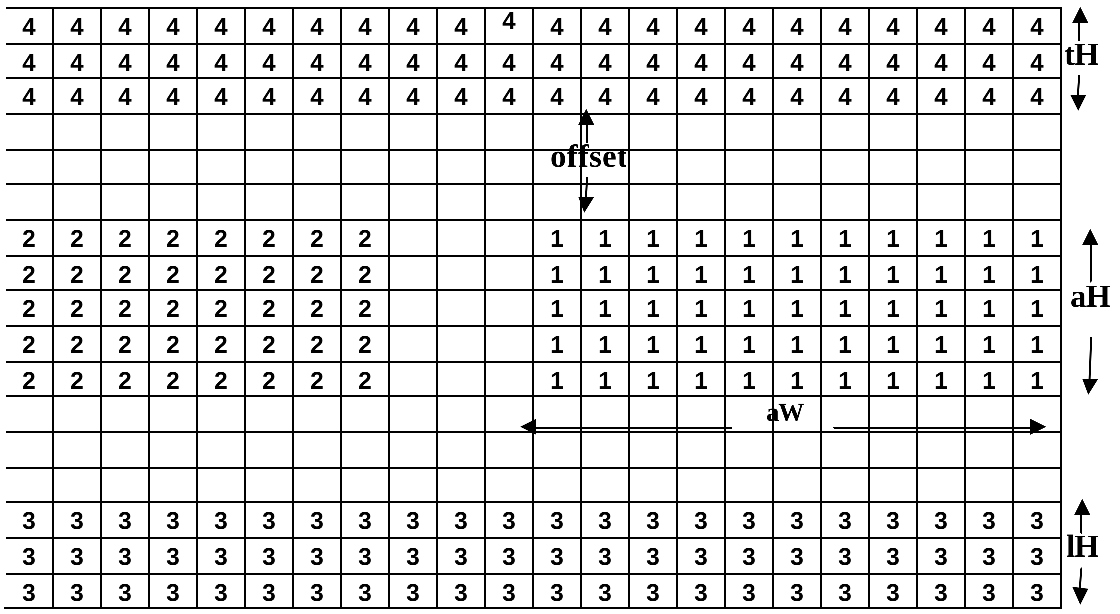


Figure 4.22 The Hough transform classifier matrix

- The dimensions of the matrix were: the height (90 = max possible angle) X the width (2120 = the max possible number of hits).
- The parameters of the classifier matrix were: tH = 4 bits, lH = 4 bits, aW = 8 bits, and offset = 3 bits.

We used the same classification model we used in the projection method. We followed the systematic procedure followed in the MLP approach to evolve the classifier systems for both the Hough and the projection methods as following:

1. Each of the 10 training files (5 for Hough and 5 for Projection) was used to evolve the classifier matrix. The Software (GAD [25]) parameters (consulted the author of GAD [25]) used for training were:
    - Probability of crossover: 1
    - Probability of mutation: 0.005
    - Crossover sites: 1-point
    - Swap probability: 1

- Reproduction form: monotonic
- Population size: 50
- Number of generations: 100

The evolved matrices were saved to be used for later testing.

2. Each of the 10 test files (5 for Hough and 5 for Projection) was tested on the corresponding evolved matrix. The output result files were stored for further analysis and presentation.

4.4.3. Self-Organizing Map (SOM)

We chose to include this approach as representative of a typical unsupervised technique. Supervised learning techniques, such as MLP, require an external verification that directs their learning to produce desired output by example, whereas unsupervised techniques require no external verification, they organize the input space by discovering collective properties of the input data [22]. The algorithm of SOM is discussed in appendix B-5.

To apply SOM to our problem, we used the software of [24]. This method produced a 2D matrix similar to the one produced from the GA approach, since we used the same feature vectors denoted by the two-component vectors: [X:Y]. So we used the same classification model used in the GA Section 4.3.2. The training and testing procedure was as following:

1. As described in the training and testing files selection, we had ten files for training and ten files for testing.
2. The files are properly formatted to comply with the data file SOM format.

3. Each of the ten training files was used to train a map, which was saved for later testing. The parameters (recommended by [24]) used for training all maps were as following:
    - number of trials: 50
    - topology type: hex
    - neighborhood type: Gaussian
    - x-dimension: 30
    - y-dimension: 20
    - training length of first part: 1000
    - training rate of first part: 0.1
    - radius in first part: 30
    - training length of second part: 10000
    - training rate of second part: 0.01
    - radius in second part: 1
4. The produced SOM maps were used to generate their corresponding classifier matrices, and the same concept of classifier model designed for the GA approach, was used.
5. Each of the ten test files was tested using the corresponding trained map, and the resulted files were saved for further analysis and presentation.

To illustrate the concept of the produced SOM maps, we would like to analyze a typical example, and for a complete reference to the produced maps, please refer to appendix C-3. The example is illustrated in Figure 4.23. The map shown in Figure 4.23 was produced by the SOM/Hough method. We can see how the SOM was able to cluster the different types of cracks into four separate regions one per each crack type. The different regions are manually separated by a dotted line, to make it easier to distinguish the borders of the regions. It was interesting to notice that the produced maps by the SOM resembled the maps suggested in the GA approach.

We illustrated some of the misclassified images on the map. The misclassified images are circled by a dotted circle. The SOM color shadings have significance in interpreting the relations among clustered classes. The darkness level between adjacent nodes/regions reflects the distance between vectors represented by these nodes/regions, therefore the lighter the area between them, the similar the nodes/regions. We can see that the map tends to build a boundary around each cluster as shown in Figure 4.23.

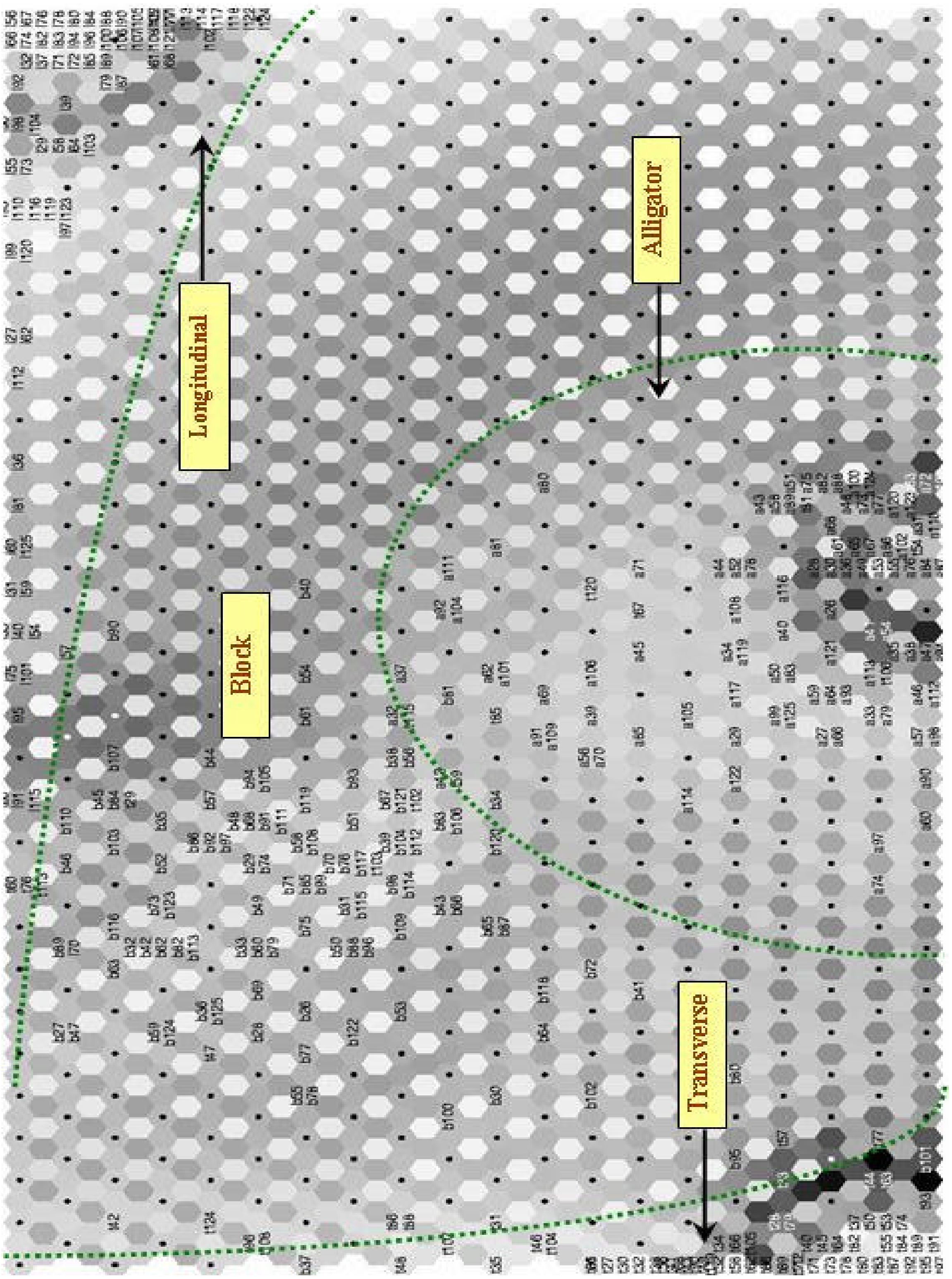


Figure 4.23 Typical sample of maps produced by the SOM

5. RESULTS

The preceding section described the approach we followed to develop our methods to address the problem of pavement crack classification. Three different approaches were investigated: Multilayer Perceptron (MLP), Genetic Algorithms (GA), and Self Organizing Maps (SOM). We used two different image representation techniques: Hough Transform, and Crack Hits Projection. We used both representation techniques with each of the classification approaches. We divided the input images into two subsets: training subset of 400 images and testing subset of 100 images which produced five different (training/test) combinations. This setup introduced a rich experimental situation of 30 different possible combinations to test, and in this section we are going to present them in details as following:

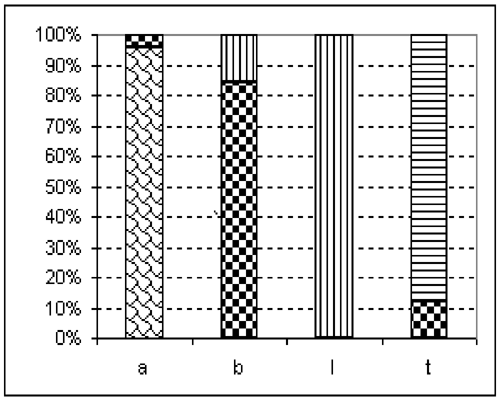
Before presenting results, we would like to explain the standard format we used to tabulate our results for all experiments. As a sample we analyzed Table 5.1. A complete reference to the results is included in Appendix D. The table is a typical format of results of each individual (classifier/image representation/training set/testing set) experiment, which contains the following information:

- MLP_0: is the label of the table which combines the name of the technique (MLP) used to test the given subset of images, and a serial number (_0) to distinguish between the different trials.
- hghTrain0.dat (400)/hghTest.dat (100): are the training and testing files, respectively, used in this trial. As described in the training and testing procedure in section 3.

- For each crack type, the table lists the classification distribution of the 25 test images, and gives the C/T per each crack type. Finally gives the aggregate (average) C/T for the experiment. The column chart, plots the crack type Vs. The C/T per that crack type.

Table 5.1 A sample of MLP results

MLP_0	hghTrain0.dat					400 samples
	hghTest0.dat					100 samples
	C/T	**a**	**b**	**L**	**t**	**Total**
A	0.96	24	1	0	0	25
B	0.84	0	21	4	0	25
L	1	0	0	25	0	25
T	0.88	0	3	0	22	25
Over all	**0.92**					100



▨ Alligator Cracks (**a**) ▦ Block Cracks (**b**) ▥ Longitudinal Cracks (**l**) ▤ Transverse Cracks (**t**)
C/T = Number of correctly classified images / Total number of tested images.

In order to provide an overview of the individual experiments, we combined all results of each (Classifier/Image representation) combination in an aggregate table. An example is illustrated in Table 5.2, which is a typical format for all of the aggregate tables. The table contains the following information:

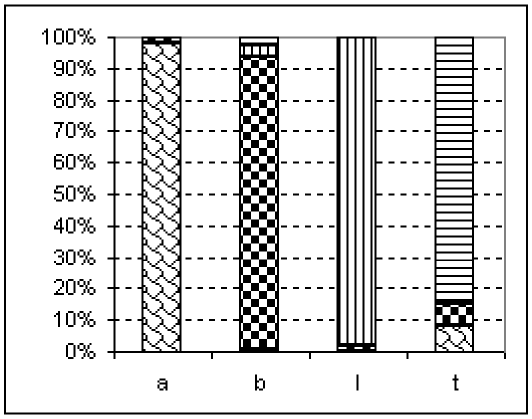
- MLP$_{\text{\_hough}}$: is the label of the table which combines the name of the classifier (MLP), and the image representation method used in this particular experiment.
- For each crack type, the table lists the classification distribution of the 125 test images used in all of the experiments per that particular (Classifier/Image representation) method, and gives the aggregate C/T per each crack type. Finally gives the overall aggregate (average) C/T for the (Classifier/Image representation)

method. The column chart plots the crack type Vs. The aggregate C/T per that crack type.

Table 5.2 A sample of the aggregate results

MLP_Hough

	C/T	a	b	l	t	total
A	0.984	**123**	2	0	0	125
B	0.928	1	**116**	5	3	125
L	0.984	0	2	**123**	0	125
T	0.848	10	9	0	**106**	125
Over all	**0.936**					**500**



▨ Alligator Cracks (**a**) ▦ Block Cracks (**b**) ▥ Longitudinal Cracks (**l**) ▤ Transverse Cracks (**t**)
C/T = Number of correctly classified images / Total number of tested images.

5.1. MLP

The Hough Method

The overall classification accuracy of this approach was 93.6% Table D4.1, Appendix D4. The lowest accuracy was 92.0% Table D1.1, and the highest was 96% Table D1.4. The classification accuracies per crack-type were as following: for the alligator was 98.4%, for the block was 92.8%, for the longitudinal was 98.4%, and for the transverse was 84.8%. The most recongnizable observation on this method was the relatively low classification accuracy for the transverse cracks which was 84.8%, which can be seen across the result tables in appendices D1 and D4.

The Projection Method

The overall classification accuracy of this approach was 98.6% Table D4.2, Appendix D4. The lowest accuracy was 97% Table D1.7, and the highest was 100% Table D1.10.

The classification accuracies per crack-type were as following: for the alligator was 100%, for the block was 97.6%, for the longitudinal was 98.4%, and for the transverse was 98.4%. The results of the projection method were clearly significantly better than those of the Hough method, and they were consistent for all crack types. Detailed results are found in the appendices D1 and D4.

5.2. GA

The Hough Method

The overall classification accuracy of this approach was 89.2% Table D4.3, Appendix D4. The lowest accuracy was 87% Table D2.3, and the highest was 91% Table D2.4. The classification accuracies per crack-type were as following: for the alligator was 96%, for the block was 92.8%, for the longitudinal was 86.4%, and for the transverse was 81.6%. We noticed that the performance relatively was the poorest with the transverse cracks at 81.6%, which was the case in the MLP/Hough method above.

The Projection Method

The overall classification accuracy of this approach was 98.2% Table D4.4, Appendix D4. The lowest accuracy was 97% Table D2.9, and the highest was 99% Table D2.6. The classification accuracies per crack-type were as following: for the alligator was 96.8%, for the block was 97.6%, for the longitudinal was 100%, and for the transverse was 98.4%. Again in this method we see the significantly better accuracy and consistency of the projection method compared to the Hough method.

5.3. SOM

The Hough Method

The overall classification accuracy of this approach was 86% Table D4.5, Appendix D4. The lowest accuracy was 82% Table D3.2, and the highest was 87% Table D3.4. The classification accuracies per crack-type were as following: for the alligator was 79.2%, for the block was 83.2%, for the longitudinal was 97.6%, and for the transverse was 84%. We noticed that the performance relatively was the poorest with the alligator cracks at 79.2%.

The Projection Method

The overall classification accuracy of this approach was 98.4% Table D4.6, Appendix D4. The lowest accuracy was 97% Table D3.9, and the highest was 99% Table D2.10. The classification accuracies per crack-type were as following: for the alligator was 100%, for the block was 96.8%, for the longitudinal was 99.2%, and for the transverse was 97.6%. Again in this method we see the significantly better accuracy and consistency of the projection method compared to the Hough method.

5.4. Automated Thresholding Experiments

As explained before in Section 4.2, after we had studied some generic automatic thresholding, we concluded that we should postpone the automatic-thresholding phase, and focus on the image representation and crack classification. After we had promising results, it was logical to try to test our developed system, using one of the generic image segmentation software [25] to assess the importance of this phase. The complete results are included in Appendix D5. The highest classification accuracy was produced by the method of SOM/Hough at 34%. These results strongly suggest that our system is sensitive to segmentation accuracy, and that work on developing an adaptive segmentation algorithm that integrates properly with the existing system, should form an important part of future work.

6. DISCUSSION

The results from Section 5, show that the best image representation for this study was the Projection, which worked significantly better than the Hough method with all of the classifier systems. This suggests that simple representations should be considered among alternatives when exploring classification. The best classifier model was the MLP/Projection at an over all average accuracy of 98.6%, but the other classifiers (GA/Projection), and SOM/Projection were not far behind at an over all accuracy of 98.2%, and 98.4 respectively. From the point of view of comparing the three classifier models as supervised MLP & GA vs. non-supervised SOM learning, there no significant difference in the results to be reported, they gave very close results and they both were satisfactory when used with the Projection method as an image representation technique. The aggregate results showed that the worst performance of Hough method was with the transverse cracks where the accuracies of the three systems were: 84%, 81.6%, and 84% respectively. The transverse cracks were misclassified as alligator and block cracks.

In an attempt to investigate the cases where misclassification occurred, we present twelve samples, four per each classifier:

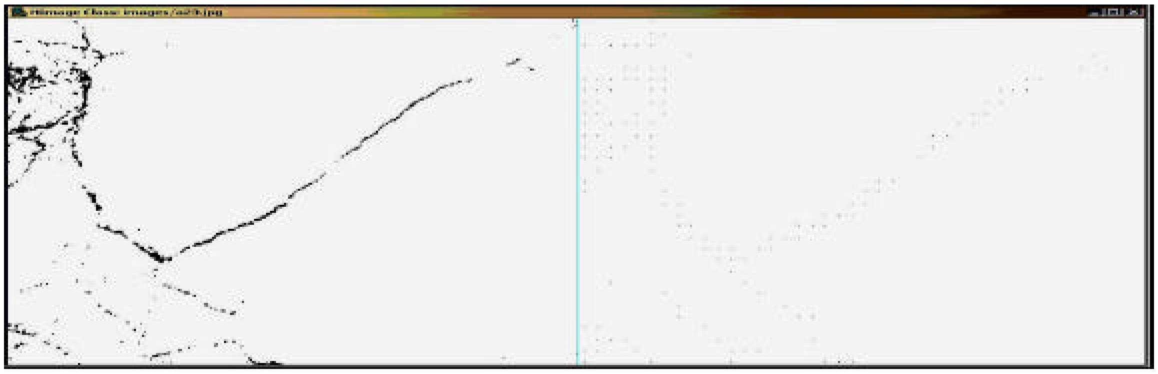


Figure 6.1 Misclassified image: type alligator, classified as block.

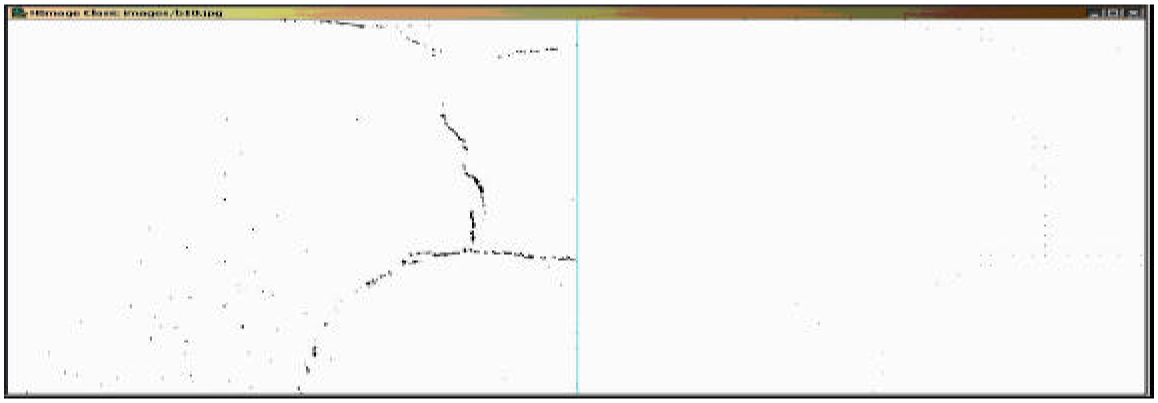


Figure 6.2 Misclassified image: type block, classified as longitudinal

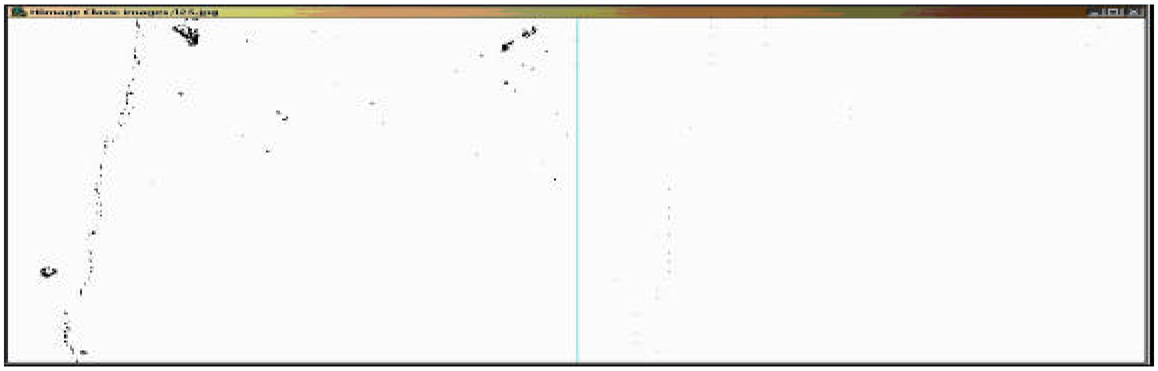


Figure 6.3 Misclassified image: type longitudinal, classified as transverse

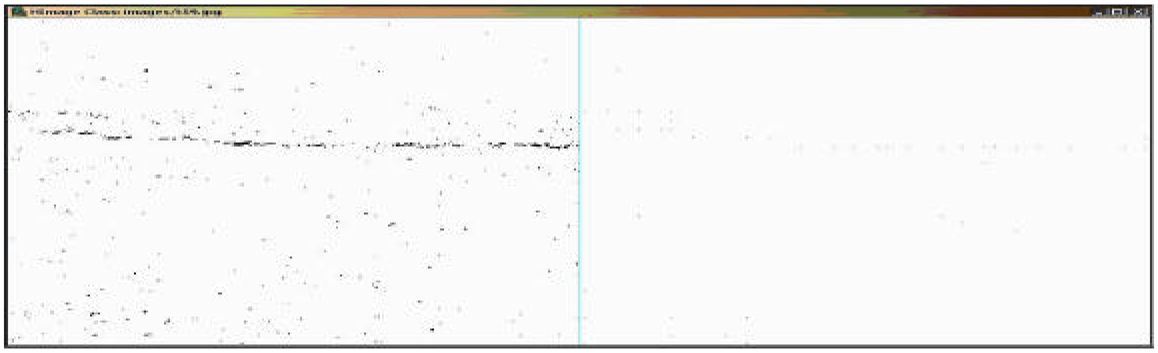


Figure 6.4 Misclassified image: type transverse, classified as longitudinal

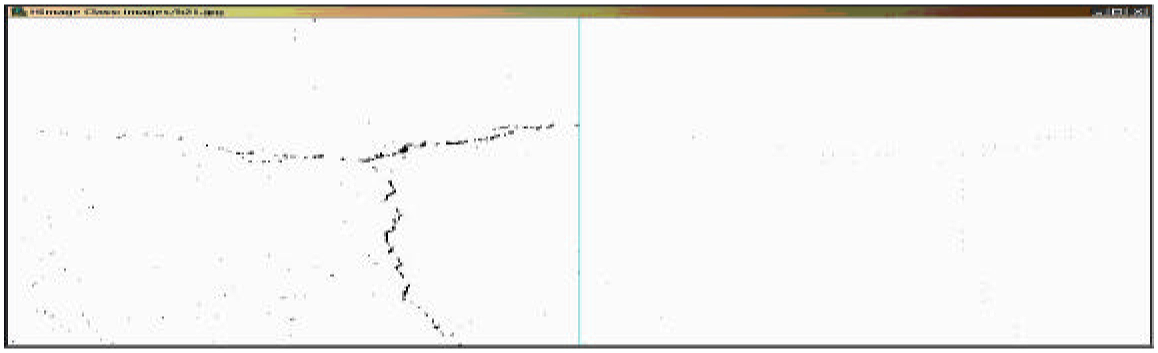


Figure 6.5 Misclassified image: type block, classified as transverse

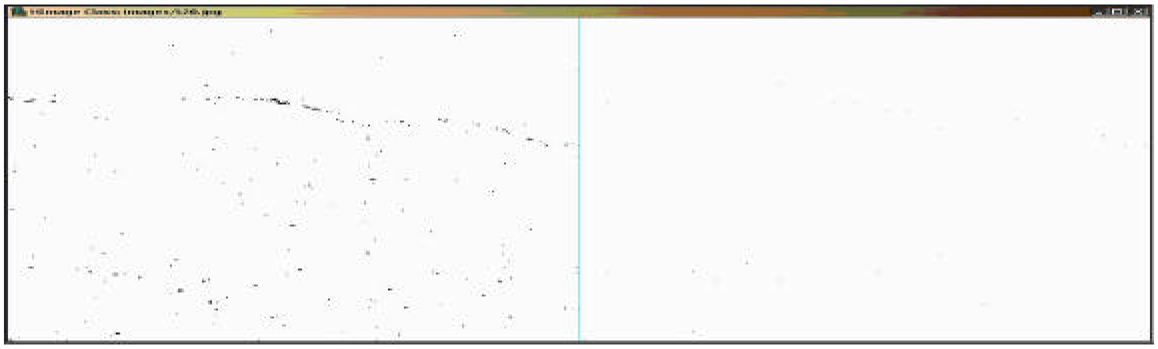


Figure 6.6 Misclassified image: type transverse, classified as block

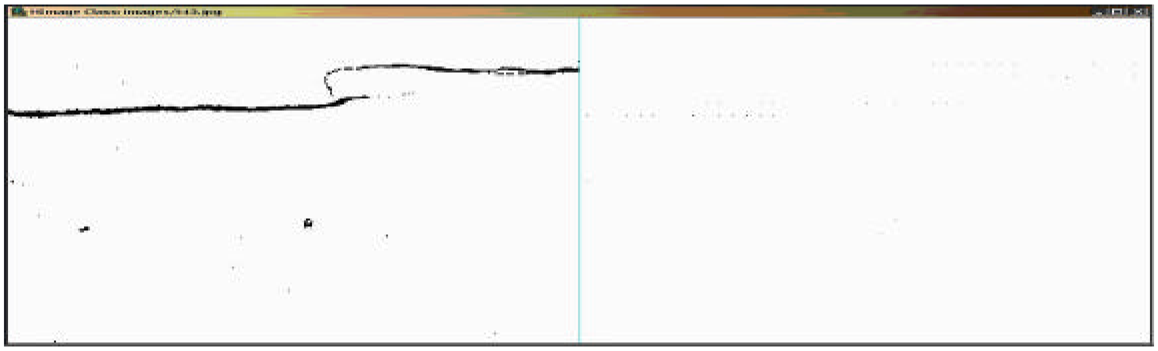


Figure 6.7 Misclassified image: type transverse, classified as longitudinal

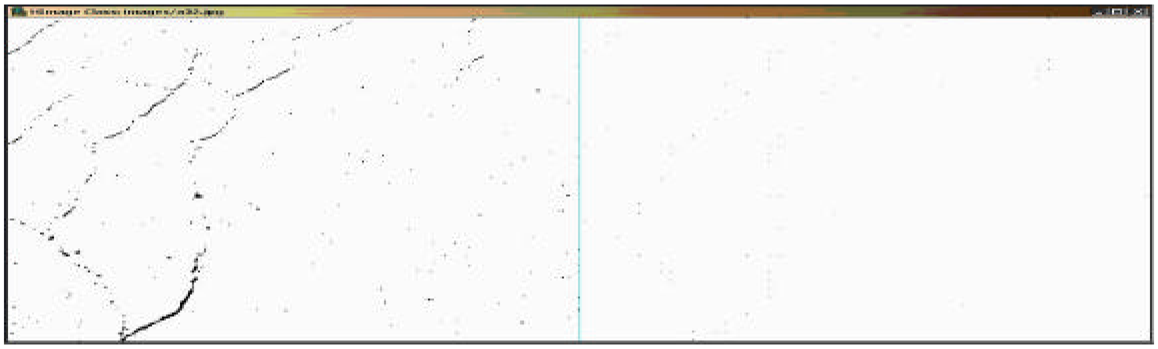


Figure 6.8 Misclassified image: type alligator, classified as block

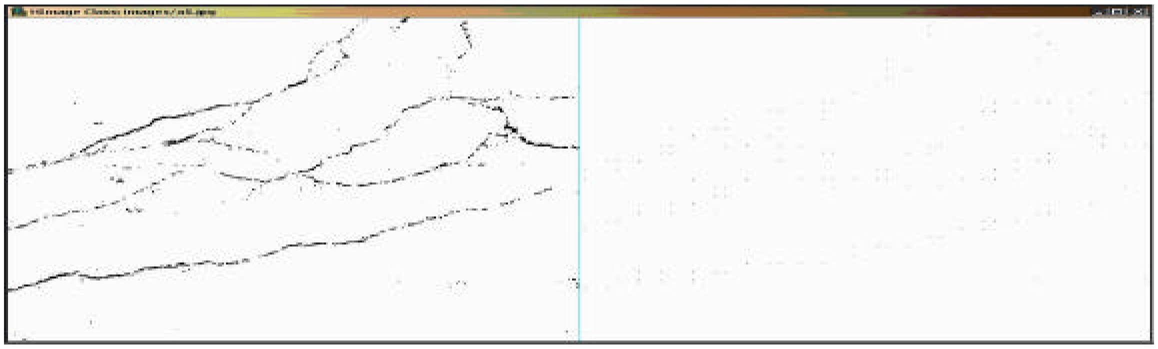


Figure 6.9 Misclassified image: type alligator, classified as transverse

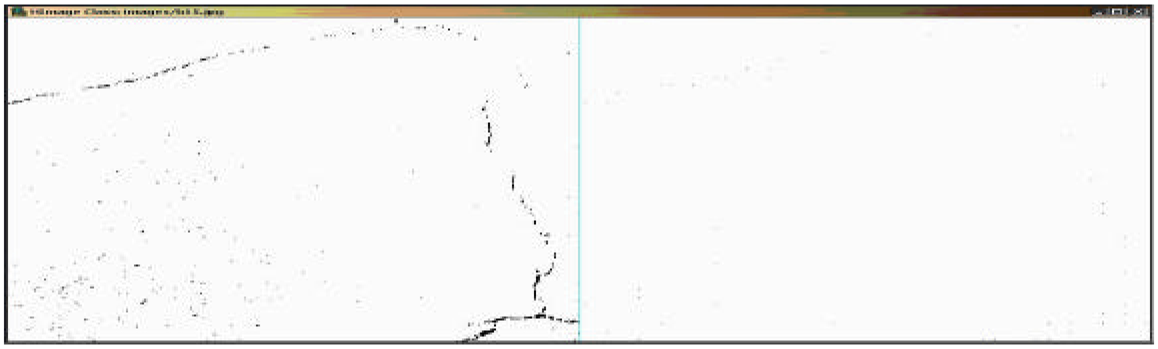


Figure 6.10 Misclassified image: type block, classified as longitudinal

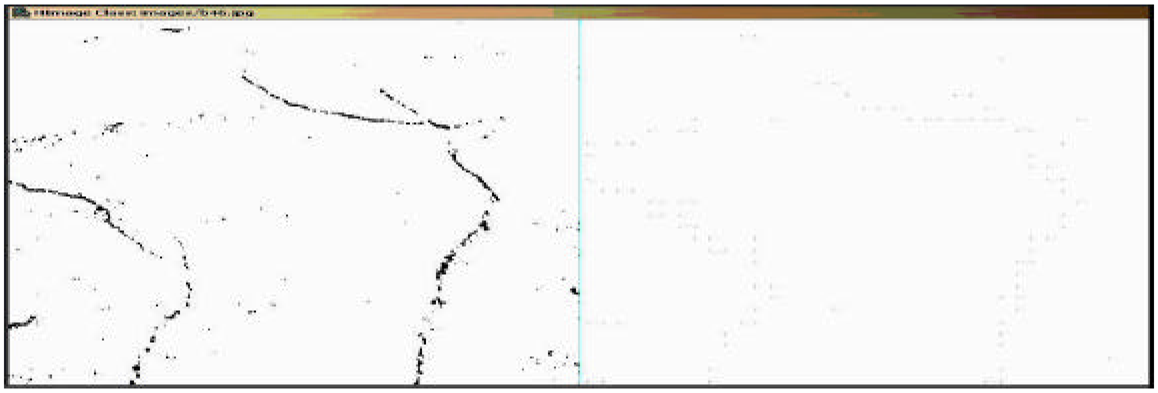


Figure 6.11 Misclassified image: type block, classified as transverse

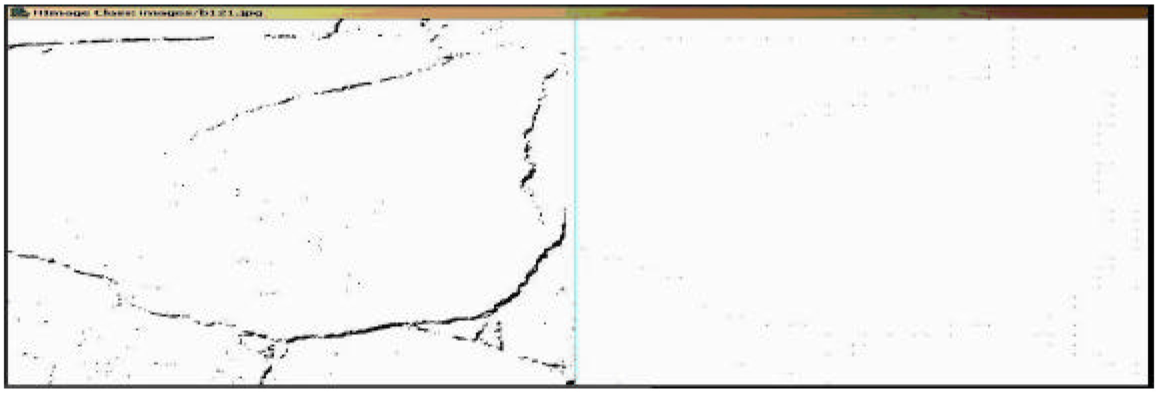


Figure 6.12 Misclassified image: type block, classified as transverse

Table 6.1 Samples of misclassified images
A: alligator **B**: block **L**: longitudinal **T**: transverse

Case No.	Classifier	Feature Method	Figure	Type	Classified	Nearest Neighbor
1	MLP	Hough	6.1.	A	B	NO
2	MLP	Hough	6.2.	B	L	NO
3	MLP	Projection	6.3.	L	T	NO
4	MLP	Projection	6.4.	T	L	NO
5	GA	Hough	6.5.	B	T	NO
6	GA	Hough	6.6.	T	B	NO
7	GA	Projection	6.7.	T	L	NO
8	GA	Projection	6.8.	A	B	YES
9	SOM	Hough	6.9.	A	T	YES
10	SOM	Hough	6.10.	B	L	YES
11	SOM	Projection	6.11.	B	T	YES
12	SOM	Projection	6.12	B	T	NO

We analyzed the images illustrated in Figure 6.1 through 6.12, and Table 6.1 We had the following observations on the possible causes behind misclassification:

1. When the crack objects are weak, due to thinness or strong noise. This case can be seen in Figures 6.5, and 6.6.
2. Although the crack patterns appear to be quite strong, the persistent noise can confuse the classifier. Figures 6.3, and 6.4.
3. The nearest-neighbor method is not guaranteed to give accurate results. Table 6.1, cases 8 through 11.

In this section it is helpful to compare our approach and results to other approaches presented in Section 2.

In [12], although their results were perfect, we noticed that their feature extraction method was computational expensive. It involves producing a vector of eighteen features computed by variety of moments. These moments range from 1st to 3rd order equations. In

[27] the overall accuracy of the system was 85%, whereas our lowest accuracy was 89.2%.

In [5] we observed that their best classifier network included 60 nodes in the hidden layer compared to our system which has only 3 nodes with corresponding less computation load. So we believe our system should perform better in real time processing. Further more, even though, we adopted their tiling method and followed similar training and testing procedures, the overall of their best classier network had an accuracy of 93.7%, whereas our best classifier has 98.6%. In addition, a comparison by crack type is given in table 6.2.

Table 6.2. A comparison by crack type between [5] and our system

	Their system	**Our system**
Alligator	88.00%	100.00%
Block	88.00%	97.60%
Longitudinal	98.00%	98.40%
Transverse	90.00%	98.40%

Table 6.2, indicates that our system produced higher classification accuracy, especially with alligator and block cracks, which they formed a constraint for the system developed in [5].

7. FUTURE POSSIBILITIES

During our study, we encountered several potential possibilities for future work, to improve our system. Theses possibilities include:

1. As mentioned earlier, the proposed system would not be fully automated unless we achieve automatic thresholding. So this phase is highly recommended to be

considered in future work. If we find it suitable, we may design an algorithm to automatically find the optimum threshold value for each input image, and do the thresholding based on that to extract the cracks objects from the entire image. Knowledge about the objects in the scene, the application, and the environment should be used in the segmentation algorithm such knowledge includes:

- Intensity characteristics of objects.
- Sizes of the objects.
- Fractions of an image occupied by the objects.
- Number of different types of objects appearing in an image. [13]

In the domain of our problem, these characteristics are not defined, so we need to devise an adaptive algorithm to achieve the automatic thresholding.

2. Since our study showed that the approach we used to address the problem was reliable, and repeatable, therefore we think our system has the potential to be expanded to include other subsets of pavement distresses as described in the introduction.
3. Pavement indices require not only classification of distresses, but also computing geometric properties such as: width, length and area, etc., so we think it would be helpful to investigate the possibility of integrating such features in the future.
4. Pavement surface may contain numerous foreign objects, such as oil residue, dirt, lane markings, vehicle's tire mark, tree limbs, and other non-distress related items. It is important to develop algorithms to correctly distinguish the distresses or cracks from these non-distress items. We did not consider this problem in the scope of the current study, so we think it is one of the future possibilities that can be considered.

5. The special case of cracks oriented at 45^o, addressed in image representation section, is a potential improvement to be considered in future work. To do that we have to consider the following:
   - In the GA classifier system, considering a third component in the feature vector will require us to evolve a three-dimensional matrix and to redesign the classification algorithm to adapt this new component.
   - In MLP and SOM classifier systems, the only change have to be made is to add a new component at the input layer.
   - We can use cascading strategy by using first SOM to reduce the dimensionality of the feature vectors from 3D to 2D then use the resulting map to evolve the GA classifier matrix.
6. It is important to address the minimum crack width our system can classify, and to work on enhancing this capability, since it was addressed as a problem in the discussion section.
7. It was suggested by a colleague [29], to improve the efficiency of our version of nearest-neighbor algorithm in the GA and SOM classification model. The suggestion was to pre-fill the null-region in the classifier matrix, do the system don't have to waist time resolving classification in case of a null hit.
8. It was suggested by [3] that our system may be capable of pavement texture classification. Further investigation is required to test the system for this capability.

8. SUMMARY

In summary, we presented a study designed to improve an automated pavement crack classification system, by targeting a subset of pavement crack types: longitudinal, transverse, block, and alligator cracking. We approached this problem by exploring the capabilities of three different approaches: multilayer perceptron (MLP), genetic algorithms (GA), and self-organizing maps (SOM). We used computer vision methods for image preprocessing (image representation and feature extraction). For crack objects segmentation and image filtering we used manual thresholding followed by median filtering. We used two different techniques for image representation: 1. image tiling with local linear regression to generate a hit pattern that reduces and approximates the original crack pattern followed by projecting these hits onto the two principal axis (X & Y). The number of projected hits on each axis is added to form a component in the 2D feature vectors which characterized the crack patterns. 2. The Hough transform: this approach was used on the hit pattern images, which produced a Hough output image that was used to characterize crack patterns by detecting the slopes or the orientation of the detected Hough lines and then to produce a feature vector that can characterize the crack patterns. In this case, the vectors have the form of (Average angle of Hough space: Total number of hits). An extensive testing was conducted following a systematic procedure of training and testing for all developed classifier using 400 training images and 100 images for testing with all possible combinations of (Projection/Hough : MLP/GA/SOM : training/testing sets), and the best classifiers gave the following accuracy: Hough/MLP 93.6%, Projection/MLP 98.6%, Hough/GA 89.2%, Projection/GA 98.2%, Hough/SOM 86%, and Projection/SOM 98.4%. Since we performed the image thresholding manually, we wanted to test the performance of the best developed classifier systems by testing them with automatically-thresholded images using arbitrary image segmentation software [26]. The best result among the three classifier systems was: Hough/SOM at 34%. The results showed that our system is sensitive to thresholding suggesting that an adaptive segmentation technique will be necessary to fully automate the system.

9. BIBLIOGRAPHY

1. David Timm, and Jason McQueen. 2004. *A Study of Manual vs. Automated Pavement Condition Surveys*. Auburn Alabama.

http://www.eng.auburn.edu/users

2. Gregory Cline, Mohamed Shahin and Jeffrey Burkhalter. 2003. Automated Data Collection for Pavement Condition Index Survey. *TRB 2003*.

http://www.pavementconsultants.com

3. M. H. Mohajeri and P. J. Manning. 1991. ARIA™, An Operating System of Pavement Distress Diagnosis by Image Processing. *Transportation Research Record 1311,* TRB, National Research Council, Washington, D.C., pp. 120-130.

4. Kelvin Wang and Robert Elliot. 1999. *Investigation of Image Archiving for Pavement Surface Distress Survey*. University of Arkansas, Department of Civil Engineering Fayetteville.

http://www.mackblackwell.org/research

5. Byoung Lee, Hosin "David" Lee. 2003. A Robust Position Invariant Artificial Neural Network for Digital Pavement Crack Analysis. *Technical Report TRB2003-000996.*

http://www.ltrc.lsu.edu

6. Kelvin Wang, We-Yih Tee Quintin Watkins and Subash Kuchikulla. 2002. Digital Distress Survey of Airport Pavement Surface. *Federal Aviation Administration Airport Technology Transfer Conference* 2002.

http://www.airporttech.tc.faa.gov

7. US department of Transportation, Federal Highway Adminstration. 2003. *Distress Identification Manual*. FHWA-03-031. Mclean, VA.

http://www.tfhrc.gov

8. Bruce Vandre. *Pavement Management & Design*.

http://www.udot.utah.gov

9. Scott E Umbaugh. 1998. *Computer Vision and Image Processing A practical Approach Using CVIPtools*. New Jersey: Prentice Hall PTR.

10. Stephen A. Book. 1977. *Statistics, Basic Techniques for Solving Applied Problems*. California: McGRAW-HILL.

11. Hosin Lee. 1991. Application of Machine Vision Techniques for the Evaluation of Highway Pavements in Unstructured Environments. *IEEE-91 ICAR, 7803-0078/91/0600-1425*.

12. JaChing Chou, Wende O'Neill and H.D. Cheng. 1994. Pavement Distress Classification Using Neural Networks. *IEEE-94, 0-7803-2129-4/94*.

13. Ramesh Jain, Rangachar Kasturi and Brian Schunck. 1995. *Computer Vision*. San Diego: MIT Press and McGraw-Hill.

14. Don R. Hush And Bill G. Horne. 1993. Progress in Supervised Neural Networks. *IEEE Signal Processing Magazine, January 1993*.

15. Simon Haykin . 1994. *Neural Networks A Comprehensive Foundation*. New Jersey: Mcmillan Publications.

16. William F. Punch III, Behrouz Minaei-Bidgoli1. *Using Genetic Algorithms for Data Mining Optimization in an Educational Web based System*. Michigan State University Department of Computer Science & Engineering.

http://www.lon-capa.org

17. William A. Greene. *Unsupervised Hierarchical Clustering via a Genetic Algorithm.* University of New Orleans, Computer Science Department.

http://www.uno.edu.

18. Marek Obitko. 1998. *Introduction to Genetic Algorithms.* University of Applied Sciences.

http://cs.felk.cvut.cz/~xobitko/ga/.

19. Teuvo Kohonen. 1990. The Self-Organizing Map. *Proceedings of the IEEE, vol. 78, No. 9, September 1990.*

20. John A. Bullianaria. 2002. *Neural Networks.* The University of Birmingham. http://www.bham.ac.uk/~jxb/nn

21. University of Virginia, Dept. of Materials Science and Engineering. 2001. *Introduction to Materials Science, Materials Failure.*

http://www.people.virginia.edu/

22. Fredric M. Ham, Ivica Kostnic. 2001. *Principles of Neurocomputing for Science & Engineering*.

McGraw-Hill publications.

23. University of Stuttgart - Applied Computer Science, University Of Tubingen - Department of Computer Architecture. 19989. *Stuttgart Neural Network Simulator (SNNS-4.2).*

http://www-ra.informatik.uni-tuebingen.de/SNNS/

24. Helsinki University of Technology, Finland. SOM_PACK, the Self-Organizing Map, version 3.1 (April 7, 1995).

http://www.cis.hut.fi/research/som_pak/

25. Dana Frajitoru. 2002. *GAD, general purpose Genetic Algorithm Software*. Indiana University South Bend, Computer Science Department.

26. University of Oslo. 2002. *XITE X based Image processing Tools and Environment.* http://www.ifi.uio.no/forskning/grupper/dsb/Software/Xite/

27. Chung-Jung Hsu, Chi-Farn Chen, Chau Lee, and Shu-Meng Huang. 2001. Airport Pavement Distress Image Classification Using Moment Invariant Neural Network. *22nd Asian Conference on Remote Sensing and processing (CRISP), Singapore*.

28. University of Edinburgh, School of Informatics. http://homepages.inf.ed.ac.uk/rbf/CVonline/

29. Ibrahim Chaaban. Department of Computer and Information Sciences, Indiana University South Bend. Private conversation.

Appendix A MLP Algorithm

A1. The algorithm of MLP:

```
backProbagation()
{
     initializeWeights();
     while( training vectors not finished ) {
          getNextVector();
          feedForward();
          computeGradient();
          updateWeights();
          testTerminationCondition();
     }
}


feedForward()
{
     // L = number of layers in the network
     for( layer = 1 to L, ++layer ){
          // N = number of nodes per each layer
          for( node = 1 to N, ++node )
               computeNodeOutput();

     }
}


computeNodeOutput()
{
     // N = number of nodes of preceding layer
     for( node = 1 to N, ++node ){
          findWeightedSum();
          computeSigmoid( weightedSum );
     }
}
```

```
computeGradient()
{
  for( layer = L to 1, --layer){
    for( node =  to N, ++node ){
      if( layer == L )
        err[node] = output[node] – desired[node];
      else
        err[node] =
          sumof{ err[node,layer+1]*output[node,layer+1]*
          (1-output[node,layer+1])*weight[node,layer+!]};
    }

     for( node = 1 to N, ++node )
       gradient[node] = err[node] * output[node]
      (1-output[node]) * output[node, layer+1];

  }
}


updateWeights()
{
  for( layer = 1 to L, ++layer )
    for( node = 1 to N, ++node )
      weight[layer,node] = weight[layer,node] -
      (learningRate * gradient[layer,node];
}
```

Appendix B Description of common methods

B-1. Median Filter

The Median Filter is a nonlinear filter which has a result that is not obtained by a weighted sum of the neighborhood pixels as opposed to smoothing filters. Median filter operates on a local neighborhood by replacing the center pixel by the median value of the neighboring pixels. A typical use of median filter is noise removal [9]. An example is illustrated in Figure B1.1. The median is found by first sorting neighboring pixels in increasing order of the grayscale value and then finding the center of the sorted values using the formula: (n+1)/2, where n is the number of neighboring pixels.

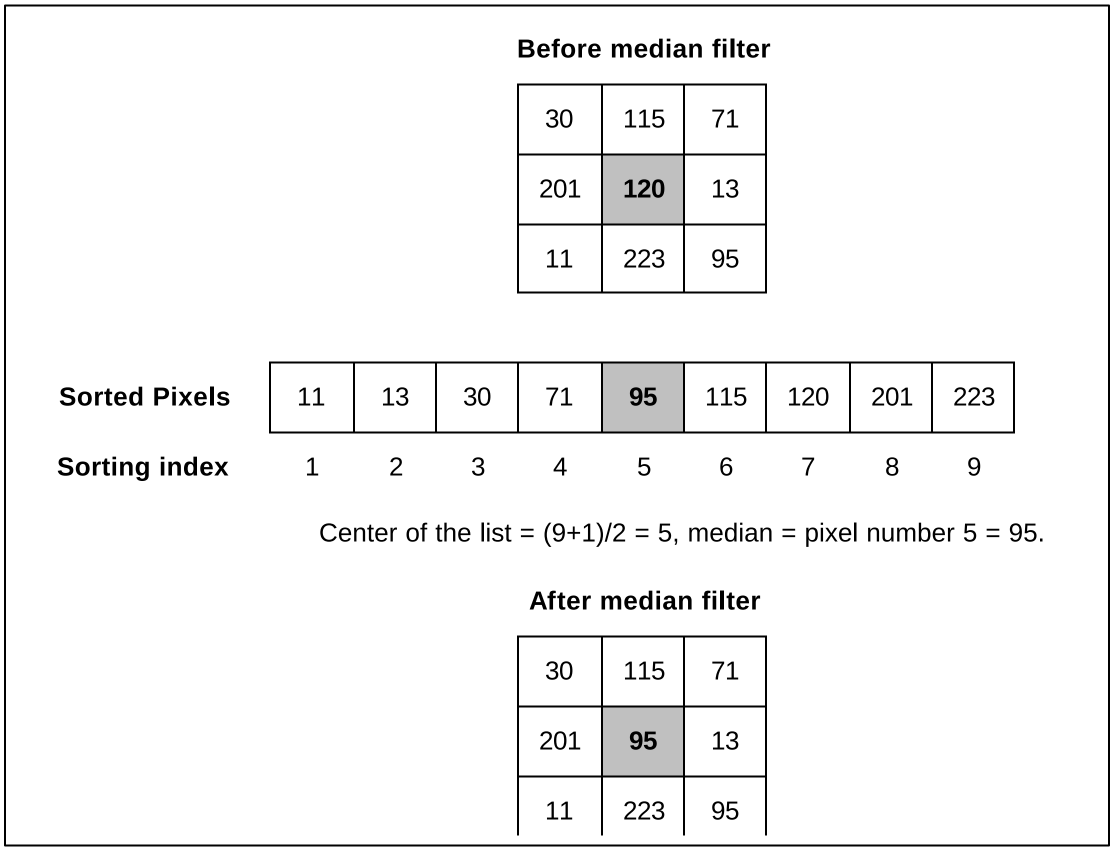


Figure B1.1 An example of median filter

B-2 Hough Transform

The Hough transform is a mathematical technique for line detection [9], which is expected to be suitable for transforming the binary image into crack lines. The Hough transform represents lines using the normal of them r as it is defined in Equation B2.1:

$$r = r\cos(J) + c\sin(J) \qquad \text{(B2.1)}$$

Where:

r = the perpendicular distance between the origin of the quantized space and the target line as illustrated in Figure B2.2. It ranges from 0 to $\sqrt{2}$ N which is the diagonal distance of the image assuming it is a square of a parameter of N pixels.

J = the angle between the vertical axis in the quantized space (which is the r-axis) and r .

r = the row coordinate of the target pixel in the image.

c = the column coordinate of the target pixel in the image.

The algorithm can be summarized as following:

1. Quantize the image space by choosing the desired Δr and ΔJ as shown in Figure B2.1.
2. For every point (pixel) find r solving equation B2.1 where (r, c) are the coordinates of the pixel in the image matrix and J varies between the lower limit and the upper limit in the Hough space.
3. For each pair of (r , J) in step 2, record the (r, c) pair in the corresponding block in the Hough space.
4. Keep track of the number of recoded pairs (hits) in step 3 per each block in the space.

5. When the entire (r, c) space is processed, the (r , J) space is processed by testing each block to check if number of hits exceeded a predefined threshold. All blocks satisfying this criterion are marked as lines in the resulting image.

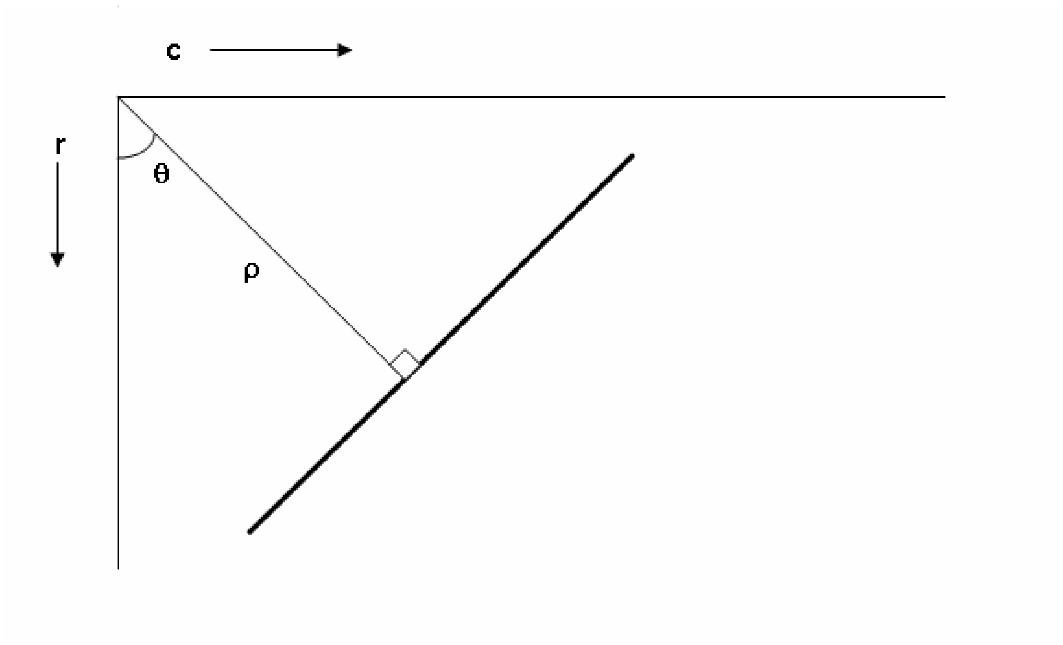


Figure B2.1 Target line representation by the normal in Hough transform

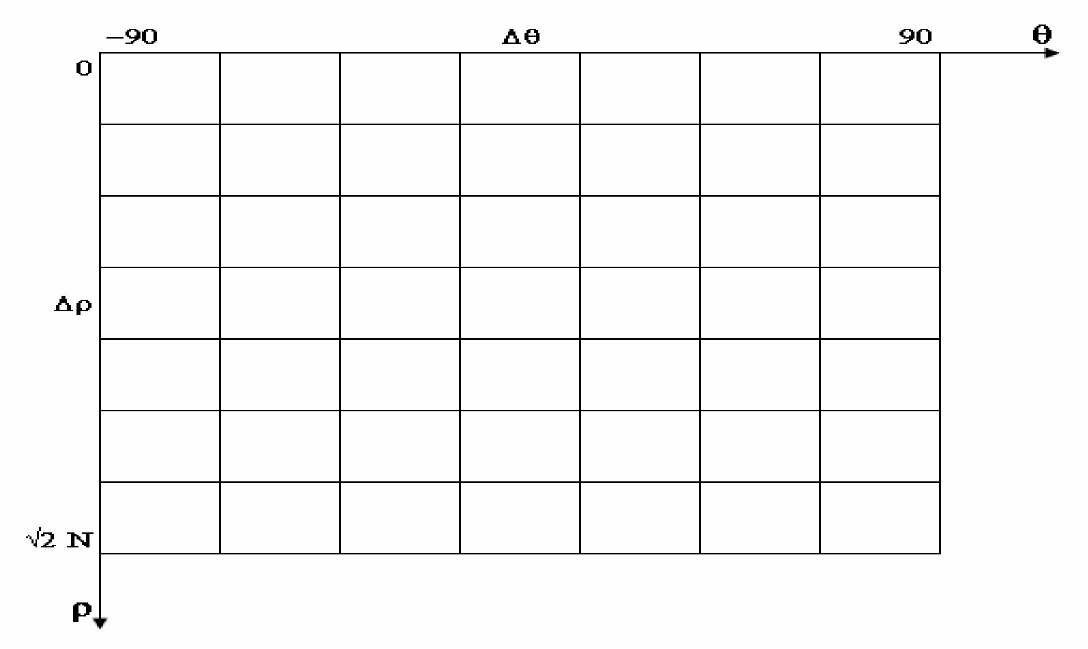


Figure B2.2 Hough quantized space

B-3 MLP

Discussing MLP requires us to have some background about their basic functional element, the perceptron, which is a concept biologically inspired from the neuron in the human nervous system.

The perceptron (Figure B3.1) is the basic structural unit of an MLP, which computes a weighted sum of the components of the input vector and subtracts a threshold value (q) from it. The result is then passed to an activation function which can be Hard-limiting Figure B3.2-a, or Sigmoid Figure B3.2-b.

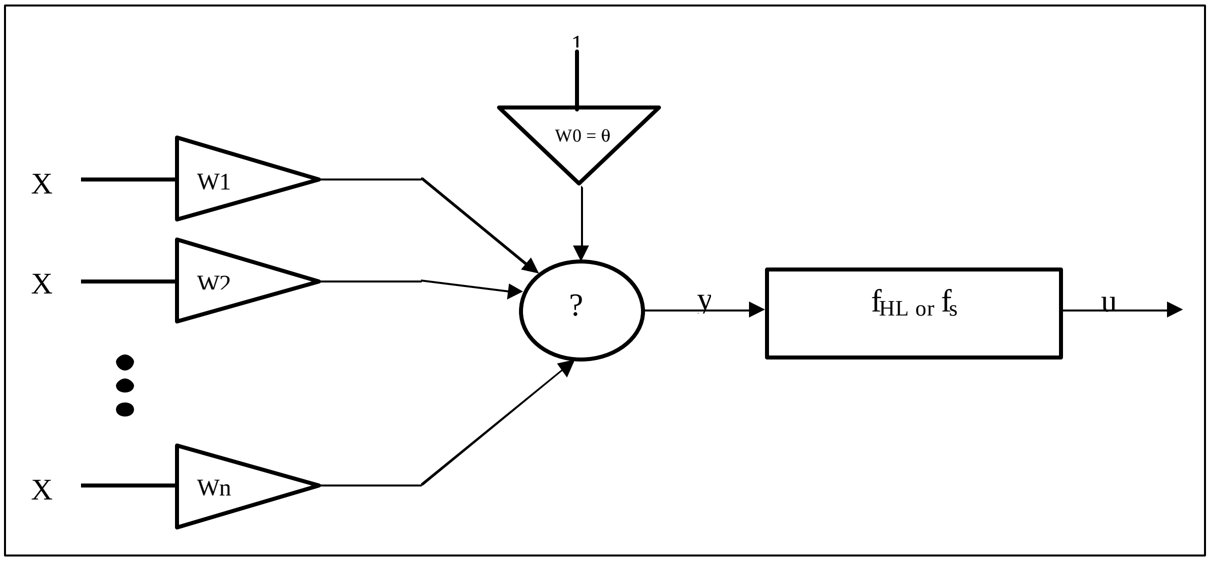


Figure B3.1 The perceptron

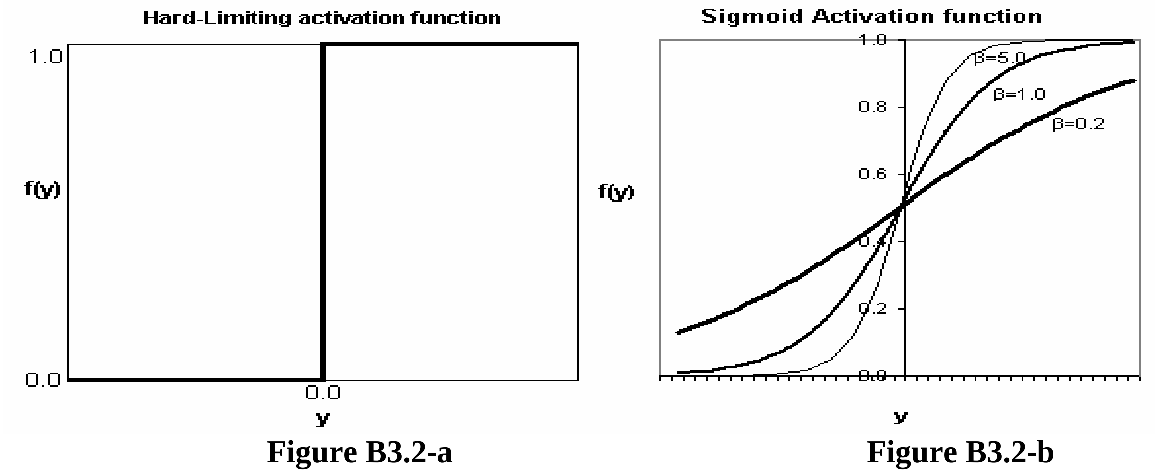


Figure B3.2-a **Figure B3.2-b**

The sigmoid function is defined by Equation B3.1. This function is continuous and the steepness of the transition region is determined by the parameter (ß).

$$f_s(y) = (1 + e^{-ßy})^{-1} \qquad \text{(B3.1)}$$

Figure B3.2-b shows the effect of ß using three different values: 0.2, 1.0, 5.0. One of the advantages of the sigmoid function is its differentiability. This property is essential for the gradient search learning algorithm of the MLP as we will discuss later.

The operation of the perceptron can be viewed in two ways:

- Discriminant function: A pattern recognition problem, where the perceprton performs a nonlinear transformation from the input space to the output space which is the set of the two recognized classes {0, 1}.
- Binary logic unit: it is capable of implementing numerous logic functions, including the three fundamental operations of the Boolean algebra: AND, OR, and NOT.

The capabilities of a single perceptron model are limited to linear decision boundaries and simple logic functions. Linear decision boundaries mean that the input space has to be linearly separable in order for the single perceptron model to work. However, by cascading perceptrons in layers, complex decision boundaries and arbitrary Boolean expressions can be achieved.

The Multilayer Perceptron (MLP)

The structure of the MLP (Figure B3.3) consists of three parts:

- Input layer: the first layer of perceptrons that receives the input vector.
- Hidden Layers: Located between the input and the output layer. The output of the input layer is fed into the first hidden layer; the output of the first layer is fed into

the next hidden layer and so on. The number of hidden layers needed varies based on the complexity of the application. Often the nodes (perceptrons) of the adjacent layers are fully connected.

- Output Layer: The multiple nodes in the output layer typically correspond to multiple classes for multi-class pattern recognition problem.

MLP Learning Algorithm (Back Probagation):

The most common approach is the gradient descent algorithm, in which a gradient search technique is used to find the network weights that minimize a criterion function. The criterion function to be minimized is the Sum-of-Square-Error. A complete derivation of the algorithm can be found in [14].

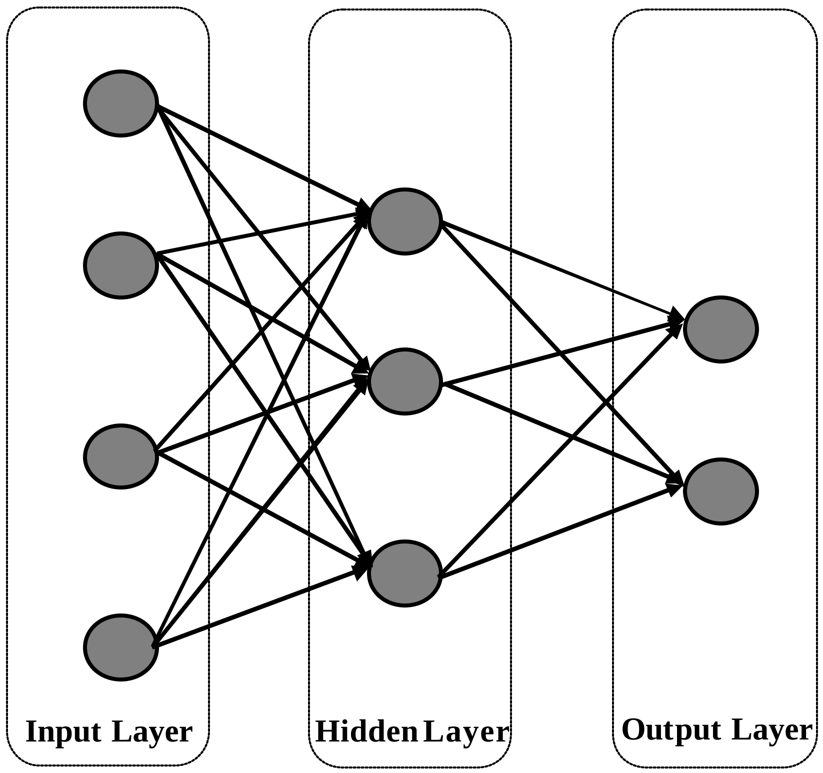


Figure B3.3 MLP Multilayer Perceptron Model

The algorithm can be summarized as following [15]:

Parameters definitions in the algorithm. Parameters are defined for a neuron in layer*(l):*

$w^{(l)}$: Synaptic weight vector of a neuron.

$q^{(l)}$: Threshold of a neuron.

$v^{(l)}$: Vector of net internal activity levels of neurons.

$y^{(l)}$: Vector of function signals of neurons.

$d^{(l)}$: Vector of local gradients of neurons.

e : Error vector represented by e_1, e_2, etc.

1. *Initialization*: all synaptic weights and thresholds are set to small random numbers.
2. *Presentations of Training Examp*les: Present the network with an epoch of training examples. For each example in the set, step 3 and 4 are repeated.
3. *Forward Feed:* let a training example in the epoch be [x(n), d(n)], where x(n) is the input vector, and d(n) is the desired output vector on the output layer. The activation potential and function signals of the network are computed, proceeding forward through the network, layer by layer, using the following relation system:

$$v_j^{(l)}(\text{n}) = \sum_{i=0}^{p} w_p^{(l)}(n) y_i^{(l-1)}(n) \qquad \text{(B3.2)}$$

$$y_i^{(l)}(\text{n}) = \frac{1}{1+\exp(-v_j^{(l)}(n))} \qquad \text{(B3.3)}$$

If the neuron is the first hidden layer, then

$$y_i^{(0)}(\mathrm{n}) = x_j(\mathrm{n}) \qquad \text{(B3.4)}$$

If the neuron is in the output layer (*l* = L), then

$$y_i^{(L)}(\mathrm{n}) = o_j(n) \qquad \text{(B3.5)}$$

Hence the error is computed as

$$e_j(n) = d_j(n) - o_j(n) \qquad \text{(B3.6)}$$

4. *Back Propagation*: compute the local gradients of the network, proceeding backward, layer by layer.

   In the output layer

$$d_j^{(L)}(n) = e_j^{(L)}(n) o_j(n)[1 - o_j(n)] \qquad \text{(B3.7)}$$

   In a hidden layer

$$d_j^{(l)}(n) = y_j^{(l)}(n)[1 - y_j^{(l)}(n)] \sum_k d_j^{(l+1)}(n) w_{kj}^{(l+1)}(n) \qquad \text{(B3.8)}$$

   Hence adjust the weights:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + a[w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)] + h d_j^{(l)}(n) y_j^{(l-1)}(n) \qquad \text{(B3.9)}$$

   where:
   h : is the learning-rate parameter
   a : is the momentum constant.

   The tradeoff of h is a rough approximation for faster processing, or a Better approximation for slower processing. In the case where a high learning-rate is chosen, a is introduced to stabilize the system.

5. *Iteration:* Iterate the computation by presenting new epochs of training examples to the network until the free parameters of the network stabilize their values and the average square error computed over the entire training set is at minimum or acceptable small value. The order of presentation of training examples should be

randomized from epoch to epoch. The momentum and the learning-rate parameter are typically adjusted (and usually decreased) as the number of training iterations increases.

Some points need to be considered in MLP model

1. The weights typically are initialized to small random values, which gives the algorithm a safe start.
2. A simple heuristic technique is used to choose learning rates, which is to make the learning rate for each node inversely proportional to the average magnitude of vectors feeding to that particular layer.
3. Termination criteria include:
    - A target minimum gradient is reached.
    - The Sum-of-Square-Error falls below a fixed threshold.
    - When all of the training samples have been correctly classified.
    - After a fixed number of iterations have been performed.
    - Cross Validation technique: As usual, the available input space is randomly partitioned into a training set and a test set. The training set is further partitioned into two subsets: a subset used for estimation the model (model training), and a subset used for evaluation the performance of the model (model validation); the validation subset is typically 10 to 20% of the training set. The goal of this technique is to validate the model on a data different from the one used for model estimation. The best model is chosen after this validation phase, then the chosen model is trained using the full training set [15].

It is worth mentioning that the last approach (Cross Validation) in contrast to all of the other approaches is not sensitive to the choice of the parameters. It not only avoids premature termination, but can improve the generalization performance. However it is computationally intensive.

B-4 Genetic Algorithm (GA)

Background

Given some problem, a genetic algorithm (GA) seeks to find solutions to it, as follows. Start with an initial generation of candidate solutions to the problem, and then subject this generation to evolutionary forces such as survival of the fittest, mating followed by crossover, and mutation. The hope is that, over time, better and better candidate solutions will surface in the generation. Use the best solution as the working solution. Genetic algorithms were devised by (Holland, 1975) and popularized by (Goldberg, 1989) [17].

The algorithm begins with a set of solutions (represented by chromosomes) called generation. Solutions from one generation are taken and used to form a new generation. This is motivated by a hope, that the new generation will be better than the old one. Solutions which are then selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example reaching a given number of new generations or improvement of the best solution) is satisfied [18].

An abstract GA is described below:

1. *Start:* Generate a random generation of *n* chromosomes (potential solutions for the problem)
2. *Fitness:* Evaluate the fitness *f(x)* of each chromosome *x* in the generation
3. *New generation:* Create a new generation by repeating the following steps until the new generation is complete
   - *Selection:* Select two parent chromosomes from a generation according to their fitness (the better fitness, the bigger chance to be selected)
   - *Crossover:* With a crossover probability, crossover the parents to form new offspring (children). If no crossover is performed, the offspring is the exact copy of parents.
   - *Mutation:* With a mutation probability mutate new offspring at each locus (position in chromosome).
   - *Accepting:* Place new offspring in the new generation.
4. *Replace:* Use new generated generation for a further run of the algorithm
5. *Test:* If the end condition is satisfied, stop, and return the best solution in current *generation.*
6. *Loop:* Go to step 2 [18].

Producing generations by crossing over two parents may cause loosing the best chromosome from the last generation. Elitism is often used to eliminate this problem. This means, that at least one of a generation's best solution, is copied without changes to a new generation, so the best solution can survive to the succeeding generation [18].

Genetic Algorithms have been shown to be an effective tool to use in data mining and pattern recognition. An important aspect of Gas in a learning context is their use in pattern recognition [16]. There are two different approaches to applying GA in pattern recognition:

- Apply a GA directly as a classifier.
- Use a GA as an optimization tool for resetting the parameters in other classifiers.

Most applications of GAs in pattern recognition optimize some parameters in the classification process. Many researchers have used GAs in feature selection. GAs has been applied to find an optimal set of feature weights that improve classification accuracy [16].

B-5 Self-organizing Maps (SOM)

Background

In contrast to other supervised ANN architectures like MLP, SOM is an unsupervised ANN. In this category of ANN, neighboring cells in a NN compete and develop adaptively into specific detectors of different input patterns. It can be pictured as a feature map where each cell or local spatial region becomes tuned to a specific domain of the input space and so the 2D coordinate system of the developed map spatially becomes meaningful in detecting the presence or the absence of a certain input pattern. In his paper [19], Kohonen suggest a strong relationship between brain maps and SOM. He describes much research that has been done to support his idea of the similarity between them. This research has proved that different regions of the brain are mapped to specific functions.

One of the key components of SOMs is the Competitive Learning so we wish to give a rough description of its algorithm:

1. Input space is described as:

   $x = x(t) \in \Re^n$, where t stands for time (15)

2. The set of reference vectors are:

   $\{m_i(t) : m_i \in \Re^n, i = 1,2,3,..., k\}$ (16)

   where $m_i(0)$ is initialized randomly.

3. At each instance (t), x(t) is compared with each vector in the space described in step 2. Then the best matching $m_i(t)$ is updated in away that guarantees it to better match the current x(t). The comparison is based on some distance measure d(x, m_i), which should be decreased to satisfy the previous condition.

4. If the index of the best matching vector was found to be (c), then the only altered reference vector is m_c. The rest of the reference space is left intact.

5. The reference space becomes spatially tuned to match different pattern domains in the input space. If the input space has a particular probability density distribution *pdf*, then the resulting reference space is located in the input space in such a way that approximates this *pdf*.

SOM Structure and Algorithm

- Selection of the best matching cell

As shown in Figure B5.1 below, the structure of the SOM consists of a 2D matrix of perceptrons (m_i : i=1, 2, 3, …, k). The input vector x is connected in parallel to all of these perceptrons. For each perceptron m_i, the weight vector is denoted by:

$m_i = [m_{i1},\ m_{i2},\ m_{i3},\ \ldots, m_{in}]$. As mentioned above, to select the best matching cell in the map, a comparison criteria is given by: either the inner product between (x . m_i) or the Euclidean distance between the two vectors. Then the winner with the shortest distance is selected to be the best match for the current input vector x.

- Adaptation (Updating) of the weight vectors

Two essential effects of the adaptation learning leading to spatially organized maps are to be emphasized:

  - Spatial concentration of the network activity on the best-match cell and its neighborhood.
  - Further tuning of the best-matching cell and its topological neighbors to the current input vector.

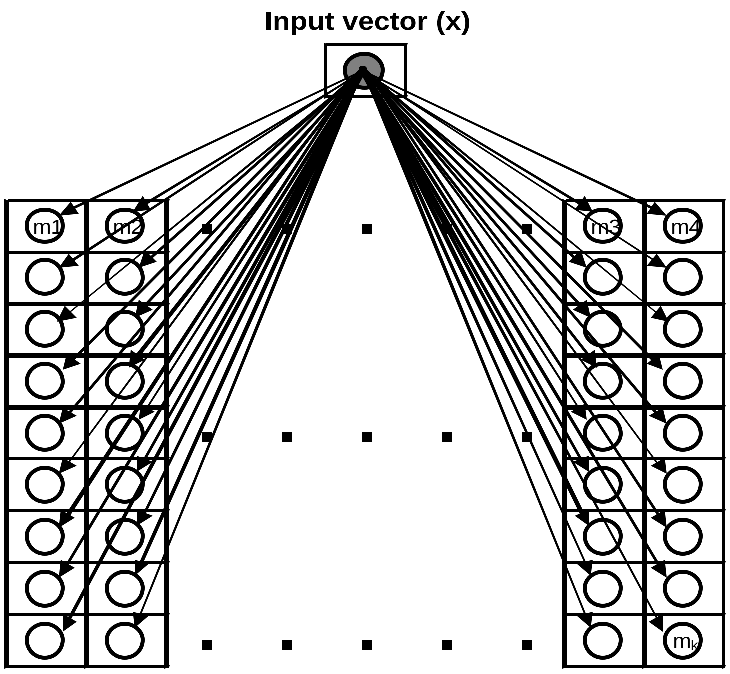


Figure B5.1 SOM Structure

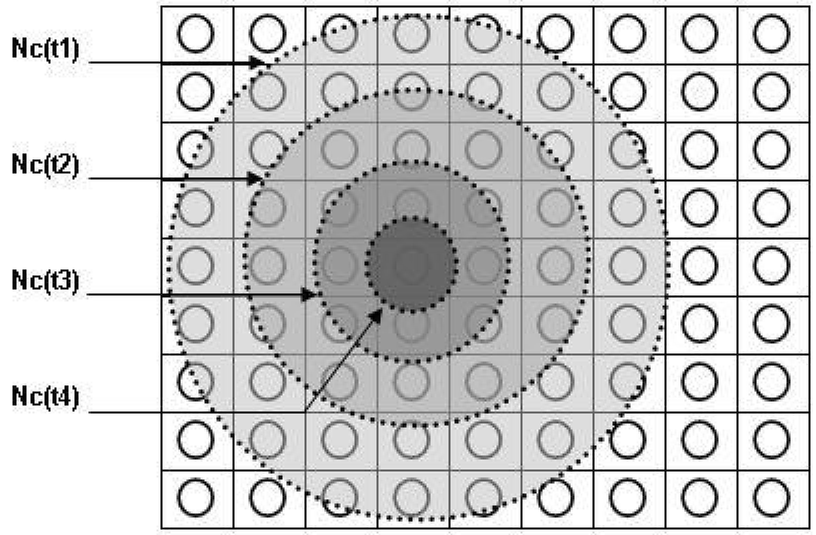


Figure B5.2 SOM neighborhood

Figure B5.2 illustrates the definition of the spatial neighborhood N_c around the best-matching cell at the index c. At each learning step, all cells within N_c are updated, whereas cells outside N_c are left intact. Best match-cell is found by using the following formula:

$$\| x - m_c \| = \min\{\| x - m_i \|\} \tag{17}$$

The radius of N_c varies with time; it starts wide and shrinks monotonically with time (Figure B5.2). Eventually the radius will shrink to 0, which means N_c = {c}.

The updating process can be expressed as:

$$m_i(t+1) = \begin{cases} m_i(t) + a(t)[x(t) - m_i(t)] & i \in N_c(t) \\ m_i(t) & i \notin N_c(t) \end{cases} \tag{18}$$

Where $a(t)$ is a scalar representing the adaptation gain range: 0< $a(t)$ <1. $a(t)$ decreases with time during the learning process. As an alternative to the binary behavior of the updating function, a bell-shaped distribution can be employed. This function is typically used in biological models to represent the normal distribution phenomena.

The goal of the SOM algorithm is to learn a feature map from the spatially continuous input space to the low dimensional spatial discrete output space, which is formed by arranging the computational neurons into a grid. Once the SOM algorithm has converged, the feature map displays important statistical characteristics of the input space. Given an input vector x, the feature map ϕ provides the coordinates of the image of that neuron in the output space. The following give an overview of several properties of the SOM:

- Approximation of the input space

We can state the aim of the SOM as storing a large set of input vectors {x} by finding a smaller set of prototypes {mi} so as to provide a good approximation to the original input space. The theoretical basis of this idea is rooted in vector quantization theory. In effect the goodness of the approximation is given by the total squared distance $D = \sum(x-mi)^2$, which we try to minimize. The weight updating algorithm guarantees generating a good approximation to the input space.

- Topological ordering

The topological ordering property is a direct consequence of the weight update equation that forces the weight vector $m_{i(x)}$ of the winning neuron I(x) to move toward the input vector x. The crucial factor is that the weight updates also move the weight vector Wj of the closest neighboring neurons j along with the winning neuron I(x). Together these weight changes cause the whole output space to become appropriately ordered.

- Density Matching

The feature map reflects variations in the statistics of the input distribution regions in the input space from which the sample training vectors x are drawn with high probability of occurrence are mapped onto larger domains of output space. And therefore with better resolution than regions of input space from which training vectors are drawn with low probability. So the SOM algorithm doesn't match the input density exactly

- Feature selection:

Given data from an input space with a non-linear distribution, the SOM is able to select a set of best features for approximating the underlying distribution. The SOM provides a discrete approximation of finding so-called principal curves or principal surfaces, and may therefore be viewed as a non-linear generalization of PCA principal component analysis.

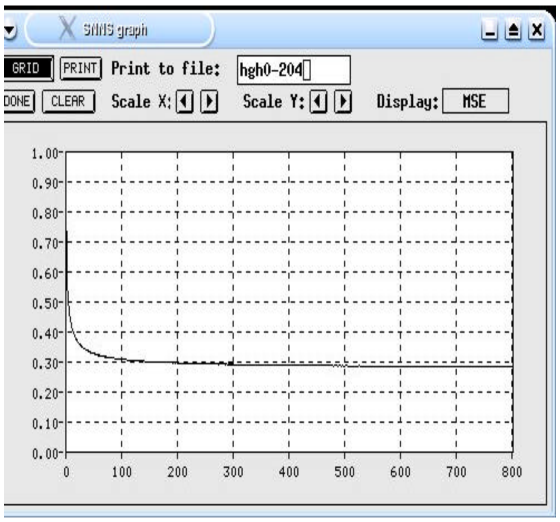
Practical hints for applying SOM algorithm

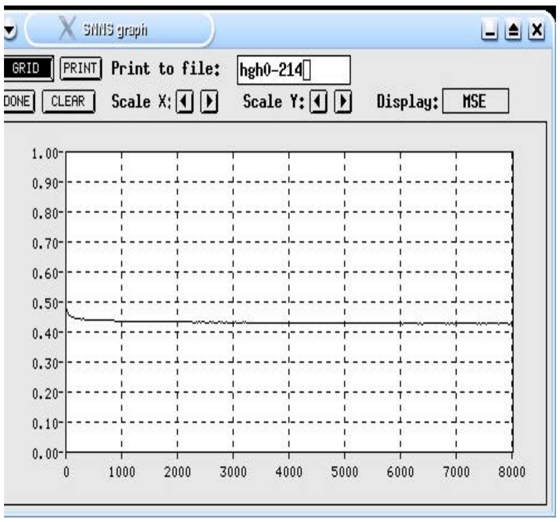
Kohonen gave experimental and practical hints in his paper to safely and efficiently apply his algorithm these hints include:

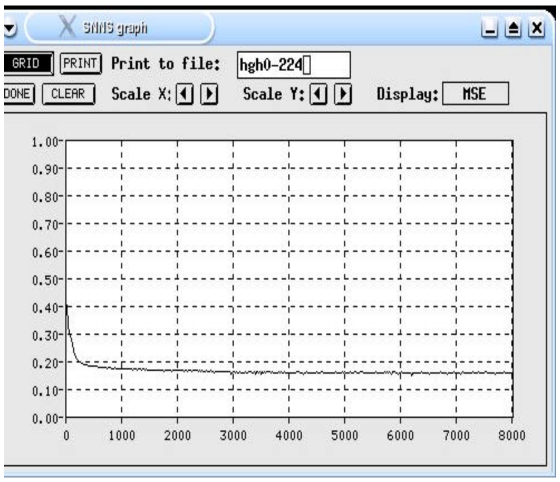
- Number of input vectors in the training set.
- Number of iterations of the algorithm.
- $a(t)$ and its model.
- N_c and its different topologies.

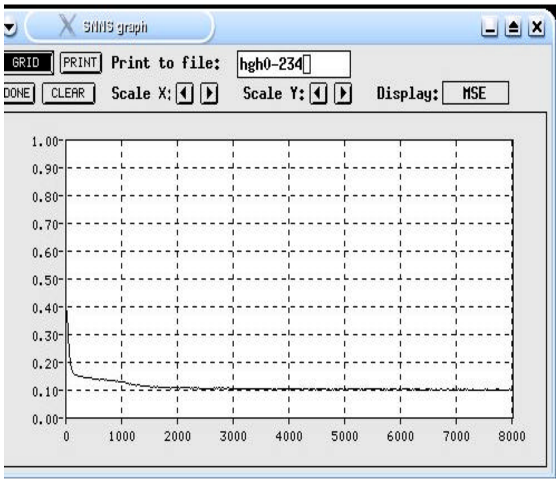
Appendix C Training Results

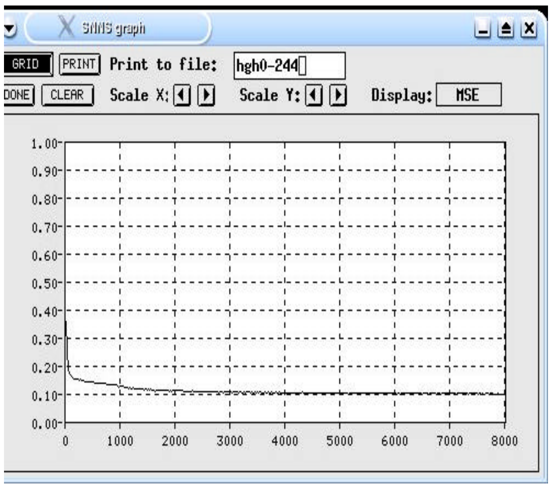
C-1 MLP Finding the optimal number of hidden nodes

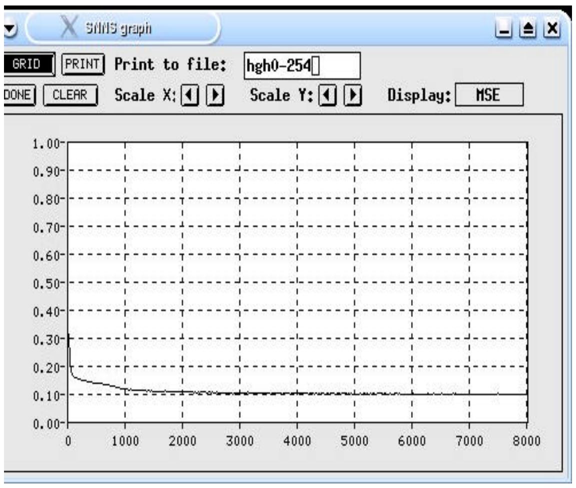

Print to file: hgh0-204
Display: MSE

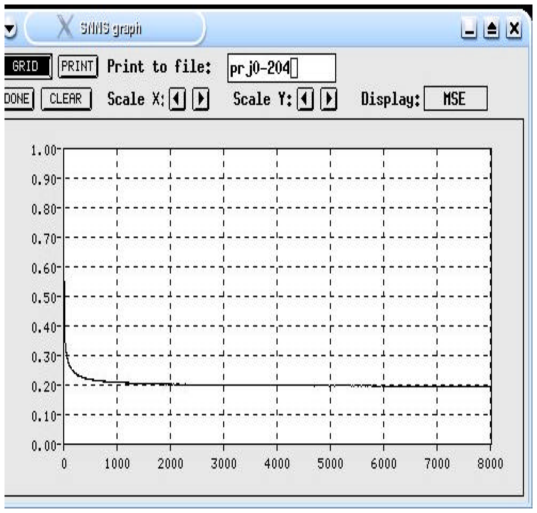

Print to file: hgh0-214
Display: MSE


Print to file: hgh0-224
Display: MSE

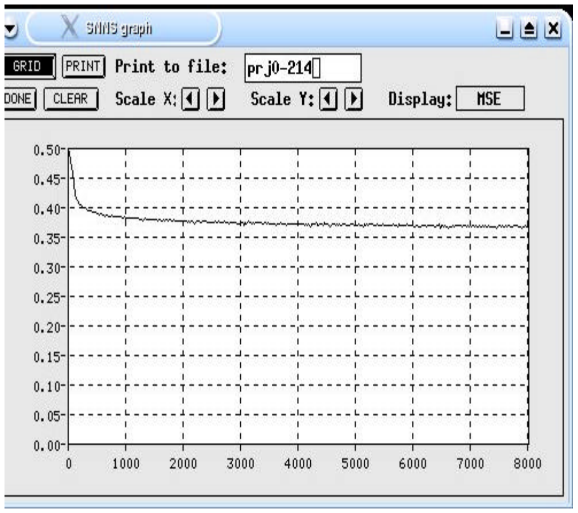

Print to file: hgh0-234
Display: MSE


Print to file: hgh0-244
Display: MSE

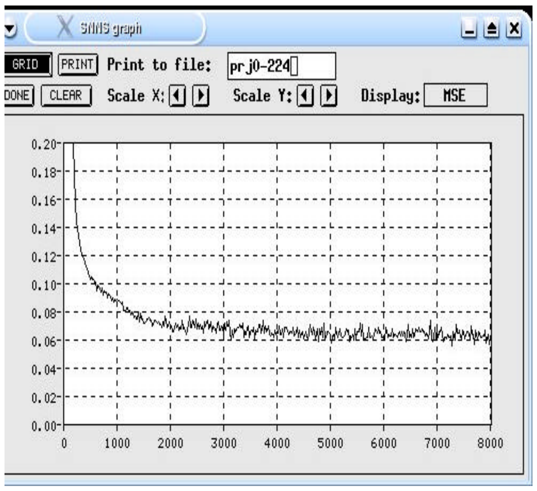

Print to file: hgh0-254
Display: MSE



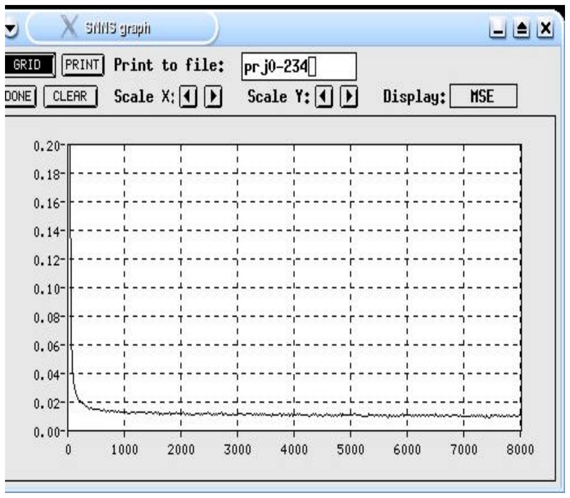
Print to file: prj0-204
Display: MSE



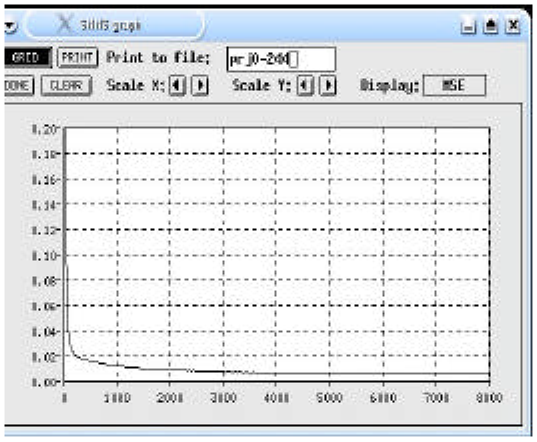
Print to file: prj0-214
Display: MSE



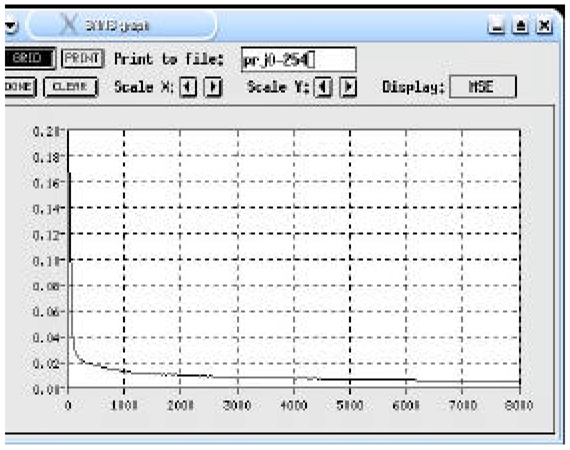
Print to file: prj0-224
Display: MSE



Print to file: prj0-234
Display: MSE

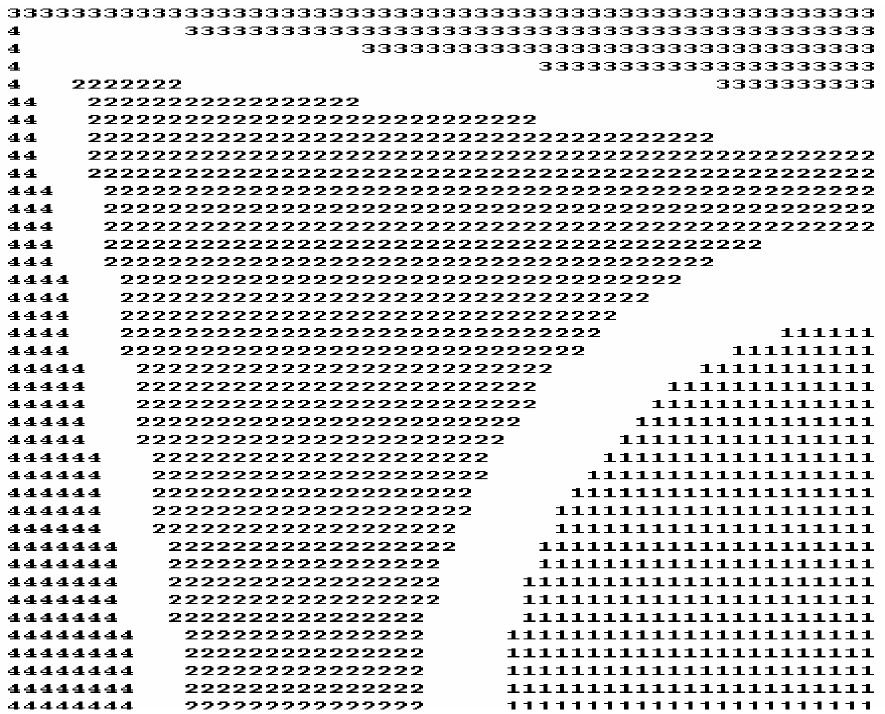


Print to file: prj0-244
Display: MSE



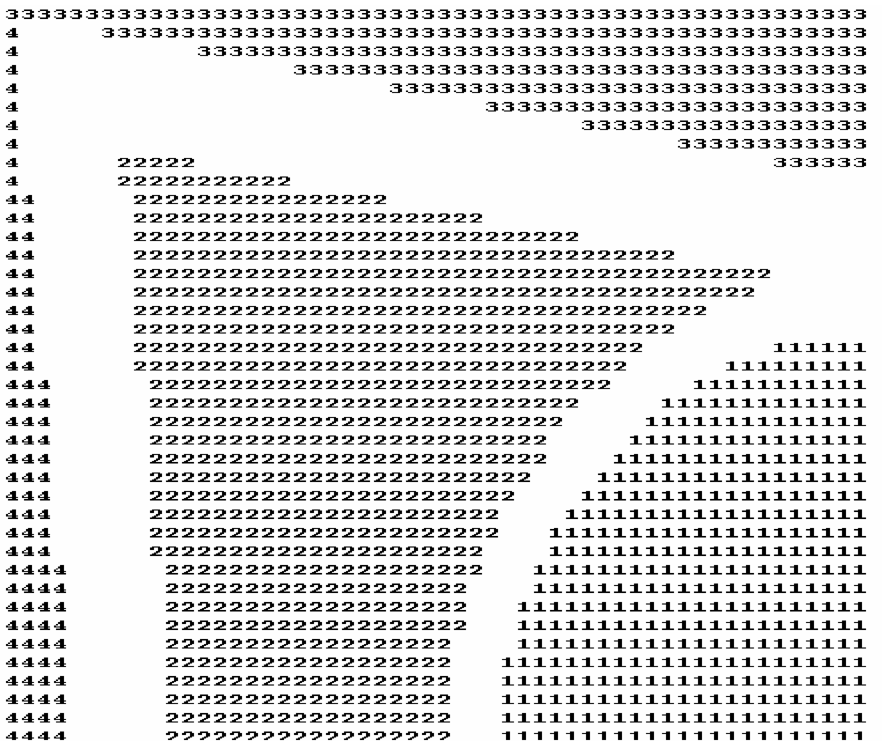
Print to file: prj0-254
Display: MSE

C-2 GA Best evolved chromosomes



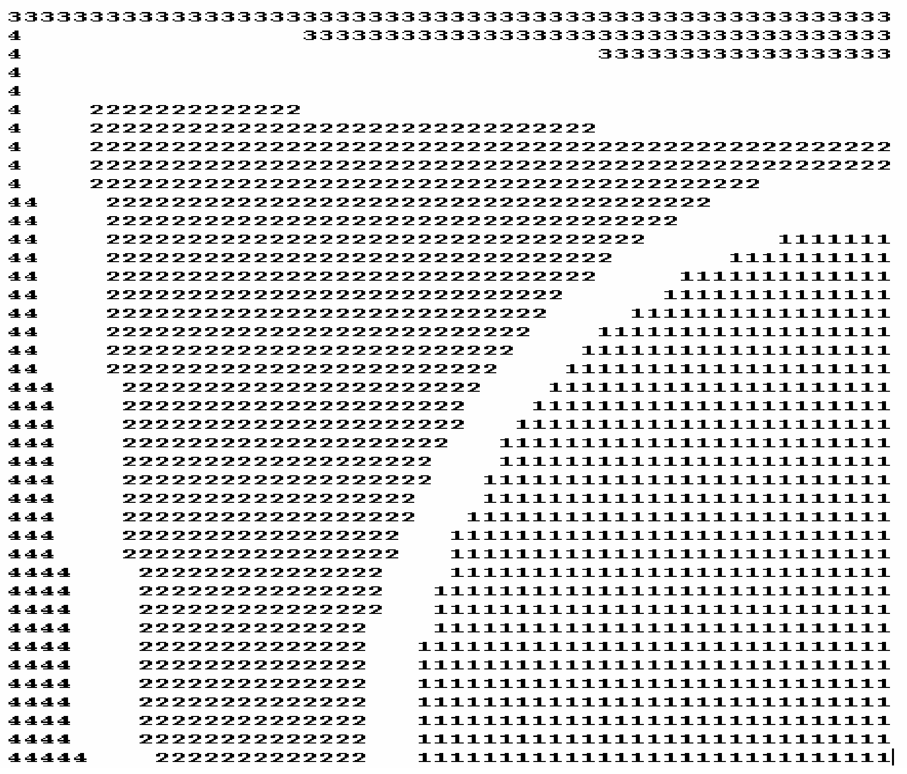
```
333333333333333333333333333333333333333333333333333333
4          3333333333333333333333333333333333333333333
4                     33333333333333333333333333333333
4                                333333333333333333333
4   2222222                                 3333333333
44   22222222222222222
44   2222222222222222222222222222
44   222222222222222222222222222222222222222
44   2222222222222222222222222222222222222222222222222
44   2222222222222222222222222222222222222222222222222
444   222222222222222222222222222222222222222222222222
444   222222222222222222222222222222222222222222222222
444   222222222222222222222222222222222222222222222222
444   22222222222222222222222222222222222222222
444   22222222222222222222222222222222222222
4444   22222222222222222222222222222222222
4444   222222222222222222222222222222222
4444   2222222222222222222222222222222
4444   222222222222222222222222222222           111111
4444   22222222222222222222222222222         111111111
44444   22222222222222222222222222         11111111111
44444   2222222222222222222222222        1111111111111
44444   2222222222222222222222222       11111111111111
44444   222222222222222222222222       111111111111111
44444   22222222222222222222222       1111111111111111
444444   222222222222222222222       11111111111111111
444444   222222222222222222222      111111111111111111
444444   22222222222222222222      1111111111111111111
444444   22222222222222222222     11111111111111111111
444444   2222222222222222222      11111111111111111111
4444444   222222222222222222     111111111111111111111
4444444   22222222222222222      111111111111111111111
4444444   22222222222222222     1111111111111111111111
4444444   22222222222222222     1111111111111111111111
4444444   2222222222222222      1111111111111111111111
44444444   222222222222222     11111111111111111111111
44444444   222222222222222     11111111111111111111111
44444444   222222222222222     11111111111111111111111
44444444   222222222222222     11111111111111111111111
44444444   222222222222222     11111111111111111111111
```

Prj0



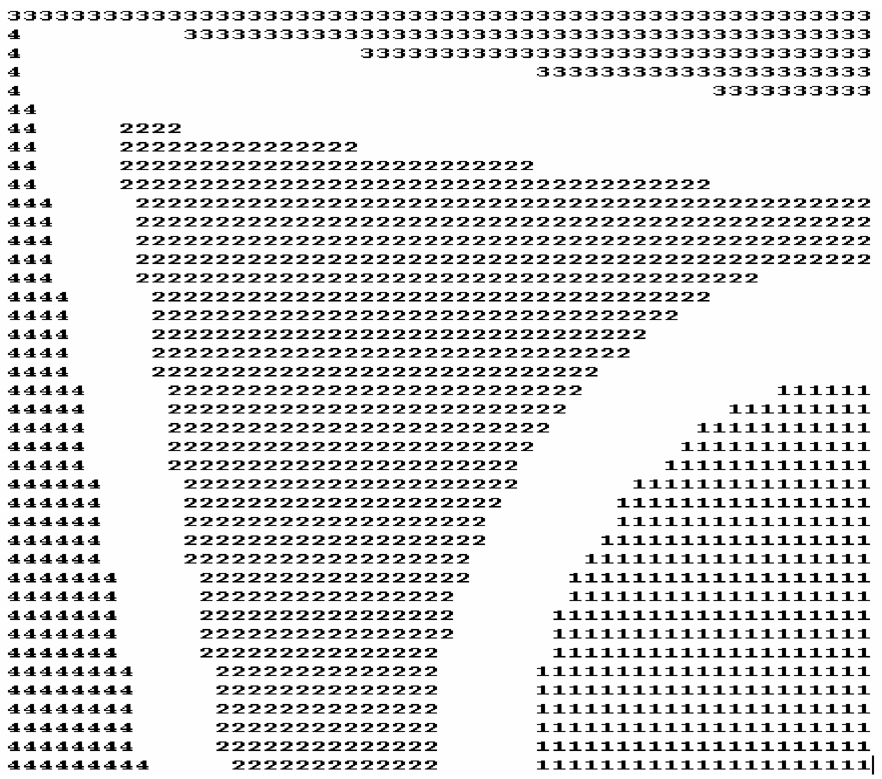
```
333333333333333333333333333333333333333333333333333333
4     333333333333333333333333333333333333333333333333
4           333333333333333333333333333333333333333333
4                 333333333333333333333333333333333333
4                       333333333333333333333333333333
4                             333333333333333333333333
4                                   333333333333333333
4                                         333333333333
4      22222                                    333333
4      22222222222
44      2222222222222222
44      2222222222222222222222
44      2222222222222222222222222222
44      2222222222222222222222222222222222
44      2222222222222222222222222222222222222222
44      222222222222222222222222222222222222222
44      222222222222222222222222222222222222
44      2222222222222222222222222222222222
44      22222222222222222222222222222222        111111
44      2222222222222222222222222222222      111111111
444      22222222222222222222222222222     11111111111
444      222222222222222222222222222     1111111111111
444      22222222222222222222222222     11111111111111
444      2222222222222222222222222     111111111111111
444      2222222222222222222222222    1111111111111111
444      222222222222222222222222    11111111111111111
444      22222222222222222222222    111111111111111111
444      2222222222222222222222    1111111111111111111
444      2222222222222222222222   11111111111111111111
444      222222222222222222222    11111111111111111111
4444      22222222222222222222   111111111111111111111
4444      2222222222222222222    111111111111111111111
4444      2222222222222222222   1111111111111111111111
4444      2222222222222222222   1111111111111111111111
4444      222222222222222222    1111111111111111111111
4444      222222222222222222   11111111111111111111111
4444      222222222222222222   11111111111111111111111
4444      222222222222222222   11111111111111111111111
4444      222222222222222222   11111111111111111111111
4444      222222222222222222   11111111111111111111111
```

Prj1



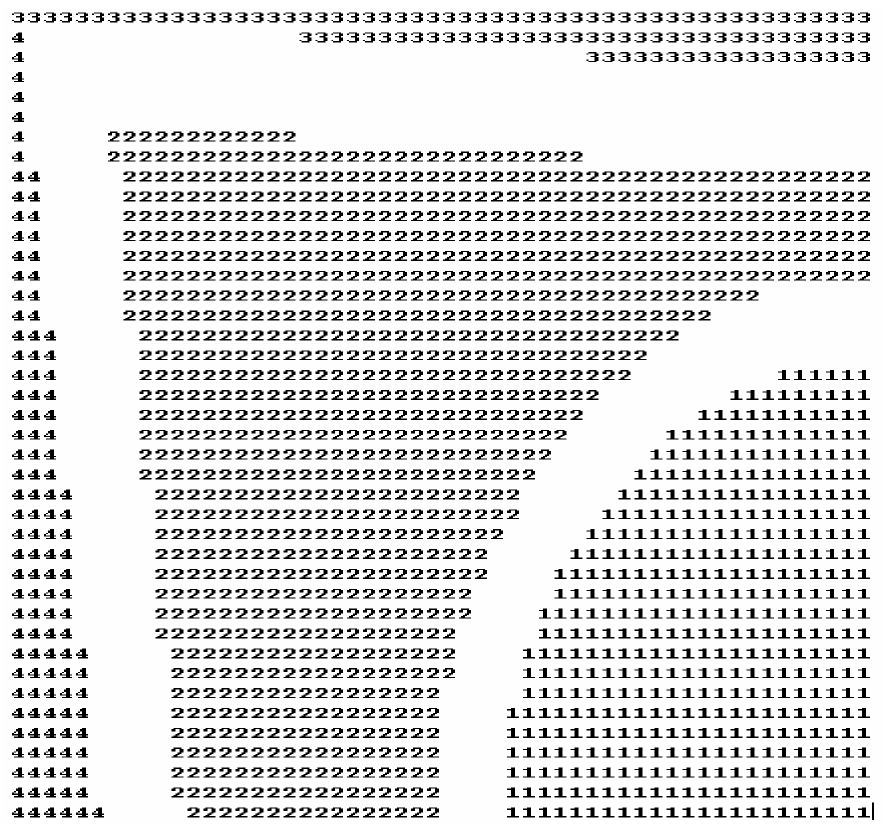
```
3333333333333333333333333333333333333333333333333333333
4                 3333333333333333333333333333333333333
4                                    333333333333333333
4
4
4    2222222222222
4    22222222222222222222222222222222
4    22222222222222222222222222222222222222222222222222
4    22222222222222222222222222222222222222222222222222
4    222222222222222222222222222222222222222222
44    22222222222222222222222222222222222222
44    222222222222222222222222222222222222
44    2222222222222222222222222222222222        1111111
44    22222222222222222222222222222222       1111111111
44    2222222222222222222222222222222     1111111111111
44    22222222222222222222222222222      11111111111111
44    2222222222222222222222222222     1111111111111111
44    222222222222222222222222222    111111111111111111
44    22222222222222222222222222    1111111111111111111
44    2222222222222222222222222    11111111111111111111
444    22222222222222222222222    111111111111111111111
444    2222222222222222222222    1111111111111111111111
444    2222222222222222222222   11111111111111111111111
444    222222222222222222222   111111111111111111111111
444    22222222222222222222    111111111111111111111111
444    22222222222222222222   1111111111111111111111111
444    2222222222222222222    1111111111111111111111111
444    2222222222222222222   11111111111111111111111111
444    222222222222222222   111111111111111111111111111
444    222222222222222222   111111111111111111111111111
4444    2222222222222222    111111111111111111111111111
4444    2222222222222222   1111111111111111111111111111
4444    2222222222222222   1111111111111111111111111111
4444    222222222222222    1111111111111111111111111111
4444    222222222222222   11111111111111111111111111111
4444    222222222222222   11111111111111111111111111111
4444    222222222222222   11111111111111111111111111111
4444    222222222222222   11111111111111111111111111111
4444    222222222222222   11111111111111111111111111111
4444    222222222222222   11111111111111111111111111111
44444    22222222222222   11111111111111111111111111111|
```

Prj2



```
333333333333333333333333333333333333333333333333333333
4          3333333333333333333333333333333333333333333
4                     33333333333333333333333333333333
4                                333333333333333333333
4                                           3333333333
44
44     2222
44     222222222222222
44     22222222222222222222222222
44     2222222222222222222222222222222222222
444     2222222222222222222222222222222222222222222222
444     2222222222222222222222222222222222222222222222
444     2222222222222222222222222222222222222222222222
444     2222222222222222222222222222222222222222222222
444     222222222222222222222222222222222222222
4444     22222222222222222222222222222222222
4444     222222222222222222222222222222222
4444     2222222222222222222222222222222
4444     222222222222222222222222222222
4444     2222222222222222222222222222
44444     22222222222222222222222222            111111
44444     2222222222222222222222222          111111111
44444     222222222222222222222222         11111111111
44444     22222222222222222222222         111111111111
44444     2222222222222222222222         1111111111111
444444     222222222222222222222       111111111111111
444444     22222222222222222222       1111111111111111
444444     2222222222222222222        1111111111111111
444444     2222222222222222222       11111111111111111
444444     222222222222222222       111111111111111111
4444444     22222222222222222      1111111111111111111
4444444     2222222222222222       1111111111111111111
4444444     2222222222222222      11111111111111111111
4444444     2222222222222222      11111111111111111111
4444444     222222222222222       11111111111111111111
44444444     22222222222222      111111111111111111111
44444444     22222222222222      111111111111111111111
44444444     22222222222222      111111111111111111111
44444444     22222222222222      111111111111111111111
44444444     22222222222222      111111111111111111111
444444444     2222222222222      111111111111111111111|
```

Prj3



```
333333333333333333333333333333333333333333333333333333
4                 333333333333333333333333333333333333
4                                   333333333333333333
4
4
4
4     222222222222
4     222222222222222222222222222222
44     22222222222222222222222222222222222222222222222
44     22222222222222222222222222222222222222222222222
44     22222222222222222222222222222222222222222222222
44     22222222222222222222222222222222222222222222222
44     22222222222222222222222222222222222222222222222
44     22222222222222222222222222222222222222222222222
44     2222222222222222222222222222222222222222
44     2222222222222222222222222222222222222
444     2222222222222222222222222222222222
444     22222222222222222222222222222222
444     2222222222222222222222222222222         111111
444     22222222222222222222222222222        111111111
444     2222222222222222222222222222       11111111111
444     222222222222222222222222222      1111111111111
444     22222222222222222222222222      11111111111111
444     2222222222222222222222222      111111111111111
4444     22222222222222222222222      1111111111111111
4444     22222222222222222222222     11111111111111111
4444     2222222222222222222222     111111111111111111
4444     222222222222222222222     1111111111111111111
4444     222222222222222222222    11111111111111111111
4444     22222222222222222222     11111111111111111111
4444     22222222222222222222    111111111111111111111
4444     2222222222222222222     111111111111111111111
44444     222222222222222222    1111111111111111111111
44444     222222222222222222    1111111111111111111111
44444     22222222222222222     1111111111111111111111
44444     22222222222222222    11111111111111111111111
44444     22222222222222222    11111111111111111111111
44444     22222222222222222    11111111111111111111111
44444     22222222222222222    11111111111111111111111
44444     22222222222222222    11111111111111111111111
444444     2222222222222222    11111111111111111111111|
```

Prj4

C-3 SOM Best generated maps

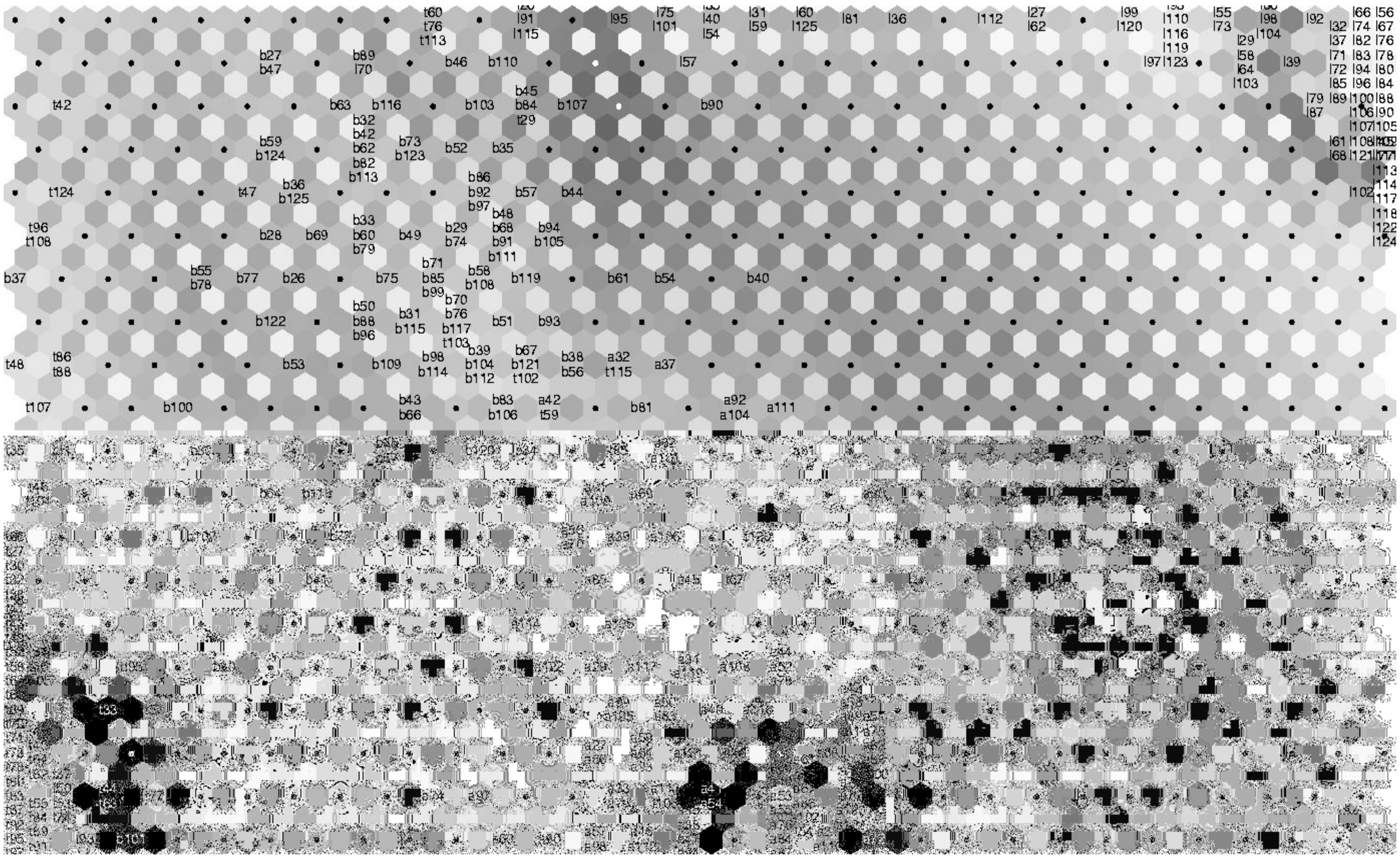


Figure C3.1 The 1st SOM map

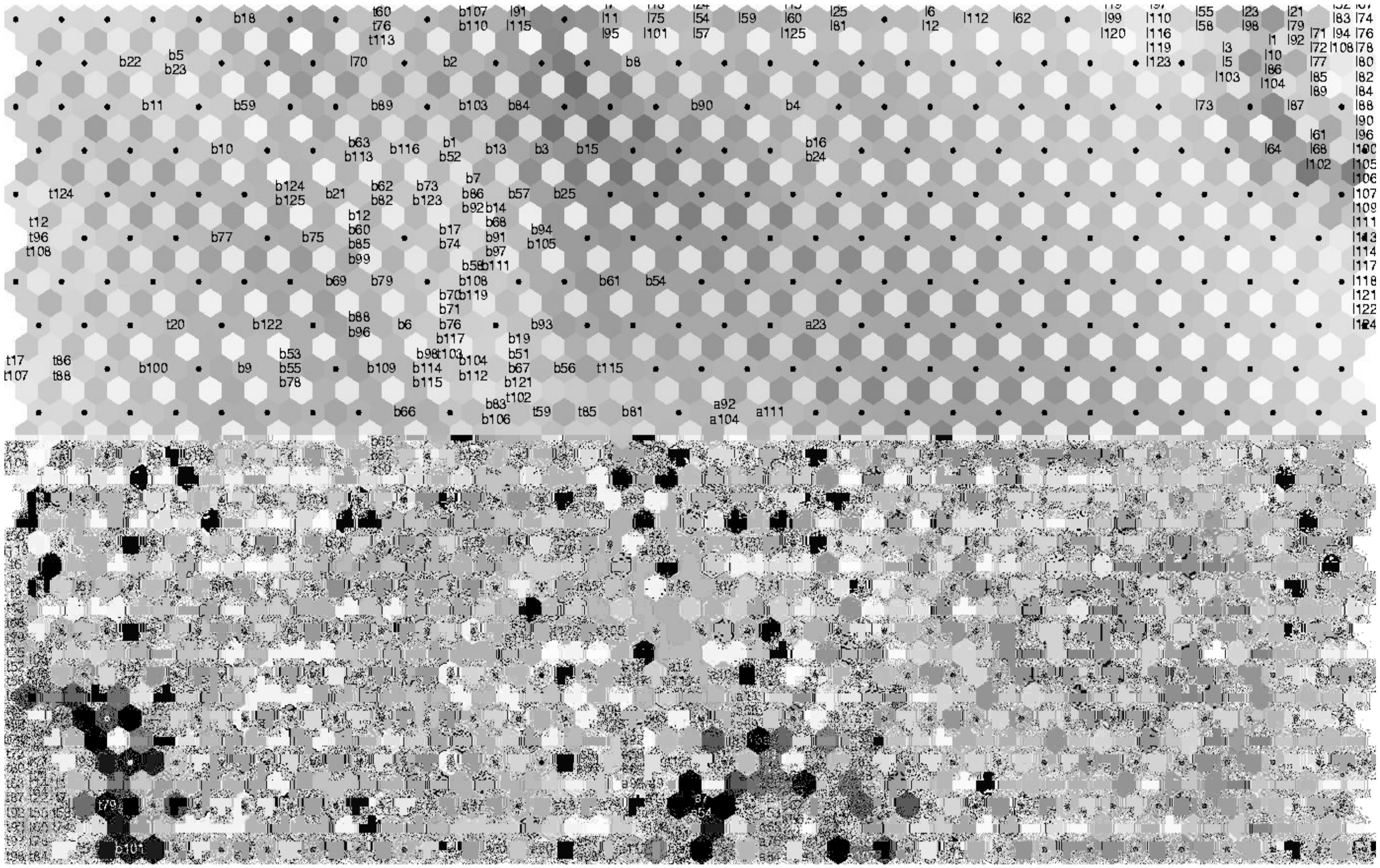


Figure C3.2 The 2nd SOM map

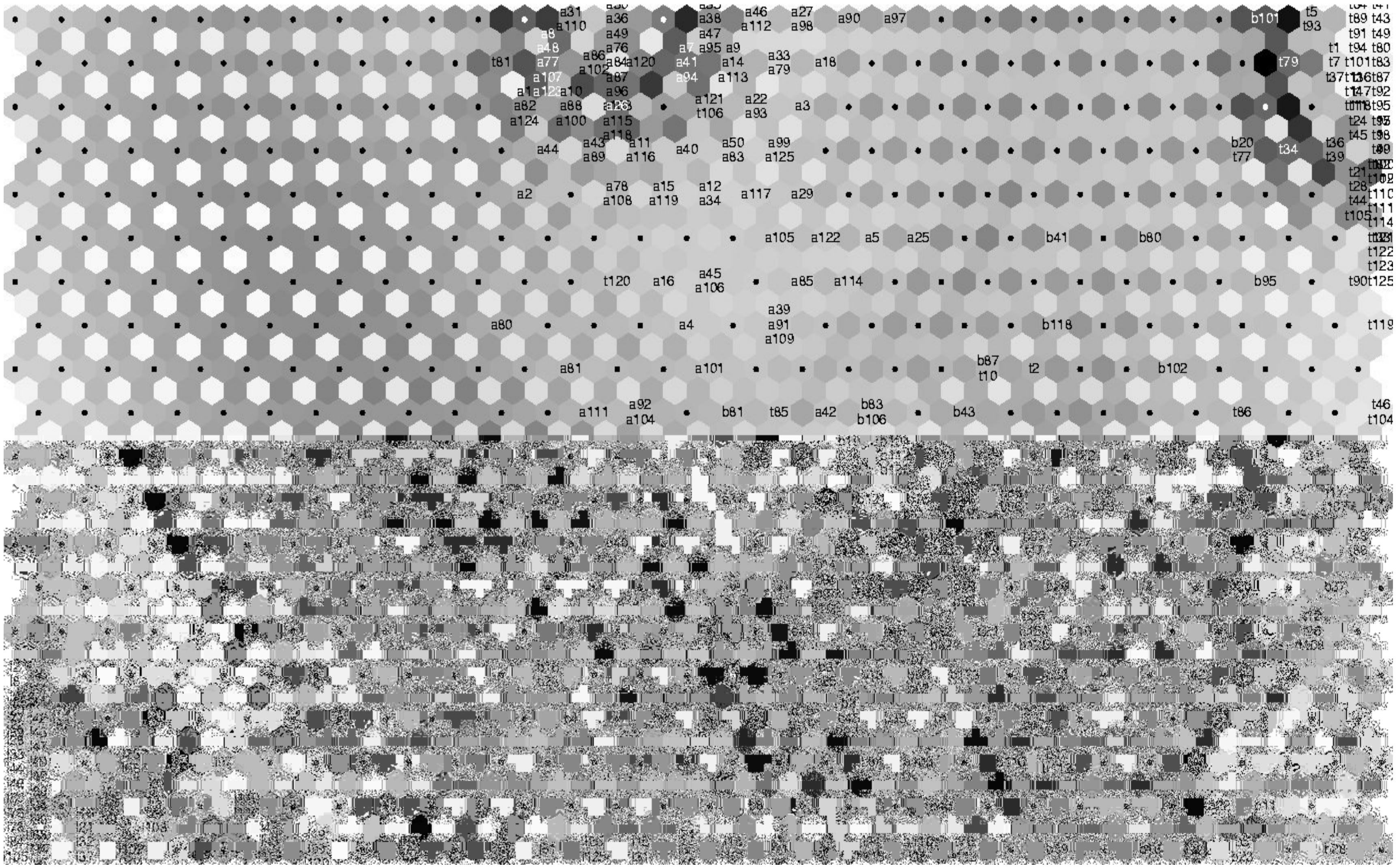


Figure C3.3 The 3rd SOM map

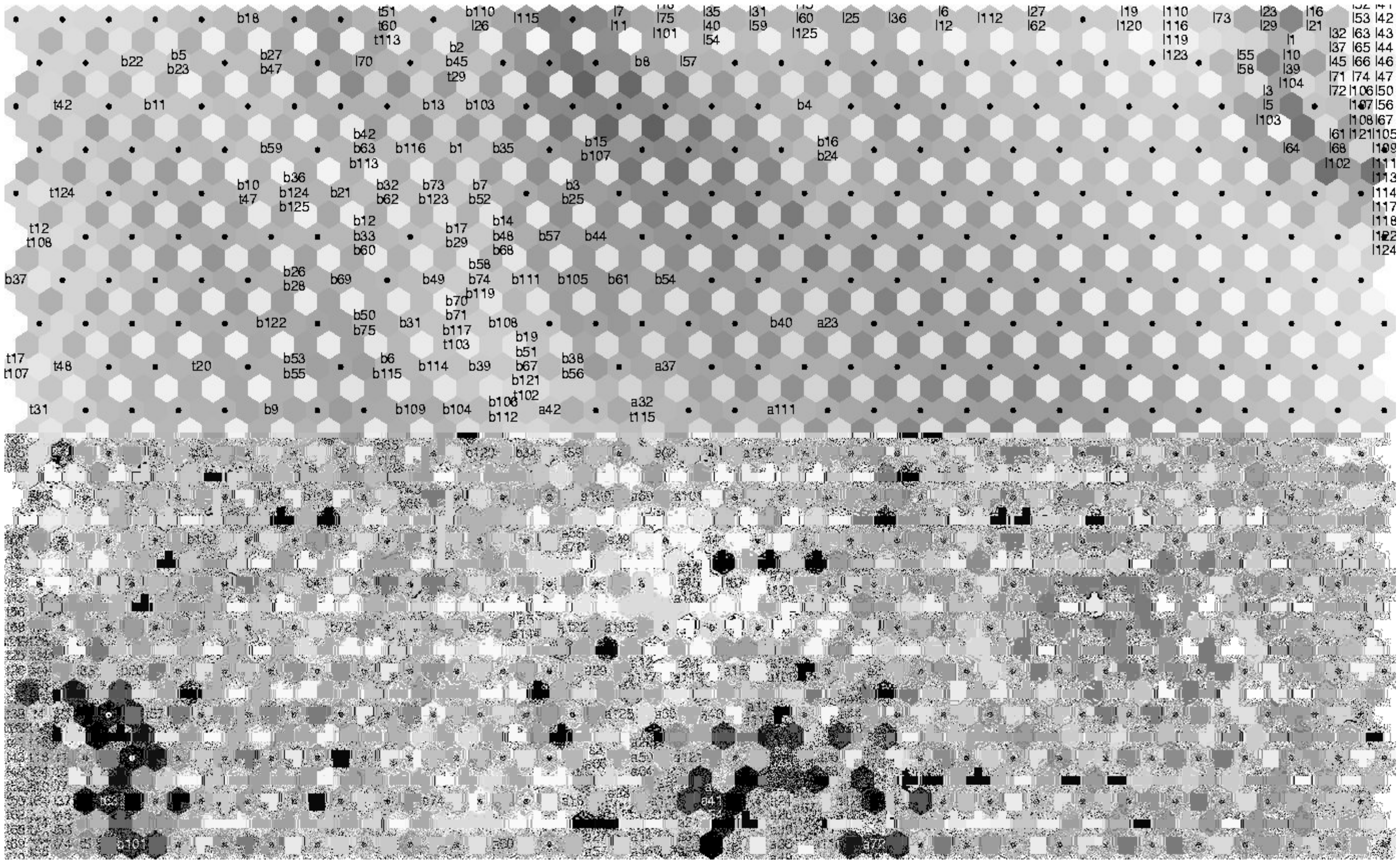


Figure C3.4 The 4th SOM map

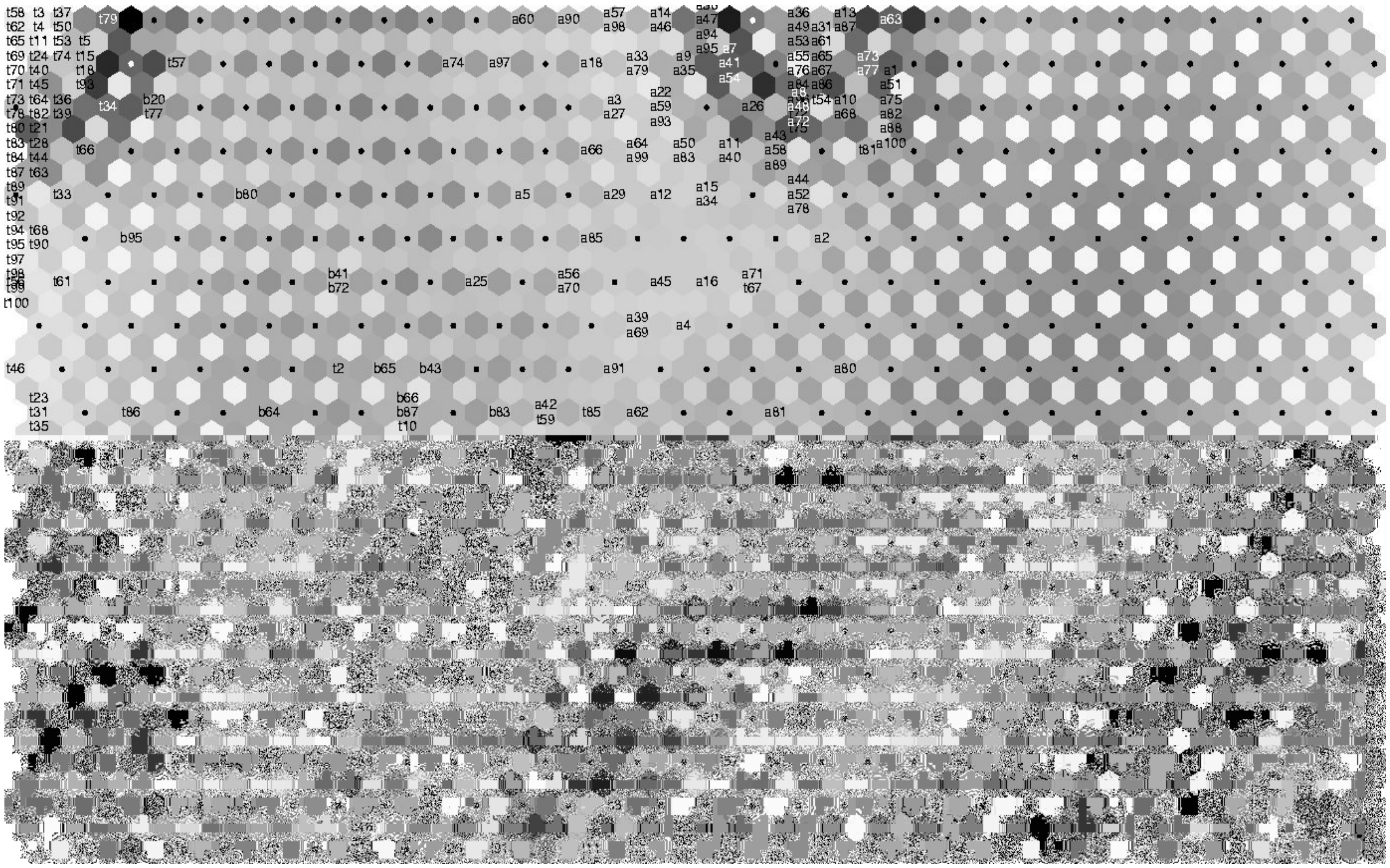


Figure C3.5 The 5th SOM map

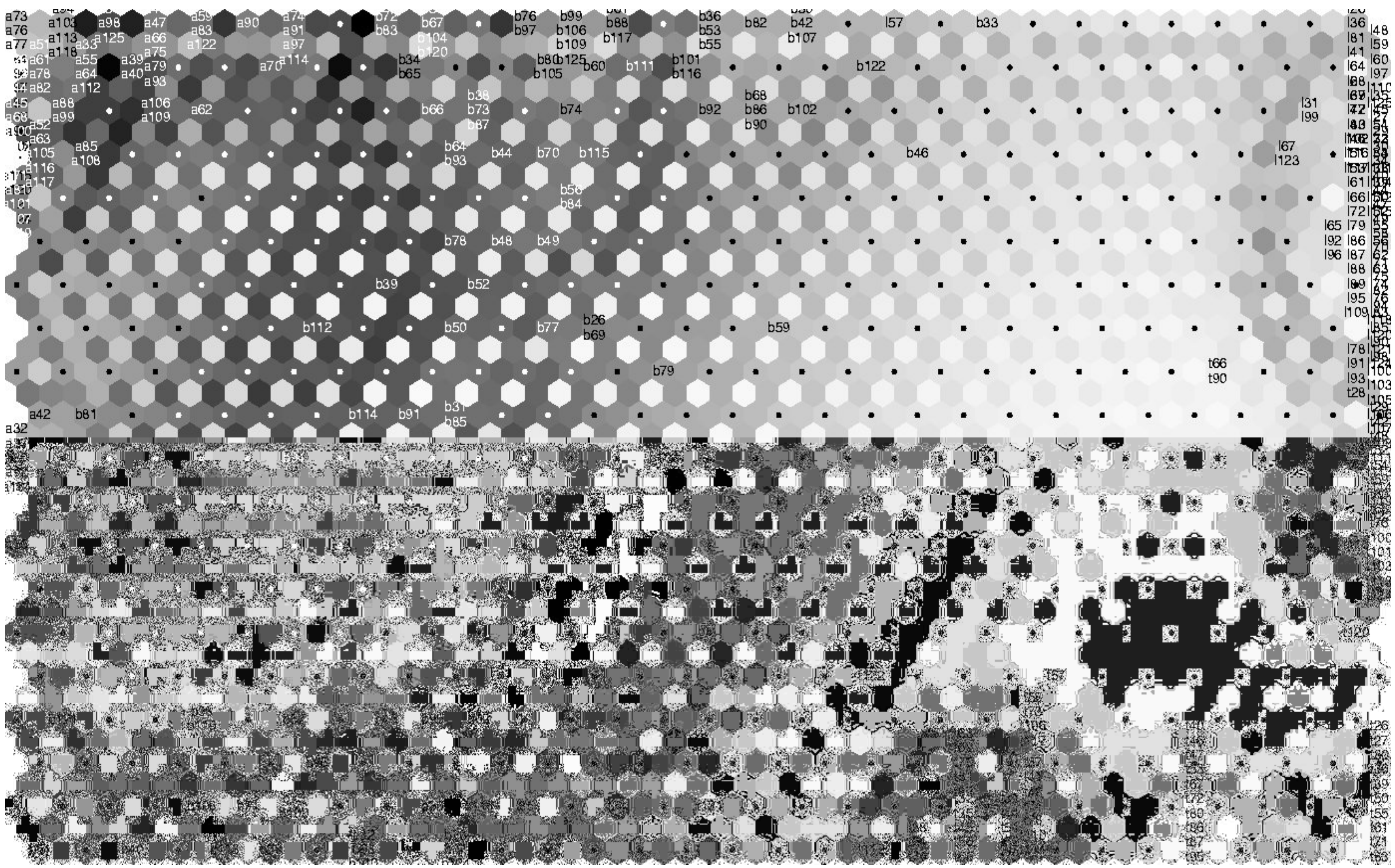


Figure C3.6 The 6th SOM map

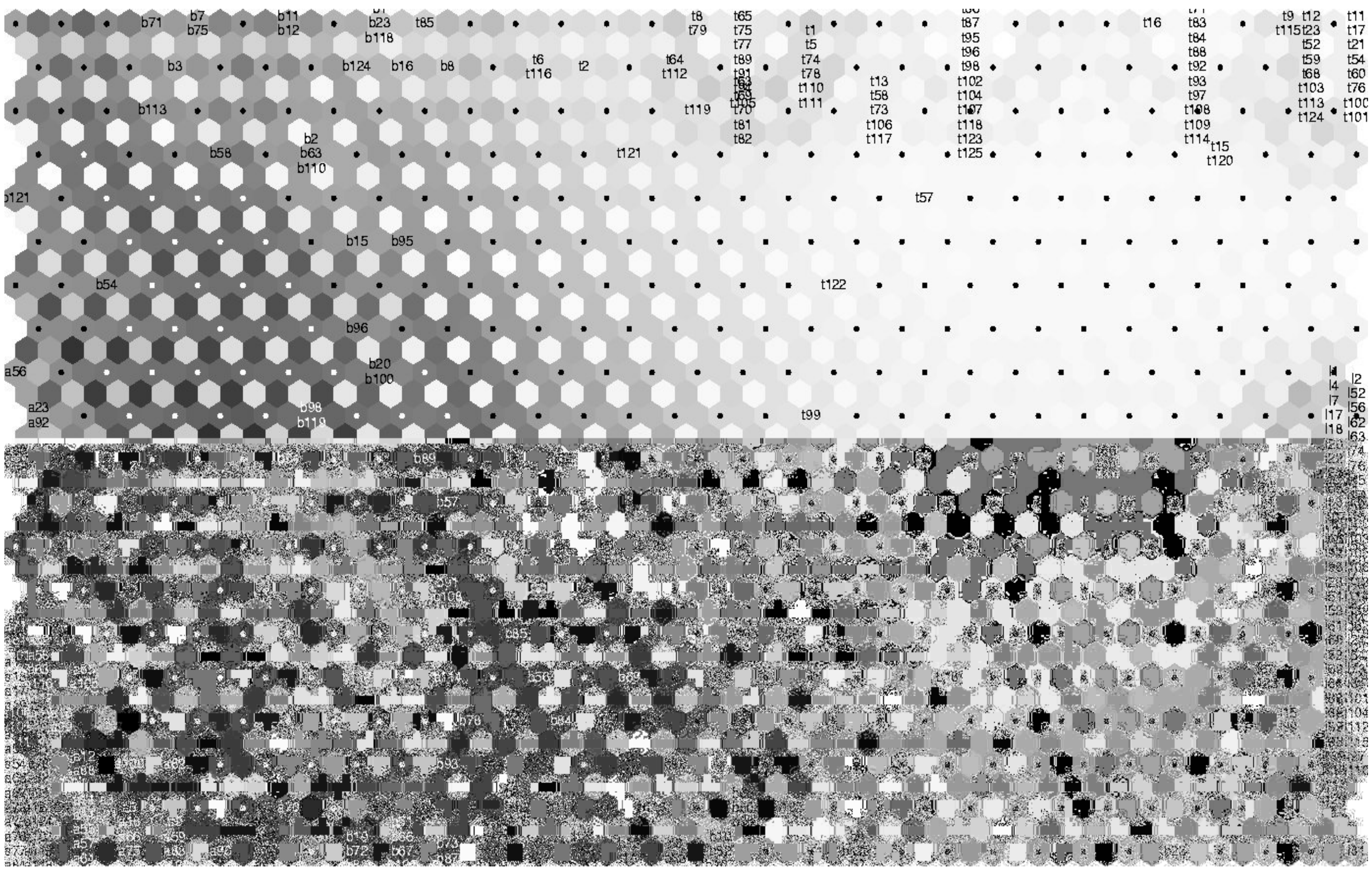


Figure C3.7 The 7th SOM map

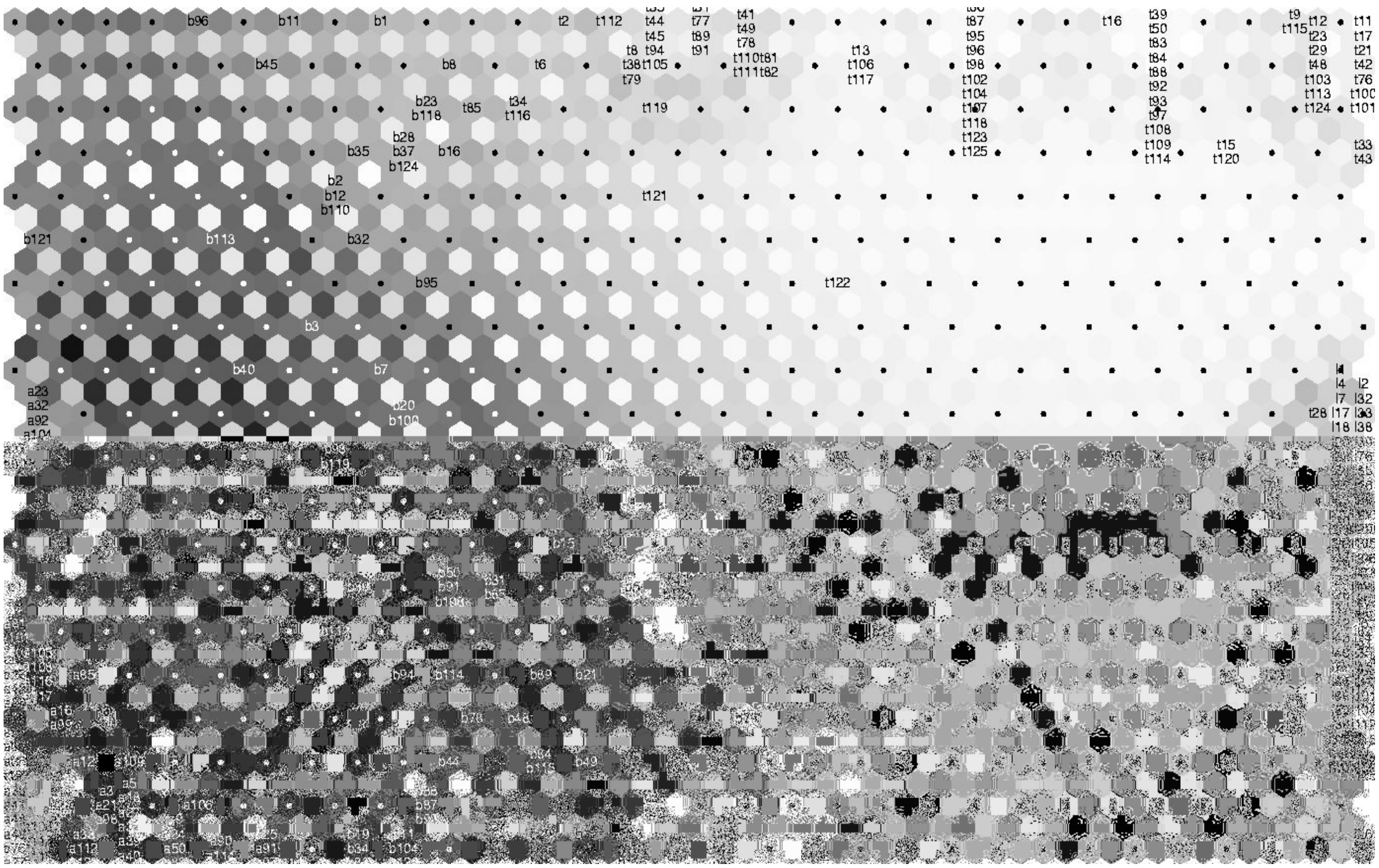


Figure C3.8 The 8th SOM map

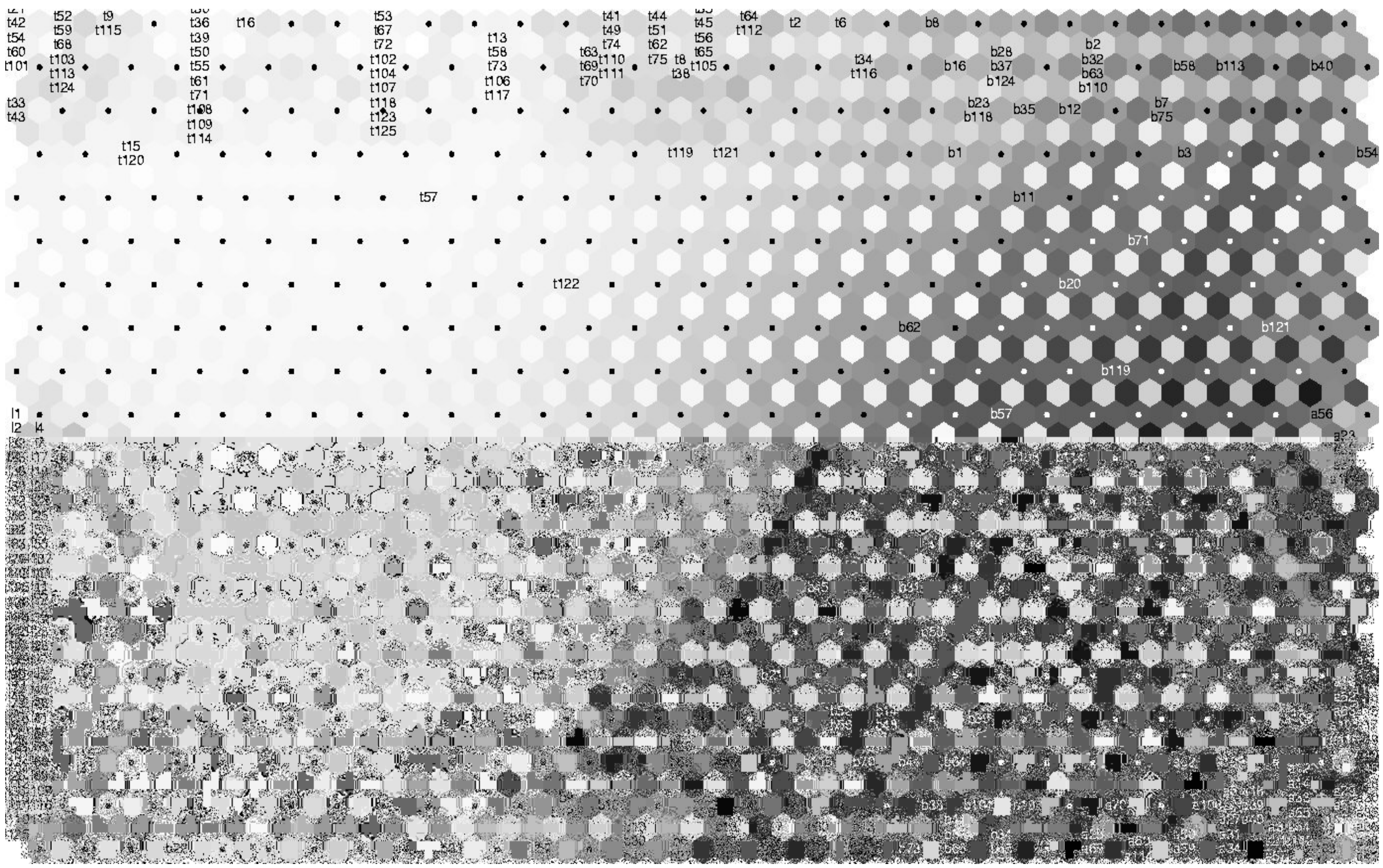


Figure C3.9 The 9th SOM map

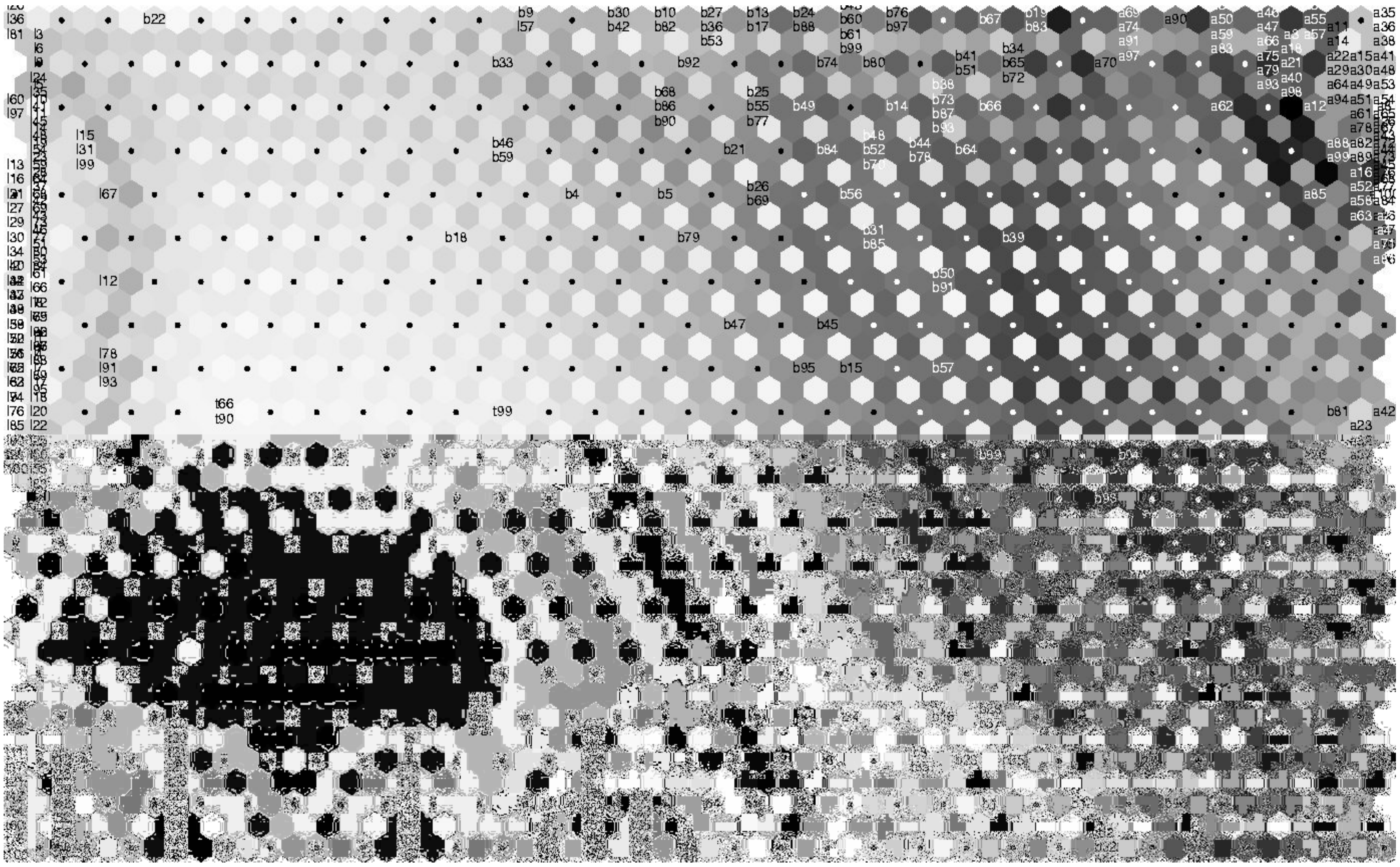


Figure C3.10 The 10th SOM map

Appendix D Testing Results

D-1 MLP Results

Legend Applies to all tables in the results appendix

▨ Alligator Cracks (**a**) ▦ Block Cracks (**b**) ▥ Longitudinal Cracks (**l**) ▤ Transverse Cracks (**t**)

C/T = Number of correctly classified images / Total number of tested images.

Table D-1.1 Results of the 1st MLP network

MLP_1	hghTrain0.dat	400 samples
	hghTest0.dat	100 samples

	C/T	a	b	l	t	total
A	0.96	24	1	0	0	25
B	0.84	0	21	4	0	25
L	1	0	0	25	0	25
T	0.88	0	3	0	22	25
Over all	**0.92**					100

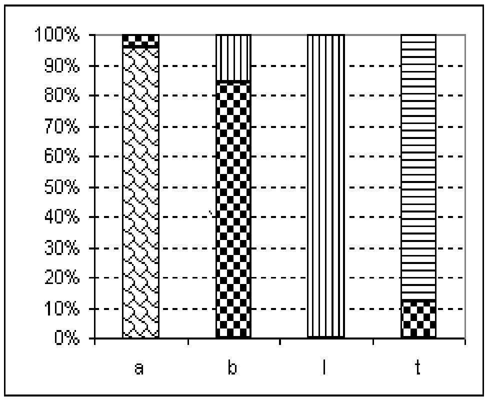


Table D-1.2 Results of the 2nd MLP network

MLP_2	hghTrain1.dat	400 samples
	hghTest1.dat	100 samples

	C/T	a	b	l	t	total
A	0.96	24	1	0	0	25
B	0.92	0	23	1	1	25
L	1	0	0	25	0	25
T	0.88	0	3	0	22	25
Over all	**0.94**					100

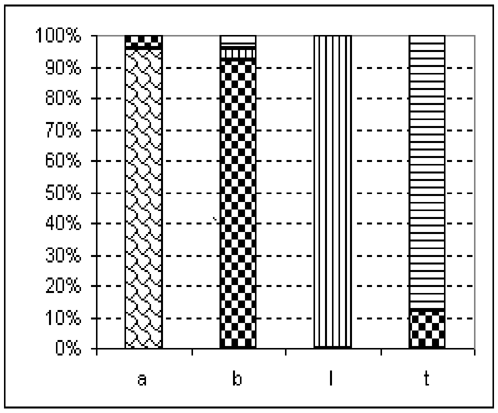


Table D-1.3 Results of the 3rd MLP network

MLP_3	hghTrain2.dat	400 samples
	hghTest2.dat	100 samples

	C/T	a	B	l	t	total
a	1	25	0	0	0	25
b	1	0	25	0	0	25
l	0.96	0	1	24	0	25
t	0.76	5	1	0	19	25
Over all	0.93					100

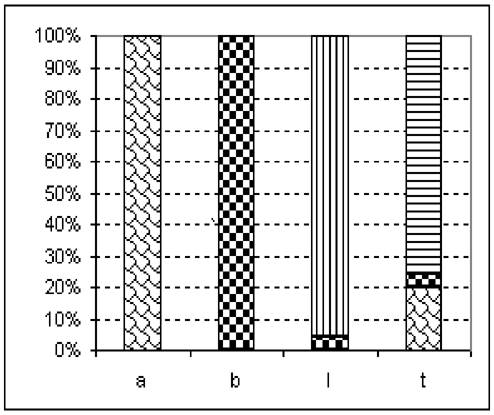


Table D-1.4 Results of the 4th MLP network

MLP_4	hghTrain3.dat	400 samples
	hghTest3.dat	100 samples

	C/T	a	B	l	t	total
a	1	25	0	0	0	25
b	0.92	1	23	0	1	25
l	1	0	0	25	0	25
t	0.92	2	0	0	23	25
Over all	0.96					100

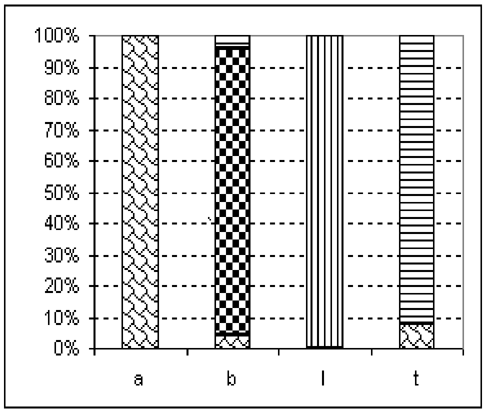


Table D-1.5 Results of the 5th MLP network

MLP_5	hghTrain4.dat	400 samples
	hghTest4.dat	100 samples

	C/T	a	b	l	t	total
a	1	25	0	0	0	25
b	0.96	0	24	0	1	25
l	0.96	0	1	24	0	25
t	0.8	3	2	0	20	25
Over all	0.93					100

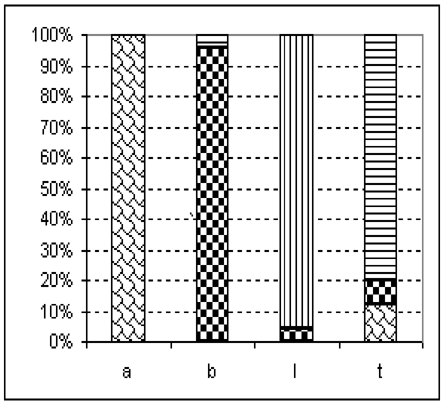


Table D-1.6 Results of the 6th MLP network

MLP_6	prjTrain0.dat 400 samples prjTest0.dat 100 samples					
	C/T	a	b	l	t	total
a	1	25	0	0	0	25
b	1	0	25	0	0	25
l	0.96	0	0	24	1	25
t	1	0	0	0	25	25
Over all	0.99					100

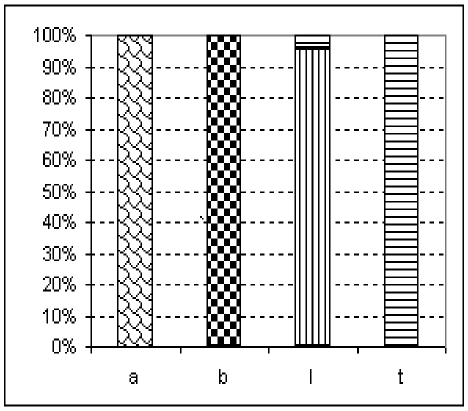


Table D-1.7 Results of the 7th MLP network

MLP_7	prjTrain1.dat 400 samples prjTest1.dat 100 samples					
	C/T	a	b	l	t	total
a	1	25	0	0	0	25
b	0.92	0	23	0	2	25
l	1	0	0	25	0	25
T	0.96	0	0	1	24	25
Over all	0.97					100

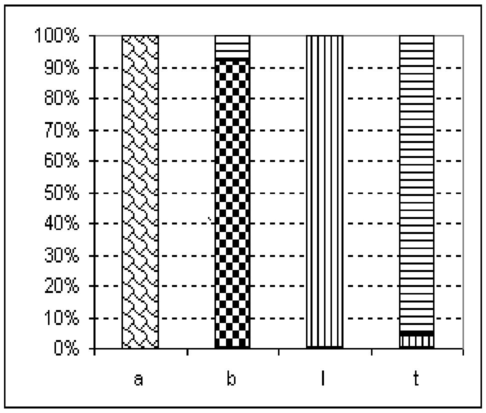


Table D-1.8 Results of the 8th MLP network

MLP_8	prjTrain2.dat 400 samples prjTest2.dat 100 samples					
	C/T	a	b	l	t	total
a	1	25	0	0	0	25
b	1	0	25	0	0	25
l	0.96	0	1	24	0	25
t	1	0	0	0	25	25
Over all	0.99					100

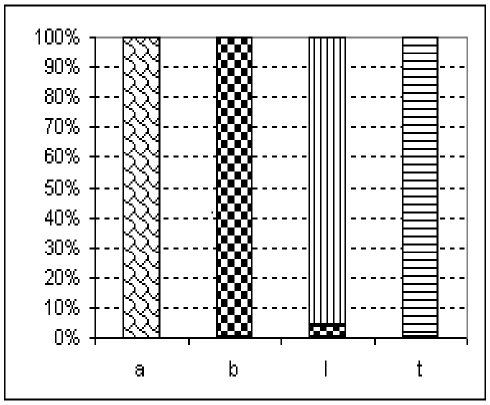


Table D-1.9 Results of the 9th MLP network

MLP_9		
	prjTrain3.dat	400 samples
	prjTest3.dat	100 samples

	C/T	a	b	l	t	total
a	1	25	0	0	0	25
b	0.96	1	24	0	0	25
l	1	0	0	25	0	25
t	0.96	0	1	0	24	25
Over all	**0.98**					100

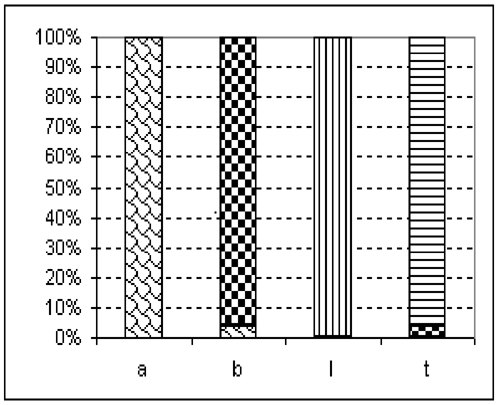
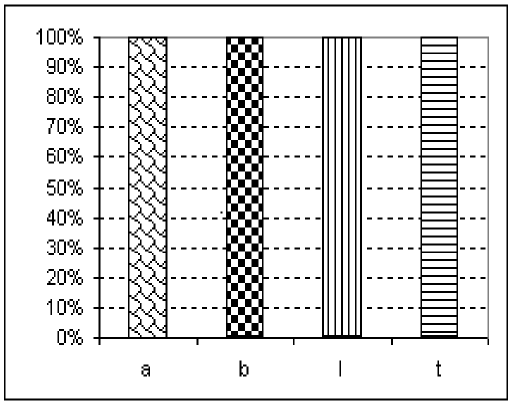


Table D-1.10 Results of the 10th MLP network

MLP_10		
	prjTrain4.dat	400 samples
	prjTest4.dat	100 samples

	C/T	a	B	l	t	total
a	1	25	0	0	0	25
b	1	0	25	0	0	25
L	1	0	0	25	0	25
T	1	0	0	0	25	25
Over all	**1**					100



D-2 SOM Results

Table D-2.1 Results of the 1st SOM

SOM_1		
	hghTrain0.dat	400 samples
	hghTest0.dat	100 samples

	C/T	a	B	L	t	total
A	0.68	17	0	0	8	25
B	0.72	0	18	3	4	25
L	1	0	0	25	0	25
T	0.88	0	3	0	22	25
Over all	**0.82**					100

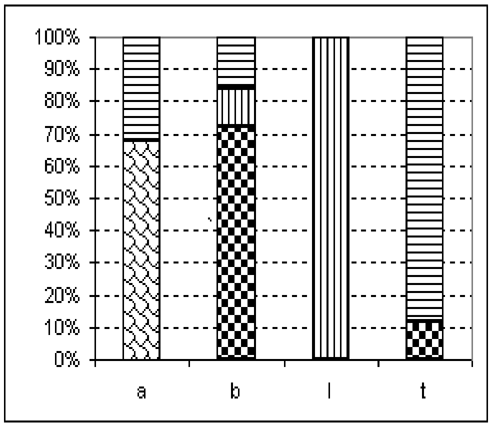


Table D-2.2 Results of the 2nd SOM

SOM_2	hghTrain1.dat	400 samples
	hghTest1.dat	100 samples

	C/T	a	b	l	t	total
a	0.68	17	1	0	7	25
b	0.88	1	22	0	2	25
l	0.96	0	1	24	0	25
t	0.88	0	3	0	22	25
Over all	**0.85**					100

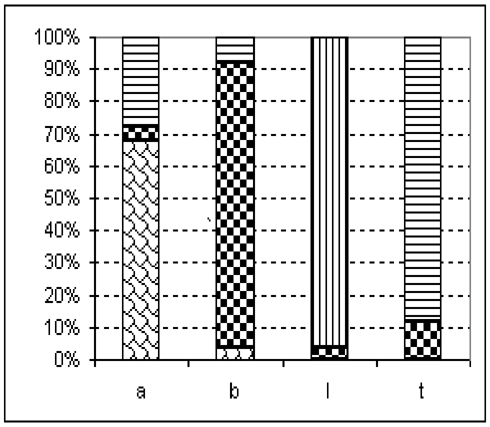


Table D-2.3 Results of the 3rd SOM

SOM_3	hghTrain2.dat	400 samples
	hghTest2.dat	100 samples

	C/T	a	b	l	t	total
a	0.96	24	0	0	1	25
b	0.8	0	20	0	5	25
l	0.92	0	2	23	0	25
t	0.8	5	0	0	20	25
Over all	**0.87**					100

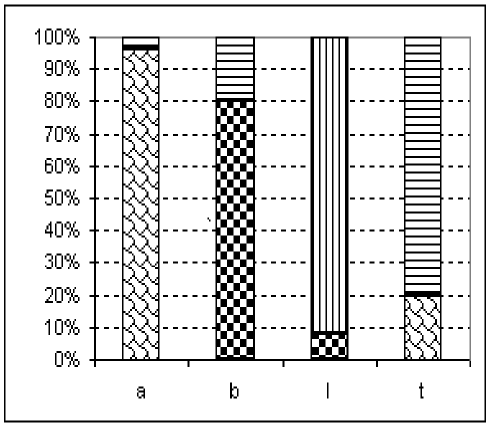


Table D-2.4 Results of the 4th SOM

SOM_4	hghTrain3.dat	400 samples
	hghTest3.dat	100 samples

	C/T	a	b	l	t	total
a	0.76	19	1	0	5	25
b	0.8	0	20	0	5	25
l	1	0	0	25	0	25
t	0.88	2	1	0	22	25
Over all	**0.86**					100

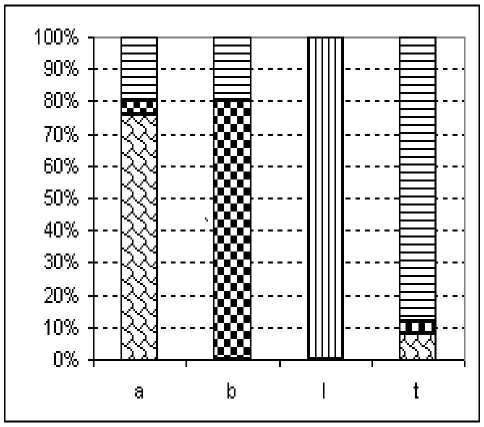


Table D-2.5 Results of the 5th SOM

SOM_5	hghTrain4.dat	400 samples
	hghTest4.dat	100 samples

	C/T	a	b	l	t	total
a	0.88	22	0	0	3	25
b	0.96	0	24	0	1	25
l	1	0	0	25	0	25
t	0.76	2	4	0	19	25
Over all	**0.9**					100

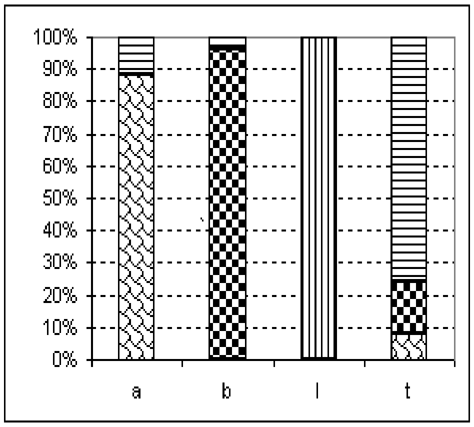


Table D-2.6 Results of the 6th SOM

SOM_6	prjTrain0.dat	400 samples
	prjTest0.dat	100 samples

	C/T	a	b	l	t	total
a	1	25	0	0	0	25
b	0.92	0	23	1	1	25
l	1	0	0	25	0	25
t	1	0	0	0	25	25
Over all	**0.98**					100

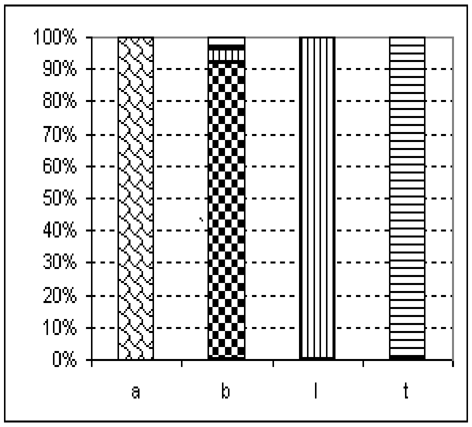


Table D-2.7 Results of the 7th SOM

SOM_7	prjTrain1.dat	400 samples
	prjTest1.dat	100 samples

	C/T	a	b	l	t	total
a	1	25	0	0	0	25
b	1	0	25	0	0	25
l	1	0	0	25	0	25
t	0.96	0	0	1	24	25
Over all	**0.99**					100

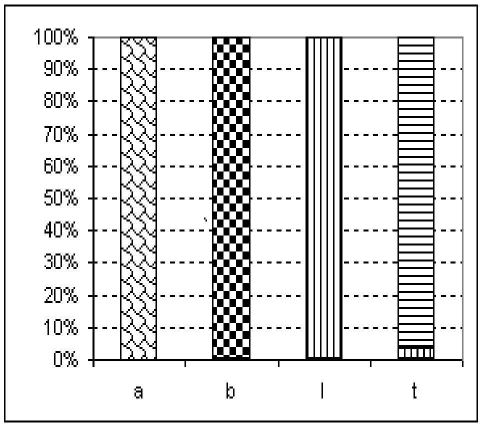


Table D-2.8 Results of the 8th SOM

SOM_8	prjTrain2.dat	400 samples
	prjTest2.dat	100 samples

	C/T	a	B	l	t	total
a	1	25	0	0	0	25
b	1	0	25	0	0	25
l	0.96	0	1	24	0	25
t	1	0	0	0	25	25
Over all	0.99					100

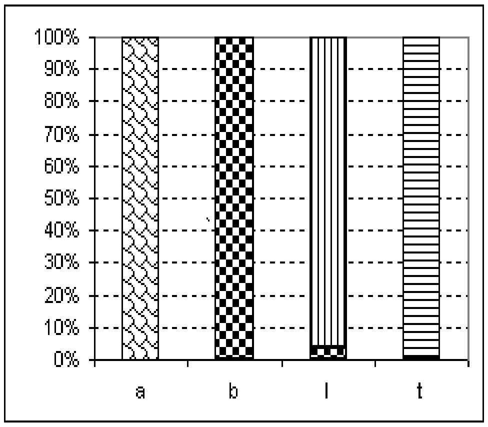


Table D-2.9 Results of the 9th SOM

SOM_9	prjTrain3.dat	400 samples
	prjTest3.dat	100 samples

	C/T	a	B	l	t	total
a	1	25	0	0	0	25
b	0.96	1	24	0	0	25
l	1	0	0	25	0	25
t	0.92	0	2	0	23	25
Over all	0.97					100

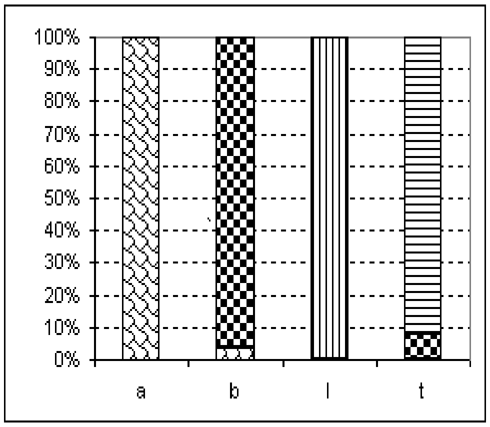
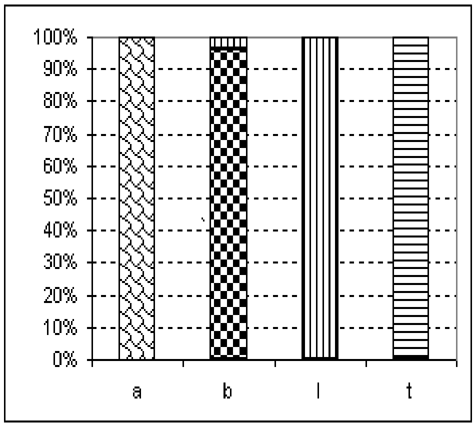


Table D-2.10 Results of the 10th SOM

SOM_10	prjTrain4.dat	400 samples
	prjTest4.dat	100 samples

	C/T	a	B	l	t	total
a	1	25	0	0	0	25
b	0.96	0	24	1	0	25
l	1	0	0	25	0	25
t	1	0	0	0	25	25
Over all	0.99					100



D3 GA Results

Table D-3.1 Results of the 1st GA Matrix

GA_1		
	hghTrain0.dat	400 samples
	hghTest0.dat	100 samples

	C/T	a	b	l	t	total
a	0.96	24	1	0	0	25
b	0.96	0	24	0	1	25
l	0.84	0	4	21	0	25
t	0.88	0	3	0	22	25
Over all	0.91					100

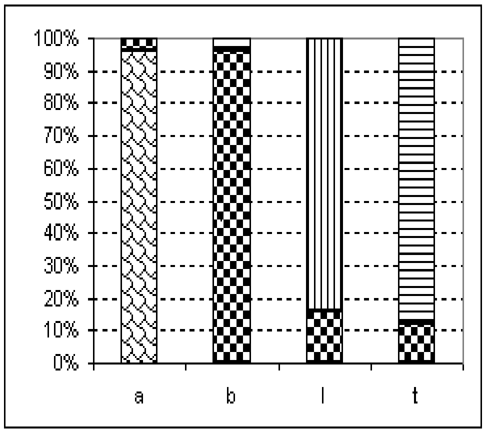


Table D-3.2 Results of the 2nd GA Matrix

GA_2		
	hghTrain1.dat	400 samples
	hghTest1.dat	100 samples

	C/T	a	B	l	t	total
a	0.88	22	3	0	0	25
b	0.92	0	23	0	2	25
l	0.84	0	4	21	0	25
t	0.88	0	3	0	22	25
Over all	0.88					100

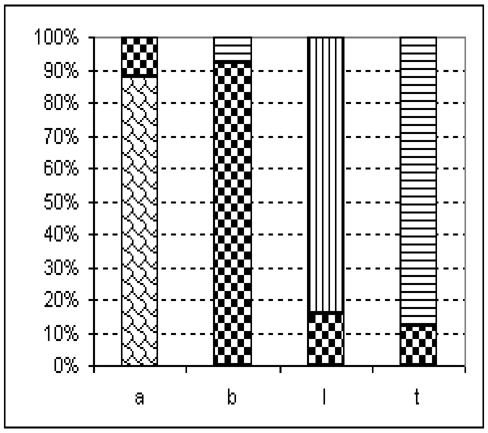


Table D-3.3 Results of the 3rd A Matrix

GA_3		
	hghTrain2.dat	400 samples
	hghTest2.dat	100 samples

	C/T	a	B	l	t	total
a	1	25	0	0	0	25
b	0.96	0	24	0	1	25
l	0.8	0	5	20	0	25
t	0.72	4	3	0	18	25
Over all	0.87					100

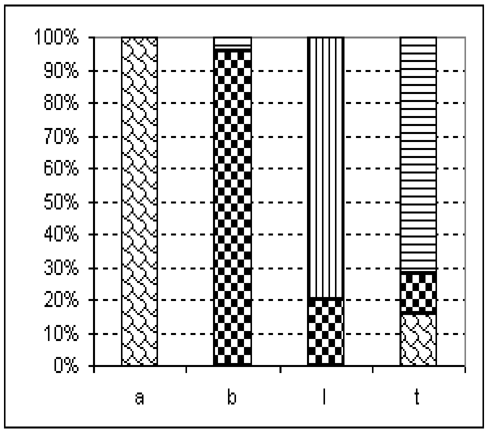


Table D-3.4 Results of the 4th GA Matrix

GA_4	hghTrain3.dat	400 samples
	hghTest3.dat	100 samples

	C/T	**a**	**b**	**l**	**t**	**total**
a	0.96	24	1	0	0	25
b	0.88	0	22	0	3	25
l	0.92	0	2	23	0	25
t	0.88	1	2	0	22	25
Over all	**0.91**					100

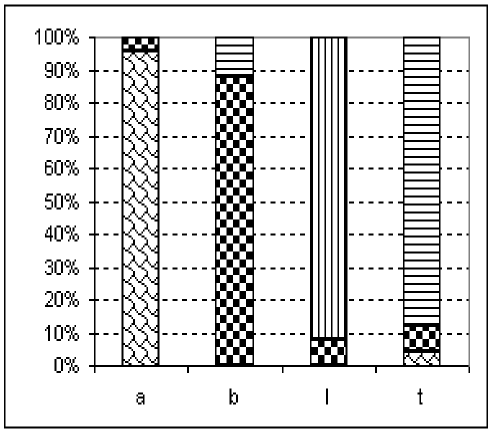


Table D-3.5 Results of the 5th GA Matrix

GA_5	hghTrain4.dat	400 samples
	hghTest4.dat	100 samples

	C/T	**a**	**b**	**l**	**t**	**total**
a	1	25	0	0	0	25
b	0.92	0	23	0	2	25
l	0.92	0	2	23	0	25
t	0.72	2	5	0	18	25
Over all	**0.89**					100

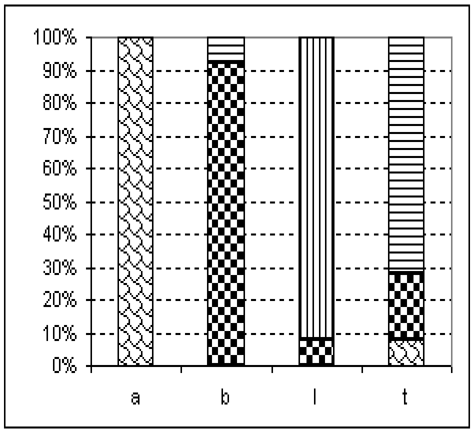


Table D-3.6 Results of the 6th GA Matrix

GA_6	prjTrain0.dat	400 samples
	prjTest0.dat	100 samples

	C/T	**a**	**B**	**l**	**t**	**total**
a	0.96	24	1	0	0	25
b	1	0	25	0	0	25
l	1	0	0	25	0	25
t	1	0	0	0	25	25
Over all	**0.99**					100

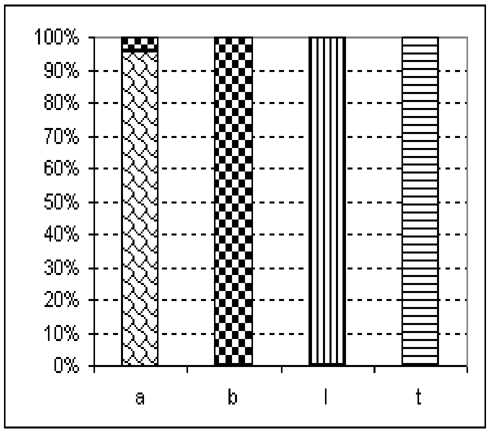


Table D-3.7 Results of the 7th GA Matrix

GA_7		
	prjTrain1.dat	400 samples
	prjTest1.dat	100 samples

	C/T	a	B	l	t	total
a	0.96	24	1	0	0	25
b	1	0	25	0	0	25
l	1	0	0	25	0	25
t	0.96	0	0	1	24	25
Over all	0.98					100

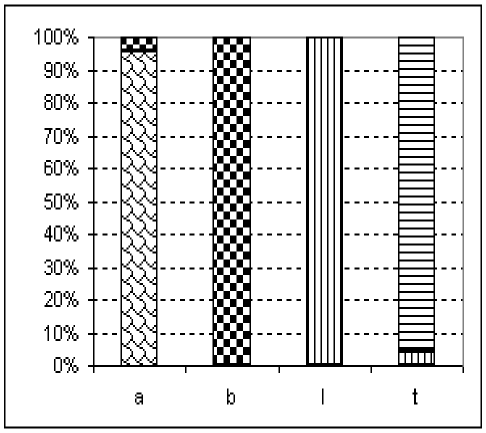


Table D-3.8 Results of the 8th GA Matrix

GA_8		
	prjTrain2.dat	400 samples
	prjTest2.dat	100 samples

	C/T	a	B	l	t	total
a	1	25	0	0	0	25
b	0.92	2	23	0	0	25
l	1	0	0	25	0	25
t	1	0	0	0	25	25
Over all	0.98					100

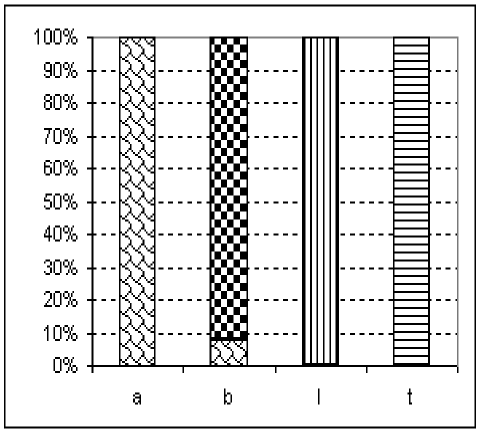


Table D-3.9 Results of the 9th GA Matrix

GA_9		
	prjTrain3.dat	400 samples
	prjTest3.dat	100 samples

	C/T	a	b	l	t	total
a	0.96	24	1	0	0	25
b	0.96	1	24	0	0	25
l	1	0	0	25	0	25
t	0.96	0	1	0	24	25
Over all	0.97					100

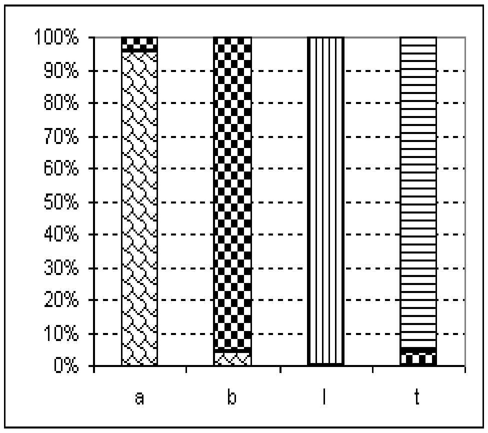
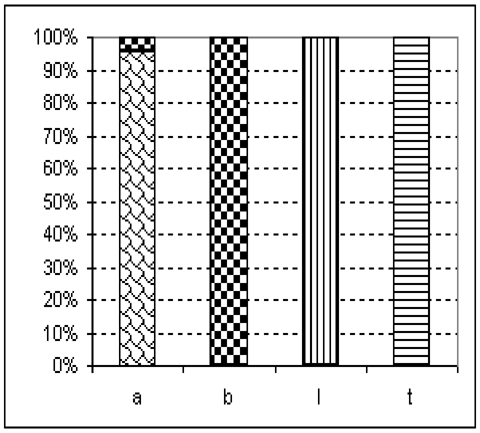


Table D-3.10 Results of the 10th GA Matrix

GA_10	prjTrain4.dat	400 samples
	prjTest4.dat	100 samples

	C/T	a	b	l	t	total
a	0.96	24	1	0	0	25
b	1	0	25	0	0	25
l	1	0	0	25	0	25
t	1	0	0	0	25	25
Over all	0.99					100



D4 Aggregate Results

Table D4.1 Aggregate results of MLP/Hough method

MLP_Hough

	C/T	a	b	l	t	total
A	0.984	**123**	2	0	0	125
B	0.928	1	**116**	5	3	125
L	0.984	0	2	**123**	0	125
T	0.848	10	9	0	**106**	125
Over all	0.936					500

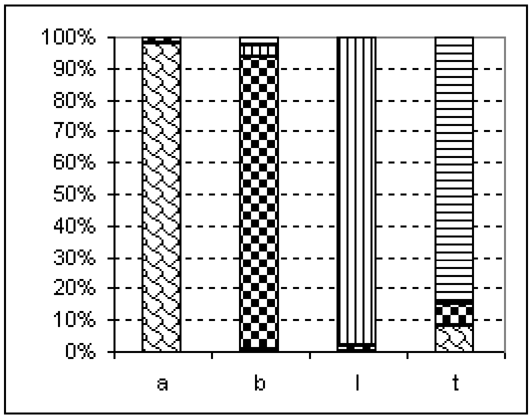


Table D4.2 Aggregate results of MLP/Projection method

MLP_Projection

	C/T	a	b	l	t	total
A	1	**125**	0	0	0	125
B	0.976	1	**122**	0	2	125
L	0.984	0	1	**123**	1	125
T	0.984	0	1	1	**123**	125
Over all	0.986					500

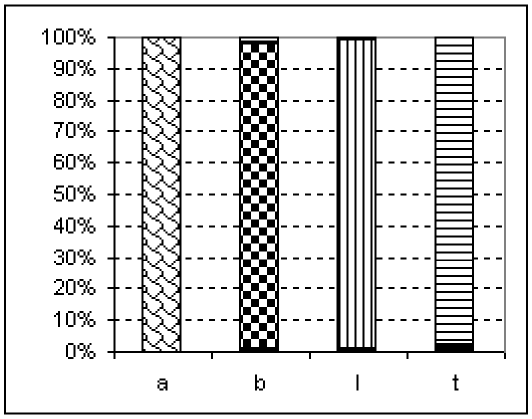


Table D4.3 Aggregate results of GA/Hough method

GA_Hough

	C/T	a	b	l	t	total
A	0.96	**120**	5	0	0	125
B	0.928	0	**116**	0	9	125
L	0.864	0	17	**108**	0	125
T	0.816	7	16	0	**102**	125
Over all	**0.892**					**500**

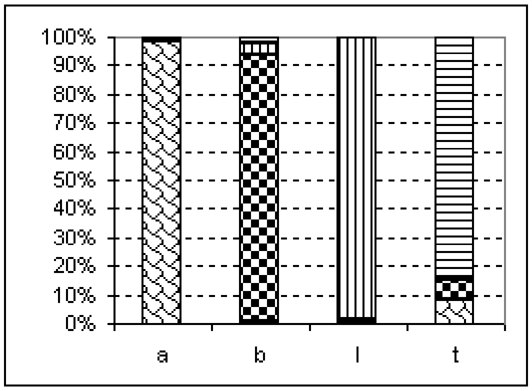


Table D4.4 Aggregate results of GA/Projection method

GA_Projection

	C/T	a	b	l	t	total
A	0.968	**121**	4	0	0	125
B	0.976	3	**122**	0	0	125
L	1	0	0	**125**	0	125
T	0.984	0	1	1	**123**	125
Over all	**0.982**					**500**

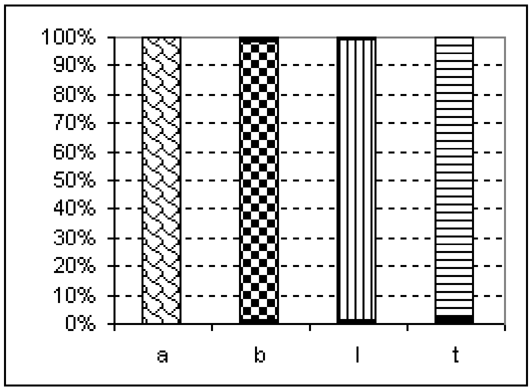


Table D4.5 Aggregate results of SOM/Hough method

SOM_Hough

	C/T	a	b	l	t	total
A	0.792	**99**	2	0	24	125
B	0.832	1	**104**	3	17	125
L	0.976	0	3	**122**	0	125
T	0.84	9	11	0	**105**	125
Over all	**0.86**					**500**

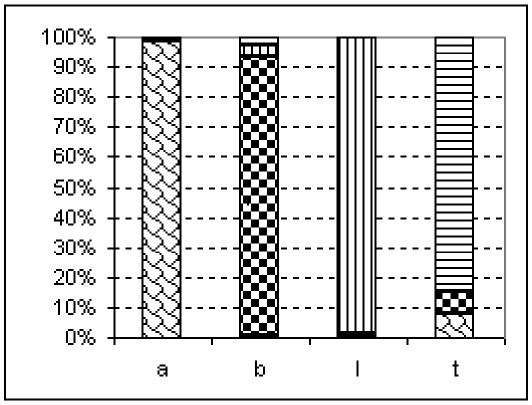
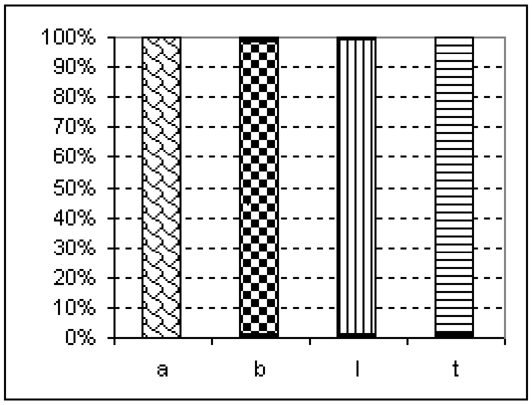


Table D4.6 Aggregate results of SOM/Projection method

SOM_Projection

	C/T	a	b	l	t	total
A	1	**125**	0	0	0	125
B	0.968	1	**121**	2	1	125
L	0.992	0	1	**124**	0	125
T	0.976	0	2	1	**122**	125
Over all	**0.984**					**500**



D-5 The experiments of automatically thresholded images

Table D-1 MLP/Hough tested with the auto-thresh 500 images

MLPhgh_BIN500

	correct/total	a	b	l	t	total
A	0	0	0	0	125	125
B	0.072	0	9	0	116	125
L	0.016	0	4	2	119	125
T	0.904	0	12	0	113	125
Over all	**0.248**					**500**

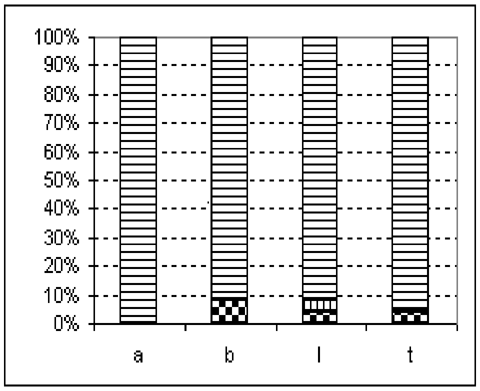


Table D-2 MLP/Projection tested with the auto-thresh 500 images

MLPprj_BIN500

	correct/total	a	b	l	t	total
a	1	125	0	0	0	125
b	0	120	0	0	5	125
l	0	123	0	0	2	125
t	0	123	2	0	0	125
Over all	**0.25**					**500**

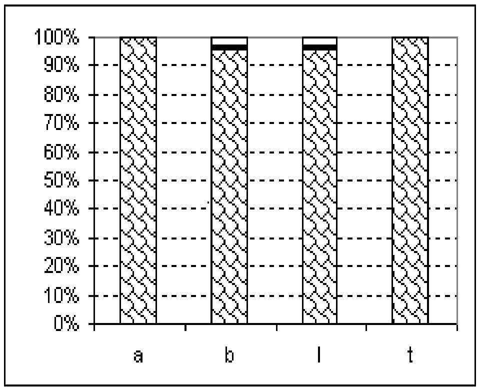


Table D-3 GA/Hough tested with the auto-thresh 500 images

GA$_{hgh}$_BIN500

	correct/total	a	b	l	t	total
a	0.992	124	1	0	0	125
b	0.096	111	12	0	2	125
l	0.016	116	7	2	0	125
t	0.08	103	12	0	10	125
Over all	0.296					500

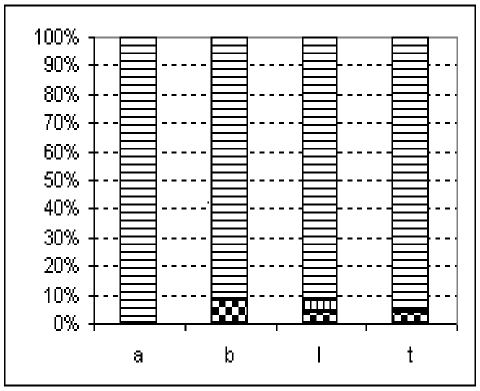


Table D-4 GA/Projection tested with the auto-thresh 500 images

GA$_{prj}$_BIN500

	correct/total	a	b	l	t	total
a	0.992	124	1	0	0	125
b	0.072	111	9	5	0	125
l	0.008	114	9	1	1	125
t	0.008	106	17	1	1	125
Over all	0.27					500

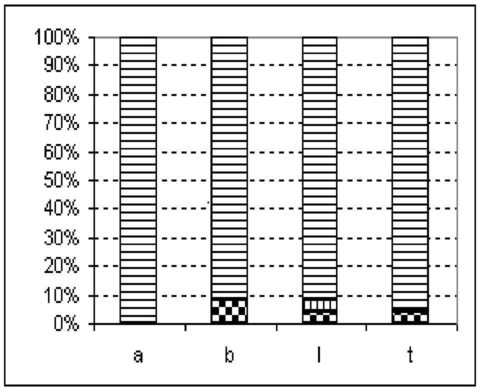


Table D-5 SOM/Hough tested with the auto-thresh 500 images

SOM$_{hgh}$_BIN500

	correct/total	a	b	l	t	total
a	0.72	90	0	0	35	125
b	0.08	60	10	0	55	125
l	0	75	0	0	50	125
t	0.56	55	0	0	70	125
Over all	0.34					500

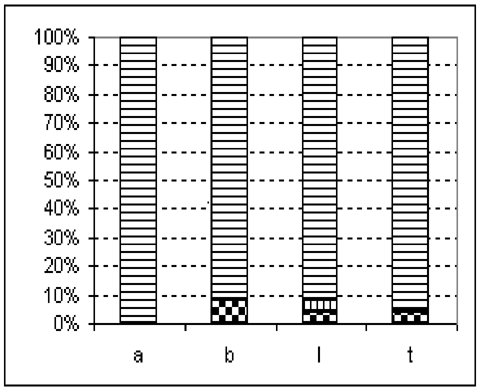
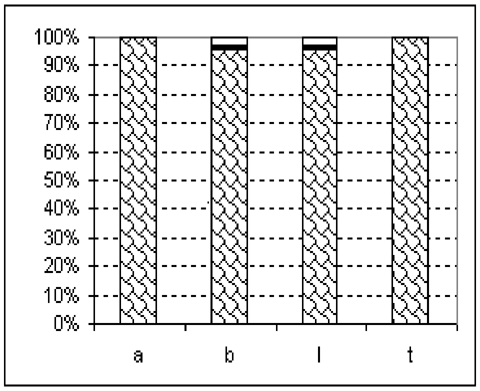


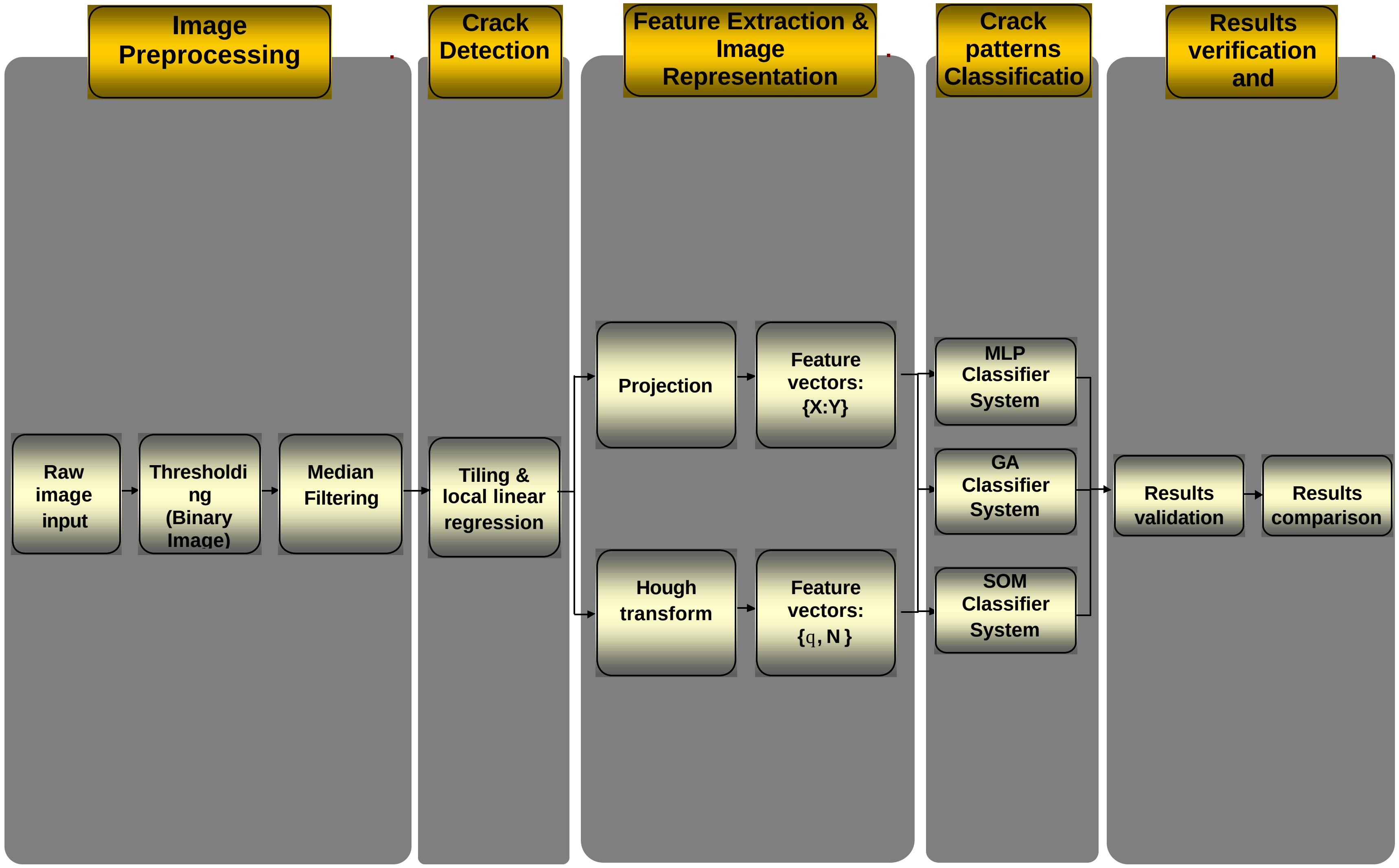
Table D-6 SOM/Projection tested with the auto-thresh 500 images

SOM$_{prj}$_BIN500

	correct/total	a	b	l	t	total
a	1	125	0	0	0	125
b	0.08	110	10	0	5	125
l	0.008	115	8	1	1	125
t	0.008	105	18	1	1	125
Over all	**0.274**					**500**



Appendix E System Architecture



VITA

Haroun R. A. Rababaah was born in Kufr Rakib, Irbid, Jordan, on May 10, 19972. The sun of Rasheed A. Rababaah and Ameneh M. Rababaah. After completing his work at Kufr Awan High School, Irbid, Jordan, in 1990, he entered the University of Jordan (UOJ) at Amman, Jordan. He received the degree of Bachelor of Science in Industrial Engineering from the University of Jordan (UOJ) in August , 1995. During the following two years he was employed as production and design engineer at Al-Majd Auto Spare Parts Industries, Amman, Jordan. During the following three years, he was employed as production and quality manager at Century Standard Textile, Irbid, Jordan. In 2001, he moved to South Bend, IN, USA, where he entered Indiana University South Bend (IUSB) at the computer science department, pursuing a degree of Master of Science in Applied Math and Computer Science, majoring in Computer Science. He worked at IUSB as an adjunct faculty and as research assistance.