The Performance of Training Pattern Sets in A New ART-Based Neural Architecture for Image Enhancement

By

Fu-Chun Chang

Thesis submitted to the faculty of the Graduate School of New Jersey Institute of Technology in partial fulfillment of the requirement for the degree of Master of Science in Computer and Information Science

1991

APPROVAL SHEET

Title of Thesis **:** The Performance of Training Pattern Sets in A New ART-Based Neural Architecture for Image Enhancement

Name of Candidate **:** Fu-Chun Chang
Master of Science in Computer and Information Science, 1991

Thesis and Abstract Approved

Dr. Frank Y. Shih — Date
Assistant Professor of
Department of Computer and Information Science

VITA

Name	:	Fu-Chun Chang
Permanent address	:	1 Birchwood Court, Apt. 2J, Mineola, NY 11501
Degree and date to be conferred	:	Master of Science , 1991
Date of birth	:	
Place of birth	:	

Collegiate institutions attended	Date	Degree	Date of Degree
New Jersey Institute of Technology	1990-1991	M.S.	1991
New York Institute of Technology	1989-1990		
National Taiwan University	1980-1984	B.S.	1984

Major	:	Computer and Information Science

Abstract

Title of the Thesis : The Performance of Training Pattern Sets in A New ART-Based Neural Architecture for Image Enhancement

Name : Fu-Chun Chang

Thesis Directed by : Dr. Frank Y. Shih

Neural network can be applied on the image enhancement after adding another two layers into the Adaptive Resonance Theory architectures (ART 1). The analysis for selecting a nice training pattern set associate the appropriate vigilance values is the main concerns in this thesis. For a single training pattern ,the network can act as a mathematical morphology operators such as erosion , dilation, opening and closing. With more than one training patterns in the network, 16 experiments are tested and are compared to each other in order to find the best selection for doing the image enhancement work. With both the training pattern set and vigilance values fixed, the first iteration always show the best performance than the other iterations. The comparison between 16 experiments explores the criteria for selecting the better training pattern set. Increasing the local features in the network training and a little bit "flexibility" of the vigilance parameter can increase the performance of the image enhancement. With the vigilance changed in different iterations also can be the good way to choose the training pattern set.

ACKNOWLEDGEMENT

I have great pleasure in expressing my grateful acknowledgements to Dr. **Frank Y. Shih**, Professor, Computer and Information Science Department for his valuable guidance and continued encouragement through this thesis work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I. INTRODUCTION

Adaptive Resonance Theory (ART) architectures are neural networks that self-organize stable recognition codes in real time in response to arbitrary sequences of input patterns. Theorems have been proved that trace the real-time dynamics of ART networks in response to arbitrary sequences of input patterns. Until now, there are three classes of ART model has since been introduced by Carpenter and Grossberg. The first class, ART 1, self-organizes recognition categories for arbitrary sequences of binary input patterns (Carpenter & Grossberg, 1987a). A second class, ART 2, does the same for either binary or analog inputs (Carpenter & Grossberg, 1987b). The most recent developed class, ART 3, incorporates a model of the chemical synapse into a new Adaptive Resonance Theory (ART) neural network architecture, which dynamics model a simple, robust mechanism for parallel search of a learned pattern recognition code.

This report implements the ART 1 model to deal with the image enhancement problems and discusses more details about the training patterns operations for relating with the mathematical morphology operations involve erosions and dilations, which has been defined on binary images. For the image enhancement topic, many different training patterns were tested in order to find the best training pattern to do the image enhancement job. The results, then, were compare to each other for getting a better idea to select a good set of learning patterns, which can do a better performance for dealing with the image enhancement problem.

In the second chapter, the architecture and the learning processes of ART 1 model were reviewed. Then , in chapter III, a review for the ART 1 implementation for image enhancement (Frank Y. Shih, Jenlong Moh and Fu-Chun Chang, 1991) were also included for understanding the necessary researches in this report. The fourth chapter discusses the training patterns and vigilance parameter analysis in image enhancement, which implements the analysis for selecting a nice training pattern and vigilance values in the image enhancement. At the fifth chapter, the experiments for single pattern training are included, where the relations between the neural network training pattern operations (ART 1 based) and the mathematical morphology operations were discussed. In the sixth chapter, the experiments for the effects using different training pattern sets on image enhancement were included. Chapter VII, the effect of different vigilance values in iterations for the same training pattern is discussed. Chapter VIII, some conclusions are summarized. Chapter IX, the references are listed. Chapter X , list the figures . Chapter eleven, list the tables. Appendix I list the ART 1 simulator program which was firstly written by Henry Bourne has been improved and debugged to perform the necessary experiments.

CHAPTER II. ART 1 ARCHITECTURE AND LEARNING PROCESSES REVIEW

An ART 1 model presents two subsystems in the architecture. One is the attentional subsystem and the other is the orienting subsystem, which is also called novelty detector (Fig.1).

In the attentional subsystem, there are two layers of neurons , feature representation layer (F) and exemplar (category) representation layer. The neurons between these two layers are fully connected, which means every neuron in one layer is connected with all the neurons in another layer. The strength of connections (i.e. weights) between neurons in the two layers , including bottom-up adaptive filter and top-down adaptive filter , determines the long-term memory (LTM) traces. The activation of neurons inside the F and E layers forms the short-term memory (STM). The weight of connections in LTM can be adjusted only during the training process, which is also the reason for calling it the long term memory trace. However, the activation of neuron in STM can be changed by input pattern codes while in training and recognition processes, which is the reason for naming short term memory. An ART 1 system is fully defined by a system of differential equations that determines STM and LTM dynamics in response to an arbitrary temporal sequence of binary input patterns. Theorems characterizing these dynamics have been proved in the case where fast learning occurs; that is, where each trial is long enough for the LTM traces to approach equilibrium values.

In the orienting subsystem, a matching threshold (ρ) called vigilance parameter (ranged between 0.0 and 1.0) was introduced to determine how close a new input pattern must be to a stored exemplar to be considered similar. In the Fig.1 , an orienting subsystem is represented by a circle with ρ , which acts as a novelty detector and handles the vigilance testing tasks.

The learning processes in ART 1 system active both subsystems together and classify every random input pattern into either a classified category if one category can be found similar or a new created category if there is no category can be found similar. First , we initialize the bottom-up filter and top-down filter connection according the eq.(1)&(2), supposing N neurons are in the F layer.

$$B^{(0)}_{F_i,E_j} = \frac{1}{1+N} \tag{1}$$

$$T^{(0)}_{E_j,F_i} = 1.0 \tag{2}$$

The stimuli of input patterns lead to the activities of neurons in F. The signal then pass through the connections to neurons in E and are multiplied by their corresponding weights. Each neuron in E sums up all of those weighted activities according the eq.(3).

$$\mu^{(t)}_{E_j} = \sum_i B^{(t)}_{F_i,E_j} \cdot x_i \tag{3}$$

The layer E is designed as a competitive network where each neuron has lateral inhibitory connections to the others. This is capable of choosing the neuron which receives the largest total input, i.e., winner-take-all strategy is used. After this , the input pattern goes through the vigilance test according eq.(4).

$$\frac{\sum_{i} T^{(t)}_{E_p,F_i} \cdot x_i}{||X||} > \rho \tag{4}$$

If the input pattern satisfies the above inequality, then it is classified into the category corresponding to the winning neuron. Otherwise, the orienting subsystem sends a reset signal through the excitatory connection to the winning neuron in order to temporarily disable its output so that the system may choose another winning neuron. Such a task will be repeated until either an exemplar similar to the input pattern being chosen or no more exemplar to choose, then a new neuron is created in E and assigned to represent the new category. Finally, the winning neuron represents a categories and triggers its associative pattern throughout the connections to F according to eq.(5)&(6).

$$T^{(t+1)}_{E_p,F_i} = T^{(t)}_{E_p,F_i} \cdot x_i \tag{5}$$

$$B^{(t+1)}_{E_p,F_i} = \frac{T^{(t)}_{E_p,F_i} \cdot x_i}{0.5 + \sum_{i=0}^{N-1} T^{(t)}_{E_p,F_i} \cdot x_i}. \tag{6}$$

These processes repeat until each input pattern has been classified into a category.

This is very useful for applications with categorical perception - that is, classifying each input pattern as belonging under one and only one category.

CHAPTER III. IMAGE ENHANCEMENT ART 1 ARCHITECTURE REVIEW

A. Overall Architecture

Image enhancement can be regarded as selective emphasis and suppression of information in the picture, with the aim of increasing the picture's usefulness. When a picture is converted from one form to another, e.g., imaged, copied, scanned, transmitted , or displayed, the "quality" of the output picture may be lower than that of the input. So, image enhancement skill is necessary and important. With ART 1 model application , image enhancement can be easily performed by adding another two layers to ART 1 model. The overall architecture for image enhancement depicted in Fig.2 & Fig.3. The Fig.2 is an unit ART 1 module , which is included in the Fig.3 as a block for dealing with every pixel in the image. The first two layers in the ART 1 module intend to determine whether or not the input pattern is matched with the exemplars of certain features previously learned and stored. The first layer consists of 25 neurons F_i since the local 5x5 neighborhood is considered. The second layer contains N exemplar neurons E_n, where N is the total number of features necessary for image enhancement (Fig.2). The third layer , or called region detection layer, made up of 5 neurons D_k ($k = 1, 2, \ldots ,5$) corresponding to the pixel P and its four neighbors, is used to determine whether P should be illuminated or not (Fig.3). Each region detection neuron receives and sums the responses from all the exemplar neurons in the second layer. If any one of the exemplar neurons has a positive response, indicating that p is classified into an exemplar category and should be illuminated as part of the object region, then the corresponding detection neuron will output "1" according to eq.(7).

$$g(D_k) = \begin{cases} 1 & \text{if } D_k \geq 0 \\ 0 & \text{otherwise} \end{cases} . \tag{7}$$

The fourth layer contains only one neuron corresponding to the pixel P. It has the input connections from five neurons related to the pixel P and its four neighbors so that a simple binary edge detection can be performed. The output neuron implements a simple binary edge detection algorithm whereby a pixel is illuminated only if it has no neighbor having value 1, indicating that it is a noise, and if it has all four neighbors having value 1, indicating that it is an interior point. The four neighbors labeled as P_1, P_2, P_3, and P_4, are located in the east, north, west, and south of the pixel P, respectively. The output neuron h(O) is given by eq.(8).

$$h(O) = h\left[\sum_{i=1}^{5} g(D_i)\right], \quad \text{where } h(u) = \begin{cases} 1 & \text{if } 1 \leq u \leq 3 \\ 0 & \text{if } u = 0, 4 \end{cases} . \tag{8}$$

Each exemplar neuron E_j (j =1,...,M) has its inputs from each neighboring neurons F_i through the bottom-up connections and a lateral inhibitory input from itself and the other exemplar neurons. The lateral inhibitory weight e_{ab} from E_a to E_b is defined as follows :

$$e_{ab} = \begin{cases} 1 & \text{for } a=b \\ -e & \text{for } a \neq b' \end{cases} \tag{9}$$

where a,b = 1, ..., M and e < 1/M. The purpose of this lateral inhibitory input made by the exemplar neuron is to suppress all the other exemplar neuron outputs toward zero but itself positive. This process will result an exemplar output greater than zero and that is the exemplar with the closest category to the input neighborhood of the pixel P. The updated output f(E_j) for exemplar neuron E_j is designed as follows :

$$f_{E_j}^{(t+1)} = \frac{f^{(t)}\left[f_{E_j}^{(t)} - e \sum_{a \neq j} f_{E_a}^{(t)} \right]}{V_{E_j}} \tag{10}$$

where $f_{Ea}^{(t)}$ is the output of exemplar neurons other than E_j and V_{Ej} is the vigilance parameter for exemplar neuron E_j. The function f(u) is a threshold logic type, i.e.,

$$f^t(u) = \begin{cases} u & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

The incorporation of the vigilance parameter into eq.(10) is to force the network to attempt classification into exemplar categories of lower vigilance first. This prevents the pixel neighborhood which does not meet with a high vigilance test but meets with a low vigilance test from misclassifying.

B. Algorithm Description

(1) Network initialization. Top-down weights $T_{Ej,Fi}{}^{(0)}$ are set to 1, bottom-up weights $B_{Fi,Ej}{}^{(0)}$ are set to $1/(1+5^2)$, and vigilance parameters V_{Ej} are set to 1.0. Both weights are updated by using eqs.(5) and (6). The detection and output layers are disabled.

(2) A set of feature patterns are trained into the network and the initial output of the exemplar neuron E_j is calculated by

$$f_{E_j}^{(0)} = \sum_{i=1}^{25} B_{F_i,E_j}^{(0)} \cdot x_i . \tag{12}$$

(3) The "Maxnet" procedure as described in Eq. (10) is performed until only one exemplar neuron is positive.

(4) The exemplar neuron E_j pre-stored in the top-down weights $T_{Ej,Fi}$ is compared with the input pattern X by

$$\frac{\sum_i T_{E_j,F_i} \cdot x_i}{\|X\|} \geq V_{E_j}, \tag{13}$$

where $||X||$ is the norm of the input pattern vector,i.e., the total number of one bits in the input vector. If eq. (13) is true, the input pattern belongs to the exemplar category represented by E_j and no new category is established. Otherwise, the input pattern does not belong to the most likely category and a new unused exemplar neuron if available is selected and updated by using eqs. (5) and (6).

(5) Steps (1) to (4) are repeated until all feature patterns are trained completely. Those steps can be done off-line. After learning, the task of image enhancement is executed.

(6) Weight adjustment is disabled and suitable vigilance parameter V_{Ej} is selected for each region exemplar. The different vigilance parameter reflects a different degree of importance of features with respect to image enhancement. Specific regions or contours could be easily extracted by the incorporation of a very high vigilance for the particular exemplar in question.

(7) A binary input image is placed into the network and the processing proceeds in parallel for all pixels. One exemplar is selected as in steps (3), (4) and (5) in the learning phase for each pixel and the corresponding output will be positive.

(8) Each detection neuron D_k outputs a 1 if any f_{Ej} is positive and 0 otherwise.

(9) Each output neuron O outputs a value 1 or 0 according to eq.(8).

CHAPTER IV. TRAINING PATTERNS & VIGILANCE VALUES ANALYSIS

The ART 1 training patterns & vigilance analysis is a very important and interesting subtopic for dealing with the image enhancement problem , especially for the feature extraction and noise removing from an image.

The vigilance parameter is a threshold factor for determining whether the input pattern can be categorized into the exemplar pattern or not. If we disable the learning process and freeze the LTM weight after a single category was trained, a matching factor between the input pattern and the exemplar can be easily calculated only by the matched pixels with pixel value "1" between both input pattern and the only exemplar pattern divided by the norm of the exemplar pattern, which is a total number of pixel "1" in the exemplar pattern. By comparing the matching factor with the vigilance parameter, it can be easily decided whether or not the tested central pixel in local input pattern, at this report we use 5x5 local region, should be turned ON. The simplification in the equation is because there is only one exemplar in the E layer, and there is no other choice for categorizing the input pattern into the other exemplars. So, after passing the input signals through the bottom-up connections, the only category in E layer always receives the highest response in the E layer and is chosen as the winner neuron for doing the vigilance test through eq.(13).

However, the condition doesn't look like all the same while the exemplar field has more than one categories after the learning process has been shut down. In this case, whether the input pattern with 5x5 local feature can be categorized into a chosen exemplar is not just that simple for considering the matching factor which is calculated by the matched pixels with pixel value "1" divided by the norm of the exemplar pattern and then for considering whether the matching factor is greater than vigilance parameter or not. The reason is that vigilance test is performed only after a winner neuron has been found. When there are more than one neurons in the E layer after the learning process has been disabled, categorizing an input pattern into an exemplar can't be easily calculated just through the eq.(13) at first time. There may have more than one exemplars which pass the vigilance test in E layer. It just can't make so quick decision as long as you find the first exemplar passed the vigilance test and categorize the input pattern into that exemplar. In this case, the lateral inhibition and winner-take-all in the E layer must be considered first in order to chose a winner for doing the vigilance test.

As so, a single pattern training in the ART 1 network model is just a special case in the system. But, it can be used to perform the feature extraction from an image, which means we can train a local feature as a single pattern and try to extract this local feature in an image by using the ART 1 model.

Through the single pattern training in the ART 1 model and input an image for test, we found some more interesting similarity between the mathematical morphology operations such as erosion and dilation , and the feature extract ART 1 model application. We analyze the responses of network activation by using a single template pattern. In eq (13), the term $\Sigma_i \; T_{Ej \cdot Fj} \cdot X_i$ is simply the inner product of the two vectors: input pattern X and template pattern T retrieved, which indicates the number of matched elements between the two patterns. By dividing the inner product with the norm of input pattern , we obtains a fractional number ranging from 0.0 to 1.0 which indicates the degree of matching. By choosing a vigilance parameter equal to or less than $1 / ||T||$, any occurrence of a pixel "1" of input pattern matched with the template will result in passing the vigilance test. In this case, the networks acts like a expanding operator, erosion , in image processing. On the other hand, choosing a higher vigilance parameter, e.g., greater than $(||T||-1)/||T||$, makes the vigilance tests difficult to pass. Even though the total pixels $||T||-1$ of input pattern matched with the template will result in failing the vigilance test. Thus, this situation is similar to shrinking operator, dilation. Also , when the vigilance value greater than $(||T||-1)/(2x||T||)$ and smaller than $(||T||+3)/(2x||T||)$, the output result will have no change with the original input pattern. While the vigilance value equal to or greater than $(||T||+3)/(2x||T||)$ but equal to or smaller than $(||T||-1)/||T||$, the network has partial dilation effect. While the vigilance value equal to or smaller than $(||T||-1)/(2x||T||)$ but greater than $1/||T||$, the network has partial erosion effect.

The vigilance parameter can be viewed as making a "fuzzy" decision on the pixel concerning the probability of being noisy points. The method depends on first distinguishing noise from non-noise in the picture, then removing the noise and "mending" the picture by interpolation.

With more than one training patterns learned in the E layer after the learning process has been disabled, the meaning in the image processing seems to extract several local features at the same time. At this case, there may be N neurons in the E layer after learning process be disabled. Let us consider the 5x5 local region as an input pattern for each pixel in the image. To decide a pixel should be treated as a picture pixel or a noisy pixel , we first need to consider its neighborhood, suppose a 5x5 local region is considered throughout the report. Each pixel in the image along with its neighborhood pixels pass through the ART 1 module (Fig.2). If one winner neuron in E layer can be found from the input pattern, the network process the winning neuron and the input pattern to pass the vigilance test (Fig.1). If the vigilance is passed, the input pattern is similar with winning neuron and is categorized into the exemplar. The third layer, the region detection layer, then receives and sums the responses from all the exemplar neurons in the second layer. Through the eq.(7), if any one of the exemplar has a positive response, central pixel of the input pattern p is classified into an exemplar category and should be illuminated as part of the object region. Then , the corresponding detection neuron will output "1", which means the pixel p is a picture pixel. If there is not any neuron in E layer can be found matching with the input pattern for passing through the vigilance test, the pixel in the local 5x5 region must be a noisy point needed to be removed. In this case, the corresponding detection will output "0". Through this report, the output "1"s are symbolized as "*" for clearly viewing. From the above analysis, the image enhancement can be performed well through the new ART1-based model which were developed by (Frank Y. Shih, Jenlong Moh and Fu-Chun Chang, 1991).

Because the training pattern and vigilance parameter value can be selected in many different combinations, the removing noise effects in the image enhancement are very different. What to choose and how to choose are the best concern in the thesis. Without a good knowledge to pick up a nice training pattern, the new ART1-based architecture cannot performed well in the image enhancement problem. Only select a good combination of the local features as a training pattern set but not considering the associated vigilance value along with each training pattern may get a worse result. For example, from the results of experiment 3 and experiment 11 , the vigilance values are set to 1.0 for each training pattern and the performance is the worst one for restoring the image in the eight combination sets. For choosing a better set of training pattern and a better vigilance value for each training pattern, the restoration of a noisy image can reach a higher performance. Several different training pattern sets and vigilance parameter value combinations have been experimented, only eight training pattern sets, Table 1 to 16, are listed in the report for showing the performance results. Also, for each training pattern set, four iteration runs are applied . The effect of several iterations on the different combination training sets are also discussed in the report.

What if we change the vigilance values in different iterations and apply the same training pattern set in every iteration is another interesting work for getting the better performance in image enhancement. The effect seems like the mathematical morphology operations opening and closing. In the chapter seven, an experiment with the different vigilance values applied in the different iteration for the same training pattern is discussed.

CHAPTER V. EXPERIMENTS FOR SINGLE PATTERN TRAINED

For single training pattern learned in the learning process, there is only one neuron in the E layer. That means no matter what kind of input patterns are fed into the system, the winner neuron in E layer is the trained pattern. Should the input patterns be categorized as the similar pattern as the trained pattern will be decided by the eq.(13).

Experiments are held on the horizontal line pattern input with 10 pixels "1". After the horizontal training pattern ,with 5 pixels "1" , is trained, different vigilance values, ranged from 1.0 to 0.0, are assigned along the trained pattern. Several experiments with single trained pattern but different vigilance value are tested. The input pattern and the experimental results are listed in the Fig.4.

As we can see from the Fig.4, the experimental results have some relationship with the vigilance value. With vigilance value equal to 1.0 ,the output has only 6 stars("1") comparing with the original input, which has 10 stars. That has the same effect with the erosion operator in the mathematic morphology with the structure element radius equal to $(||T||-1)/2 = 2$. The term $||T||$ is the norm of the training pattern with pixel "1". In Fig.4, the $||T||$ is equal to 5. After the erosion operator applied, the output has only 6 stars left. With the vigilance value equal to 0.2 , the output result shows 14 stars in the Fig.4. Comparing with the original input pattern which has 10 stars in one horizontal line, it has the same effect with the dilation operator in mathematic morphology by using the structure element with radius equal to 2. Notice that, with the vigilance value equal to 0.6 that will produce the result just the same with the original one, which means there is no any dilation or erosion operator used on the input pattern.

Applied the same method on the vertical line will show the similar results. With the analysis in the last chapter, we know if the vigilance value is equal or less than $1/||T||$, any occurrence of pixel "1" of input pattern matched with the template will result in passing the vigilance test. In this case , the network acts as a erosion operator in image processing. Choosing the higher vigilance parameter, greater than $(||T||-1) / ||T||$, makes the vigilance tests difficult to pass. The situation is similar with dilation operator in image processing. When the training pattern has odd pixel numbers, with the vigilance value greater than $(||T||-1) / (2x||T||)$ and the vigilance value smaller than $(||T||+3) / (2x||T||)$, the output result will have no change with the original input pattern. For example , in a training pattern with norm equal to 5 ($||T||=5$), when vigilance parameter has value between $0.4 < v < 0.8$, the result is the same with the original input pattern. If the training pattern has the norm equal to 7, then the vigilance value should between $3/7 < v < 5/7$ in order to have the unchanged effect in output.

We tried to use a character "B" and a bulb image to do the erosion and dilation operation with the vigilance parameter set equal to 1.0 and 0.2. Fig.5 (a) is the original "B" image and Fig.5 (b) is a original bulb image in 32x32 region. Fig.4 (b) is selected as the training pattern. Fig.6 (a) shows the erosion operation on "B" with vigilance equal to 1.0 , while Fig.6 (b) on bulb image with vigilance value equal to 1.0. The Fig.7 (a) and Fig.7 (b) show the dilation results on character image "B" and bulb image with vigilance equal to 0.2 . The conditional probabilities in those figures shows the performance degree between the output images and the original nonnoisy image, which will be discussed more detail in next chapter.

CHAPTER VI. EFFECTS OF DIFFERENT TRAINING PATTERN SETS

In order to get a better image enhancement effect on the noisy image, several different training pattern sets are provided to test the performance. Assume that the nonnoisy picture contains a large region of constant gray level, we can get some insight into the noise statistics by analyzing the gray level fluctuations in the corresponding region of the noisy picture. The region contrast is 100. Two different noisy images, character "B" and bulb image, are produced by adding the independent Gaussian noise having mean zero and standard deviation 35. Thus the signal to noise ratio (SNR -region contrast/noise deviation) of these images is 2.86. The original images are shown in Fig.5. The noisy images are shown in Fig.8.

Three different training pattern sets, A,B and C, that are more meaningful in enhancing the noisy image are selected in the report for further test (Fig.9). Given the different vigilance values , three different pattern sets can then be expanded into eight experiments to each tested image. That means there are totally 16 experiments included. For each experiment, four iterations are run in order to see the effect of the iteration. To compare the performance of different vigilance parameters and the number of iterations, we use the conditional probability of the labeled "object-point" given the true object-point, $P(O'|O^*)$ and the conditional probability of a true object-point given the labeled object-point, $P(O^*|O')$. The performance probabilities were recorded for further comparison and discussion.

The adjustable parameters of each iteration and the number of iterations are chosen to equalize these two conditional probabilities. The quality of the neural architecture to image enhancement is determined by the value of $P(O'|O^*)=P(O^*|O')$. Although this performance measure is not in general applicable on all kinds of images, it is well suited for these simulated images.

The output images through each iteration in every experiment are listed from Fig.10a to Fig.25b. Sixteen performance tables, which are labeled with the experiment sequence number and the iteration number are listed from Table 1 to Table 16. The performance value of 0th iteration from Table 1 to Table 16 means the conditional probability between the noisy image (Fig.8) and the original image (Fig.5).

Training pattern set A use eight local features to enhance the noisy image. These eight local features including the vertical line, horizontal line, right diagonal line, left diagonal line , and four corners. These are the basic local features in enhancing the image. Training pattern set B adds another four training patterns to set A in order to smooth the circle corner. The new added training patterns all posses three local features. In the training pattern set C, the first four training patterns are the same with the other two set. However, the other eight training patterns are changed, which all posses two local feature in each pattern. The change in the set C try to smooth the circle corner and the perpendicular corner at the same time (Fig.9).

The experiment 1 applies the network to the noisy image "B" after the network were trained with training pattern A with the vigilance setting A1, A2, A3 and A4 equal to 0.8 and A5, A6, A7 and A8 equal to 1.0. The performance values are recorded in the Table 1.

The experiment 2 and experiment 3 are similar to experiment 1 except the vigilance settings are different. At the experiment 2, the vigilance setting from A1 to A4 is 0.8 and from A5 to A8 are 0.9. While at experiment 3 , the vigilance setting of all eight training patterns is equal to 1.0 (Tab.2 & 3).

The experiment 4 applies the network to noisy image "B" after the training pattern set B are trained with the vigilance setting B1, B2, B3, B4 equal to 0.8 and B5,B6,B7,B8 equal to 1.0 and B9,B10,B11,B12 equal to 0.8 again (Tab.4).

The experiment 5 and 6 do the same thing as experiment except the vigilance setting is different. In experiment 5, the network is trained with the vigilance value all equal to 0.8 . In experiment 6, the first four pattern vigilance parameters are set to 0.8 but the others are set to 1.0 (Tab.5 & 6).

Training pattern set is changed to C in the experiment 7 and 8. In these two experiment, the noisy image is "B" but the vigilance settings are different. The vigilance setting in experiment 7 is C1,C2,C3,C4 equal to 0.8 and the others equal to 1.0. Vigilance setting in the patterns of experiment 8 are all equal to 0.8 (Tab.7 & 8).

The experiments from 9 to 16 apply the same testing set as the experiments from 1 to 8 except the noisy image is changed to bulb image (Tab.9-16).

From the Tab.1 to Tab.16 and from the Fig.10a to Fig.25b, we can find the performance of the first iteration almost have the highest value in both P(O'|O*) and P(O*|O') although, sometimes, the second iteration's performance is very close to the first one. That's mean we can always use the first iteration result to representing the best result in each different combination test. The more iteration on the same training pattern set with the fixed vigilance value doesn't help to enhance the image. Because the first iteration result can be treated as a representative for that test. We just average both the conditional probabilities value of P(O'|O*) and P(O*|O') together and compare the average with the other experiments. For two arbitrary images, no matter those images are noisy or nonnoisy , both the P(O'|O*) and P(O*|O') equal to one if those two images are the same. So, the more equality and the higher performance those two conditional probability values have , the more similarity between two image is. Tab.17 lists the average of P(O'|O*) and P(O*|O') in the first iteration between the different experiments and different images. From the Tab.17 , we can see the experiment 3 and experiment 11 have the worst performance among the above experiments. Experiment 3 and experiment 11 belong to the same training pattern set A and have the same vigilance setting with vigilance all equal to 1. It is clear that the A training pattern set have only the simplest training patterns. When the vigilance values all are set to 1, the network lose the ability to adjust themselves and the coming out image got the lowest performance comparing with the others. Experiment 1 & 9 and experiment 6 & 14 all have the lower performance when comparing with other experiments. The same as experiment 3 & 11, experiment

1 & 9 use the training pattern set A that has only 8 basic patterns and half of them have the vigilance parameter set to 1.0. However, experiment 6 & 14 use training pattern set B that has 12 training patterns but two third of their vigilance are set to be 1.0. For character "B" image, experiment 5 has the best performance among the 8 experiments. The network use training pattern set B that has 12 training patterns with all vigilance values set to 0.8. For bulb image, experiment 16 has the best result. At this time , the training pattern set is C which has 12 training patterns with all 12 vigilance values set to 0.8. It seems that both training pattern set B and C perform well in these test. Because they all use 12 training patterns in the network training. When you add more training patterns in the exemplar , the system performs better than those with the only a few training patterns in the system. Another important point of view is that the vigilance parameter value must have some kind of flexible in the system. Noticed that when all the vigilance value has been set to 1.0, the system lose its ability to adjust the image , to "mending" the image or to fill in the image. On the other hand, it persist so many erosion effects on the image that even the nonnoisy pixel can't pass the vigilance test and must be turn off. Experiment 5 and 16 all have these kind of "flexibility" that can sometimes fill in the gap and still posses the ability to get rid of the noisy point. The Tab. 18 list the performance order for each image.

CHAPTER VII. EFFECTS OF DIFFERENT VIGILANCE VALUES IN ITERATIONS

As we see from the chapter IV and V, the higher vigilance will perform the erosion effect on the image and the lower vigilance will perform the dilation effect on the image. For a ART 1 model applied on the image enhancement, we may first apply some higher vigilance to do more erosion effect and get rid of the noisy pixel. The second iteration , we just lower the vigilance parameter in order to perform the "mending", dilation effect. From this point of view, we select the simple training pattern set A to do the test. But, at the first iteration , the first 4 training patterns assigned the vigilance parameter equal to 0.8 and the others training patterns with vigilance value assigned 0.9. This is the experiment 2 and 10 actually. And the performance order in the last chapter shows they are in the middle of the list. Differently, we change all the vigilance values equal to 0.6 at the 2nd iteration. The results were shown in the last paper (Frank Y. Shih, Jenlong Moh and Fu-Chun Chang, 1991) and will repeatedly show in this report for the completely research report. The results can be found in the Fig.26, which has the original images , noisy images and outputs of the region detection layer for both "B" image and bulb image. The Fig.27 show the output images after the 4th layer operations in the network. The Tab.19 list the performance of this test with the 2nd, 3rd, 4th iteration vigilance values all be set to 0.6.

CHAPTER VIII. CONCLUSIONS

From the single pattern training experiment, the results clearly show that the network acts as a erosion operator in image processing if the vigilance value is equal to or less than 1/||T|| and the network acts as a dilation operator in image processing if the vigilance value greater than (||T||-1)/||T||. Also , when the vigilance value greater than (||T||-1)/(2x||T||) and smaller than (||T||+3)/(2x||T||), the output result will have no chang with the original input pattern. While the vigilance value equal to or greater than (||T||+3)/(2x||T||) but equal to or smaller than (||T||-1)/||T||, the network has partial dilation effect. While the vigilance value equal to or smaller than (||T||-1)/(2x||T||) but greater than 1/||T||, the network has partial erosion effect.

When the training pattern set has more than one neuron in the E layer, there are so many different combination training pattern sets to be selected. Three different training patterns sets are chosen to test both the character "B" and bulb noisy images. When we combine the three different training pattern set A,B,C with the different vigilance setting, totally sixteen experiments were performed. From the above 16 experiments, the results show one similar effect. That is no matter what kind of training pattern set is applied, the result from the first iteration can reach the best performance than results form the other iteration. The first iteration always shows a better performance than the other iterations. That means we can just apply a single training pattern set and get the best result from the first iteration if the vigilance setting is fixed. From this conclusion , we average the performance of P(O'|O*) and P(O*|O') from the first iteration of each experiment and compare the performance between those eight different experiments in each noisy image. The experiment 5 and 16 show the best performance in "B" and bulb image, but the experiment 3 and 11 show the worst performance in both of these two images. The comparisons between the 16 experiments show two important conclusions. First, increasing the training patterns, which means increasing the local features for enhancing the noisy image, will increase the performance for the image enhancement. Second, assign the vigilance parameter a little bit "flexible" in order to have both the erosion and dilation effect. With these two considerations, the training pattern set can get a better performance for doing the image enhancement.

Another conclusion comes from the test of using different vigilance values in different iterations. We can also get a nice performance from changing the vigilance values in different iterations. Try to use the higher vigilance in the first iteration can perform a lot of erosion effect which get rid of the noisy pixels. Then, at the second iteration, we choose a lower vigilance value for providing a little bit dilation or "mending" effect.

CHAPTER IX. REFERENCES

[1] Frank Y. Shih and Jenlong Moh, " Improved Adaptive Resonance Theory" In Proc. SPIE Conference on Intelligent Robots and Computer Vision IX: Neural, Biological, and 3-D Methods, Boston, Nov. 1990.

[2] Frank Y. Shih, Jenlong Moh and Fu-Chun Chang, " A New ART-Based Neural Architecture for Pattern Classification and Image Enhancement without Prior Knowledge" pp. 1-22, 1991. (in submitting).

[3] Frank Y. Shih and O. Robert Mitchell, " Decomposition of Gray Scale Morphological Structuring Elements," IEEE, pp 304-306, 1987.

[4] G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," Computer Vision, Graphics, and Image Processing, Vol. 37, pp. 54-115, 1987.

[5] G. A. Carpenter and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," Applied Optics, Vol. 26, pp. 4919-30, Dec. 1987.

[6] G. A. Carpenter and S. Grossberg, " The ART of Adaptive Pattern Recognition by a Self-Organization Neural Network," IEEE Computer , pp. 77-88, 1988.

[7] G. A. Carpenter, " Neural Network Models for Pattern Recognition and Associative Memory," Neural Networks, Vol. 2, No. 2, No. 4, pp. 243-57, 1989.

[8] G. A. Carpenter and S. Grossberg, " ART3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures," Neural Networks, Vol. 3, No. 2, pp. 129-152, 1990

[9] S. Grossberg, " Nonlinear Neural Networks: Principles, Mechanisms, and Architecture, " Neural Networks, Vol. 1, No. 1, pp. 17-62, 1988.

[10] T. Kohonen, " An Introduction to Neural Networks," Neural Networks, Vol. 1 , No. 1, pp. 3-16, 1988.

[11] R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, pp. 4-22, 1987.

CHAPTER X. FIGURES

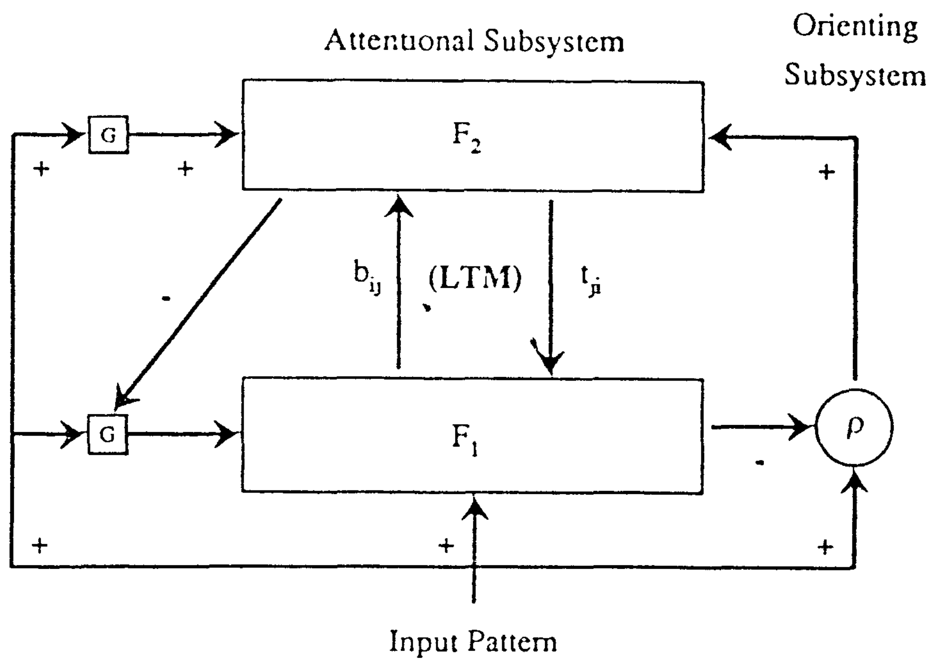

Fig.1. ART 1 model. The activation in layers F and E forms a short term memory (STM) and the connections including bottom-up and top-down construct a long term memory (LTM). Squares with the label "G" indicate the gain control devices and the orienting subsystem with vigilance parameter ρ is represented by the circle.

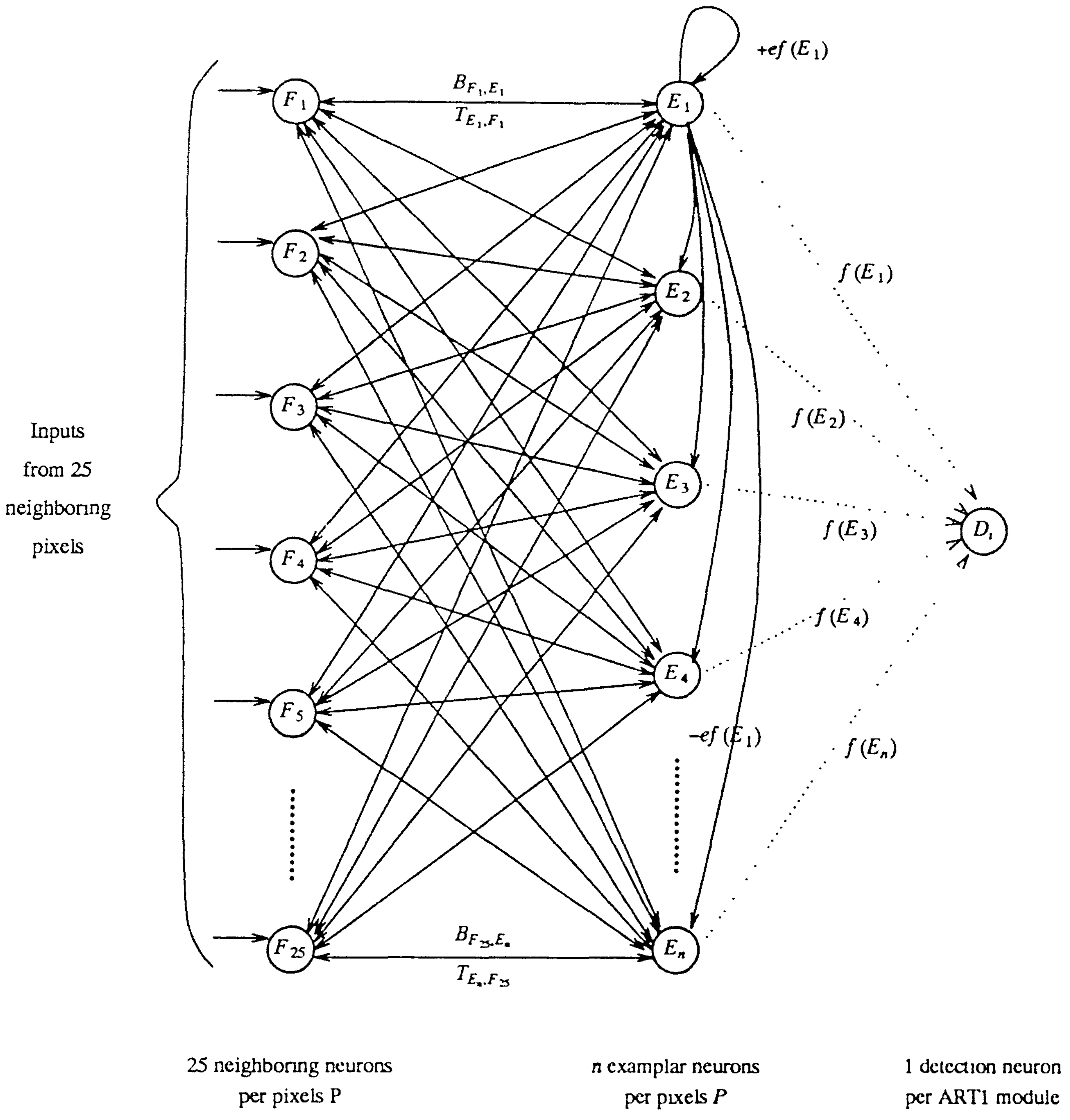

Fig.2. An ART 1 module

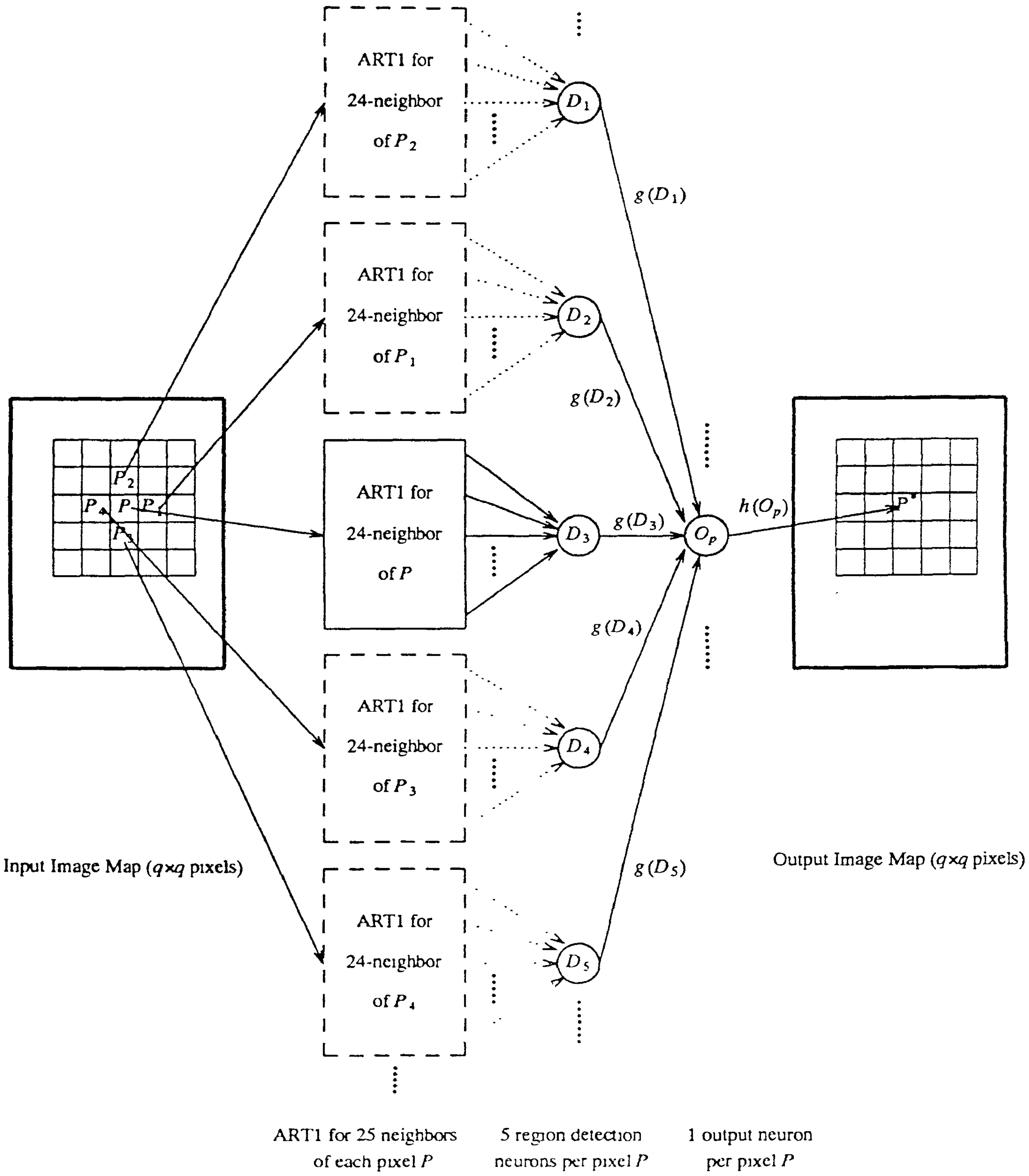
ART1 for 24-neighbor of P_2
ART1 for 24-neighbor of P_1
ART1 for 24-neighbor of P
ART1 for 24-neighbor of P_3
ART1 for 24-neighbor of P_4
D_1
D_2
D_3
D_4
D_5
$g(D_1)$
$g(D_2)$
$g(D_3)$
$g(D_4)$
$g(D_5)$
O_p
$h(O_p)$
P_2
P_4
P
P_1
P_3
P'
Input Image Map ($q \times q$ pixels)
Output Image Map ($q \times q$ pixels)
ART1 for 25 neighbors of each pixel P
5 region detection neurons per pixel P
1 output neuron per pixel P

Fig.3. An overall neural architecture for image enhancement. ART 1 modules (Fig.2) are included for each pixel in the input image.

```
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
...........**********............
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
.................................
```

(a) Original test image
(10 stars)

```
.....
.....
*****
.....
.....
```

(b) Train pattern

```
Original   **********

v = 1.0      ******

v = 0.8     ********

v = 0.6    **********

v = 0.4    ************

v = 0.2  **************

v = 0.0      fill all
```

(c) Experimental results with
different vigilance

Fig.4. Different vigilance effects on the original test image with a simple horizontal line trained.

```
................................
................................
................................
......***************...........
......*****************.........
......******************........
......*******************.......
......*****........******.......
......*****.........*****.......
......*****.........*****.......
......*****.........*****.......
......*****.........*****.......
......*****........*****........
......*****************.........
......****************..........
......*****************.........
......******************........
......*****.........*****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
......*****........*******......
......*******************.......
......******************........
......*****************.........
......***************...........
................................
................................
................................
................................
```

(a) Original "B"

```
................................
.........**************.........
........****************........
.......******************.......
......********************......
.....**********************.....
....************************....
...**************************...
..****************************..
.*********............*********.
.********..............********.
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.***********........***********.
.************......************.
.************......************.
.************......************.
.************......************.
.******************************.
.******************************.
.******************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......******************.......
........****************........
.........**************.........
................................
```

(b) Original bulb image

Fig.5. Original Image of character "B" and bulb image.

```
................................    ................................
................................    ...........**********...........
................................    ..........************..........
........***********.............    .........**************.........
........*************...........    ........****************........
........**************..........    .......******************.......
........***************.........    ......********************......
........*............**.........    .....**********************.....
........*.............*.........    ....************************....
........*.............*.........    ...*****................*****...
........*.............*.........    ...****..................****...
........*.............*.........    ...****..................****...
........*............*..........    ...****..................****...
........*************...........    ...*****................*****...
........************............    ...******..............******...
........*************...........    ...*******............*******...
........**************..........    ...********..........********...
........*.............*.........    ...********..........********...
........*..............*........    ...********..........********...
........*..............*........    ...********..........********...
........*..............*........    ...**************************...
........*..............*........    ...**************************...
........*.............**........    ...**************************...
........*............***........    ....************************....
........***************.........    .....**********************.....
........**************..........    ......********************......
........*************...........    .......******************.......
........***********.............    ........****************........
................................    .........**************.........
................................    ..........************..........
................................    ...........**********...........
................................    ................................

        P(O'¦O*) = 0.553                      P(O'¦O*) = 0.724
        P(O*¦O') = 1.000                      P(O*¦O') = 0.508

     (a) Character "B" Image                (b) Bulb Image
```

Fig.6. Similar erosion effect with vigilance equal to 1.0. Input patterns are Fig.5. The training pattern is Fig.4 (b).

```
................................
................................
................................
....*******************.........
....*********************.......
....**********************......
....***********************.....
....*********....**********.....
....*********.....*********.....
....*********.....*********.....
....*********.....*********.....
....*********.....*********.....
....*********....*********......
....*********************.......
....********************........
....*********************.......
....**********************......
....*********.....*********.....
....*********......*********....
....*********......*********....
....*********......*********....
....*********......*********....
....*********.....**********....
....*********....***********....
....***********************.....
....**********************......
....*********************.......
....*******************.........
................................
................................
................................
................................

P(O'¦O*) = 1.000
P(O*¦O') = 0.691
```

(a) Character "B" Image

```
................................
.......******************.......
......********************......
.....**********************.....
....************************....
...**************************...
..****************************..
.******************************.
********************************
************........************
***********..........***********
***********..........***********
***********..........***********
************........************
*************......*************
**************....**************
***************..***************
***************..***************
***************..***************
***************..***************
********************************
********************************
********************************
********************************
.******************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......******************.......
................................

P(O'¦O*) = 0.929
P(O*¦O') = 0.403
```

(b) Bulb Image

Fig.7. Similar dilation effect with vigilance equal to 0.2. Input patterns are Fig.5. The training pattern is Fig.4 (b).

```
...............*................
............*..................*
.............................*..
......****.**********...........
......**.***********.*..........
......**.*****.***.*****........
......******.**.****.***........
.......*.**.......****.**..*....
......*****.........*****.*.*...
......****..........*.*.*.......
....*.****.......*..****........
........***........******.......
......*.***........***..........
....*.**************.**.........
....*.****************..........
......******.**********.........
......*************.***......*..
..**..*****......*..**.**.......
......*****..........**.*.......
......*****..........*****......
......*****.......*..*****......
.......**.*..........*****......
......*****.........******......
......*****.....**.****.**......
......*..**.******.*****........
.....*.*.***.****.******........
.......****..***********.*......
......*..************...........
.............*..................
................................
....................*...........
........*.......................
```

(a) Noisy "B"

```
............*...................
.........*.************.........
........******.*******.*........
.......****.********.**.*.......
......**.****.*.*.***....*......
.......*..*******..********...*.
....**.*************.**.*.**..*.
...*****.***********.********...
..******.**.*.*.***.**********..
.****.****............*********.
.**.*.**.......*.......****.***.
.*****.**..*...........********.
.***.*.**...........*..*.*****..
.*..******......*..*..*********.
.****.***.*..........***.******.
.**********........*******.**.*.
.**.*****.***......***.********.
.*********.**.*....**.*********.
.***.********......************.
.*********.**......******.*****.
..*************.****.**********.
..*..******.***.******.****.***.
***.*****..*.**************.***.
..******.*******************.*..
...****....*********.********...
....*..*****.***************....
.....*********.************...*.
...*..**********.*********......
.......*.*.***********..*.......
........*.**.***********........
.........****.***.***...........
................................
```

(b) Noisy bulb

Fig.8. The noisy images with added Gaussian noise having mean zero and standard deviation 35.

```
..*..   .....   *....   ....*
..*..   .....   .*...   ...*.
..*..   *****   ..*..   ..*..
..*..   .....   ...*.   .*...
..*..   .....   ....*   *....
 A1      A2      A3      A4

.....   ..*..   ..*..   .....
.....   ..*..   ..*..   .....
***..   ***..   ..***   ..***
..*..   .....   .....   ..*..
..*..   .....   .....   ..*..
 A5      A6      A7      A8
```

(a) Training pattern set A

```
..*..   .....   *....   ....*
..*..   .....   .*...   ...*.
..*..   *****   ..*..   ..*..
..*..   .....   ...*.   .*...
..*..   .....   ....*   *....
 B1      B2      B3      B4

.....   ..*..   ..*..   .....
.....   ..*..   ..*..   .....
***..   ***..   ..***   ..***
..*..   .....   .....   ..*..
..*..   .....   .....   ..*..
 B5      B6      B7      B8

.....   ...*.   .*...   .....
**...   ...*.   .*...   ...**
..*..   ..*..   ..*..   ..*..
...*.   **...   ...**   .*...
...*.   .....   .....   .*...
 B9      B10     B11     B12
```

(b) Training pattern set B

```
..*..   .....   *....   ....*
..*..   .....   .*...   ...*.
..*..   *****   ..*..   ..*..
..*..   .....   ...*.   .*...
..*..   .....   ....*   *....
 C1      C2      C3      C4

.....   *....   ..*..   ....*
.....   .*...   ..*..   ...*.
***..   ..*..   ..*..   ***..
...*.   ..*..   .*...   .....
....*   ..*..   *....   .....
 C5      C6      C7      C8

*....   ..*..   ....*   .....
.*...   ..*..   ...*.   .....
..***   ..*..   ..*..   ..***
.....   ...*.   ..*..   .*...
.....   ....*   ..*..   *....
 C9      C10     C11     C12
```

(c) Training Pattern set C

Fig.9. Three training pattern sets A, B, C

```
................................
................................
................................
.......*************............
......**************.*..........
......*****************.........
......*******..*********........
......*****.......******........
......****.........******.......
.......***..........*****.......
......****..........****........
......*.***.........****........
......*.****......****..........
......*****************.........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........**.*.......
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......****.......*******.......
......******************........
.......*****************........
........****.**********.........
..........**********............
................................
................................
................................
................................

     P(O'¦O*) = 0.891
     P(O*¦O') = 0.981
```

(a) Experiment 1, 1st iteration

```
................................
................................
................................
........***********.............
.......*************............
......****************..........
......******....*******.........
......*****.......******........
......****.........*****........
......****..........****........
......****..........****........
......*.***........****.........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........***........
......*****..........****.......
......*****..........****.......
......*****..........****.......
......*****.........*****.......
.......*****.......******.......
.......*****************........
........***************.........
.........*************..........
...........********.............
................................
................................
................................
................................

     P(O'¦O*) = 0.832
     P(O*¦O') = 0.976
```

(b) Experiment 1, 2nd iteration

Fig.10a . The output image from the experiment 1.

```
................................
................................
................................
.........*********..............
........************............
.......**************...........
......******....*****...........
......*****.......*****.........
......****.........*****........
......****..........****........
......****..........***.........
......*.***........****.........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......*****************.........
......******........***.........
......*****..........***........
......*****..........***........
......*****..........****.......
......*****..........****.......
.......****.........*****.......
.......*****.......*****........
........****************........
.........**************.........
..........***********...........
............******..............
................................
................................
................................
................................
```

```
P(O'¦O*) = 0.776
P(O*¦O') = 0.974
```

(c) Experiment 1, 3rd iteration

```
................................
................................
................................
..........*******...............
........***********.............
.......**************...........
.......*****.....*****..........
......*****.......*****.........
......****.........****.........
......****..........***.........
......****..........***.........
......*.***........***..........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......****************..........
......******........***.........
......*****..........**.........
......*****..........***........
......*****..........***........
.......****..........***........
.......****.........****........
........****.......*****........
........***************.........
.........*************..........
...........*********............
.............****...............
................................
................................
................................
................................
```

```
P(O'¦O*) = 0.718
P(O*¦O') = 0.972
```

(d) Experiment 1, 4th iteration

Fig.10b . The output image from the experiment 1.

```
................................
................................
................................
......**************............
......**************.*..........
......******************........
......*******..*********........
......*****.......******........
......*****........******.......
.......***..........*****.......
......****..........****........
......*.***........*****........
......*.****......****..........
......*****************.........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........**.*.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......****.......*******.......
......******************........
.......*****************........
........****.**********.........
.........************...........
................................
................................
................................
................................

     P(O'¦O*) = 0.909
     P(O*¦O') = 0.978
```

(a) Experiment 2, 1st iteration

```
................................
................................
................................
......**************............
......**************............
......******************........
......*******..*********........
......******......******........
......*****........*****........
......****..........****........
......****..........****........
......*.***........*****........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......*****......*******.......
.......*****************........
.......*****************........
........***************.........
.........************...........
................................
................................
................................
................................

     P(O'¦O*) = 0.903
     P(O*¦O') = 0.972
```

(b) Experiment 2, 2nd iteration

Fig.11a . The output image from the experiment 2.

```
................................
................................
................................
......**************............
......**************............
......******************........
......*******..*********........
......******......******........
......*****........*****........
......****..........****........
......****..........****........
......*.***........*****........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......*****......*******.......
.......*****************........
.......*****************........
........***************.........
.........************...........
................................
................................
................................
................................

      P(O'¦O*) = 0.903
      P(O*¦O') = 0.972
```

(c) Experiment 2, 3rd iteration

```
................................
................................
................................
......**************............
......**************............
......******************........
......*******..*********........
......******......******........
......*****........*****........
......****..........****........
......****..........****........
......*.***........*****........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......*****......*******.......
.......*****************........
.......*****************........
........***************.........
.........************...........
................................
................................
................................
................................

      P(O'¦O*) = 0.903
      P(O*¦O') = 0.972
```

(d) Experiment 2, 4th iteration

Fig.11b. The output image from the experiment 2.

```
................................
................................
................................
......**.*.*********............
.........*********.*............
.......*.*****.***.*.***........
......******....****.*.*........
.......*.**........***..........
......*****..........****.......
.......***..........*.*.........
......****...........***........
........***........****.........
........***........***..........
......**************.*..........
......****************..........
......******.*********..........
......*************.***.........
......*****..........*.*........
......*****..........**.........
......*****..........*****......
......*****..........*****......
.......**.*..........*****......
......*****.........******......
.......****........****.........
.........**..****..*****........
.........**..****.*****.........
.........**..**********.........
.........************...........
................................
................................
................................
................................
```

```
P(O'¦O*) = 0.750
P(O*¦O') = 0.996
```

(a) Experiment 3, 1st iteration

```
................................
................................
................................
...........*******.*............
.........*********..............
.........*****.*.*.*.*..........
.......*****....****............
.......*.**........*.*..........
.......****..........***........
.......*.*...........*..........
.......***...........***........
........***........****.........
........***........***..........
......**************.*..........
......****************..........
......******.*******.*..........
......*************.**..........
......*****..........*..........
......*****..........**.........
......*****..........*****......
......*****..........*****......
.......**.*..........*****......
.......****.........******......
.......****........****.........
.........**..****..****.........
.........**..****.*****.........
.............**********.........
.........************...........
................................
................................
................................
................................
```

```
P(O'¦O*) = 0.676
P(O*¦O') = 0.996
```

(b) Experiment 3, 2nd iteration

Fig.12a. The output image from the experiment 3.

```
................................
................................
................................
...........*****.*..............
.........*******.*..............
.........*****...*..............
.......*.***.....**.............
.........*......................
.......*.**.........*...........
.........*......................
.......*.*..........*...........
........***........***..........
........***........***..........
......**************.*..........
......**************.*..........
......******.*******.*..........
......*************.**..........
......*****..........*..........
......*****..........**.........
......*****..........*****......
......*****..........*****......
.......**.*..........*****......
.......****.........******......
.......****........****.........
.............****..****.........
.............****.*****.........
.............**********.........
...........**********...........
................................
................................
................................
................................

P(O'|O*) = 0.591
P(O*|O') = 0.995
```

(c) Experiment 3, 3rd iteration

```
................................
................................
................................
...........***..................
.........*****..................
.........*.***..................
.........*.*....................
.........*......................
.........*......................
.........*......................
.........*......................
........***........*.*..........
........***........*.*..........
......**************.*..........
......**************.*..........
......******.*******.*..........
......*************.**..........
......*****..........*..........
......*****..........**.........
......*****..........*****......
......*****..........*****......
.......**.*..........*****......
.......**.*.........******......
.......**.*........****.........
.............****..****.........
.............****.*****.........
.............**********.........
.............********...........
................................
................................
................................
................................

P(O'|O*) = 0.524
P(O*|O') = 0.994
```

(d) Experiment 3, 4th iteration

Fig.12b. The output image from the experiment 3.

```
................................
................................
................................
.......*************............
......****************..........
......******************........
......*******..*********........
......*****.......*******.......
......*****........******.......
......****..........*****.......
......****..........****........
......*****........*****........
......******.....*****..........
......*****************.........
......****************..........
.....******************.........
......******************........
......******........****........
......*****..........**.*.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
......******.....*******........
......*******************.......
.......*****************........
........***************.........
.........************...........
................................
................................
................................
................................

        P(O'|O*) = 0.938
        P(O*|O') = 0.973
```

(a) Experiment 4, 1st iteration

```
................................
................................
................................
.......*************............
......****************..........
......******************........
......*******..*********........
......******......*******.......
......*****........*****........
......****..........*****.......
......****..........****........
......*****........*****........
......******.....*****..........
......*****************.........
......*****************.........
......*****************.........
......******************........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
......******.....*******........
......*******************.......
.......*****************........
........***************.........
.........************...........
................................
................................
................................
................................

        P(O'|O*) = 0.938
        P(O*|O') = 0.970
```

(b) Experiment 4, 2nd iteration

Fig.13a. The output image from the experiment 4.

```
................................
................................
................................
.......*************............
......****************..........
......******************........
......*******..*********........
......******......******........
......*****........*****........
......****..........****........
......****..........****........
......*****........*****........
......******......*****.........
......*****************.........
......*****************.........
......*****************.........
......******************........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
......******......*******.......
......*******************.......
.......*****************........
........***************.........
.........************...........
................................
................................
................................
................................

     P(O'|O*) = 0.932
     P(O*|O') = 0.969
```

(c) Experiment 4, 3rd iteration

```
................................
................................
................................
.......*************............
......****************..........
......******************........
......*******..*********........
......******......******........
......*****........*****........
......****..........****........
......****..........****........
......*****........*****........
......******......*****.........
......*****************.........
......*****************.........
......*****************.........
......******************........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
......******......*******.......
......*******************.......
.......*****************........
........***************.........
.........************...........
................................
................................
................................
................................

     P(O'|O*) = 0.932
     P(O*|O') = 0.969
```

(d) Experiment 4, 4th iteration

Fig.13b. The output image from the experiment 4.

```
................................
................................
................................
......****************..........
......****************..........
......******************........
......********.**********.......
......******....*********.......
......*****........******.......
......*****.........*****.......
......*****........******.......
......*****........******.......
......******......*****.........
......*****************.........
......*****************.........
.....******************.........
......******************........
......******........*****.......
......*****.........*****.......
......*****..........*****......
......*****..........*****......
......*****.........******......
......*****........*******......
......******...**.********......
......*******************.......
.......*****************........
.......*****************........
.........*************..........
................................
................................
................................
................................
```

```
P(O'¦O*) = 0.974
P(O*¦O') = 0.940
```

(a) Experiment 5, 1st iteration

```
................................
................................
................................
......****************..........
......*****************.........
......*******************.......
......*******************.......
......*******...*********.......
......******......*******.......
......*****........******.......
......*****........******.......
......******......*******.......
......*******....*******........
......*****************.........
......*****************.........
......******************........
......*******************.......
......*******......******.......
......******........******......
......*****.........******......
......*****.........******......
......*****........*******......
......******......********......
......*******.************......
......********************......
......*******************.......
.......*****************........
........***************.........
................................
................................
................................
................................
```

```
P(O'¦O*) = 0.991
P(O*¦O') = 0.880
```

(b) Experiment 5, 2nd iteration

Fig.14a. The output image from the experiment 5.

```
................................
................................
................................
......*****************.........
......******************........
......*******************.......
......*******************.......
......********.**********.......
......*******....********.......
......******......*******.......
......******......*******.......
......*******....********.......
......********..*********.......
......******************........
......******************........
......*******************.......
......*******************.......
......********....********......
......*******......*******......
......******........******......
......*****........*******......
......******......********......
......*******....*********......
......********************......
......********************......
......********************......
......*******************.......
.......*****************........
................................
................................
................................
................................

     P(O'¦O*) = 0.997
     P(O*¦O') = 0.811
```

(c) Experiment 5, 3rd iteration

```
................................
................................
................................
......******************........
......*******************.......
......*******************.......
......*******************.......
......*******************.......
......********..*********.......
......*******....********.......
......*******....********.......
......********..*********.......
......*******************.......
......*******************.......
......*******************.......
......*******************.......
......********************......
......*********..*********......
......********....********......
......*******......*******......
......******......********......
......*******....*********......
......********..**********......
......********************......
......********************......
......********************......
......********************......
......*******************.......
................................
................................
................................
................................

     P(O'¦O*) = 1.000
     P(O*¦O') = 0.757
```

(d) Experiment 5, 4th iteration

Fig.14b. The output image from the experiment 5.

```
................................
................................
................................
.......*************............
......**************.*..........
......*****************.........
......*******..*********........
......*****.......******........
......*****........******.......
.......***..........*****.......
......****..........****........
......*.***........*****........
......*.****......****..........
......*****************.........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........**.*.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......****.......*******.......
......******************........
.......*****************........
........****.**********.........
.........************...........
................................
................................
................................
................................

     P(O'¦O*) = 0.903
     P(O*¦O') = 0.978
```

(a) Experiment 6, 1st iteration

```
................................
................................
................................
.......************.............
......**************............
......*****************.........
......*******..*********........
......******......******........
......*****........*****........
......****..........****........
......****..........****........
......*.***........*****........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......*****......*******.......
.......*****************........
.......*****************........
........***************.........
.........************...........
................................
................................
................................
................................

     P(O'¦O*) = 0.894
     P(O*¦O') = 0.971
```

(b) Experiment 6, 2nd iteration

Fig.15a. The output image from the experiment 6.

```
................................    ................................
................................    ................................
................................    ................................
.......************.............    .......************.............
......**************............    ......**************............
......*****************.........    ......*****************.........
......*******..*********........    ......*******..*********........
......******......******........    ......******......******........
......*****........*****........    ......*****........*****........
......****..........****........    ......****..........****........
......****..........****........    ......****..........****........
......*.***........****.........    ......*.***........****.........
......*.****......****..........    ......*.****......****..........
......****************..........    ......****************..........
......****************..........    ......****************..........
......****************..........    ......****************..........
......*****************.........    ......*****************.........
......******........****........    ......******........****........
......*****..........****.......    ......*****..........****.......
......*****..........*****......    ......*****..........*****......
......*****..........*****......    ......*****..........*****......
......*****..........*****......    ......*****..........*****......
......*****.........******......    ......*****.........******......
.......*****......*******.......    .......*****......*******.......
.......*****************........    .......*****************........
.......*****************........    .......*****************........
........***************.........    ........***************.........
.........************...........    .........************...........
................................    ................................
................................    ................................
................................    ................................
................................    ................................

    P(O'¦O*) = 0.891                    P(O'¦O*) = 0.891
    P(O*¦O') = 0.971                    P(O*¦O') = 0.971
```

(c) Experiment 6, 3rd iteration   (d) Experiment 6, 4th iteration

Fig.15b. The output image from the experiment 6.

```
................................    ................................
................................    ................................
................................    ................................
.......*************............    ........***********.............
......**************.*..........    .......*************............
......*****************.........    ......****************..........
......*******..*********........    ......*******..*********........
......*****.......******........    ......******......******........
......*****........******.......    ......*****........*****........
.......***..........*****.......    ......****..........****........
......****..........****........    ......****..........****........
......*.***.........****........    ......*.***........****.........
......*.****......****..........    ......*.****......****..........
......*****************.........    ......****************..........
......****************..........    ......****************..........
......****************..........    ......****************..........
......*****************.........    ......*****************.........
......******........****........    ......******........****........
......*****..........**.*.......    ......*****..........****.......
......*****..........*****......    ......*****..........*****......
......*****..........*****......    ......*****..........*****......
......*****..........*****......    ......*****..........*****......
......*****.........******......    ......*****.........******......
.......****.......*******.......    .......*****......*******.......
......******************........    .......*****************........
.......*****************........    .......****************.........
........****.**********.........    ........**************..........
.........************...........    .........************...........
................................    ................................
................................    ................................
................................    ................................
................................    ................................

      P(O'|O*) = 0.903                    P(O'|O*) = 0.876
      P(O*|O') = 0.981                    P(O*|O') = 0.971

(a) Experiment 7, 1st iteration     (b) Experiment 7, 2nd iteration
```

Fig.16a. The output image from the experiment 7.

```
................................
................................
................................
........***********.............
........************............
......****************..........
......*******..********.........
......******......******........
......*****........*****........
......****..........****........
......****..........****........
......*.***........****.........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......*****......*******.......
.......*****************........
.......****************.........
.........*************..........
.........************...........
................................
................................
................................
................................

      P(O'|O*) = 0.868
      P(O*|O') = 0.970
```

(c) Experiment 7, 3rd iteration

```
................................
................................
................................
.........**********.............
........************............
.......***************..........
......*******..*******..........
......******......******........
......*****........*****........
......****..........****........
......****..........****........
......*.***........****.........
......*.****......****..........
......****************..........
......****************..........
......****************..........
......*****************.........
......******........****........
......*****..........****.......
......*****..........*****......
......*****..........*****......
......*****..........*****......
......*****.........******......
.......*****......*******.......
.......*****************........
........***************.........
.........*************..........
..........***********...........
................................
................................
................................
................................

      P(O'|O*) = 0.853
      P(O*|O') = 0.970
```

(d) Experiment 7, 4th iteration

Fig.16b. The output image from the experiment 7.

```
................................    ................................
................................    ................................
................................    ................................
.......**************...........    .......**************...........
......****************..........    ......****************..........
......*****************.........    ......*****************.........
......******************........    ......******************........
......******.....********.......    ......*******...*********.......
......*****........******.......    ......******......*******.......
......*****.........*****.......    ......*****........******.......
......*****........*****........    ......*****........******.......
......*****........*****........    ......******......******........
......******......******........    ......*******....*******........
......*****************.........    ......******************........
......*****************.........    ......*****************.........
......*****************.........    ......*****************.........
......*****************.........    ......******************........
......******.......******.......    ......*******.....*******.......
......*****.........*****.......    ......******........*****.......
......*****..........*****......    ......*****.........******......
......*****..........*****......    ......*****.........******......
......*****.........******......    ......*****.........******......
......*****.........******......    ......******......*******......
......******...***********......    ......*******.************......
......*******************.......    ......*******************.......
.......*****************........    .......*****************........
.......****************.........    ........***************.........
.........*************..........    .........*************..........
................................    ................................
................................    ................................
................................    ................................
................................    ................................

P(O'¦O*) = 0.962                    P(O'¦O*) = 0.965
P(O*¦O') = 0.945                    P(O*¦O') = 0.896
```

(a) Experiment 8, 1st iteration   (b) Experiment 8, 2nd iteration

Fig.17a. The output image from the experiment 8.

```
................................
................................
................................
.......**************...........
......****************..........
......*****************.........
......******************........
......********.**********.......
......******.....********.......
......******......*******.......
......******......*******.......
......******......*******.......
......********..********........
......******************........
......******************........
......******************........
......******************........
......********...********.......
......******.......*******......
......******........******......
......*****.........******......
......******.......*******......
......******......********......
......********************......
......*******************.......
.......*****************........
........***************.........
.........*************..........
................................
................................
................................
................................

     P(O'¦O*) = 0.971
     P(O*¦O') = 0.853
```

(c) Experiment 8, 3rd iteration

```
................................
................................
................................
.......**************...........
......****************..........
......*****************.........
......******************........
......*******************.......
......*******...*********.......
......******.....********.......
......******......*******.......
......*******....********.......
......*******************.......
......******************........
......******************........
......******************........
......*******************.......
......*********.*********.......
......*******.....********......
......******.......*******......
......******.......*******......
......******.......*******......
......*******....*********......
......********************......
......*******************.......
.......*****************........
........***************.........
.........*************..........
................................
................................
................................
................................

     P(O'¦O*) = 0.971
     P(O*¦O') = 0.813
```

(d) Experiment 8, 4th iteration

Fig.17b. The output image from the experiment 8.

```
................................
...........***********..........
........**************.*........
.......****************.*.......
......*****************..*......
.....**********************.....
....************************....
...**************************...
..*********.....**************..
..********............********..
.*******...............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********........************.
.**********........************.
.************......************.
.************......************.
..***********.....*************.
..*****************************.
..*****************************.
..****************************..
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.**************.*.......
........*.*************.........
..........*****.***.............
................................

   P(O'¦O*) = 0.948
   P(O*¦O') = 0.997
```

(a) Experiment 9, 1st iteration

```
................................
............*********...........
.........*************..........
.......****************.*.......
......******************.*......
.....**********************.....
....************************....
...**************************...
...********......************...
..********............********..
..*******..............*******..
.********..............*******..
.********..............********.
.*********............*********.
.*********...........**********.
.**********.........***********.
.**********........************.
.***********.......************.
..***********......************.
..************....*************.
..*****************************.
..****************************..
..****************************..
...**************************...
...**************************...
....************************....
.....**********************.....
......********************......
.......*.**************.........
..........************..........
...........******...............
................................

   P(O'¦O*) = 0.920
   P(O*¦O') = 0.997
```

(b) Experiment 9, 2nd iteration

Fig.18a. The output image from the experiment 9.

```
................................
.............*******............
..........************..........
........***************.........
......******************.*......
.....**********************.....
....************************....
....************************....
...********.......***********...
...*******............*******...
..*******..............*******..
..*******..............*******..
.********..............*******..
.********.............*********.
.*********...........**********.
.*********..........***********.
.**********.........***********.
..**********.......************.
..***********......************.
..************....*************.
..****************************..
..****************************..
...**************************...
...**************************...
....************************....
....************************....
.....**********************.....
......*******************.......
.........**************.........
..........***********...........
............****................
................................

     P(O'¦O*) = 0.880
     P(O*¦O') = 0.997
```

(c) Experiment 9, 3rd iteration

```
................................
..............*****.............
...........**********...........
.........**************.........
.......*****************........
......********************......
.....**********************.....
....************************....
....*******........*********....
...*******............*******...
...******..............******...
..*******..............*******..
..*******..............*******..
.********.............********..
.********............**********.
.*********...........**********.
..*********.........***********.
..**********........***********.
..***********......************.
..************....************..
..****************************..
...**************************...
...**************************...
....************************....
....************************....
.....**********************.....
......********************......
.......*****************........
.........*************..........
...........*********............
.............**.................
................................

     P(O'¦O*) = 0.821
     P(O*¦O') = 0.996
```

(d) Experiment 9, 4th iteration

Fig.18b. The output image from the experiment 9.

```
................................
...........***********..........
........**************.*........
.......****************.*.......
......*****************..*......
.....**********************.....
....************************....
...**************************...
..*********.....**************..
.*********............*********.
.*******...............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********........************.
.**********.*......************.
.************......************.
.************......************.
.************.....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.**************.*.......
........*.*************.........
..........*****.***.*...........
................................

P(O'¦O*) = 0.957
P(O*¦O') = 0.997
```

(a) Experiment 10, 1st iteration

```
................................
...........***********..........
........**************..........
.......****************.*.......
......******************.*......
.....**********************.....
....************************....
...**************************...
..*********.....**************..
.*********............********..
.********..............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********........************.
.**********........************.
.************......************.
.************......************.
.*************....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.**************.........
..........*************.........
..........*******.*.............
................................

P(O'¦O*) = 0.951
P(O*¦O') = 0.995
```

(b) Experiment 10, 2nd iteration

Fig.19a. The output image from the experiment 10.

```
................................
...........***********..........
........**************..........
.......****************.........
......******************.*......
.....**********************.....
....************************....
...**************************...
..*********.....**************..
.*********............********..
.********..............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********........************.
.**********........************.
.************......************.
.************......************.
.*************....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.........**************.........
..........*************.........
..........*******...............
................................

    P(O'¦O*) = 0.946
    P(O*¦O') = 0.995
```

(c) Experiment 10, 3rd iteration

```
................................
...........***********..........
........**************..........
.......****************.........
......******************........
.....**********************.....
....************************....
...**************************...
..*********.....**************..
.*********............********..
.********..............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********........************.
.**********........************.
.************......************.
.************......************.
.*************....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.........**************.........
..........*************.........
..........*******...............
................................

    P(O'¦O*) = 0.944
    P(O*¦O') = 0.995
```

(d) Experiment 10, 4th iteration

Fig.19b. The output image from the experiment 10.

```
................................
............**********..........
.........****...****............
.......*.**.********....*.......
.........****.*.*.***....*......
.......*..*******..******.*.....
.......*************.**....*....
...*****.*********.*.********...
..******.*......*.*..*********..
.****.**.*............*********.
..*....*...............****.**..
.*****.*...............********.
.*.....**..............*.*****..
.*..*****.............*********.
.****.***............*.*.******.
.*********.........*******.**.*.
.**.*****.*.*......*.*.********.
.*********.........**.*********.
.***.********......************.
.*********.**......******.*****.
..*************.****.**********.
..*..******.***.******.****.***.
..*.*****..*.**************.*.*.
..*.****...*****************....
...****....*********.******.*...
....*..*****.***************....
.....*.*******.************.....
......**********.*********......
.......*.*.***********..........
.............*********..........
...........*..*.*.*.*...........
................................

     P(O'¦O*) = 0.761
     P(O*¦O') = 0.998
```

(a) Experiment 11, 1st iteration

```
................................
............*.******............
.........**.*...*.**............
............*.******............
.........****.*.*.***...........
..........*******..****...*.....
.......*.***********.**....*....
...*****.********....********...
..*.**.*.............*********..
..***..*..............********..
.......*...............****.**..
.***...*...............*******..
.......*...............*.*****..
.*..*.***.............*********.
.**.*.***............*.*.******.
.********..........*.*****.**.*.
.**.*****............*.********.
.*********.........*..*********.
.***.********......*.**********.
.*********.*.......******.*****.
..*************.****.**********.
..*..******.***.******.****.*.*.
....*****..*.**************.*...
....****...*****************....
....***....*********.******.....
....*..*****.**************.....
.....*.*.*****.************.....
.......*********.*********......
...........***********..........
.............*********..........
................................
................................

     P(O'¦O*) = 0.671
     P(O*¦O') = 0.998
```

(b) Experiment 11, 2nd iteration

Fig.20a. The output image from the experiment 11.

```
................................
................****............
................................
............*.******............
..........***.*.*.***...........
..........*******..****.........
..........********.*.**.........
....**.*..*******....********...
.......*.............*********..
....*..*..............********..
.......*...............****.**..
.......*...............*******..
.......*...............*.*****..
......***..............********.
......***..............*.******.
.********..............***.**.*.
.**.*****..............********.
.*********............*********.
.***.*******.........**********.
.*********.*.......******.*****.
..*************.****.**********.
.....******..**.******.****.....
....*****..*.**************.....
....****...****************.....
....*.*....*********.******.....
.......*.***.**************.....
...........***.************.....
.......*.*******.*********......
...........***********..........
.............*********..........
................................
................................

      P(O'¦O*) = 0.590
      P(O*¦O') = 1.000
```

(c) Experiment 11, 3rd iteration

```
................................
................................
................................
.............*.**.*.............
..........*.*...................
..........*******..****.........
..........*******....**.........
..........*******....********...
.....................*********..
.......*..............********..
.......*...............****.**..
.......*...............*******..
.......*...............*.*****..
......***..............*.******.
......***..............*.******.
.********..............***.**.*.
....*****..............********.
.*********............*********.
.***.*******.........**********.
.*********.*.......******.*****.
..*************.****.**********.
.....******..*..******.****.....
....*****..*.**************.....
.....***...****************.....
...........*.*******.******.....
...........*.**************.....
...........*.*.************.....
...........*****.*********......
...........***********..........
.............*********..........
................................
................................

      P(O'¦O*) = 0.535
      P(O*¦O') = 1.000
```

(d) Experiment 11, 4th iteration

Fig.20b. The output image from the experiment 11.

```
................................
...........***********..........
........**************.*........
.......****************.*.......
......******************.*......
.....**********************.....
....************************....
...***************************..
..*********...*.**************..
.*********............*********.
.*******...............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********........************.
.***********.......************.
.************......************.
.************......************.
.*************....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.****************.......
........*.*************.........
..........*****.***.*...........
................................

     P(O'¦O*) = 0.961
     P(O*¦O') = 0.994
```

(a) Experiment 12, 1st iteration

```
................................
...........***********..........
........**************..........
.......****************.*.......
......******************.*......
.....**********************.....
....************************....
...***************************..
..*********.....**************..
.*********............********..
.********..............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********........************.
.***********.......************.
.************......************.
.************......************.
.*************....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.****************.......
........*.*************.........
..........*******.*.............
................................

     P(O'¦O*) = 0.957
     P(O*¦O') = 0.994
```

(b) Experiment 12, 2nd iteration

Fig.21a. The output image from the experiment 12.

```
................................    ................................
...........***********..........    ...........***********..........
........**************..........    ........**************..........
.......****************.........    .......****************.........
......******************.*......    ......******************........
.....**********************.....    .....**********************.....
....************************....    ....************************....
...***************************..    ...***************************..
..*********.....**************..    ..*********.....**************..
.*********............********..    .*********............********..
.********..............*******..    .********..............*******..
.********..............********.    .********..............********.
.********..............********.    .********..............********.
.*********............*********.    .*********............*********.
.**********..........**********.    .**********..........**********.
.**********........************.    .**********........************.
.***********.......************.    .***********.......************.
.************......************.    .************......************.
.************......************.    .************......************.
.*************....*************.    .*************....*************.
..*****************************.    ..*****************************.
..*****************************.    ..*****************************.
..*****************************.    ..*****************************.
..****************************..    ..****************************..
...**************************...    ...**************************...
....************************....    ....************************....
.....**********************.....    .....**********************.....
......********************......    ......********************......
.......*.****************.......    .......*.****************.......
........*.*************.........    ........*.*************.........
..........*******...............    ..........*******...............
................................    ................................

     P(O'|O*) = 0.954                    P(O'|O*) = 0.952
     P(O*|O') = 0.994                    P(O*|O') = 0.994

(c) Experiment 12, 3rd iteration    (d) Experiment 12, 4th iteration
```

Fig.21b. The output image from the experiment 12.

```
................................
.........**************.........
.......***************.*........
.......****************.*.......
......******************.*......
.....***********************....
...**************************...
..****************************..
.************.*****************.
.**********..........**********.
.********.............*********.
.********..............********.
.********..............********.
.**********..........**********.
.**********.........***********.
.***********.......************.
.************......************.
.************......************.
.************......************.
.*************....*************.
.******************************.
..*****************************.
..*****************************.
..****************************..
..****************************..
....************************....
....************************....
......*********************.....
.......*.****************.......
........*****************.......
.........*************..........
................................

     P(O'¦O*) = 0.988
     P(O*¦O') = 0.965
```

(a) Experiment 13, 1st iteration

```
................................
........***************.........
.......*****************........
......*******************.......
.....*********************......
....*************************...
..****************************..
.******************************.
.******************************.
.***********........***********.
.*********...........**********.
.********.............*********.
.*********............*********.
.**********.........***********.
.***********.......************.
.************......************.
.************......************.
.************......************.
.*************....*************.
.**************..**************.
.******************************.
.******************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....***********************....
.......*******************......
.......******************.......
.........**************.........
................................

     P(O'¦O*) = 0.998
     P(O*¦O') = 0.923
```

(b) Experiment 13, 2nd iteration

Fig.22a. The output image from the experiment 13.

```
................................
.......*****************........
......*******************.......
.....*********************......
....***********************.....
...***************************..
.******************************.
.******************************.
.******************************.
.************......************.
.**********.........***********.
.*********...........**********.
.**********..........**********.
.***********.......************.
.************......************.
.************......************.
.************......************.
.*************....*************.
.**************..**************.
.******************************.
.******************************.
.******************************.
.******************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
......*********************.....
.......*******************......
........****************........
................................

P(O'¦O*) = 1.000
P(O*¦O') = 0.873
```

(c) Experiment 13, 3rd iteration

```
................................
......*******************.......
.....*********************......
....***********************.....
...*************************....
..*****************************.
.******************************.
.******************************.
.******************************.
.*************....*************.
.***********.......************.
.**********.........***********.
.***********........***********.
.************......************.
.************......************.
.************......************.
.*************....*************.
.**************..**************.
.******************************.
.******************************.
.******************************.
.******************************.
.******************************.
.******************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
.....***********************....
......*********************.....
.......******************.......
................................

P(O'¦O*) = 1.000
P(O*¦O') = 0.832
```

(d) Experiment 13, 4th iteration

Fig.22b. The output image from the experiment 13.

```
................................
...........***********..........
........**************.*........
.......****************.*.......
......*****************..*......
.....**********************.....
....************************....
...**************************...
..*********.....**************..
.*********............*********.
.*******...............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********........************.
.***********.......************.
.************......************.
.************......************.
..***********.....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.**************.*.......
........*.*************.........
..........*****.***.............
................................
```

P(O'¦O*) = 0.954
P(O*¦O') = 0.997

(a) Experiment 14, 1st iteration

```
................................
...........**********...........
........**************..........
.......****************.*.......
......******************.*......
.....**********************.....
....************************....
...**************************...
..*********.....**************..
.*********............********..
.********..............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********.........***********.
.***********.......************.
.************......************.
.************......************.
..************....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.**************.........
..........************..........
...........******.*.............
................................
```

P(O'¦O*) = 0.946
P(O*¦O') = 0.997

(b) Experiment 14, 2nd iteration

Fig.23a. The output image from the experiment 14.

```
................................
...........**********...........
........**************..........
.......****************.........
......******************.*......
.....**********************.....
....************************....
...**************************...
..*********.....**************..
.*********............********..
.********..............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********.........***********.
.***********.......************.
.************......************.
.************......************.
..************....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.........**************.........
..........************..........
...........******...............
................................

     P(O'|O*) = 0.941
     P(O*|O') = 0.997
```

(c) Experiment 14, 3rd iteration

```
................................
...........**********...........
........**************..........
.......****************.........
......******************........
.....**********************.....
....************************....
...**************************...
..*********.....**************..
.*********............********..
.********..............*******..
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.**********.........***********.
.***********.......************.
.************......************.
.************......************.
..************....*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.........**************.........
..........************..........
...........******...............
................................

     P(O'|O*) = 0.940
     P(O*|O') = 0.997
```

(d) Experiment 14, 4th iteration

Fig.23b. The output image from the experiment 14.

```
................................
...........************.........
........**************.*........
.......****************.*.......
......*****************..*......
.....**********************.....
....************************....
...**************************.*.
..***********.*.**************..
.*********............*********.
.*********.............********.
.********..............********.
.********..............********.
.*********............*********.
.**********..........**********.
.***********.......************.
.************......************.
.************......************.
.************......************.
..************....*************.
..*****************************.
..*****************************.
.******************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.**************.*.......
........*.**************........
.........************...........
................................
```

```
P(O'¦O*) = 0.975
P(O*¦O') = 0.992
```

(a) Experiment 15, 1st iteration

```
................................
...........************.........
........**************.*........
.......****************.*.......
......******************.*......
.....**********************.....
....************************....
...**************************...
..*************.**************..
.**********...........*********.
.*********.............********.
.*********.............********.
.*********.............********.
.*********............*********.
.**********..........**********.
.***********........***********.
.************......************.
.************......************.
.*************.....************.
..*************...*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.****************.......
........*.**************........
.........************...........
................................
```

```
P(O'¦O*) = 0.978
P(O*¦O') = 0.988
```

(b) Experiment 15, 2nd iteration

Fig.24a. The output image from the experiment 15.

```
................................
...........************.........
........**************.*........
.......****************.*.......
......********************......
.....**********************.....
....************************....
...**************************...
..****************************..
.***********..........*********.
.**********............********.
.*********.............********.
.*********.............********.
.**********...........*********.
.**********..........**********.
.***********........***********.
.************......************.
.*************.....************.
.*************.....************.
..**************..*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.****************.......
........*.**************........
.........*************..........
................................

      P(O'¦O*) = 0.983
      P(O*¦O') = 0.980
```

(c) Experiment 15, 3rd iteration

```
................................
...........************.........
........**************.*........
.......******************.......
......********************......
.....**********************.....
....************************....
...**************************...
..****************************..
.************.........*********.
.**********............********.
.**********............********.
.**********............********.
.**********...........*********.
.***********.........**********.
.***********........***********.
.************......************.
.*************.....************.
.**************....************.
..***************.*************.
..*****************************.
..*****************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......*.****************.......
........*.**************........
.........**************.........
................................

      P(O'¦O*) = 0.986
      P(O*¦O') = 0.971
```

(d) Experiment 15, 4th iteration

Fig.24b. The output image from the experiment 15.

```
................................    ................................
.........**************.........    .........**************.........
........**************.*........    ........**************.*........
.......****************.*.......    .......****************.*.......
......*****************..*......    ......******************.*......
.....**********************.....    .....**********************.....
....************************....    ....************************....
...**************************.*.    ...**************************...
..***********.****************..    ..****************************..
.*********...........**********.    .**********.........***********.
.*********.............********.    .*********............*********.
.********..............********.    .*********............*********.
.********.............*********.    .*********............*********.
.*********............*********.    .*********...........**********.
.**********..........**********.    .**********..........**********.
.***********.......************.    .***********........***********.
.************......************.    .************......************.
.************......************.    .************......************.
.************......************.    .*************....*************.
.*************....*************.    .**************..**************.
..*****************************.    .******************************.
..*****************************.    ..*****************************.
.******************************.    ..*****************************.
..****************************..    ..****************************..
...**************************...    ...**************************...
....************************....    ....************************....
.....**********************.....    .....**********************.....
......********************......    ......********************......
.......*.**************.*.......    .......******************.......
........*.**************........    ........*.**************........
.........************...........    .........************...........
................................    ................................

     P(O'¦O*) = 0.981                    P(O'¦O*) = 0.988
     P(O*¦O') = 0.989                    P(O*¦O') = 0.976
```

(a) Experiment 16, 1st iteration   (b) Experiment 16, 2nd iteration

Fig.25a. The output image from the experiment 16.

```
................................
.........**************.........
........**************.*........
.......****************.*.......
......********************......
.....**********************.....
....************************....
...**************************...
..****************************..
.***********.......************.
.**********..........**********.
.*********............*********.
.*********...........**********.
.**********..........**********.
.**********.........***********.
.***********........***********.
.************......************.
.*************....*************.
.*************....*************.
.******************************.
.******************************.
.******************************.
..*****************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......******************.......
........****************........
.........*************..........
................................

      P(O'¦O*) = 0.994
      P(O*¦O') = 0.960
```

(c) Experiment 16, 3rd iteration

```
................................
.........**************.........
........**************.*........
.......******************.......
......********************......
.....**********************.....
....************************....
...**************************...
..****************************..
.************.....*************.
.**********.........***********.
.**********..........**********.
.**********..........**********.
.**********.........***********.
.***********........***********.
.***********.......************.
.************......************.
.*************....*************.
.**************..**************.
.******************************.
.******************************.
.******************************.
.******************************.
..****************************..
...**************************...
....************************....
.....**********************.....
......********************......
.......******************.......
........****************........
.........**************.........
................................

      P(O'¦O*) = 0.998
      P(O*¦O') = 0.945
```

(d) Experiment 16, 4th iteration

Fig.25b. The output image from the experiment 16.

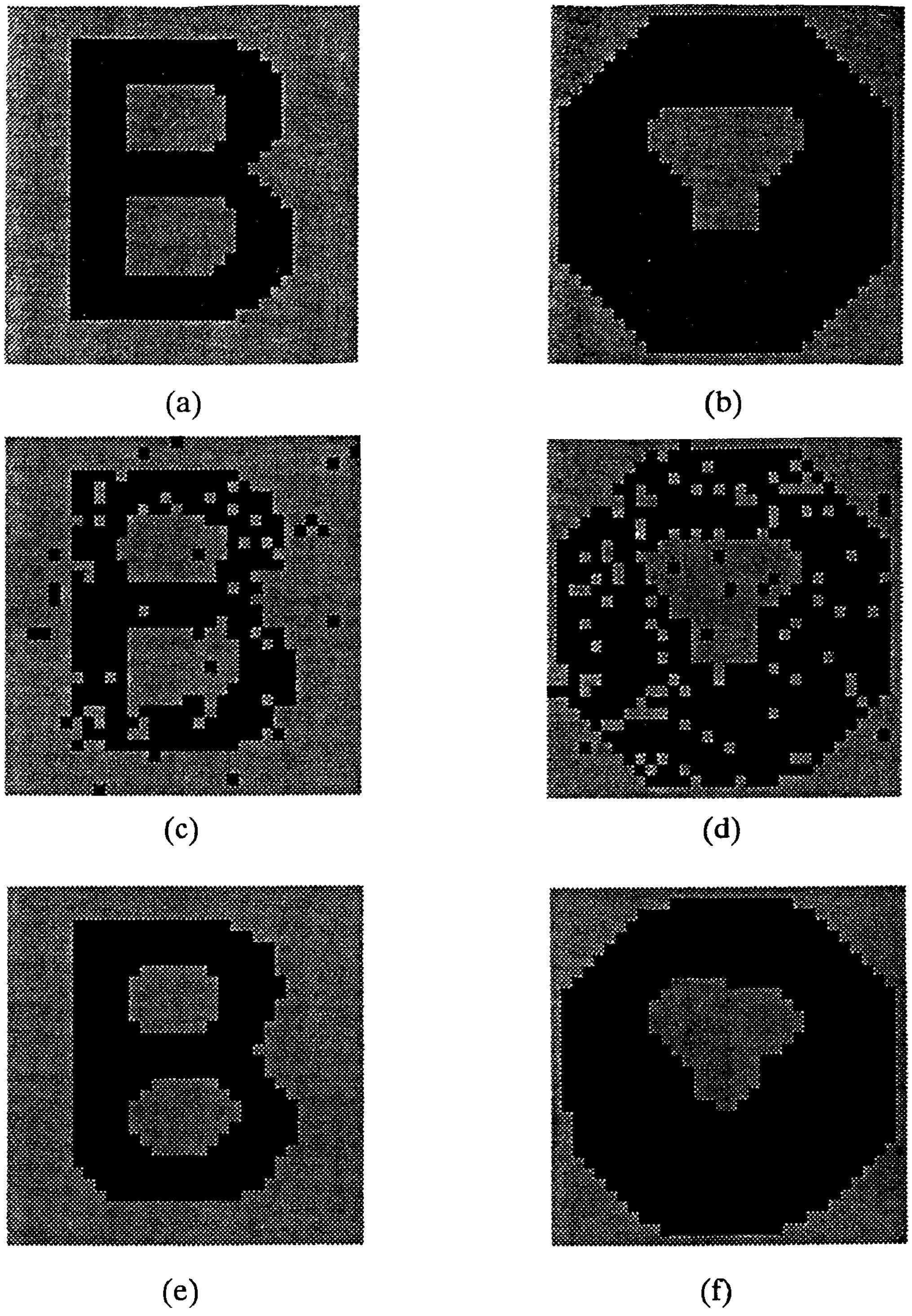

Fig.26. The effect with different vigilance setting in iterations. The training pattern set is A. The vigilance in 1st iteration are A1,A2,A3,A4 = 0.8 and A5,A6,A7,A8=0.9 , but in 2nd iteration are all be set to 0.6. (a) & (b) are original images. (c) & (d) are noisy images. (e) and (f) are the output images of the region detection layer in the network.

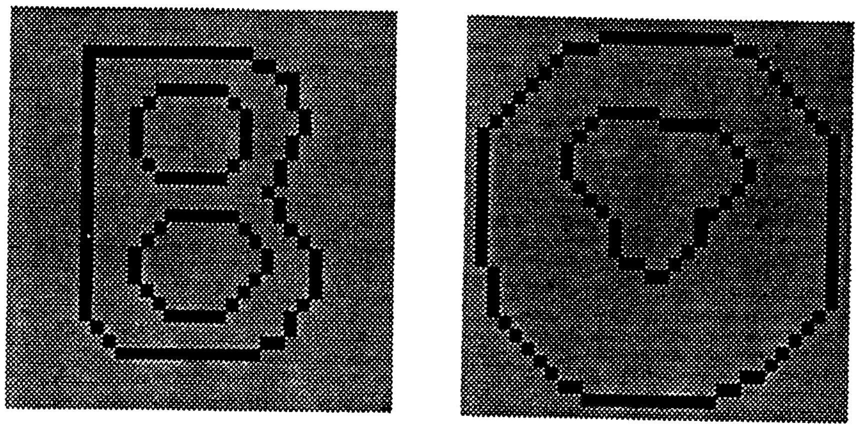

Fig.27. The output images from the 4th layer in the network.

CHAPTER XI. TABLES

Experiment 1 :

Noisy image	"B"			
Training pattern set	A			
Vigilance setting	A1=0.8	A2=0.8	A3=0.8	A4=0.8
	A5=1.0	A6=1.0	A7=1.0	A8=1.0

Table 1. Performance of the 1st experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.859	0.921
1st	0.891	0.981
2nd	0.832	0.976
3rd	0.776	0.974
4th	0.718	0.972

Experiment 2 :

Noisy image	"B"			
Training pattern set	A			
Vigilance setting	A1=0.8	A2=0.8	A3=0.8	A4=0.8
	A5=0.9	A6=0.9	A7=0.9	A8=0.9

Table 2. Performance of the 2nd experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.859	0.921
1st	0.909	0.978
2nd	0.903	0.972
3rd	0.903	0.972
4th	0.903	0.972

Experiment 3 :

Noisy image	"B"			
Training pattern set	A			
Vigilance setting	A1=1.0	A2=1.0	A3=1.0	A4=1.0
	A5=1.0	A6=1.0	A7=1.0	A8=1.0

Table 3. Performance of the 3rd experiment in different iteration runs.

Iteration	$P(O' \mid O^*)$	$P(O^* \mid O')$
0th	0.859	0.921
1st	0.750	0.996
2nd	0.676	0.996
3rd	0.591	0.995
4th	0.524	0.994

Experiment 4 :

Noisy image	"B"			
Training pattern set	B			
Vigilance setting	B1=0.8	B2=0.8	B3=0.8	B4=0.8
	B5=1.0	B6=1.0	B7=1.0	B8=1.0
	B9=0.8	B10=0.8	B11=0.8	B12=0.8

Table 4. Performance of the 4th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.859	0.921
1st	0.938	0.973
2nd	0.938	0.970
3rd	0.932	0.969
4th	0.932	0.969

Experiment 5 :

Noisy image	"B"			
Training pattern set	B			
Vigilance setting	B1=0.8	B2=0.8	B3=0.8	B4=0.8
	B5=0.8	B6=0.8	B7=0.8	B8=0.8
	B9=0.8	B10=0.8	B11=0.8	B12=0.8

Table 5. Performance of the 5th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.859	0.921
1st	0.974	0.940
2nd	0.991	0.880
3rd	0.997	0.811
4th	1.000	0.757

Experiment 6 :

Noisy image	"B"			
Training pattern set	B			
Vigilance setting	B1=0.8	B2=0.8	B3=0.8	B4=0.8
	B5=1.0	B6=1.0	B7=1.0	B8=1.0
	B9=1.0	B10=1.0	B11=1.0	B12=1.0

Table 6. Performance of the 6th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.859	0.921
1st	0.903	0.978
2nd	0.894	0.971
3rd	0.891	0.971
4th	0.891	0.971

Experiment 7 :

Noisy image	"B"			
Training pattern set	C			
Vigilance setting	C1=0.8	C2=0.8	C3=0.8	C4=0.8
	C5=1.0	C6=1.0	C7=1.0	C8=1.0
	C9=1.0	C10=1.0	C11=1.0	C12=1.0

Table 7. Performance of the 7th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.859	0.921
1st	0.903	0.981
2nd	0.876	0.971
3rd	0.868	0.970
4th	0.853	0.970

Experiment 8 :

Noisy image	"B"			
Training pattern set	C			
Vigilance setting	C1=0.8	C2=0.8	C3=0.8	C4=0.8
	C5=0.8	C6=0.8	C7=0.8	C8=0.8
	C9=0.8	C10=0.8	C11=0.8	C12=0.8

Table 8. Performance of the 8th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.859	0.921
1st	0.962	0.945
2nd	0.965	0.896
3rd	0.971	0.853
4th	0.971	0.813

Experiment 9 :

Noisy image	Bulb			
Training pattern set	A			
Vigilance setting	A1=0.8	A2=0.8	A3=0.8	A4=0.8
	A5=1.0	A6=1.0	A7=1.0	A8=1.0

Table 9. Performance of the 9th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.855	0.977
1st	0.948	0.997
2nd	0.920	0.997
3rd	0.880	0.997
4th	0.821	0.996

Experiment 10 :

Noisy image	Bulb			
Training pattern set	A			
Vigilance setting	A1=0.8	A2=0.8	A3=0.8	A4=0.8
	A5=0.9	A6=0.9	A7=0.9	A8=0.9

Table 10. Performance of the 10th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.855	0.977
1st	0.957	0.997
2nd	0.951	0.995
3rd	0.946	0.995
4th	0.944	0.995

Experiment 11 :

Noisy image	Bulb			
Training pattern set	A			
Vigilance setting	A1=1.0	A2=1.0	A3=1.0	A4=1.0
	A5=1.0	A6=1.0	A7=1.0	A8=1.0

Table 11. Performance of the 11th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.855	0.977
1st	0.761	0.998
2nd	0.671	0.998
3rd	0.590	1.000
4th	0.535	1.000

Experiment 12 :

Noisy image	Bulb			
Training pattern set	B			
Vigilance setting	B1=0.8	B2=0.8	B3=0.8	B4=0.8
	B5=1.0	B6=1.0	B7=1.0	B8=1.0
	B9=0.8	B10=0.8	B11=0.8	B12=0.8

Table 12. Performance of the 12th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.855	0.977
1st	0.961	0.994
2nd	0.957	0.994
3rd	0.954	0.994
4th	0.952	0.994

Experiment 13 :

Noisy image	Bulb			
Training pattern set	B			
Vigilance setting	B1=0.8	B2=0.8	B3=0.8	B4=0.8
	B5=0.8	B6=0.8	B7=0.8	B8=0.8
	B9=0.8	B10=0.8	B11=0.8	B12=0.8

Table 13. Performance of the 13th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.855	0.977
1st	0.988	0.965
2nd	0.998	0.923
3rd	1.000	0.873
4th	1.000	0.832

Experiment 14 :

Noisy image	Bulb			
Training pattern set	B			
Vigilance setting	B1=0.8	B2=0.8	B3=0.8	B4=0.8
	B5=1.0	B6=1.0	B7=1.0	B8=1.0
	B9=1.0	B10=1.0	B11=1.0	B12=1.0

Table 14. Performance of the 14th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.855	0.977
1st	0.954	0.997
2nd	0.946	0.997
3rd	0.941	0.997
4th	0.940	0.997

Experiment 15 :

Noisy image	Bulb			
Training pattern set	C			
Vigilance setting	C1=0.8	C2=0.8	C3=0.8	C4=0.8
	C5=1.0	C6=1.0	C7=1.0	C8=1.0
	C9=1.0	C10=1.0	C11=1.0	C12=1.0

Table 15. Performance of the 15th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.855	0.977
1st	0.975	0.992
2nd	0.978	0.988
3rd	0.983	0.980
4th	0.986	0.971

Experiment 16 :

Noisy image	Bulb			
Training pattern set	C			
Vigilance setting	C1=0.8	C2=0.8	C3=0.8	C4=0.8
	C5=0.8	C6=0.8	C7=0.8	C8=0.8
	C9=0.8	C10=0.8	C11=0.8	C12=0.8

Table 16. Performance of the 16th experiment in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.855	0.977
1st	0.981	0.989
2nd	0.988	0.976
3rd	0.994	0.960
4th	0.998	0.945

Table 17. Average of P(O'|O*) and P(O*|O') in 1st iteration

Exp. No.	"B" Image	Exp. No.	Bulb Image
1	0.936	9	0.973
2	0.944	10	0.977
3	0.873	11	0.880
4	0.956	12	0.978
5	0.957	13	0.977
6	0.941	14	0.976
7	0.942	15	0.984
8	0.954	16	0.985

Table 18. The performance order for "B" and bulb image.

"B"	$5 > 4 > 8 > 2 > 7 > 6 > 1 > 3$
Bulb	$8 > 7 > 4 > 2 = 5 > 6 > 1 > 3$

Test of different vigilance value in iterations

Noisy image	"B"			
Training pattern set	A			
Vigilance setting				
1st iteration	A1=0.8	A2=0.8	A3=0.8	A4=0.8
	A5=0.9	A6=0.9	A7=0.9	A8=0.9
2nd - 4th iterations	A1=0.6	A2=0.6	A3=0.6	A4=0.6
	A5=0.6	A6=0.6	A7=0.6	A8=0.6

Table 19. Performance of the different vigilance in different iteration runs.

Iteration	P(O'|O*)	P(O*|O')
0th	0.859	0.921
1st	0.909	0.978
2nd	0.947	0.944
3rd	0.947	0.915
4th	0.947	0.899

APPENDIX I.

The ART 1 SIMULATOR PROGRAM LISTING

This program will simulate the entire network as described in this paper, from noisy binary input image to edge detected output. The input training patterns must be stored in a previously created text file of the form:

```
<data><data><data><data><data>
<data><data><data><data><data>
<data><data><data><data><data>
<data><data><data><data><data>
<data><data><data><data><data>
```

<up to twelve training patterns as shown above are provided for>

Where: <data> is a one character representation of a binary value with the character '*' representing a value of one and any other character, except a space, representing a value of zero. The input image file must also be of the above form except that it will be 32 x 32 instead of 5 x 5. The program is interactive with the user being prompted for the necessary inputs when required. Flexibility is also provided for altering vigilance parameters and re-running, writing results to files, etc.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

/***********************************************************************/
                         EXTERNAL CONSTANTS
/***********************************************************************/

#define CATEGORIES 12
#define FEATURES 25
#define ALPHA 0.01
#define EPSILON 0.05

float VIGILANCE;


/***********************************************************************/
                         EXTERNAL VARIABLES
/***********************************************************************/

struct pattern
   {
    int template[FEATURES];           /* 5 x 5 pixel array              */
    struct pattern *nxtptr;
   };
struct pattern *INPUT_PATTERNS;

struct category_neuron
   {
    float vigilance;
    int active;              /* 0=inactive, 1=active               */
    float bottom_up[FEATURES];
    float output;
   };
struct category_neuron F_2[CATEGORIES];

struct feature_neuron
   {
    int top_down[CATEGORIES];
    int feature_value;            /* 1 or 0                             */
   };
struct feature_neuron F_1[FEATURES];

int X, T_X, H;
float criteria, per1, per2;
int active_nodes, match1, match2;
char inp_pic[36][36];
char oup_pic[32][32], org_pic[32][32];
/****************************************************************************/
                              FUNCTION DECLARATIONS
/****************************************************************************/

void train_network();
void get_training_patterns();
void get_input_picture();
int view_pixel(int i, int j);
int maxnet();
void compute_weighted_sums();
int apply_input(struct pattern *input_pointer);
void update_exemplar(int node);
void print_categories();
void ori_input();
void no_classification();
int activate_new_node();
void initialize_weighted_links();
void print_output();
void matching();
void see_input();
void write_to_file();
void filter();

main()
{
 char buffer[8], ans;


 get_training_patterns();
 initialize_weighted_links();
 train_network();
 print_categories();
 ori_input();
 get_input_picture();
 matching();
```

```
print_output();
buffer[0] = 3;
while (buffer[2] != '1')
  {
   gotoxy(1, 25);
   clreol();
   printf("<1>done      <2>write file <3>new vigilance\n");
   printf("<4>new file <5> retrain   <6>run           <7> edge detect:");
   cgets(buffer);
   switch (buffer[2])
     {
      case '1':
        clrscr();
        printf(" Are you sure ? (y / n)");
        ans= getch();
        if(ans=='Y' || ans=='y') exit(1);
        else buffer[2] = '0';
        break;
      case '2':
        write_to_file();
        buffer[2] = '0';
        break;
      case '3':
        print_categories();
        buffer[0] = 3;
        buffer[2] = '0';
        break;
      case '4':
        get_input_picture();
        print_output();
        buffer[2] = '0';
        break;
      case '5':
        get_training_patterns();
        initialize_weighted_links();
        train_network();
        print_categories();
        buffer[2] = '0';
        break;
      case '6':
        see_input();
        print_output();
        printf("\a\a\a");
```

```
            matching();
            buffer[2] = '0';
            break;
         case '7':
            filter();
            print_output();
            buffer[2] = '0';
        }
    }
}


void get_training_patterns()
/*****************************************************************************/
This function will get the 5 x 5 training patterns from a file and store them in a linked list
pointed to by INPUT_PATTERNS. It will also set the vigilance parameter to 1.0 to disable
updating of previously committed exemplars during the training phase.
/*****************************************************************************/
{
 struct pattern *temp_ptr, *prev_ptr=0;
 int i, ch=0;
 FILE *inp;
 char path[12];

 clrscr();
 system("dir/w *.pat");
 printf("\n\n");
 printf("Training file name >> ?");
 printf("\n\n");
 gotoxy(wherex()-1, 1);
 i = -1;
 do
    {
     i = i + 1;
     if (i == 13)
        {
         printf ("\nFilename too long! Shorten it and rerun...XXXXXXXX.XXX\n");
         exit(0);
        }
     path[i] = getche();
     if (path[i] == '\x08')
        i = i - 2;
    }
 while (path[i] != '\x0d');
 path[i] = NULL;
 INPUT_PATTERNS = (struct pattern *) malloc(sizeof(struct pattern));
 temp_ptr = INPUT_PATTERNS;
 inp = fopen(path, "r");
 if (inp == NULL)
    {
     printf ("\n\nTraining file %s doesn't exist!", path);
     exit(0);
    }
 else
    {
     while (ch != EOF)
        {
         for (i=0; i<FEATURES; i++)
            {
             while ((ch = getc(inp)) == '\x20' || ch == '\x0d' || ch == '\x0a');
             if (ch == EOF)
                {
                 prev_ptr->nxtptr = (struct pattern *)NULL;
                 break;
                }
             if (ch == '*')
                temp_ptr->template[i] = 1;
             else
                temp_ptr->template[i] = 0;
            }
         temp_ptr->nxtptr = (struct pattern *) malloc(sizeof(struct pattern));
         prev_ptr = temp_ptr;
         temp_ptr = temp_ptr->nxtptr;
        }
     fclose(inp);
    }
 for (i=0; i<CATEGORIES; i++)
    {
     F_2[i].vigilance = 1.0;
    }
 VIGILANCE = 1.0;
}
```

```
int maxnet()
/*****************************************************************************/
This function will select the category neuron with the highest exemplar match using lateral
inhibition and return an integer pointer into the array F_2 of category neurons.
/*****************************************************************************/
{
 int i, j, k, positive_nodes, last_positive_node=0;
 float temp, temp_alpha, temp_epsilon, temp_output[CATEGORIES];

 if (active_nodes == 0)
    return(activate_new_node());
 temp_alpha = ALPHA;
 temp_epsilon = EPSILON;
 positive_nodes = 0;
 while (positive_nodes != 1)
    {
     positive_nodes = 0;
     for (j=0; j<CATEGORIES; j++)
        {
         if (F_2[j].active == 1)
            {
             temp = 0;
             for (i=0; i<CATEGORIES; i++)
                {
                 if ((j != i ) && (F_2[i].active == 1))
                      temp = temp + F_2[i].output;
                }
             if (temp < 0)
                temp = temp * -1.0;

             if(F_2[j].vigilance!=0)
                 temp_output[j] = temp_alpha * (F_2[j].output - temp_epsilon * temp)
                      * (1.0/F_2[j].vigilance);
             else temp_output[j] = 0;

             if (temp_output[j] > 0)
                {
                 positive_nodes = positive_nodes + 1;
                 last_positive_node = j;
                }
             else
```

```
              temp_output[j] = 0;
           }
        }
     for (k=0; k<CATEGORIES; k++)
        {
         if (F_2[k].active == 1)
            F_2[k].output = temp_output[k];
        }
     if (positive_nodes == 0)
        {
         return(last_positive_node);
        }
    }
 for (j=0; j<CATEGORIES; j++)
    {
     if ((F_2[j].active == 1) && (F_2[j].output > 0))
        return(j);
    }
}


void compute_weighted_sums()
/*****************************************************************************/
This function will compute the outputs for the category neurons based upon the bottom
up weights for each category neuron and also the input pattern loaded into the
corresponding feature neurons.
/*****************************************************************************/
{
 int i, j;
 float temp_alpha;

 temp_alpha = ALPHA;
 for (i=0; i<CATEGORIES; i++)
    {
     if (F_2[i].active == 1)
        F_2[i].output = 0;
    }
 for (j=0; j<CATEGORIES; j++)
    {
     if (F_2[j].active == 1)
        {
         for (i=0; i<FEATURES; i++)
            {
```

```
            F_2[j].output = F_2[j].output + F_2[j].bottom_up[i] *
               (float)F_1[i].feature_value;
            }
          F_2[j].output = F_2[j].output * temp_alpha;
         }
     }
}


void update_exemplar(node)
/******************************************************************************/
This function will update both the top down and the bottom up weights for the selected
exemplar category "node".
/******************************************************************************/
int node;
{
 int i, coincedences;

 coincedences = 0;
 for (i=0; i<FEATURES; i++)
    {
     F_1[i].top_down[node] = F_1[i].top_down[node] * F_1[i].feature_value;
     coincedences = coincedences + F_1[i].top_down[node];
    }
 for (i=0; i<FEATURES; i++)
    {
     F_2[node].bottom_up[i] = (float)F_1[i].top_down[node] / (0.5 + (float)coincedences);
    }
}


int apply_input(input_pointer)
/******************************************************************************/
This function will get the input pattern stored in pattern->template and pointed to by
input_pointer and load it into the array of feature neurons->feature_value. A value of 0
will be returned if there are no more input patterns and a value of 1 will be returned if
there are more input patterns.
/******************************************************************************/
struct pattern *input_pointer;
{
 int i;

 if (input_pointer == (struct pattern *)NULL)
```

```
   return(0);
 else
   {
    for (i=0; i<FEATURES; i++)
      {
       F_1[i].feature_value = input_pointer->template[i];
      }
    return(1);
   }
}


void print_categories()
/*****************************************************************************/
This function will type the exemplar categories established during the training phase on the
screen and it will also get the individual vigilance parameters desired for each category.
/*****************************************************************************/
{
 int i, j, k, l;
 char ch, buffer[7];
 char *p;

 clrscr();
 gotoxy(23, 1);
 printf("Network Contour Template Exemplars");
 gotoxy(23, 2);
 printf("==================================");
 gotoxy(4, 3);
 for (i=0; i<CATEGORIES; i++)
   {
    printf("E[%d]", i);
    gotoxy(4+i*6, 4);
    printf("~~~~~");
    gotoxy(wherex()+1, 3);
   }
 gotoxy(4, 5);
 for (i=0; i<CATEGORIES; i++)
   {
   if (F_2[i].active == 1)
      {
      l = 0;
      for (j=0; j<5; j++)
         {
```

```
            for (k=0; k<5; k++)
               {
                if (F_1[l].top_down[i] == 1)
                   {
                    printf("*");
                   }
                else
                   {
                    printf(" ");
                   }
                l = l + 1;
               }
            gotoxy(wherex()-5, wherey()+1);
           }
        gotoxy(wherex()+6, wherey()-5);
       }
    else
       gotoxy(wherex()+6, wherey());
   }
 gotoxy(1, 25);
 printf("Enter the vigilance parameter for each exemplar {0 --> 1.0}");
 buffer[0] = 5;
 for (i=0; i<CATEGORIES; i++)
   {
    if (F_2[i].active == 1)
       {
        gotoxy(4+i*6, 11);
        p = cgets(buffer);
        F_2[i].vigilance = (float)atof(p);
       }
   }
 gotoxy(1, 25);
 clreol();
 printf("Enter >> <ret> to abort; <any key> to continue");
 ch = getch();
 if (ch == '\x0d')
   {
    clrscr();
    exit(0);
   }
}
```

```
void no_classification()
/*****************************************************************************/
This function will indicate when all the available category neurons have been allocated.
/*****************************************************************************/
{
 int ch, i, j, k;

 window(1, 6, 80, 25);
 clrscr();
 gotoxy(1, 1);
 k = 0;
 for (i=0; i<5; i++)
    {
     for (j=0; j<5; j++)
        {
         if (F_1[k].feature_value == 1)
            printf("*");
         else
            printf(" ");
         k = k + 1;
        }
     gotoxy(1, wherey()+1);
    }

 gotoxy(9, 1);
 printf("UNABLE TO CLASSIFY THIS INPUT PATTERN WITH THE AVAILABLE
CATEGORY");
 gotoxy(9, 2);
 printf("NEURONS AND THE SPECIFIED VIGILANCE PARAMETER. EITHER
LOWER THE");
 gotoxy(9, 3);
 printf("VIGILANCE PARAMETER OR ADD CATEGORY NEURONS TO CLASSIFY
THIS");
 gotoxy(9, 4);
 printf("PATTERN.");
 window(1, 1, 80, 1);
 gotoxy(58, 1);
 printf("p = %d / %d = %6.3f", T_X, X, criteria);
 window(1, 2, 80, 25);
 gotoxy(10, 1);
 printf("Hit any key to view next pattern or return to abort.");
 ch = getch();
 if (ch == '\x0d')
```

```
    {
     window(1, 1, 80, 25);
     clrscr();
     exit(0);
    }
}


int activate_new_node()
/*************************************************************************/
This function will activate a new category node and return an integer pointer into the array
F_2 of category neurons corresponding to that node. If no more category nodes are
available a value of -1 will be returned.
/*************************************************************************/
{
 int i;

 if (active_nodes != CATEGORIES)
    {
     active_nodes = active_nodes + 1;
     for (i=0; i<CATEGORIES; i++)
        {
         if (F_2[i].active == 0)
            {
             F_2[i].active = 1;
             return(i);
            }
        }
    }
 else
    return(-1);
}


void train_network()
/*************************************************************************/
This function will train the network by establishing exemplar categories for the training
patterns pointed to by INPUT_PATTERNS.
/*************************************************************************/
{
 struct pattern *pointer;
 int new_node, largest_node, i;
```

```
pointer = INPUT_PATTERNS;
active_nodes = 0;
new_node = 0;
while (apply_input(pointer) != 0)        /* go through all patterns     */
   {
    criteria = 0;
    while (criteria < VIGILANCE)
       {
        compute_weighted_sums();
        if (new_node == 0)
           largest_node = maxnet();
       new_node = 0;
        X = 0;
        for (i=0; i<FEATURES; i++)
           {
            X = X + F_1[i].feature_value;
           }
        T_X = 0;
        for (i=0; i<FEATURES; i++)
           {
            T_X = T_X + F_1[i].feature_value * F_1[i].top_down[largest_node];
           }

        criteria = (float)T_X/(float)X;
        if (criteria < VIGILANCE)
           {
            new_node = 1;
            largest_node = activate_new_node();
           }
        if (largest_node == -1)
           {
            no_classification();
            break;
           }
       }
    if (largest_node >= 0)
       {
           update_exemplar(largest_node);
       }
    pointer = pointer->nxtptr;
   }
}
```

```
void ori_input()
{
 int i,j;
 FILE *org;
 char ch, buf[14], *ptr;

     buf[0] = 13;
     clrscr();
     org = NULL;
     while (org == NULL)
        {
         gotoxy(1, 1);
         clreol();
         system("dir/w *.bin");
         printf("\n\n");
         printf("Enter the original binary input picture filename >> ");
        printf("\n\n");
        ptr = cgets(buf);
        org = fopen(ptr, "r");
        if (org == NULL)
        {
                gotoxy(1, 1);
                clreol();
                printf("File \"%s\" doesn't exist!  <space> retry    <ret> abort", ptr);
                ch = getch();
                if (ch == '\x0d')
                {
                        window(1, 1, 80, 25);
                        clrscr();
                        fclose(org);
                        exit(0);
                }
        }
    }

 for (i=0; i<32; i++)
    {
     if (ch == EOF)
        break;
     for (j=0; j<32; j++)
        {
         if (ch == EOF)
            break;
        while ((ch = getc(org)) == '\x0d' || ch == '\x0a');
        if (ch != EOF)
           {
            if (ch == '*')
               org_pic[i][j] = '*';
            else
               org_pic[i][j] = '.';
           }
        }
     }
fclose(org);

}



void get_input_picture()
/*************************************************************************/
This function will get the input picture from a file and store it in the array inp_pic.
/*************************************************************************/
{
 int i, j;
 FILE *inp;
 char ch, buffer[14];
 char *p;

 buffer[0] = 13;

 clrscr();
 inp = NULL;
 while (inp == NULL)
    {
     gotoxy(1, 1);
     clreol();
     system("dir/w *.bin");
     printf("\n\n");
     printf("Enter the binary input picture filename >> ");
     printf("\n\n");
     p = cgets(buffer);
     inp = fopen(p, "r");
     if (inp == NULL)
        {
         gotoxy(1, 1);
        clreol();
        printf("File \"%s\" doesn't exist!  <space> retry    <ret> abort", p);
        ch = getch();
        if (ch == '\x0d')
           {
            window(1, 1, 80, 25);
            clrscr();
            fclose(inp);
            exit(0);
           }
       }
    }

 for (i=2; i<34; i++)
    {
     if (ch == EOF)
        break;
     for (j=2; j<34; j++)
        {
         if (ch == EOF)
            break;
         while ((ch = getc(inp)) == '\x0d' || ch == '\x0a');
         if (ch != EOF)
            {
             if (ch == '*')
                inp_pic[i][j] = '*';
             else
                inp_pic[i][j] = '.';
            }
        }
    }
fclose(inp);

for (i=0; i<36; i++)
   {
    for (j=0; j<2; j++)
       {
        inp_pic[j][i] = '.';
        inp_pic[34+j][i] = '.';
        inp_pic[i][j] = '.';
        inp_pic[i][34+j] = '.';
       }
   }
```

```
for (i=0; i<32; i++)
   {
    for (j=0; j<32; j++)
       {
        oup_pic[i][j] = inp_pic[i+2][j+2];
       }
   }
}

int view_pixel(i, j)
/*****************************************************************************/
This function will determine, from the 5 x 5 neighborhood, if a pixel is close enough to
any of the exemplars and their corresponding vigilance. If so then a value of 1 will be
returned otherwise a value of 0 will be returned.
/****************************************************************************/
int i;
int j;
{
 int l, m, n, largest_node, x, p, q;

 n = 0;
 X = 0;
 for (l=i-2; l<i+3; l++)
    {
     for (m=j-2; m<j+3; m++)
        {
         if (inp_pic[l][m] == '*')
            {
             F_1[n].feature_value = 1;
             X = X + 1;
            }
         else
            F_1[n].feature_value = 0;
         n = n + 1;
        }
    }
if (X >= 0)
   {
    compute_weighted_sums();
    largest_node = maxnet();
    T_X = 0;
    for (l=0; l<FEATURES; l++)
       T_X = T_X + F_1[l].feature_value * F_1[l].top_down[largest_node];
    /*if (X < 5)
       X = 5;*/

    if (F_2[largest_node].active == 1)
       {
        H = 0;
        for (l=0; l<FEATURES; l++)
            if(F_1[l].top_down[largest_node] == 1)  H++;}

    criteria = (float)T_X/(float) H;
    if (criteria < F_2[largest_node].vigilance)
       return(0);
    else
       return(1);
   }
else
   return(0);
}


void initialize_weighted_links()
/*************************************************************************/
This function will initialize all top down links to 1 and all bottom up links to 1/(25+1).
/*************************************************************************/
{
 int i, j;
 float temp_features;

 temp_features = FEATURES;
 for (j=0; j<CATEGORIES; j++)
    {
     F_2[j].active = 0;
     for (i=0; i<FEATURES; i++)
        {
         F_2[j].bottom_up[i] = 1/(1+temp_features);
         F_1[i].top_down[j] = 1;
        }
    }
}


void print_output()
/*************************************************************************/
```

```
This function will type the current contents of oup_pic to the screen.
/*****************************************************************************/
{
 int i, j;
 char char_pic[33];

 clrscr();
 for (i=0; i<32; i++)
    {
     for (j=0; j<32; j++)
        {
         char_pic[j] = oup_pic[i][j];
        }
     char_pic[j] = oup_pic[i][j];
     char_pic[32] = '\0';
     printf("%s\n", char_pic);
    }
}


void matching()
{
     int i,j, t1, t2;

     match1=0; t1=0;
     match2=0; t2=0;
     for (i=0; i<32; i++)
       {
        for (j=0; j<32; j++)
        {    if(org_pic[i][j]=='*')
                {
                     t1++;
                     if(oup_pic[i][j]==org_pic[i][j]) match1++;
                }

             if(oup_pic[i][j]=='*')
                {
                     t2++;
                     if(oup_pic[i][j]==org_pic[i][j]) match2++;
                }
        }
       }
     per1 = (float)match1 / t1;
     per2 = (float)match2 / t2;
     printf("   P(O'|O*) = %5.3f\n",per1);
     printf("   P(O*|O') = %5.3f\n",per2);
     getch();
}


void see_input()
/******************************************************************************/
This function will scan the entire input image located in inp_pic, applying the network
architecture to each pixel and its 5 x 5 neighborhood,to determine if the pixel should be
s          e          t o          r n          o          t          .
/******************************************************************************/
{
 int i, j, q;

 gotoxy(1, 25);
 printf("\n");
 for (i=2; i<34; i++)
   {
    for (j=2; j<34; j++)
      {
       q = view_pixel(i, j);
       if (q == 1)
          oup_pic[i-2][j-2] = '*';
       else
          oup_pic[i-2][j-2] = '.';
      }
    gotoxy(1, 25);
    printf("Input picture lines scanned >> %d", i-1);
   }
}


void write_to_file()
/******************************************************************************/
This function will simply write the current contents of oup_pic to a file.
*******************************************************************************/
{
 char buffer[14], ch, char_pic[33];
 int i, j;
 FILE *oup;
 char *p;
```

```
buffer[0] = 13;
gotoxy(1, 25);
clreol();
printf("Enter the output picture filename >> ");
p = cgets(buffer);
oup = fopen(p, "r");
if (oup != NULL)
   {
    gotoxy(1, 25);
    clreol();
    printf("File \"%s\" exists, overwrite it <y or n>? >> ", p);
    if ((ch=getch()) == 'n' || ch == 'N')
       {
        fclose(oup);
        return;
       }
   }
 fclose(oup);
 oup = fopen(p, "w");
 for (i=0; i<32; i++)
   {
    for (j=0; j<32; j++)
       char_pic[j] = oup_pic[i][j];
    char_pic[32] = '\0';
    fprintf(oup, "%s\n", char_pic);
   }
 fprintf(oup, "   P(O'|O*) = %5.3f\n",per1);
 fprintf(oup, "   P(O*|O') = %5.3f\n",per2);
 fclose(oup);
}


void filter()
/****************************************************************************/
This function will apply the edge detetion operation mentioned in this paper to a pixel and
set or reset that pixel accordingly.
/****************************************************************************/
{
 char buffer[34][34], char_pic[33];
 int i, j, k, l, count;

 for (i=0; i<34; i++)
```

```
   {
    buffer[0][i] = ' ';
    buffer[33][i] = ' ';
    buffer[i][0] = ' ';
    buffer[i][33] = ' ';
   }
 for (i=1; i<33; i++)
   {
    for (j=1; j<33; j++)
      {
       buffer[i][j] = oup_pic[i-1][j-1];
      }
   }
 for (i=1; i<33; i++)
   {
    for (j=1; j<33; j++)
      {
       if (buffer[i][j] == '*')
         {
          count = 0;
          for (k=i-1; k<i+2; k++)
            {
             if (buffer[k][j] == '*')
               count = count + 1;
            }
          for (l=j-1; l<j+2; l++)
            {
             if (buffer[i][l] == '*')
               count = count + 1;
            }
          if (count == 2 || count == 6)
            oup_pic[i-1][j-1] = '.';
          else
            oup_pic[i-1][j-1] = '*';
         }
       else
         oup_pic[i-1][j-1] = '.';
      }
   }
}
```