ACTIVE VISION THROUGH INVARIANT REPRESENTATIONS AND SACCADE MOVEMENTS

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Yue Li

June 2006

This thesis entitled

ACTIVE VISION THROUGH INVARIANT REPRESENTATIONS AND SACCADE MOVEMENTS

by

YUE LI

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Janusz A. Starzyk

Professor of Electrical Engineering and Computer Science

Dennis Irwin

Dean, Russ College of Engineering and Technology

Abstract

YUE LI, M.S., June 2006, Electrical Engineering and Computer Science

ACTIVE VISION THROUGH INVARIANT REPRESENTATIONS AND SACCADE MOVEMENT (143 pp.)

Director of Thesis: Janusz A. Starzyk

This thesis presents an innovative approach to pattern recognition, by using self-organized, invariant representations integrating continuous observation and saccade movements. This biologically motivated approach can achieve visual perception through a retina like sampling of high resolution images with lower resolution artificial retina.

The neural network uses hierarchical feedback structures to build object representations, self-organizes invariant transformations, while iterates on the images received from the retina model. The network identifies the whole image by using winner-take-all scheme through temporal association of sufficiently accurate saccades. By using our invariance building scheme, the network can identify different views of the same object.

Approved:

Janusz A. Starzyk

Professor of Electrical Engineering and Computer Science

Acknowledgements

My thesis could not be accomplished without so many people's help and support. I would like to take this moment to express my gratitude to all those who inspired me to complete this thesis.

First of all, I would like to express my deepest gratitude to my advisor, Professor Janusz A. Starzyk for his extraordinary help and guidance during my master studies. It was a great pleasure for me to conduct this thesis under his supervision. His wide knowledge and integral view on research and working attitude have been of great value for me. I am deeply grateful for all his help in doing research and valuable advice, which have made a remarkable influence on my life. I could not achieve so much accomplishment without him.

I would like to thank my committee members: Professor Jeffery Dill, Professor Robert Curtis and Professor Sergiu Aizicovici for reviewing my thesis and providing helpful comments.

During many discussions and computer simulations of my thesis, I got a lot of valuable comments from my labmates: Yinyin, Haibo, Mingwei, Zhen Zhu, James and Yiming. More important, I am thankful for their great friendship and the encouragement that all of them gave me during hard times.

I owe my loving thanks to my beloved parents. They have always been there supporting me, and they have made this great achievement in my life possible. I am very grateful for my boyfriend Hui, for all his love and support.

Finally, my deepest thanks goes to my friend, Sterling, whose consistent support, encouragement and help during these years has made my dream come true.

Thank you all.

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Background and Motivation

Human vision is a complicated process which has not been completely understood yet, despite hundreds of years of study and research. [Tri00] has showed that human recognition will be critical in future applications that will allow computers to interact with human and environment more efficiently. This raises interests in building computer vision, which is a branch of artificial intelligence that deals with image recognition and understanding. It focuses on providing computers with the functions typical to human vision [Pic03], thus providing the communication interface between human and computers. At present, state of technology, computer vision has provided important applications in commercial, security, and military fields, such as industrial automation, quality inspection, robotics, target recognition and surveillance systems, etc.

As one of the most remarkable and successful applications of computer vision, face recognition, has recently received significant attention due to variety of potential applications. These potential applications include identification, verification, anticipation, etc. For the identification problem, the face to be recognized will be compared to a group of faces of known individuals. For the verification problem, the face to be verified will be confirmed or rejected by the system [Tor04]. For the anticipation problem, the system will anticipate a visual input (of the observed person) based on its accumulated past learning experience. A face recognition system generally includes steps of face detection, feature extraction, and face recognition.

It has been widely recognized that feature extraction is important in face recognition. Features can be either local features such as lines or points, or basic facial features. For instance, OpenCV [Bra05] detects and recognizes objects by extracting local features such as edges and textures, and uses some heuristics to find configurations of those features specific to the object of interest. Basic facial features, such as eyes, nose and mouth, and their spatial arrangement, are important for discrimination among face images that are all quantitatively similar to each other [Lia00]. [Kan73] developed one of the earliest face recognition algorithms based on automatic facial feature detection. In [Iva04], authors use component facial feature based face recognition that trains recognition classifiers on each of the components separately and then combines their outputs to recognize faces.

[Ait03] developed a neural network based facial recognition program. While the system can only handle the image with both mouth and both eyes visible, it cannot deal with every facial image in all possible poses. However, a successful and efficient facial recognition system should also be able to obtain “invariant” recognition characteristics, including facial recognition under different viewing perspectives [Lee02]. This results in another challenging situation when certain facial features are invisible due to large variation in head pose. The best solution might be to visualize the missing features by using the symmetry of the face or using past learned information [Zha03].

Other problem in the existing face recognition systems is the resolution reduction. A memory-based face recognition algorithm (ARENA) [Sim00] is implemented to identify building visitors. However, it reduces the resolution of the visitor’s image to get best match against the images stored in the database, which may decrease the recognition accuracy.

To summarize, there is still no technology that provides a robust solution to all situations, conditions and different applications that face recognition algorithms may encounter [Tor04]. In other words, current systems are still far away from the capability of the human perception system [Zha03], and it is still a problem to robustly extract facial regions and features out of complex scenes [Kar96].

1.2 Research Goals

Based on the recent results in computer vision, there is a need to develop invariant image interpretation and recognition in a natural and dynamic way. This natural and dynamic approach to image recognition is biologically motivated and has a penitential to accomplish higher levels of recognition and image understanding than the existing systems. It is aimed at building active vision in a learning machine that actively analyzes the perceived image by directing its sensors (retina sampling) to most interesting and informative parts of the input image. This thesis aims at building an artificial vision model, which can achieve high-resolution image recognition, can support continuous observation, can perform feature extraction, and can build internal image representations for machine learning. The task is to maintain observation and correct identification of each person, even when a current view of the person cannot be uniquely recognized.

The feature extraction in this thesis is based on an artificial retina sampling model. The perception of an object in its entirety is built up from active scanning and feature extraction in various areas of the object image, known as saccades. Human constantly scan the visual world using saccade movements [Dor04]. Saccades are rapid, ballistic eye movements trigged by different cues, such as visual, auditory, and planned cues [Gre99]. During these saccades, the image moves across the human retina. By using the artificial retina sampling, our vision model is capable of extracting features by sampling any interesting area of the observed image. Our artificial retina model is built up by following the model of the rods and cones distributions in human retina. By repeating saccade movements and building invariant temporal correlations of sampled image features at sufficient detail level, an object can be successfully perceived.

We use a neural network with hierarchical feedback structures to build object representations, self-organize network topology to build invariant transformations, iterate on the images received from the artificial retina model, and control the retina model to sample details of the observed objects. The network identifies the whole image by using winner-take-all scheme after sufficiently accurate saccades. By using our invariance building scheme, the network can identify different views of the same object. When certain facial features are invisible due to large variation in the head pose, the network uses its previously learned information, and when new facial features appear, the network learns the new features, expanding its topological structure.

A reference for our active vision model will be the learning-based computer vision-Open Source Computer Vision Library (OpenCV) from Intel Corporation. OpenCV is an open source collection of computer vision routines geared mainly towards human-computer interaction, robotics, security, and other vision applications [Bra05]. OpenCV detects and recognizes objects by extracting local features such as edges and textures, and using some heuristics to find configurations of those features specific to the object of interest. Our active vision model not only extracts local features, but also extracts facial features such as eyes, mouth and nose which are important for discrimination among face images that are all quantitatively similar to each other [Lia00]. We will compare the learning quality, learning efficiency and learning capacity between our active vision and OpenCV.

1.3 Thesis Organization

This thesis is organized into six chapters. Chapter 2 presents the network structure of the bidirectional feedback self-organization winner-take-all classifier. In this chapter, the network operation is detailed described, which includes the supervised learning and unsupervised learning schemes. This chapter also covers the results analysis and learning performance of the self-organization winner-take-all classifier.

Chapter 3 discusses the continuous observation network that can maintain observation and correct identification of each person, even when a current view of the person cannot be uniquely recognized. This continuous observation network is supported by the dual neurons that generate feedback prediction signals.

Chapter 4 introduces the four different retina sampling models based on the theoretical model and experimental data. In addition, it combines the retina sampling with ganglion cells implementations. Motion detection, micro-saccades and substantial simulation results are discussed in this chapter. This chapter also presents the saccade movement network that applies the retina sampling model discussed in Chapter 4 to achieve accurate recognition for higher resolution images. This network can identify the whole high resolution image using the low resolution retina after several saccades. Architecture and algorithm for this saccade movement network is detailed described.

Chapter 5 gives a conclusion of the work done, summarizes major results obtained from the thesis and discusses the future work.

2. Self-Organizing Winner-Take-All Classifier

2.1 Bidirectional Feedback Network Structure

In this section, we propose a novel bidirectional neural network structure for self-organizing classifier design. This neural network uses hierarchical feedback WTA structures with bidirectional connections between hierarchical levels to build object representations and uses self-organization to obtain invariant transformations. It is assumed that the network receives continuous input representing the same object over a period of time. The network anticipates that the input represents the same object by using feedback prediction signals. The network maintains the same internal representation for the received input, unless there is a strong disagreement between the feedback predictions and received input signals. In such case, the network switches to the internal representation of different object and maintains this representation until the next strong disagreement. The mechanism of building these predictions and checking the level of the disagreement is described in the following sections.

Suppose that a hierarchical level h receives its inputs from WTA blocks on the previous level $h-1$. For 2D input sampling, each block on level h contains inputs a_{pq}^{h} from a 2D subarray of output signals from the lower level blocks.

$$a_{pq}^{h} = \begin{bmatrix} \hat{b}_{i,j}^{h-1} & \hat{b}_{i,j+1}^{h-1} & \cdots & \hat{b}_{i,j+k}^{h-1} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{b}_{i+m,j}^{h-1} & \hat{b}_{i+m,j+1}^{h-1} & \cdots & \hat{b}_{i+m,j+k}^{h-1} \end{bmatrix}_{m\times(k\cdot n_{h-1})} \tag{2.1}$$

where $\hat{b}_{pq}^{h-1}$ are $1\times n_{h-1}$ vectors of outputs from $h-1$ level WTA blocks. Thus a_{pq}^{h} block receives total $m\times(k\cdot n_{h-1})$ scalar inputs from the lower level blocks. Fig. 2.1 shows three levels of hierarchical structure illustrating this 2D sampling scheme. In this figure, a single element $\hat{b}_{pq}^{h}$ on level h is connected to a rectangular array of elements $\hat{b}_{pq}^{h-1}$ on the lower level. Different elements $\hat{b}_{pq}^{h}$ may receive inputs from different overlapping rectangular arrays of elements $\hat{b}_{pq}^{h-1}$ on the lower level.

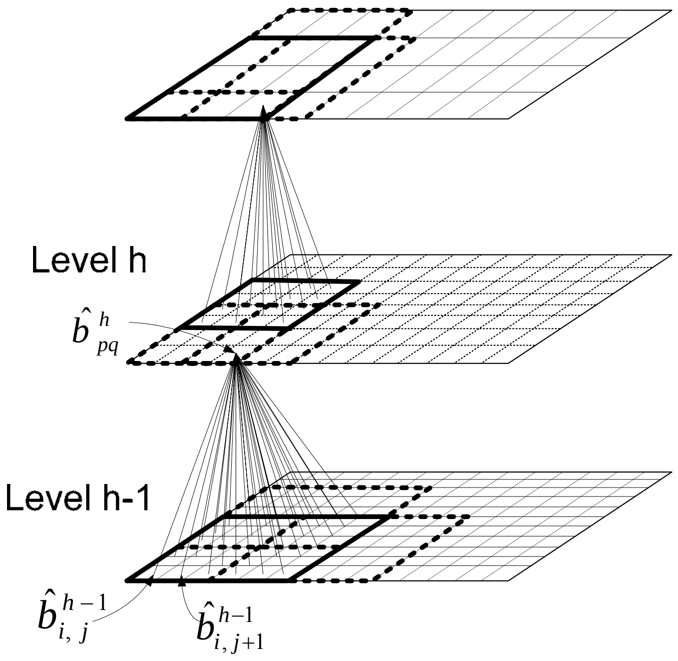


Fig. 2.1 Hierarchical structure of 2D sampling

In detail, the input signals to hierarchical blocks a_{pq}^{h} on level h are obtained from blocks of lower level signals $\hat{b}_{pq}^{h-1}$ by specifying four parameters p,q,m,k, where p,q are the coordinates of the origin (e.g, $p = i\,, q = j$) and m, k are respectively the number of rows and columns in each block of signals $\hat{b}^{h-1}_{pq}$. For the uniform sampling (with overlap) of blocks of $\hat{b}^{h-1}_{pq}$, p and q may be computed. For instance, if we specify a 2D input signal sampling with steps Δx and Δy in horizontal and vertical direction respectively, then

$$a^{h}_{pq} = \begin{bmatrix} \hat{b}^{h-1}_{(p-1)\Delta x+1,(q-1)\Delta y+1} & \hat{b}^{h-1}_{(p-1)\Delta x+1,(q-1)\Delta y+2} & \cdots & \hat{b}^{h-1}_{(p-1)\Delta x+1,(q-1)\Delta y+k} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{b}^{h-1}_{(p-1)\Delta x+m,(q-1)\Delta y+1} & \hat{b}^{h-1}_{(p-1)\Delta x+m+1,(q-1)\Delta y+1} & \cdots & \hat{b}^{h-1}_{(p-1)\Delta x+m,(q-1)\Delta y+k} \end{bmatrix}_{m\times(k\cdot n_{h-1})} . \quad (2.2)$$

As seen in Fig. 2.2, the inputs of h level block are obtained from first taking the product of $\tilde{a}^{h}_{pq}$ and the interconnect weight matrix W^{h}.

$$\tilde{b}^{h}_{pq} = \tilde{a}^{h}_{pq} W^{h}, \quad (2.3)$$

where $\tilde{a}^{h}_{pq}$ is a $1\times(n_{h-1}\cdot m\cdot k)$ vector which is reshaped from a^{h}_{pq} , and W^{h} is $(n_{h-1}\cdot m\cdot k)\times n_{h}$ weight matrix. Thus, as shown in Fig. 2.2, outputs $\hat{b}^{h-1}_{pq}$ are first sampled and formulate a vector $\tilde{a}^{h}_{pq}$ that is multiplied by the weight matrix W^{h} to obtain inputs to level h $\tilde{b}^{h}_{pq}$. In a continuous observation of an object, each scalar signal in hierarchical structure shown in Fig. 2.1 is vectorized to represent a sequence of input images or other time domain input signals. Thus vectors $\tilde{b}^{h}_{pq}$ and $\tilde{a}^{h}_{pq}$ will be replaced by matrices with number of rows corresponding to the number of samples. After $\tilde{b}^{h}_{pq}$ are obtained using initial weight matrix W^{h}, a new W^{h} can be computed from equation (2.3) by least square method:

$$W^{h} = pinv(\tilde{a}^{h}_{pq})\times\tilde{b}^{h}_{pq}, \quad (2.4)$$

where $pinv(\tilde{a}^{h}_{pq})$ is a pseudo inverse of the input signal matrix $\tilde{a}^{h}_{pq}$.

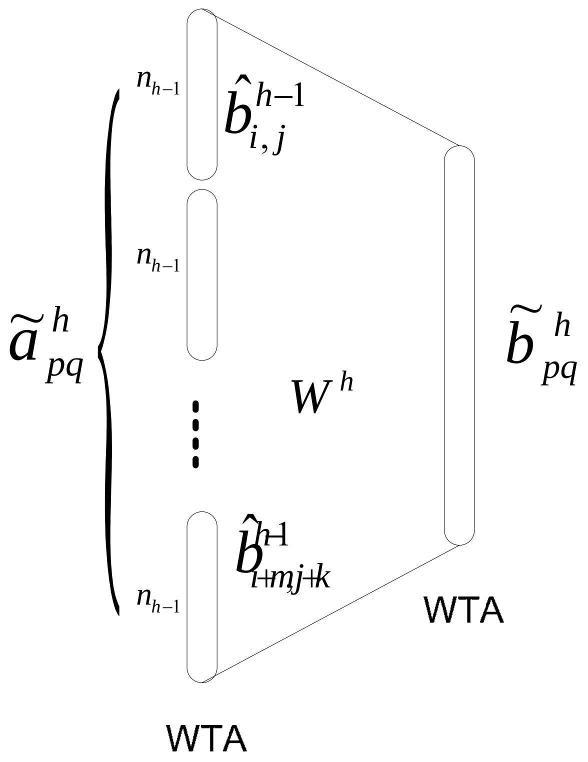


Fig. 2.2 Outputs of level (h-1) formated as inputs to level h

Finally, by taking a nonlinear transformation (e.g., sigmoidal transformation followed by WTA operation) of $\tilde{b}^h_{pq}$, we will obtain the output of level h block $\hat{b}^h_{pq}$ as

$$\hat{b}^h_{pq} = f(\tilde{b}^h_{pq}), \tag{2.5}$$

where for each input signal, $\hat{b}^h_{pq}$ is a $1 \times n_h$ vector that is an input to the next layer. Subsequently, we will adjust the weight matrix $\hat{W}^h$ to satisfy

$$\hat{b}^h_{pq}(\hat{W}^h)^T = a^h_{pq}. \tag{2.6}$$

Equation (2.6) implements a feedback signal idea where the output $\hat{b}^h_{pq}$ of the level h is fed back to the input of this level through the same interconnection weights. Thus $\hat{W}^h$ represents weights of the bidirectional links between two levels. $\hat{W}^h$ can be updated iteratively after each new input signal is received by this network, by using the recursive least square method which is described in the next section.

2.2 Recursive Least Square (RLS) Weights Adjustment

Suppose that we have a linear system of equations

$$\Phi_{n-1}W_{n-1} = z_{n-1}, \tag{2.7}$$

where W_{n-1} is the $k \times m$ weight matrix, Φ_{n-1} is $(n-1)\times k$ matrix of received signal values, and z_{n-1} is $(n-1)\times m$ matrix of desired signal values. W_{n-1} can be obtained by using the least square method as follows.

$$W_{n-1} = (\Phi_{n-1}^T \Phi_{n-1})^{-1} \Phi_{n-1}{}^T z_{n-1}. \tag{2.8}$$

When a new input signal x^T is received, we need to update the weight matrix to a new value W_n to satisfy

$$\Phi_n W_n = z_n, \tag{2.9}$$

where $\Phi_n = \begin{vmatrix} \Phi_{n-1} \\ x^T \end{vmatrix}$, (2.10)

and $z_n = \begin{bmatrix} z_{n-1} \\ z \end{bmatrix}$. (2.11)

Our intension is to do it iteratively without inverting the new coefficient matrix Φ_n. Finding the inverse of $\Phi_n^T \Phi_n$ is a key to recursive least square (RLS) weight update. From the matrix inverse lemma, we know that for a positive definite matrix A,

$$(A + xx^T)^{-1} = A^{-1} - \frac{A^{-1}xx^T A^{-1}}{1 + x^T A^{-1} x}. \tag{2.12}$$

Thus using $(\Phi_n^T \Phi_n)^{-1} = P_n$ and the matrix inverse lemma, we have

$$P_n = P_{n-1} - \frac{P_{n-1} x x^T P_{n-1}}{1 + x^T P_{n-1} x}. \tag{2.13}$$

Let us define vector $v = P_{n-1} x$, so we can simplify notation for P_n

$$P_n = P_{n-1} - \frac{v v^T}{1 + x^T v}, \tag{2.14}$$

and the updated weight matrix W_n is obtained from

$$W_n = P_n \Phi_n^T z_n = \left(P_{n-1} - \frac{v v^T}{1 + x^T v} \right) \left(\Phi_{n-1}^T z_{n-1} + x z \right). \tag{2.15}$$

Let $\Delta P = \frac{v v^T}{1 + x^T v}$, (2.16)

Then the updated weight matrix is

$$W_n = (P_{n-1} - \Delta P)(\Phi_{n-1}^T z_{n-1} + x z). \tag{2.17}$$

In this way, we can update the weights by using the recursive least square method each time a new input comes in. The signal flow diagram of the feed forward network that implements this iterative scheme is shown in Fig. 2.3.

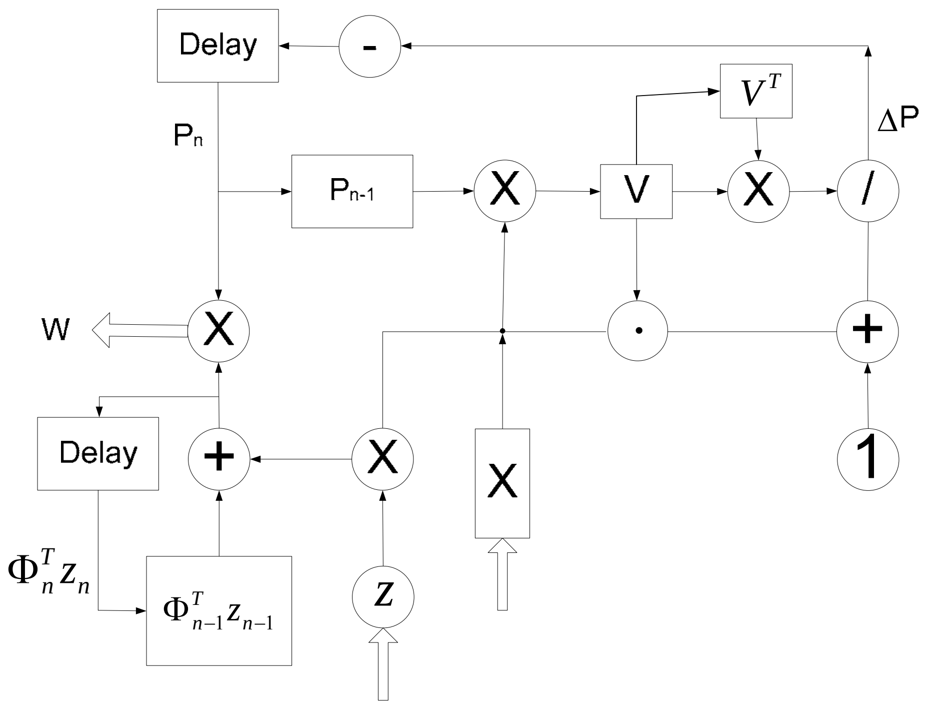


Fig. 2.3 Signal flow diagram for RLS

This network requires storage of a matrix P_{n-1} and vector $\Phi_{n-1}^T z_{n-1}$. It requires two matrix-vector, one vector-vector multiplication, and one scalar-vector, one matrix scalar division and add/subtract operations as shown in Fig. 2.3.

2.3 2-layer Self-organizing WTA Classifier

Let's first consider a 2-layer network without sampling of the input image. This network has a fully connected input and output layers and the interconnection weights are described by a single interconnection weight matrix W_0. As illustrated in Fig. 2.4, the 2-layer output is computed by first adding the weighted input signals at each output node b_k

$$b_k = \sum_{i=1}^{n1} W_{ik} a_i \qquad (2.18)$$

and then performing the WTA operation to obtain $\hat{b}_k = f(b_k)$.

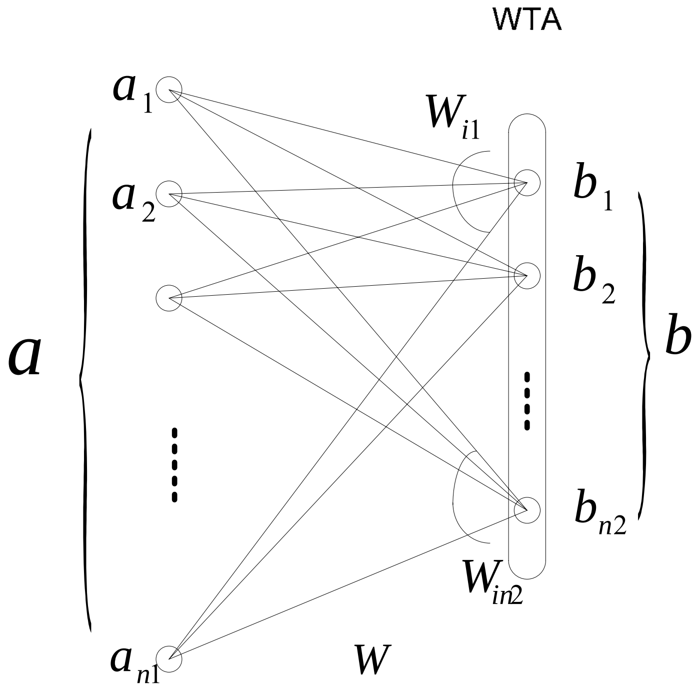


Fig. 2.4 A 2-layer classification network

We developed both supervised learning and unsupervised learning schemes to illustrate how the RLS method can be used to classify different categories.

There are two ways to achieve this classification. First, unsupervised learning uses the initial weight setting for W_0, and another one uses supervised leaning with initial output setting as discussed next.

2.3.1 Initial Weight Setting

Let us assume that we have a database, which needs to be classified in n categories. We will first choose n samples from the database to represent these n categories. The classification will be performed by the network with n_1 inputs (input signal size) and n_2 outputs ($n_2 = n$). The interconnection weights between $h-1$ and h levels will be initially set based on these n samples to W_0 as follows. Each of the n_2 output nodes in WTA will be connected to all input nodes through weights equal to the input signal values. All the input signal values were initially normalized. This results in such weights that the product of each sample signal with the weight matrix is equal to one for one of the output nodes and is less than one for the other nodes. Thus, this particular node is selected to represent one category. This implements Hebbian learning idea, where weights of the winning output are adjusted towards the input signal values.

Then we will send the entire database to the network for least square weights adjustment and classification. The network will adjust its weights and classify the database by its self-organization learning scheme in an unsupervised way using

$$B = AW_0, \tag{2.19}$$

where each row of the matrix A contains a single input signal to the learning network, W_0 is the initial interconnection weight matrix (set by chosen n samples from the database) and B is the resulting output signals matrix. The interconnection weights are set to their final values by solving

$$AW = B, \tag{2.20}$$

using the RLS method.

A small database with 19 images was used to test this learning scheme with initial weight setting. Each image is an array of 14×18 8-bit pixels. A 2-layer network with 252 input nodes and 4 output nodes was used for classification with RLS method.

Fig. 2.5 shows 4 images for 4 different categories used for initial weight setting.

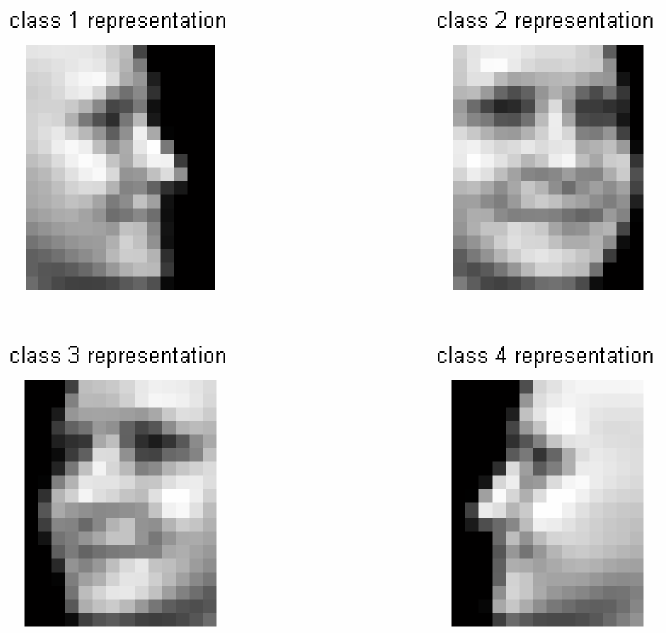


Fig. 2.5 Samples representing 4 categories

Fig. 2.6 shows the surface plot of the output layer after the unsupervised learning of the entire database. In Fig. 2.6, the x-axis represents the output node activity, y-axis represents a consecutive image from the database, and z-axis represents the output neuron value in response to each input image. Strength of a given output value x corresponds to the strength of our belief that sample y belongs to class x.

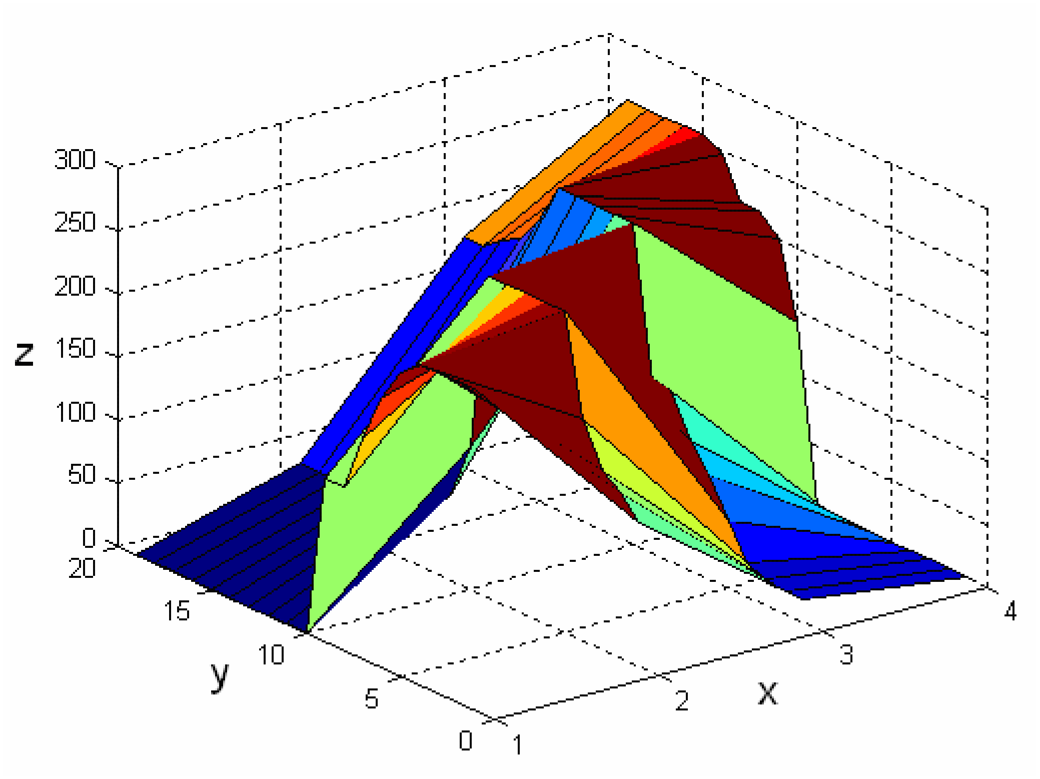
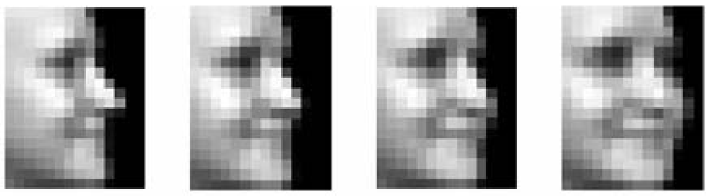
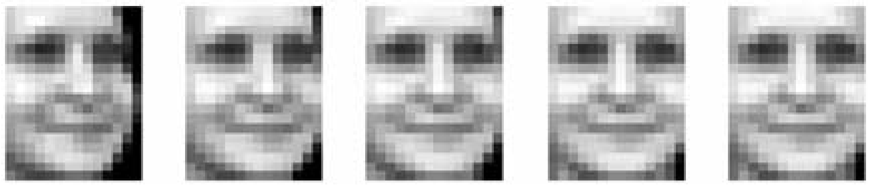


Fig. 2.6 Surface plot of the output layer

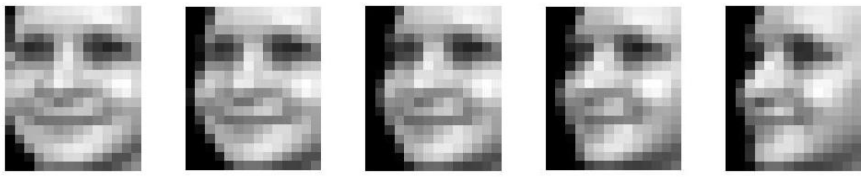
Based on the output neuron values, different images correspond to different categories. In this unsupervising stage, only 4 categories were obtained. Fig. 2.7(a)-(d) show the resulting image categories after self-organized learning.



(a) Images classified as class 1



(b) Images classified as class 2



(c) Images classified as class 3

(d) Images classified as class 4

Fig. 2.7 Images classified as different classes

As we can see, the self-organized learning using RLS method resulted in a reasonable grouping of the input images into similar profile categories. In this initial weight setting, main_LSQ_weight_setting.m file is used under RLS_WTA_Classification (see Appendix A).

2.3.2 Initial Output Setting

A more natural approach is the supervised output setting method. We will first present the network with n chosen samples to the inputs of the level h, and set the output of the h level to a $n \times n$ unit matrix. By solving equation $\tilde{a}_{pq}^{h} W_0^h = I$, we can obtain the initial set of weights W_o^h for the unsupervised learning. Next, we send the entire database to the input of the network and use equation (2.4) for the least square weight adjustment. The self-organizing network will adjust its weights and classify the database in an unsupervised way. If we choose the same 4 samples as in Fig. 2.5, the classification result is the same as shown in Fig. 2.7(a)-(d). In this initial output setting, main_LSQ_output_setting.m is used in folder RLS_WTA_Classification (see Appendix A).

2.4 3-layer Self-organizing WTA Classifier with Bidirectional Connections

We already demonstrated that by using a combination of the supervised and unsupervised learning, the 2-layer network can self-organize and learn to classify. Now we introduce the 3-layer network which uses hierarchical feedback WTA structures with bidirectional connections between hierarchical levels. Fig. 2.8 shows the signal flow for the 3-layer network.

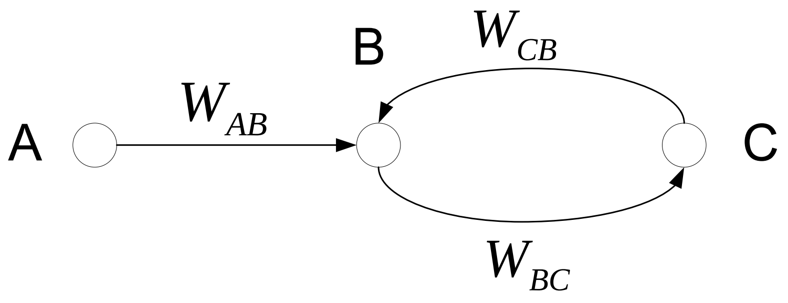


Fig. 2.8 Signal flow for the 3-layer network

Suppose that we have three layers A, B and C, where A is the input layer, C is the output layer which represents the number of categories, and B is the layer which has bidirectional connections to and from the layer C. The weights from layer A to layer B are W_{AB}, the weights from layer B to C are W_{BC} and the weights from layer C to B are W_{CB}.

Based on this 3-layer network with hierarchical feedback, we develop self-organizing winner take all classifier (SOWTAC). Architecture and basic operation of SOWTAC are presented in this section.

As discussed before, the input layer A receives inputs from the input images of the database by 2D sampling. Each block a_{pq} in layer A contains the inputs from a 2D subarray of an input image Im.

$$a_{pq} = \begin{bmatrix} \mathrm{Im}_{i,j} & \mathrm{Im}_{i,j+1} & \cdots & \mathrm{Im}_{i,j+k} \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{Im}_{i+m,j} & \mathrm{Im}_{i+m,j+1} & \cdots & \mathrm{Im}_{i+m,j+k} \end{bmatrix}_{m\times k} \tag{2.21}$$

And $\tilde{a}_{pq}$ is a $1\times(m\cdot k)$ vector which is reshaped from a_{pq}. So initially, the signal values at layer B are obtained by taking the product of $\tilde{a}_{pq}$ and the interconnect weight matrix W_{ab} between $\tilde{a}_{pq}$ and $\tilde{b}_{pq}$,

$$\tilde{b}_{pq} = \tilde{a}_{pq} W_{ab} . \tag{2.22}$$

As shown in Fig. 2.9, there is N blocks of $\tilde{a}_{pq}$, N blocks of W_{ab}, and N blocks of $\tilde{b}_{pq}$. For simplicity, we use the same notation $\tilde{a}_{pq}$ for the sampled input at different overlapping locations of the input image, and the same notation W_{ab} for the weight matrix from $\tilde{a}_{pq}$ to $\tilde{b}_{pq}$ at different locations. In reality, equation (2.22) relates different signals $\tilde{a}^{h}_{pq}$ and $\tilde{b}_{pq}$ through different weight matrices W_{ab}.

In general, we represent the weight matrix from layer A to B by W_{AB}, where

$$A = \left\{ \tilde{a}_{pq} \right\}_{p=1,2,\ldots,n,q=1,2,\ldots,n} , \tag{2.23}$$

$$B = \left\{ \tilde{b}_{pq} \right\}_{p=1,2,\ldots,n,q=1,2,\ldots,n} . \tag{2.24}$$

If there are M samples sent to the network, the rows of A are set to M input samples.

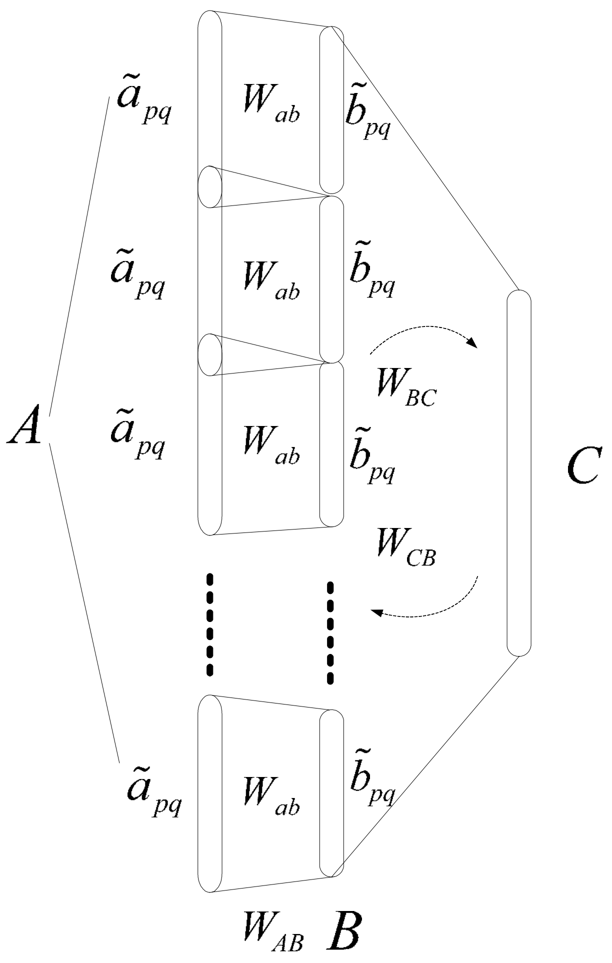


Fig. 2.9 Hierarchical structure of the 3-layer network

2.4.1 Supervised Learning and Handmade Features

Basically, as discussed above, we will first have supervised learning followed by the unsupervised learning. During supervised learning, we will send n samples from the database, each of which represents one category. The output layer C will be set to a unit matrix. In the following we will discuss how to set the initial value of W_{ab}.

Initial weight setting and weights adjustment will be explained on the same database used in section 2.3. Assume that $m = k = 4$ and $M = 20$.

First, we generate 20 different handmade features as shown in Fig. 2.10.

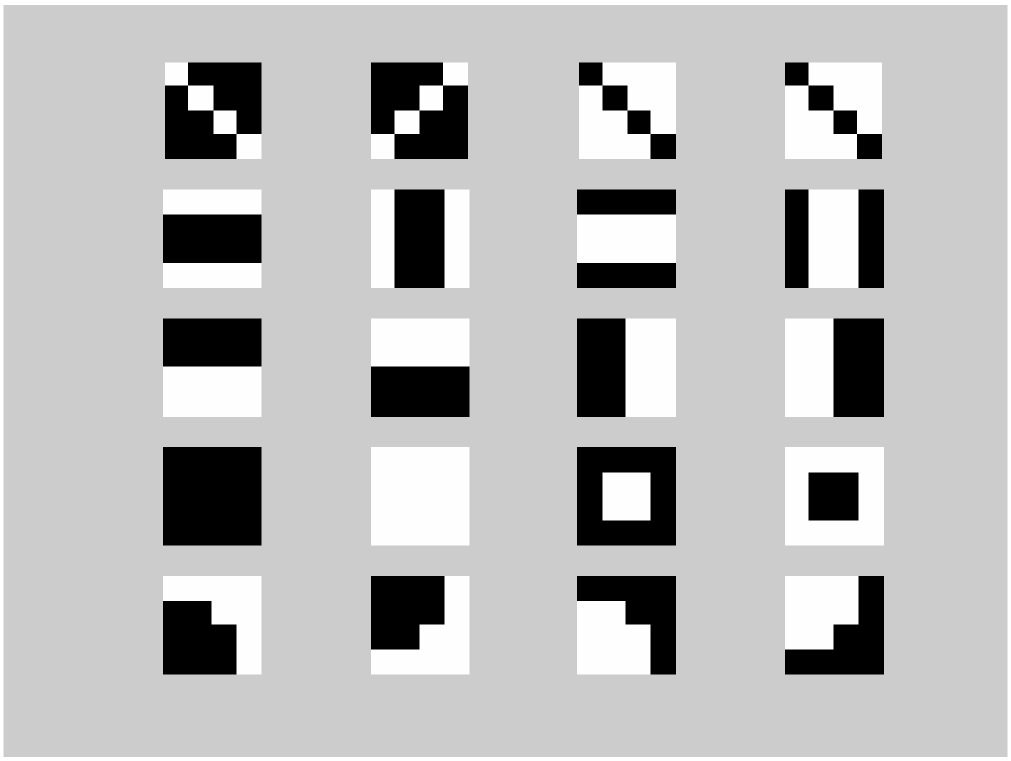
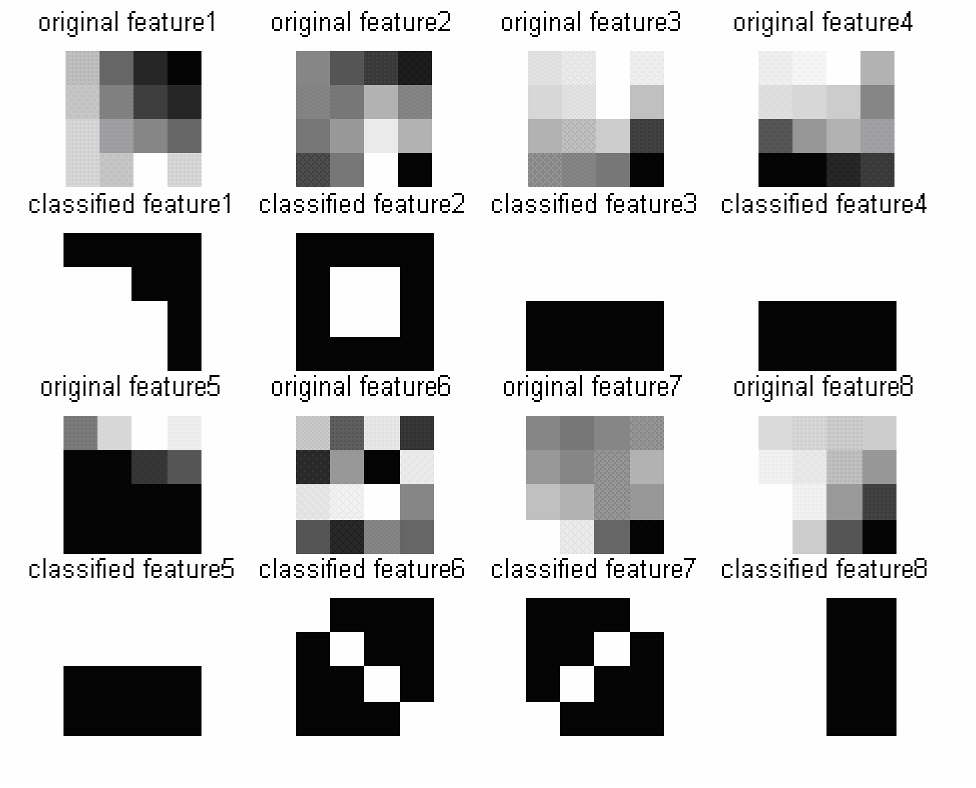
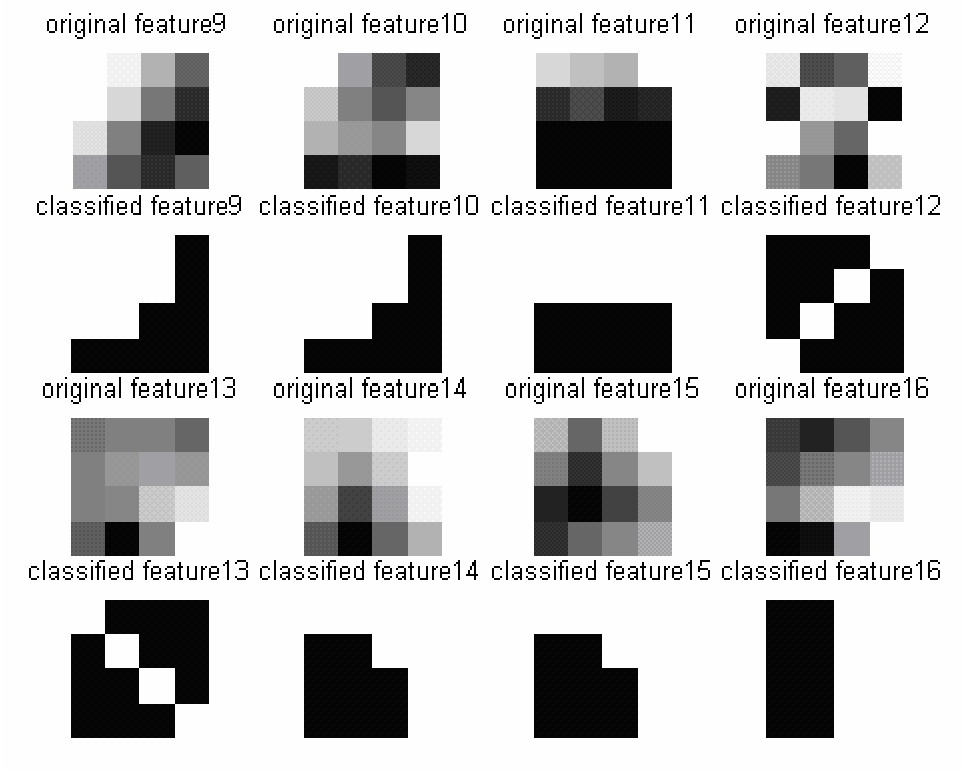


Fig. 2.10 20 handmade features

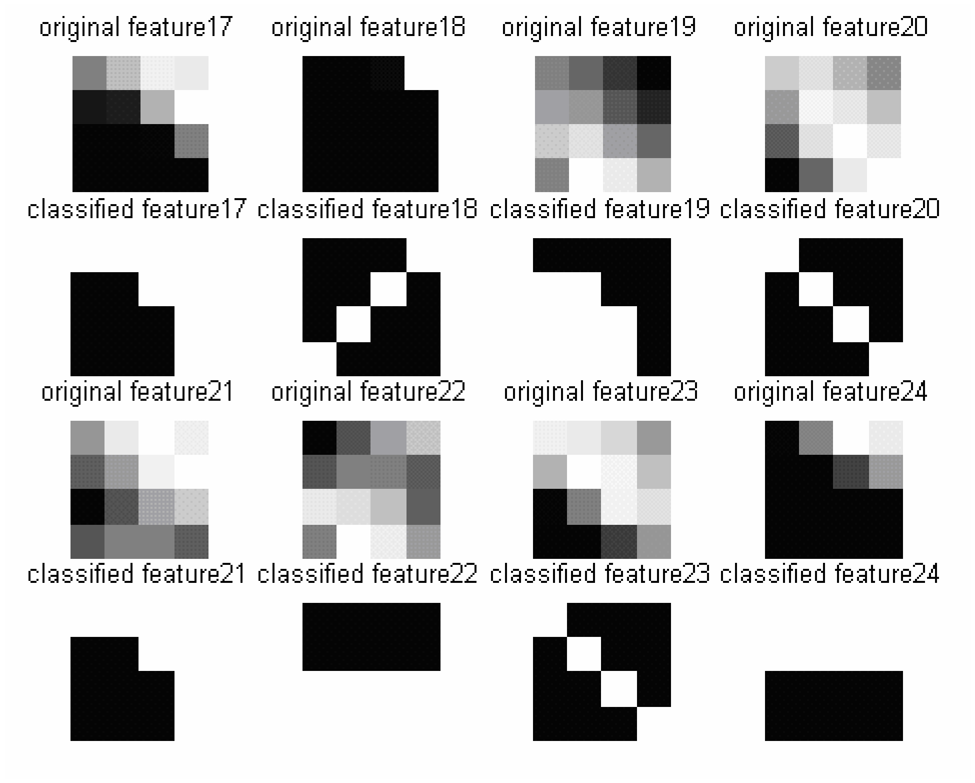
Consider, for instance, N blocks of a_{pq} obtained from one image in our database (N=48). Fig. 2.11(a)-(f) shows how these N original features from one image are classified into these 20 binary handmade features. We use the supervised output-setting method as described in section 2.3.1 to classify the 48 extracted features into 20 different classes.


original feature1
original feature2
original feature3
original feature4
classified feature1
classified feature2
classified feature3
classified feature4
original feature5
original feature6
original feature7
original feature8
classified feature5
classified feature6
classified feature7
classified feature8

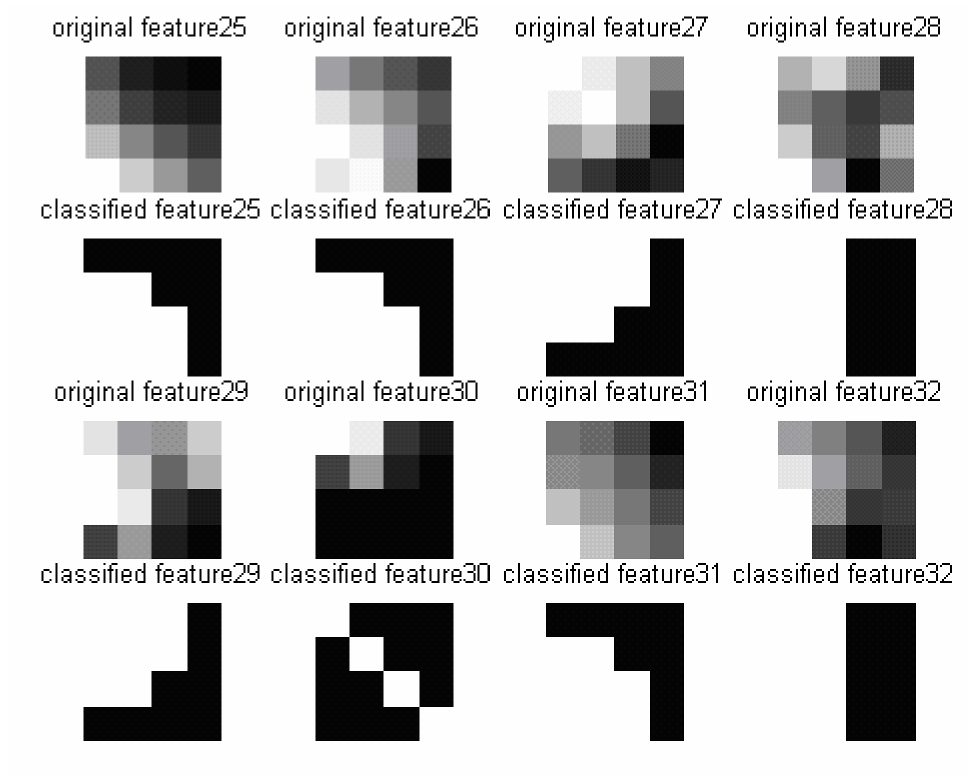
(a)


original feature9
original feature10
original feature11
original feature12
classified feature9
classified feature10
classified feature11
classified feature12
original feature13
original feature14
original feature15
original feature16
classified feature13
classified feature14
classified feature15
classified feature16

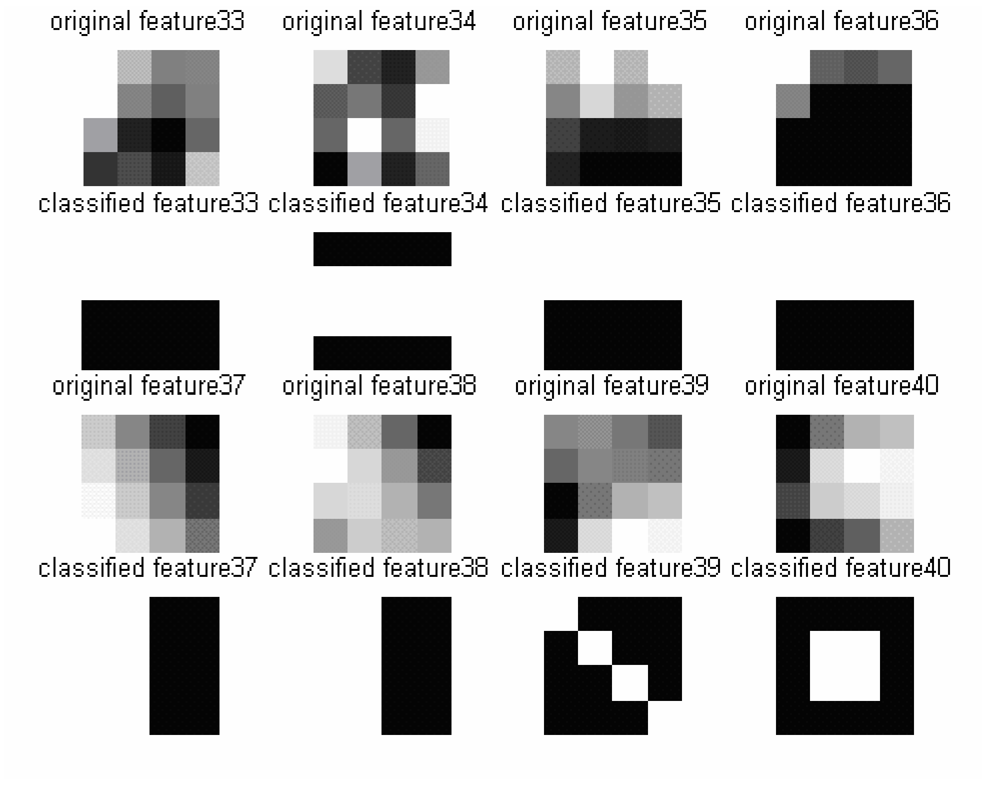
(b)


original feature17
original feature18
original feature19
original feature20
classified feature17
classified feature18
classified feature19
classified feature20
original feature21
original feature22
original feature23
original feature24
classified feature21
classified feature22
classified feature23
classified feature24

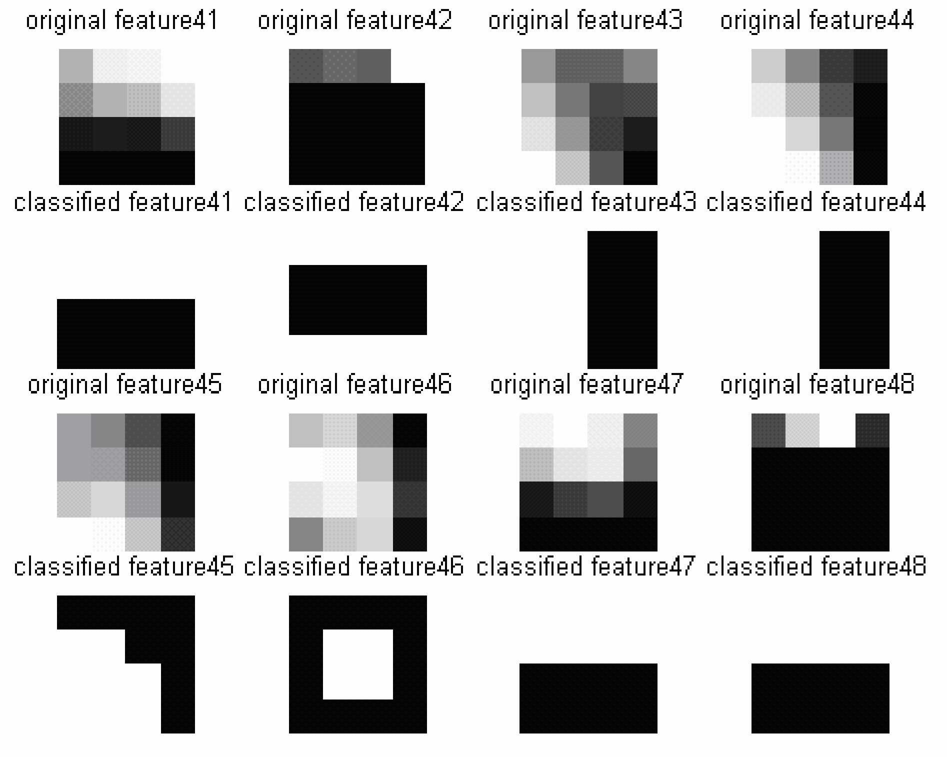
(c)


original feature25
original feature26
original feature27
original feature28
classified feature25
classified feature26
classified feature27
classified feature28
original feature29
original feature30
original feature31
original feature32
classified feature29
classified feature30
classified feature31
classified feature32

(d)



(e)



(f)

Fig. 2.11 Original features and their classified handmade features

All handmade features and can be put together in a matrix shown in Fig. 2.12(a), By using the initial output setting as described in section 2.3.1, we will send this matrix to the input layer A, and set the output of layer B to the unit matrix. By using the recursive least square weights adjustment, we can get the initial value of W_{ab0} as shown in Fig. 2.12(b). Then we will use this W_{ab0} in 3-layer supervised training as discussed below.

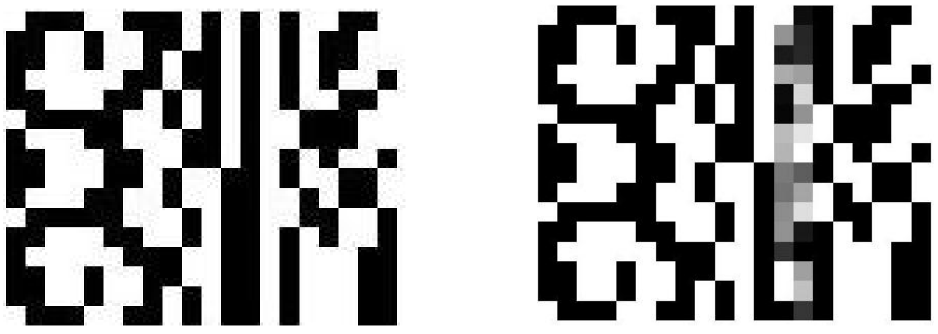

(a) (b)

Fig. 2.12 Handmade features and Initial weight matrix W_{ab0}

During the supervised learning, for each block $\tilde{a}_{pq}$, $\tilde{b}_{pq}$ we will have

$$\tilde{b}_{pq} = \tilde{a}_{pq} W_{ab0} . \tag{2.25}$$

Since $A = \left\{ \tilde{a}_{pq} \right\}_{p=1,2,\ldots,n,q=1,2,\ldots,n}$, $B = \left\{ \tilde{b}_{pq} \right\}_{p=1,2,\ldots,n,q=1,2,\ldots,n}$,

$$BW_{BC0} = C . \tag{2.26}$$

Because C is set to a unit matrix by using the initial output setting, so we will have

$$BW_{BC0} = C = I . \tag{2.27}$$

W_{BC0} can be solved by using the recursive least square method.

After the supervised learning, we get the initial value of W_{BC0}. In this example, we set the initial value of W_{ab0} to the handmade features. Later we proved that we could also set the initial value of W_{ab0} to random numbers.

2.4.2 Unsupervised Learning

After the supervised learning, we will send the whole database to the network inputs for iterative least square weights adjustment and self-organization. This unsupervised learning is accomplished in two stages: the linear iterations and the nonlinear iterations.

2.4.2.1 Linear Iterations

In the linear iterations, we simply iterate between layers B and C until convergence. Since during the unsupervised learning, we will send the whole database to the network, the initial values of signals B are computed as follows:

$$B_0 = AW_{AB0}, \tag{2.28}$$

where A is the 2D input matrix. Each row of A represents a single image from the database.

Layer C receives feed forward signals from layer B. Thus initial signals in layer C can be computed as:

$$C_0 = B_0 W_{BC0} = AW_{AB0} W_{BC0}. \tag{2.29}$$

In the next iteration, the layer B receives both feed forward signals from layer A and the feedback signals from layer C. So the signals in layer B can be updated as:

$$B_1 = \begin{bmatrix} A & C_0 \end{bmatrix} \begin{bmatrix} W_{AB0} \\ W_{CB0} \end{bmatrix}. \tag{2.30}$$

Since

$$B_1 = AW_{AB0} + C_0 W_{CB0} = AW_{AB0} + AW_{AB0} W_{BC0} W_{CB0}. \tag{2.31}$$

The successive iterations can be computed as follows:

$$\begin{aligned} B_2 &= AW_{AB0} + C_1 W_{CB0} \\ &= AW_{AB0} + (AW_{AB0} + AW_{AB0} W_{BC0} W_{CB0}) W_{BC0} W_{CB0} \\ &= AW_{AB0} + AW_{AB0} W_{BC0} W_{CB0} + AW_{AB0} (W_{BC0} W_{CB0})^2 \end{aligned} \tag{2.32}$$

subsequently,

$$\begin{aligned} B_3 &= AW_{AB0} + C_2 W_{CB0} \\ &= AW_{AB0} + (AW_{AB0} + AW_{AB0} W_{BC0} W_{CB0} + AW_{AB0} (W_{BC0} W_{CB0})^2) W_{BC0} W_{CB0} \\ &= AW_{AB0} + AW_{AB0} W_{BC0} W_{CB0} + AW_{AB0} (W_{BC0} W_{CB0})^2 + AW_{AB0} (W_{BC0} W_{CB0})^3 \end{aligned} \tag{2.33}$$

and finally, we will have

$$B_n = AW_{AB0} [I + W_{BC0} W_{CB0} + (W_{BC0} W_{CB0})^2 + (W_{BC0} W_{CB0})^3 + \cdots + (W_{BC0} W_{CB0})^n]. \tag{2.34}$$

where A is the input signal matrix.

Using matrix diagonalization technique, we can express the product of two matrices $W_{BC0} W_{CB0}$ as: $W_{BC0} W_{CB0} = (V^{-1})^T D V^T$. (2.35)

where V is the matrix of eigen vectors of $W_{BC0} W_{CB0}$ and D is the diagonal matrix of eigen values of $W_{BC0} W_{CB0}$. If the maximum eigen value in D is less than 1, B_n will gradually converge to a stable value. In order to have the eigen value of $W_{BC0} W_{CB0}$ less than 1, we normalize W_{BC0} and W_{CB0} to have the sum of each column equal to α , where α is called the convergence parameter, and $0 < \alpha < 1$. Small α forces the faster convergence of linear iterations. Not only this normalization ensures stability of the iterative process, it also makes equal sum of weights for signals that enter each neuron in the winner take all nonlinear output transformation. This removes any statistical bias in selecting a winner, and the only criteria for the winning neuron is similarity of its weights to the input signal vector. In a few iterations, we can reach stable values for all the signals in layers B and C.

2.4.2.2 Nonlinear Iterations

Once the signals in layers B and C become stable after the linear iterations, we will start the non-linear iterations, which allow the network to gradually push up the winner signal values and push down the losers.

Let us denote the signals B_i and C_i after the nonlinear sigmoidal transformation in equation (5) as $\hat{B}_i$ and $\hat{C}_i$ respectively, so for one nonlinear iteration, we will have

1. $C_i = \hat{B}_i W_{BCi}$, (2.36)

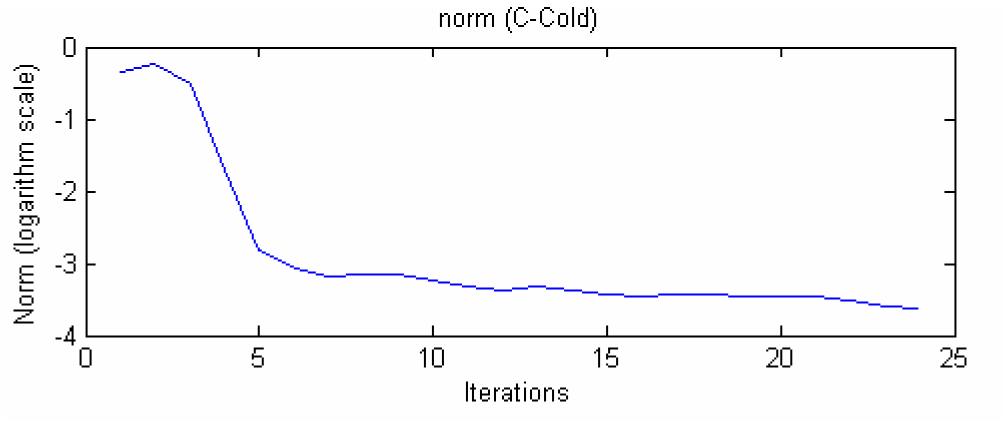
2. $B_i = \begin{bmatrix} A & \hat{C}_{i-1} \end{bmatrix} \begin{bmatrix} W_{ABi} \\ W_{CBi} \end{bmatrix}$, $i = 0,1,...n$, (2.37)

3. Adjust the weights W_{BCi}, W_{ABi}, and W_{CBi} by solving equation (2.4).

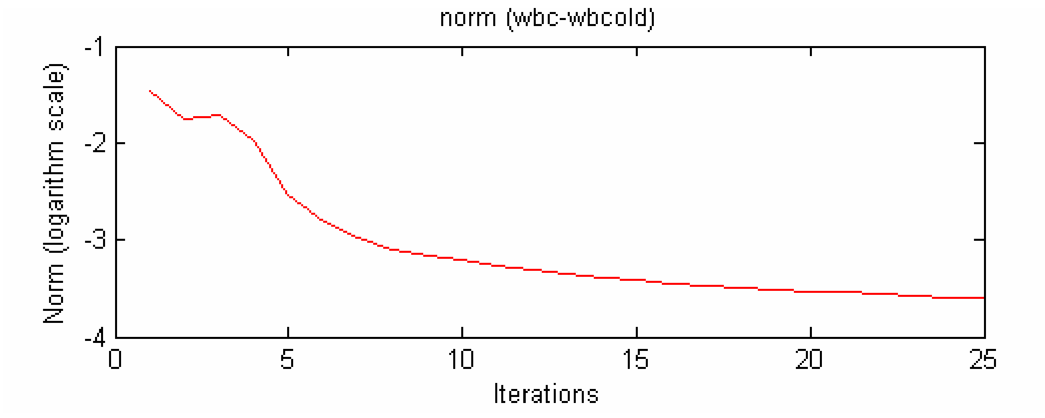
4. Normalize weights W_{BCi}, W_{ABi}, and W_{CBi}.

After a few nonlinear iterations, all the signals and interconnection weights will become stable.

Fig. 2.13 shows the convergence of the signal C_i and weights W_{BCi} during the iterations with convergence parameter α set to 0.5. Fig. 2.13 a) shows the norm of $\|C_i - C_{i+1}\|$ after each iteration, while Fig. 2.13 b) shows the norm of $\|W_{BCi} - W_{BCi+1}\|$ in logarithm scale. As we can see from the figure, they converge quickly after 15 iterations.



(a)



(b)

Fig. 2.13 Convergence of the signal and weights

2.4.3 Learning Performance and Results Analysis

Fig. 2.14 shows the surface plot of the values at the output layer C as a function of the input sample number y and the output neuron activities x. On Fig. 2.14 when function value equals 1 at x, y coordinates, this means that the sample number y was classified as class x. Based on this surface plot, we can correctly classify all 19 samples.

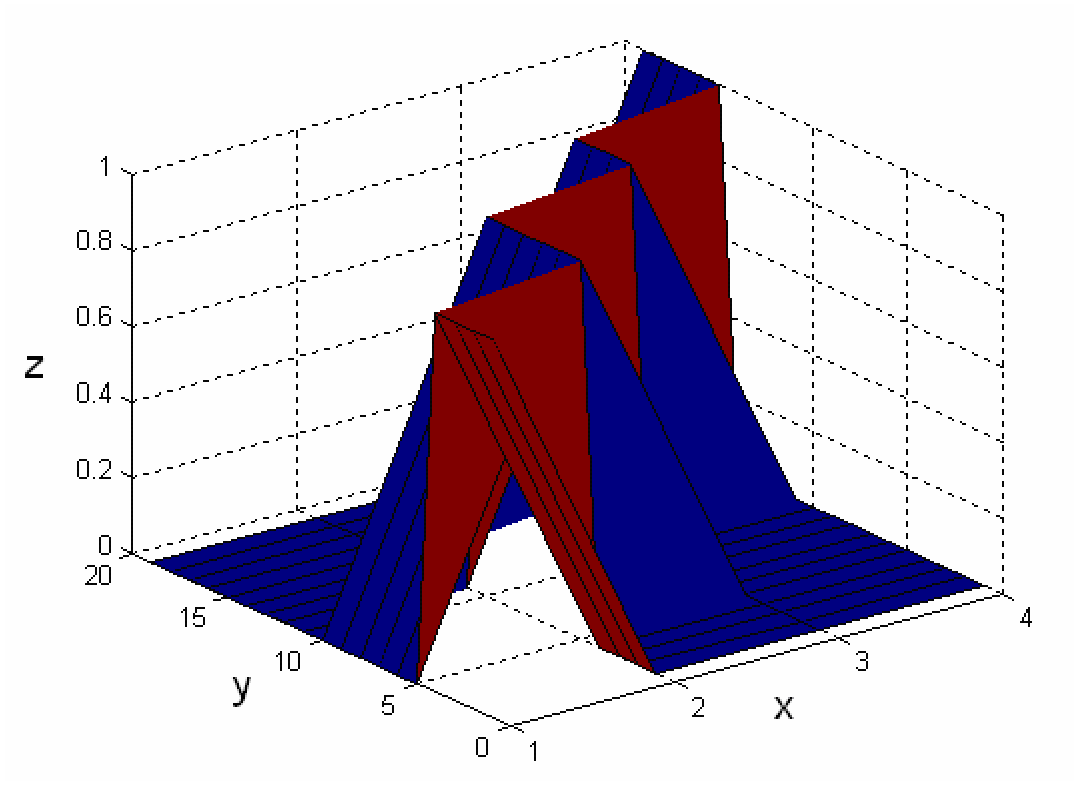
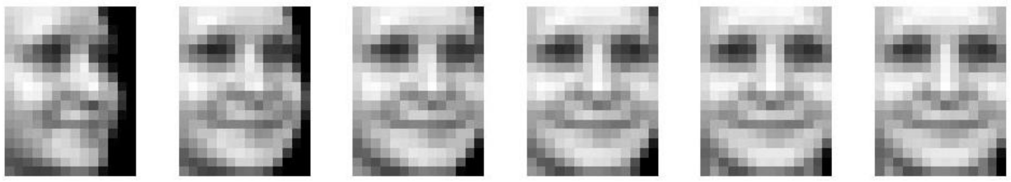


Fig. 2.14 Surface plot of the output layer

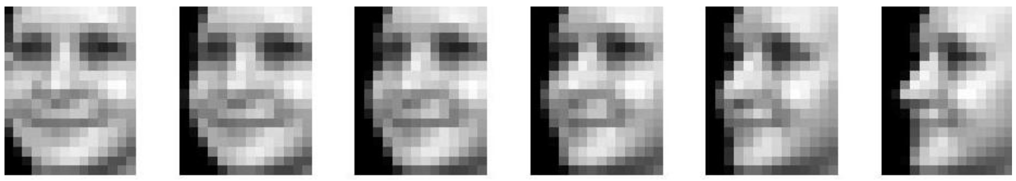
Comparing to the surface plot on Fig. 2.6, the feedback network provided clear and more desirable classification boundaries.

If we still use the same samples representing 4 categories as shown in Fig. 2.5, we will get the classification results as shown in Fig. 2.15(a)-(d). In this 3-layer SOWTAC, main_wta_output_setting_nonlinearmodel_cleanup.m is used in the folder named RLS_WTA_Classification (see Appendix A).

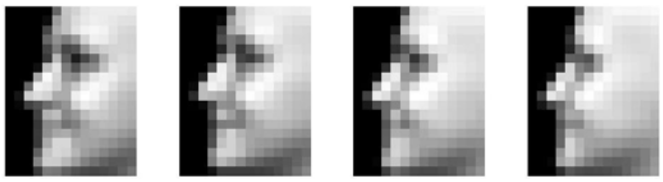
(a) Images classified as class 1



(b) Images classified as class 2



(c) Images classified as class 3



(d) Images classified as class 4

Fig. 2.15 Images classified as different classes

Now we will illustrate how the updated weights W_{ab} changed from the initial W_{ab0} after the nonlinear iterations. Fig. 2.16 shows the norm of difference between the singular values of the initial W_{ab0} matrices and updated W_{ab} matrices, of all 48 different feature matrices W_{ab} depicted in Fig. 2.9. We can see that the peak value of norm of the difference between the singular values occurs at the sample locations 8th, 16th, 27th, 34th, 39th, which indicate the locations of the most distinguishing parts of the images. These most distinguishing locations are shown in Fig. 2.17.

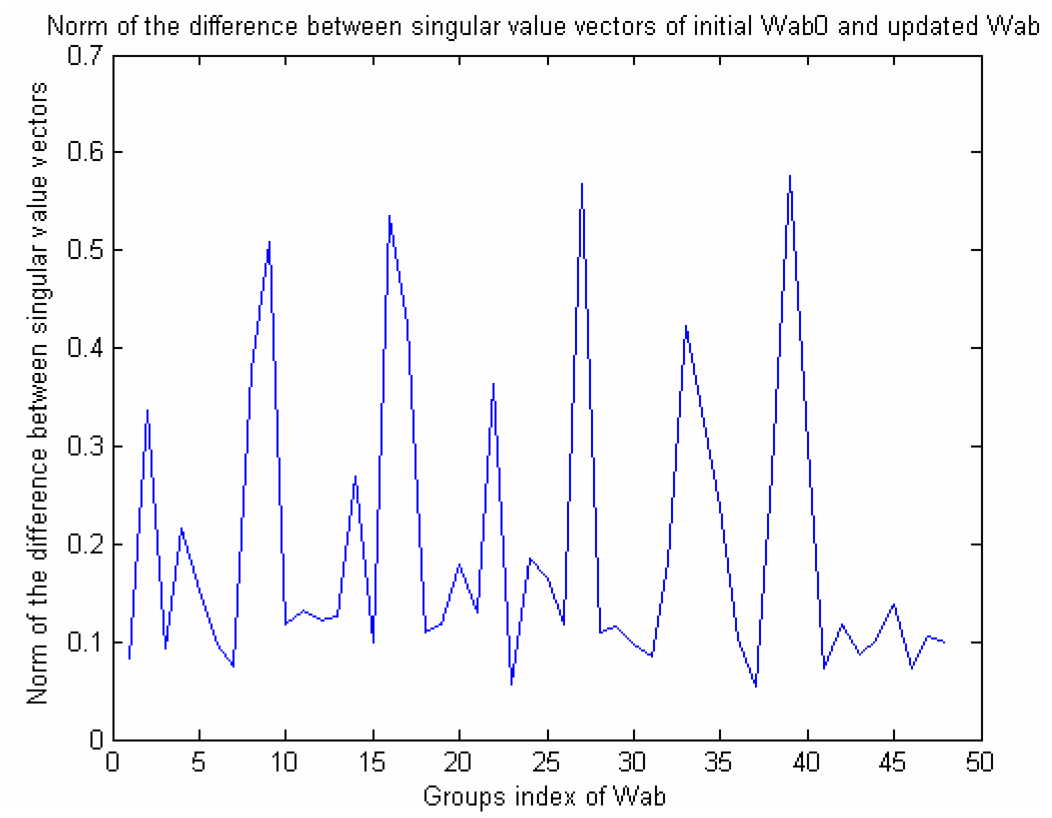


Fig. 2.16 Norm of the difference between singular value vectors of initial W_{ab0} and W_{ab}

By comparing the selected location with an eigenface, we can identify these features with eye, nose, and lip area.

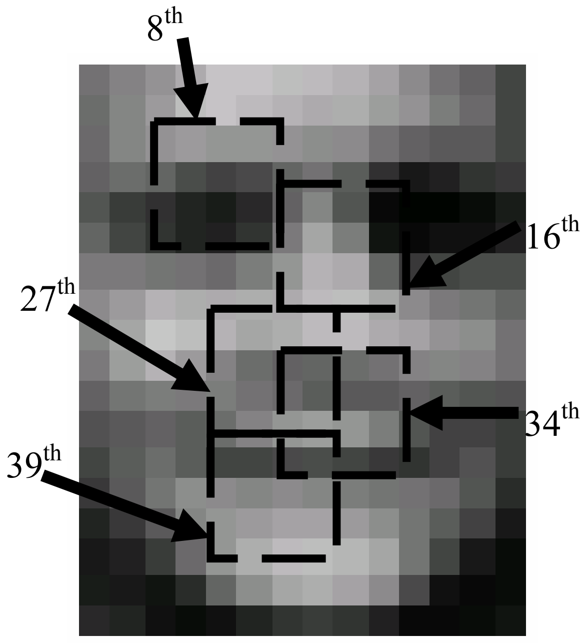


Fig. 2.17 Locations of the most distinguishing parts of the images shown on the eigenface background

Now we compare the singular value vectors of initial W_{ab0} and the updated W_{ab} at location N=8. Singular values can be used as a measure of independence of the associated feature vectors. Thus large singular values represent independent feature vectors, while small indicate dependence of the corresponding feature vector on other feature vectors. Thus, we can use SVD to measure the quality of the extracted features. Let us compare the singular values of initial W_{ab0} which is obtained by our handmade features with singular values of the updated weight matrix W_{ab} at the feature location 8. As shown in Fig. 2.18, we can see that features 11-16 of initial W_{ab0} are dependent on features 1-10, while most of the features of the updated matrix W_{ab} are independent. This may indicate that the updated features represent local requirements for self-organized learning much better than hand made features.

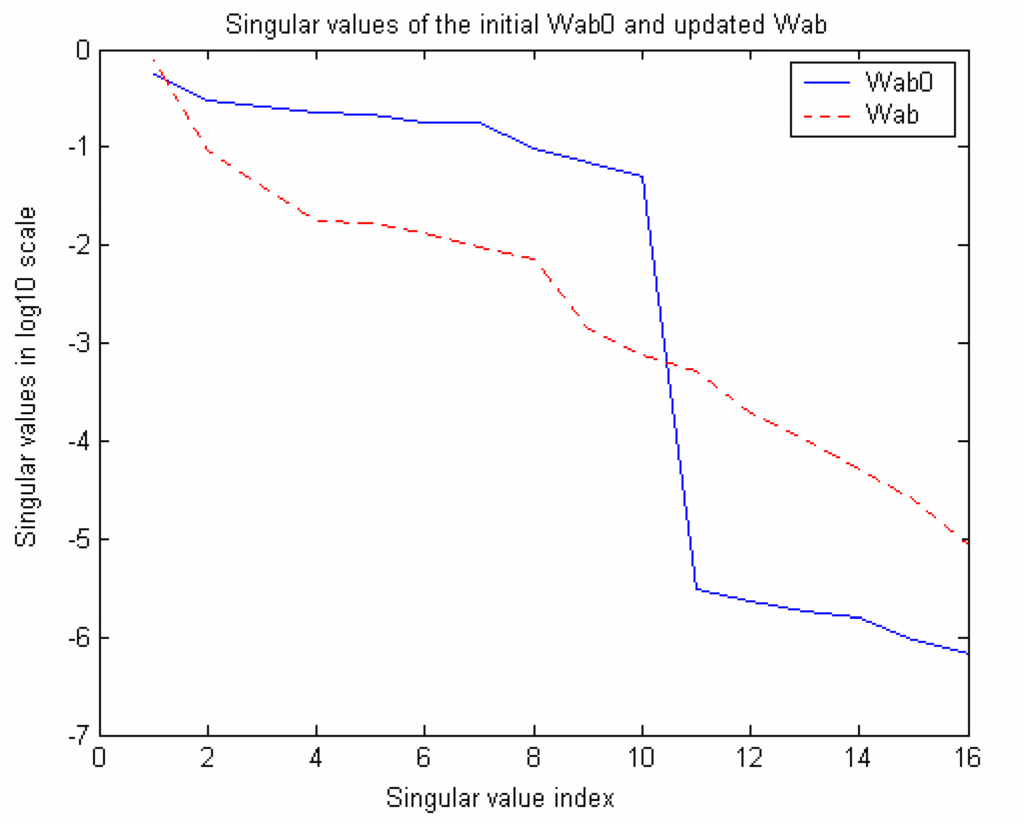


Fig. 2.18 Singular values of the initial W_{ab0} and the updated W_{ab} at location N=8

Fig. 2.19 shows the initial W_{ab0} and the updated W_{ab} at the location N=8. We can see that after the nonlinear iterations, W_{ab} changed significantly from the initial W_{ab0}.

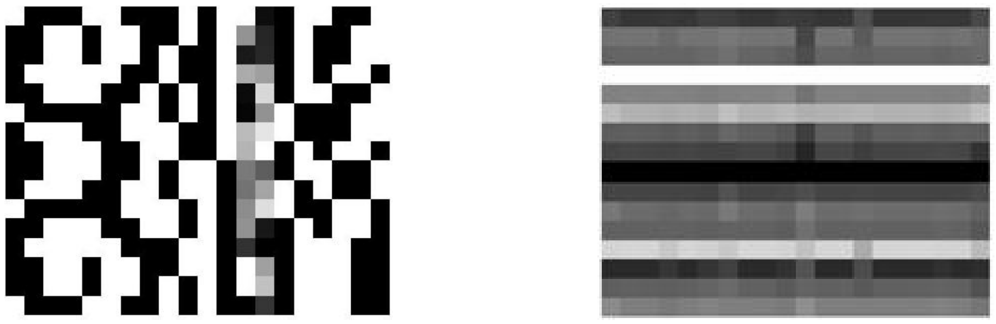


Fig. 2.19 Initial W_{ab0} and the updated W_{ab} at location N=8

Fig. 2.20 shows the 2D samplings from the 19 input images at the location N=8. Note that each row in Fig. 2.20 represents a 2D sampling from one image at this location. The columns represent the pixel location in the sampled images.

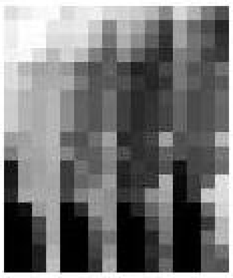


Fig. 2.20 2D samplings from the input images at location N=8

Now let us compare the singular value vectors of initial W_{ab0} and the updated W_{ab} at location N=1. From Fig. 2.21 we can see that for the updated matrix W_{ab}, features 14-16 are dependent on features 1-13. The singular value of the updated matrix W_{ab} is different from initial W_{ab0} matrix at features 11-13. Fig. 2.21 and Fig. 2.22 both indicate that W_{ab} changed little from W_{ab0}. This may confirm small significance of features at this location as was indicated by small difference in singular vectors of W_{ab} and W_{ab0} shown in Fig. 2.16.

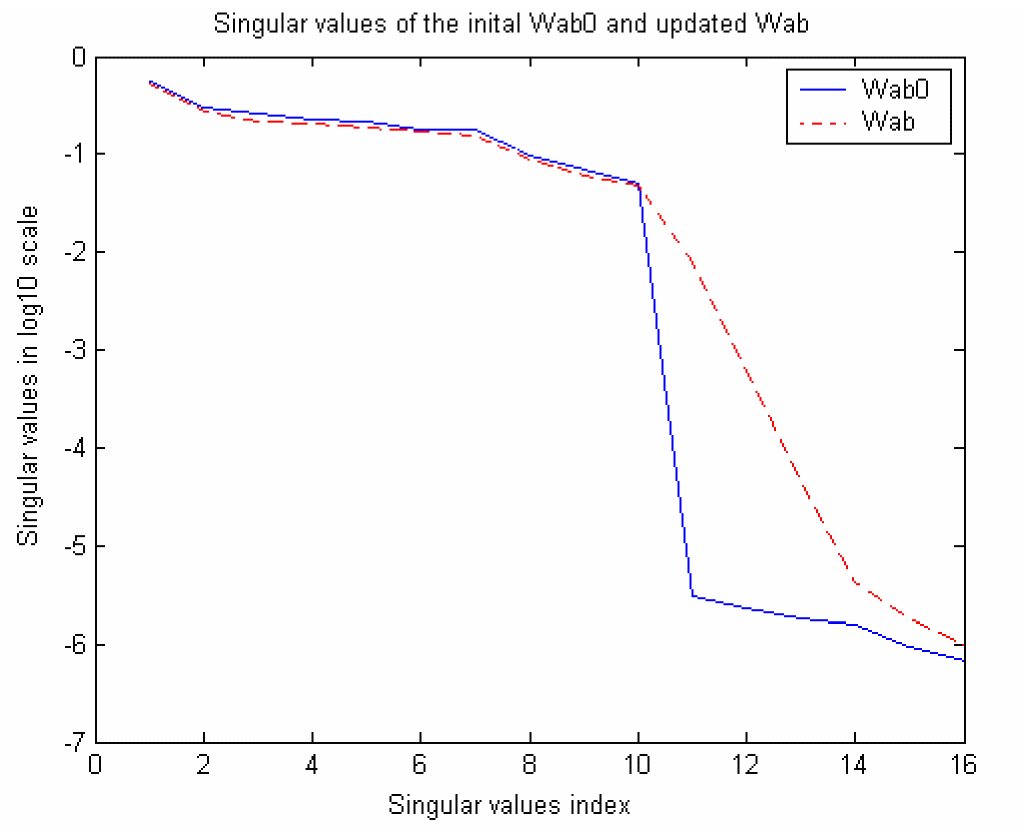
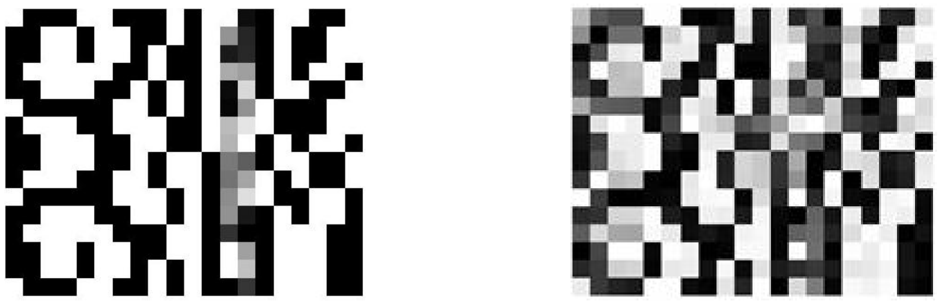


Fig. 2.21 Singular values of the initial W_{ab0} and the updated W_{ab} at location N=1



(a) Initial W_{ab0} (b) updated W_{ab}

Fig. 2.22 Initial W_{ab0} and the updated W_{ab} at location N=1

Fig. 2.23 shows the 2D sampling from the input images at N=1. While compared with Fig. 2.20, we can see that there is a much smaller number of independent image samples at this location, and the singular values of the updated W_{ab} look much more similar to those of the handmade features.

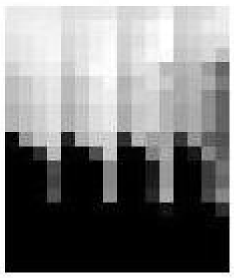


Fig. 2.23 2D samplings from the input images at location N=1

As discussed in Fig. 2.18 and 2.21, each W_{ab} only sees one certain area of all the input images. Now we assign all N=48 2D sampling areas from one image to one $\tilde{a}_{pq}$, which means that the weight W_{ab} corresponding to $\tilde{a}_{pq}$ will interact with the whole image. Fig. 2.24 shows the difference between the singular values of the initial W_{ab0} and the updated W_{ab}. Singular values of the updated W_{ab} look more uniform, which indicates that W_{ab} use different features than features represented by the initial weights.

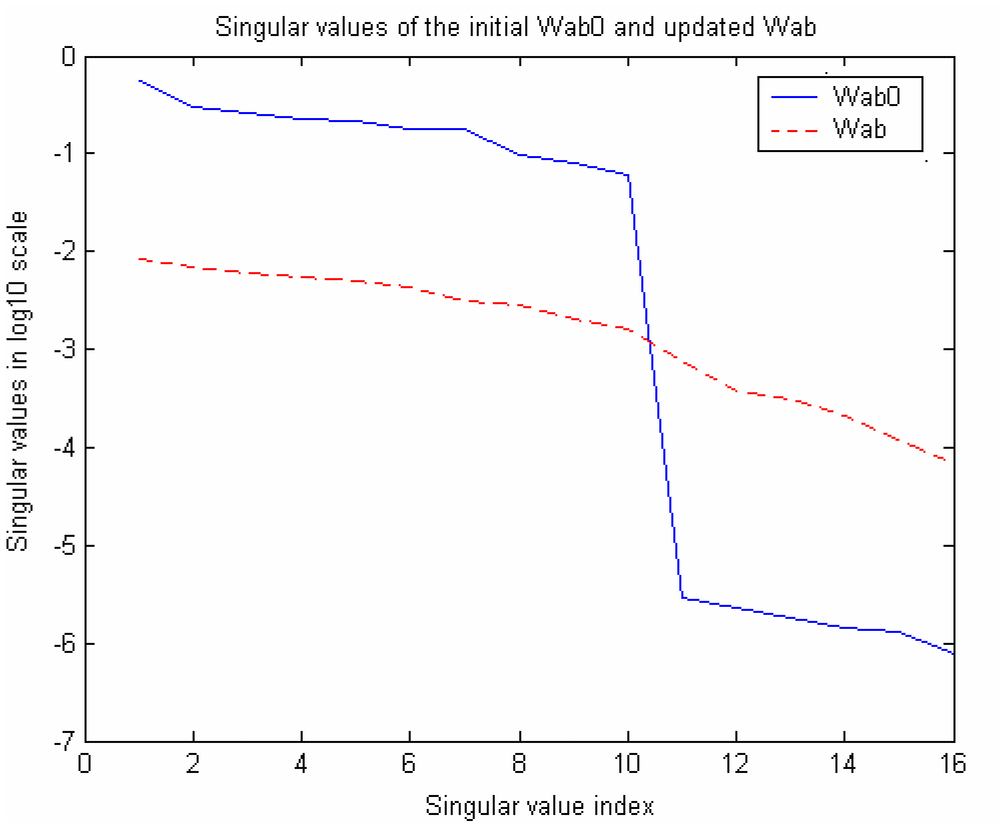
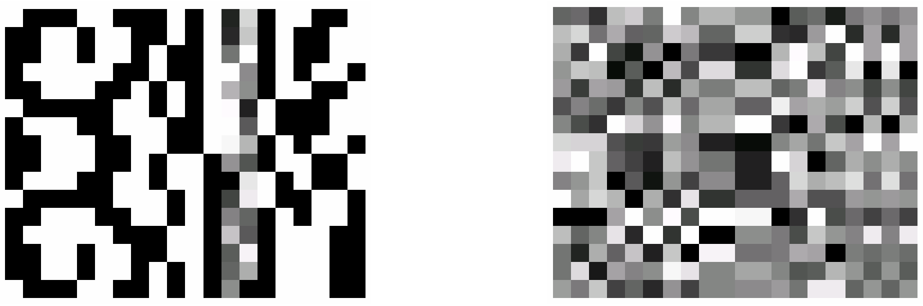


Fig. 2.24 Singular values of the initial W_{ab0} and the updated W_{ab}

Fig. 2.25 shows the initial W_{ab0} and the updated W_{ab}. Because weight W_{ab} interact with the whole image, the amount of difference between the initial W_{ab0} and the updated W_{ab} is lager than the difference as shown in Fig. 2.22 and smaller than the difference in Fig. 2.19.



(a) Initial W_{ab0} (b) updated W_{ab}

Fig. 2.25 Initial W_{ab0} and the updated W_{ab} at location N=1

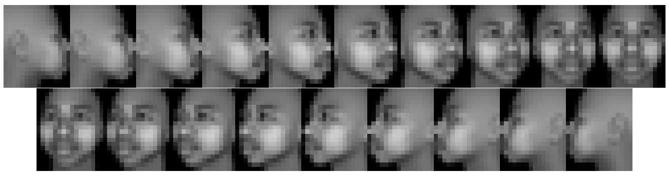
3. Continuous Observation Network

3.1 Network Structure

In the previous sections, we discussed how to use the hierarchical feedback WTA structures to classify different positions. This hierarchical WTA feedback network builds invariant transformations, such that the network can recognize one object at different positions. In this section, another application of the hierarchical WTA feedback network is described. In this application, the network receives continuous input representing the same object over a period of time. For instance, Figs 3.1(a) and (b) show a sequence of images obtained from continuous observation of two different people (called here person A and B).

The WTA network organization is modified to support continuous observation. The task of the network is to maintain observation and correct identification of each person, even when a current view of the person cannot be uniquely recognized. Such observation requires a memory of past activities, and in particular, it uses a prior assumption that the new scene is just a modification of the previous scene. We call this assumption the sameness principle. Under the sameness principle, the new input represents the same scene (or object) as the previous one. Structurally, this is supported by the introduction of dual neurons. Dual neurons generate feedback prediction signals in anticipation that the input represents the same object as before. With the help of the feedback prediction signals, the network maintains the same internal representation for the received input, unless there is a strong disagreement between the feedback prediction and the received input.

The continuous observation network (CON) structure is shown in Fig. 3.2. As the self-organizing WTA classifier (SOWTAC), the CON network uses the same organization of WTA structures between layers B and C. Fig. 3.1(a) and (b) show the database we used for the continuous observation.


(a) Continuous observation of person A

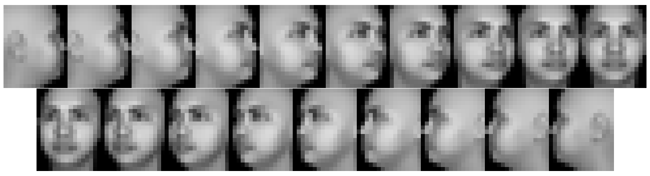

(b) Continuous observation of person B

Fig. 3.1 Continuous observation of person A and B

SOWTAC operation was described using the batch mode, however, in the CON, we use online mode. As each input image is presented, the network adjusts its weights iteratively. Dual neurons provide feed back prediction signals to the input of layer C. A CON shown in Fig. 3.2 is a 4-layer network with dual neurons. Signals in layers A, B, C, and weight matrices W_{AB}, W_{BC} and W_{CB} were described in detail in Section 2.4. Layer D represents the dual neurons with the feedback to layer C.

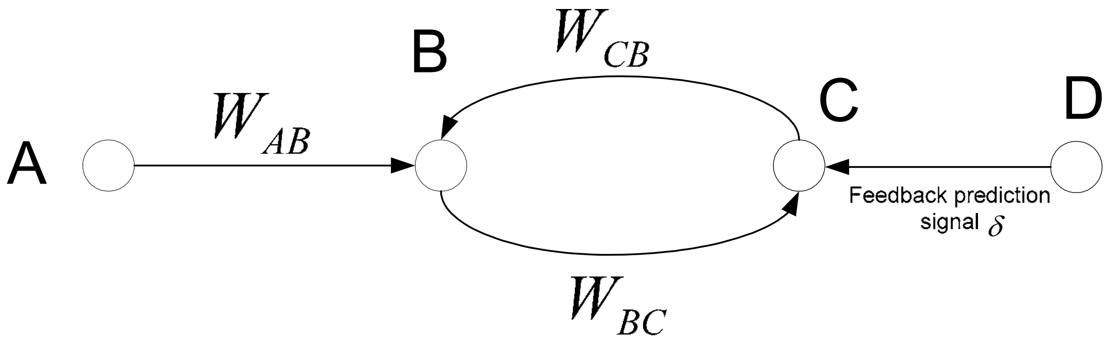
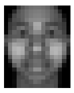


Fig. 3.2 Signal flow for the 4-layer CON with dual neurons

3.1 Network Operation

The network is first trained using clear (strong) view of each person through a supervised training (see Fig. 3.3 (a) and (b)). The initial weight settings of W_{BC} and W_{AB} are obtained as described in section 2.4.1.


(a) Strong view of person A

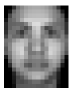

(b) Strong view of person B

Fig. 3.3 Strong view of person A and B

For the online mode learning, the input images are sent to the network one by one, and are appended to the strong view images that were used for initial weight setting. After each input image is sent to the network, the network undergoes both linear and nonlinear iterations as described in Section 2.4.2.

During the first linear iteration when the first input image comes in ($C_0 = 0$), there is no internal feedback signal from layer C. Neurons on layer B only receive signals from the neurons on layer A. Thus $B_1 = AW_{AB0}$. After that, neurons on layer B receive both signals from the neurons on layer A and layer C, and

during the linear iterations:

$$B_{i+1} = \begin{bmatrix} A & C_i \end{bmatrix} \begin{bmatrix} W_{ABi} \\ W_{CBi} \end{bmatrix} \quad i \geq 0 . \tag{3.1}$$

Initially, layer C receives only the feed forward signals from layer B. Thus the signals in layer C can be computed as:

$$C_{i+1} = B_{i+1} W_{BCi+1} . \tag{3.2}$$

After the linear iterations converge, the neurons on layer B go through the sigmoidal transformation, thus the neurons on layer C are updated as:

$$C = \hat{B} W_{BC} . \tag{3.3}$$

This result, after nonlinear sigmoidal transformation is stored in dual neurons through forward links of fixed weight equal to 1 (see Fig. 3.4). Once activated, dual neurons stay active maintaining the memory of the most recent event. When the next input is presented to the CON, the dual neurons on layer D provide feedback prediction signals to the input of the neurons on layer C. If the previous winning node of layer C was C_{winner}, then the corresponding dual neuron on layer D provides positive feedback δ to this node:

$$C'_{winner} = C_{winer} + \delta . \tag{3.4}$$

And, the neurons on layer B should be updated as:

$$B = \begin{bmatrix} A & C' \end{bmatrix} \begin{bmatrix} W_{AB} \\ W_{CB} \end{bmatrix} . \tag{3.5}$$

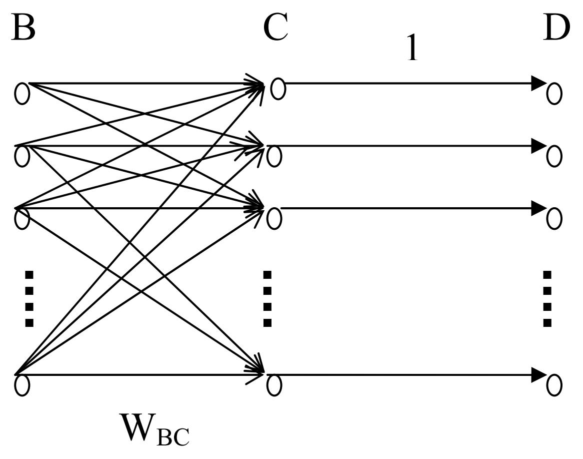


Fig. 3.4 Feed forward activation of dual neurons

During nonlinear iterations:

1. $C_i = \hat{B}_i W_{BCi}$, (3.6)

2. $B_i = \begin{bmatrix} A & \hat{C}_{i-1} \end{bmatrix} \begin{bmatrix} W_{ABi} \\ W_{CBi} \end{bmatrix}, \; i = 0,1,...n$, (3.7)

3. Adjust the weights W_{BCi}, W_{ABi}, and W_{CBi} by using the formula (2.4).

4. Normalize weights W_{BCi}, W_{ABi}, and W_{CBi}.

As a result, the feedback signal helps the network to maintain the same internal representation of the perceived input. This prediction of the expected input continues unless there is a strong disagreement between the feedback predictions and the received input, at which point different dual neuron is activated and it helps to maintain predictions for a new type of input.

3.3 Learning Performance and Results Analysis

In this experiment, we use the database shown in Fig. 3.1 to test the continuous observation. A sequence of 20 images were presented to the network one by one. Table 1 shows the results of the test by two neural networks. For comparison, the test results without the dual neurons on layer D are also provided. Network 1 has layer D with dual neurons which provided feedback signals to layer C. The feedback prediction signal δ is set to 0.08. Network 2 doesn't have dual neurons. Each row of Table 1 contains a test image, its true classification, and classification results by Network 1 and Network 2. Network 1 correctly recognized all the test images of person A even with the side view, while Network 2 confused person A with person B in images 6-10. As image 11, which represents the clear view of person B, was sent to both networks, Network 1 correctly recognized that this image is of person B and modified firing pattern of its dual neurons. Subsequently, Network 1 switched to person B during images 11-20. Again, Network 2 confused person B with person A in images 17-19.

Table 3.1 Results of the invariant test 1 by two networks

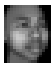
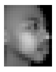
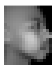
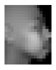
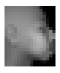
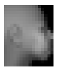
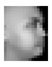
Number	Input images	True answer	Test results (Network 1)	Test results (Network 2)
1		Person A	Person A	Person A
2		Person A	Person A	Person A
3		Person A	Person A	Person A
4		Person A	Person A	Person A
5		Person A	Person A	Person A
6		Person A	Person A	Person B
7		Person A	Person A	Person B
8		Person A	Person A	Person B
9		Person A	Person A	Person B
10		Person A	Person A	Person B
11		Person B	Person B	Person B
12		Person B	Person B	Person B
13		Person B	Person B	Person B
14		Person B	Person B	Person B
15		Person B	Person B	Person B
16		Person B	Person B	Person B
17		Person B	Person B	Person A
18		Person B	Person B	Person A

Table 3.1: continued

19		Person B	Person B	Person A
20		Person B	Person B	Person B

Discussion

In Table 3.1, the presentation sequence of person A ends with a weak view, while the presentation sequence of person B begins with the strong view. In such case, when a strong view of person B comes in, although the feedback prediction signal anticipates the same object as before (person A), the received input (strong view of person B) is strongly different from the internal representation of the previous object so that the network correctly recognized this image as person B.

This example illustrates another aspect of the continuous observation with dual neurons. In Table 3.2, the representations of person A is from a strong view to a weak view (images 1-10). When presented with the weak view of person B in image 11, the feedback prediction signal caused the network 1 to continuously match the received input with the same internal representation (person A), until the network receives strong disagreement between the received input and the internal representation in image 18.

Table 3.2 Results of the invariant test 2 by two networks

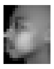
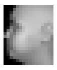
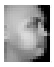
Number	Input images	True answer	Test results (Network 1)	Test results (Network 2)
1		Person A	Person A	Person A
2		Person A	Person A	Person A
3		Person A	Person A	Person A
4		Person A	Person A	Person B
5		Person A	Person A	Person B
6		Person A	Person A	Person B
7		Person A	Person A	Person B
8		Person A	Person A	Person B
9		Person A	Person A	Person B
10		Person A	Person A	Person B
11		Person B	Person A	Person B
12		Person B	Person A	Person B
13		Person B	Person A	Person A
14		Person B	Person A	Person A
15		Person B	Person A	Person A
16		Person B	Person A	Person A
17		Person B	Person A	Person A
18		Person B	Person B	Person B

Table 3.2: continued

19		Person B	Person B	Person B
20		Person B	Person B	Person B
21		Person B	Person B	Person B
22		Person B	Person B	Person B
23		Person B	Person B	Person B
24		Person B	Person B	Person B
25		Person B	Person B	Person A
26		Person B	Person B	Person A
27		Person B	Person B	Person B
28		Person B	Person B	Person A
29		Person B	Person B	Person A
30		Person B	Person B	Person A

At this point, dual neurons switch to support continuous observation of person B. This type of image recognition is in a good agreement with human vision system if a certain input is expected, based on the previously obtained images. Notice the erratic behavior of Network 2 which confused person A for person B in images 4-12, and person B for person A in images 25, 26, 28-30. In this CON, main_wta_output_setting_nonlinearmodel_cleanup_online.m is used in folder RLS_WTA_Invariance (see Appendix A).

4. Retina Sampling and Saccade Movements

4.1 Introduction

Human visual perception is achieved in two stages. First, the light entering the cornea is projected to the back of the eye and converged into electrical signals by the retina. These signals represent time correlated saccade samples of dynamically changing visual images. Second, these electrical signals received from the retina after local preprocessing by ganglion cells are sent to the cortex neurons though the optic nerve for pattern recognition and further processing.

Human retina contains two types of photoreceptors. Basically, rods are higher sensitivity, and are specialized for night vision; while cones are lower sensitivity, and are specialized for day vision. Fig. 4.1 shows the experimentally determined human rods and cones distribution diagram [Riede]. We can see that human cone density reaches peak at the fovea, and declines rapidly in the first few degrees around it and more slowly outside the fovea. Density of the photoreceptors is not symmetrical and covers larger portion of the visual field in the nasal direction (see Fig. 4.2). It represents relative density of rods and cones from the center of the fovea, at projection angle 0^o, towards nose (projection angle up to 70^o) and towards temple (projection angle up to 55^o).

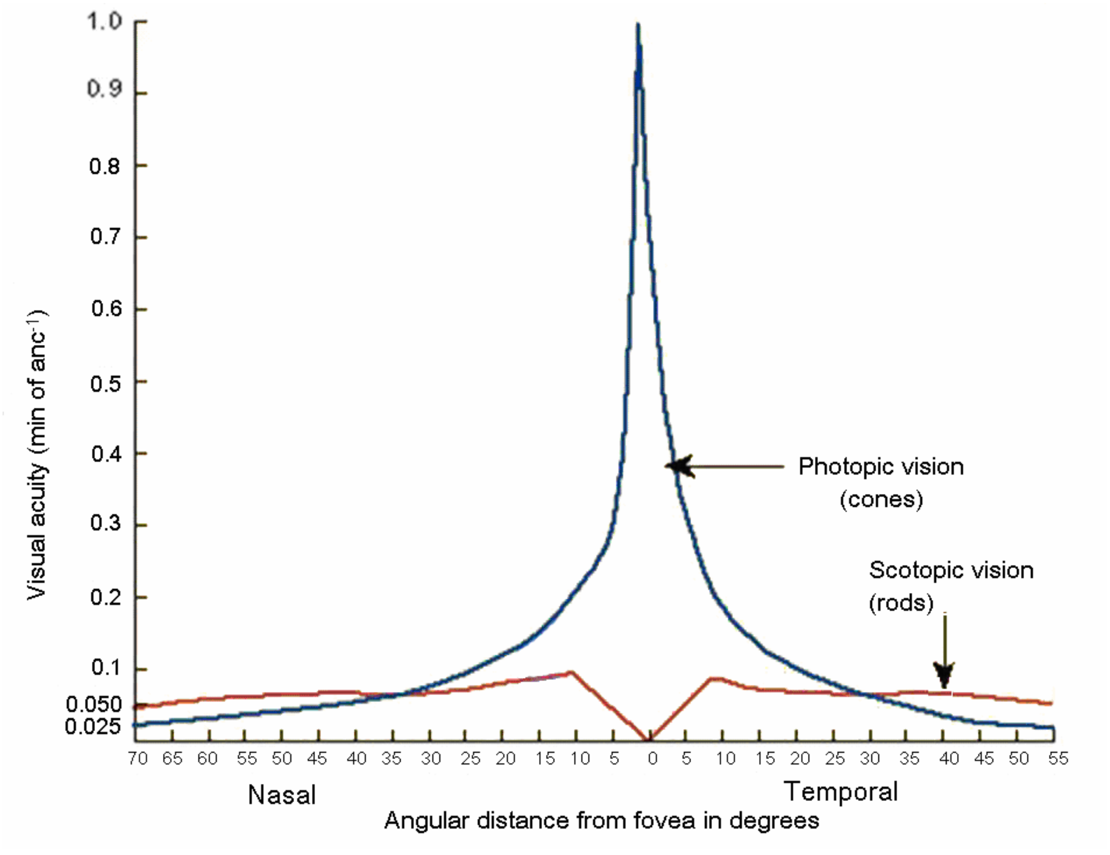


Fig. 4.1 Rods and cones distribution diagram

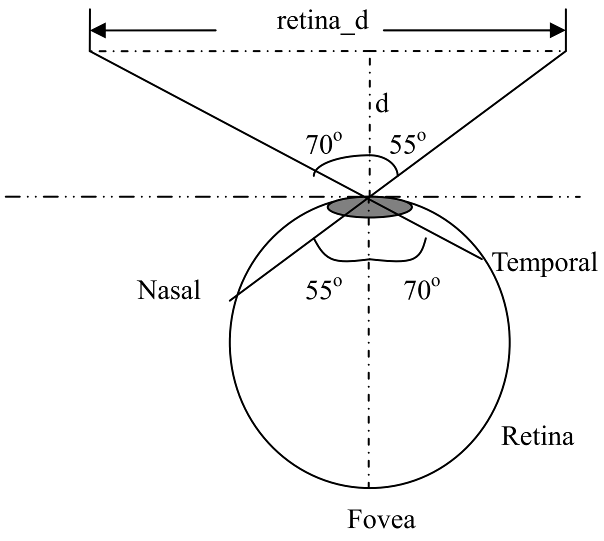


Fig. 4.2 Image projection on the retina

4.2 Retina Models

4.2.1 Retina Model 1

We first approximate the human rods and cones distribution by the superposition of Gaussian and linear distribution. The probability density of superposition of Gaussian and linear distribution is shown in Fig. 4.3.

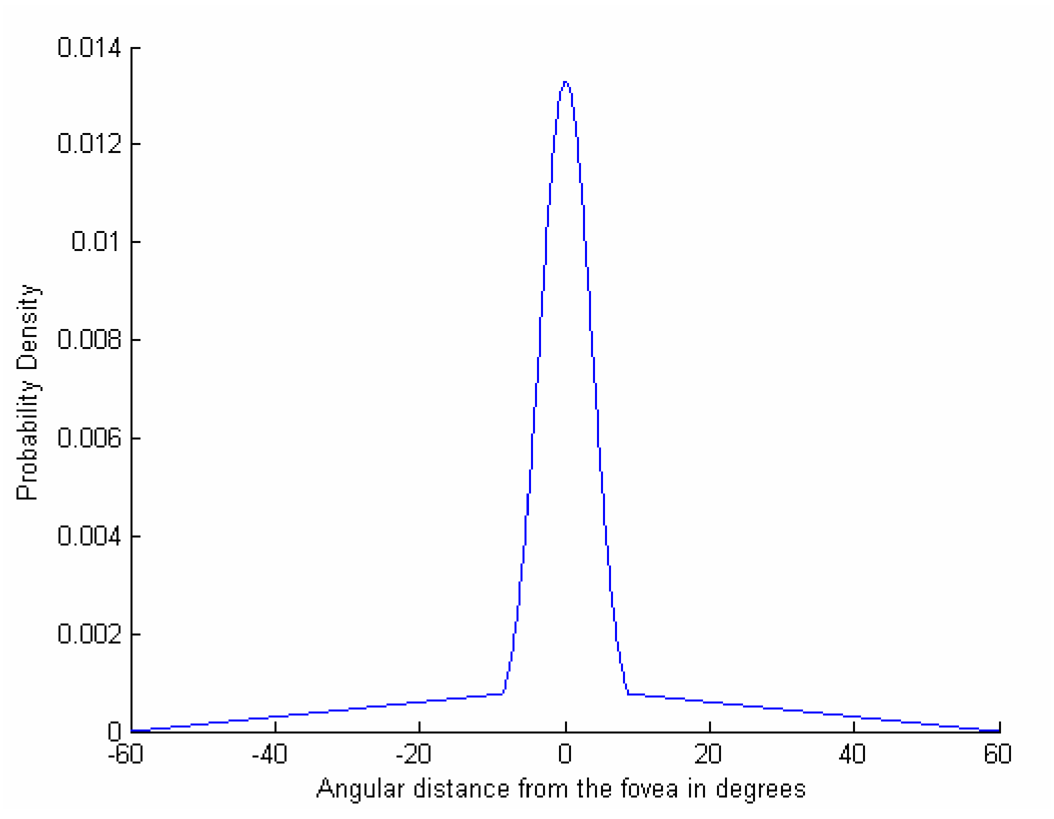


Fig. 4.3 Probability density of superposition of Gaussian and linear distribution

Since Fig. 4.3 represents only one dimensional distribution of cones and rods density, we modified the probability density in Fig. 4.3 by multiplying it by the absolute value of the distance from the fovea. Fig. 4.4 shows the resulting probability density of the modified model. By using this density function at a given angular distance and at a random zenith angle from 0 to 2π.

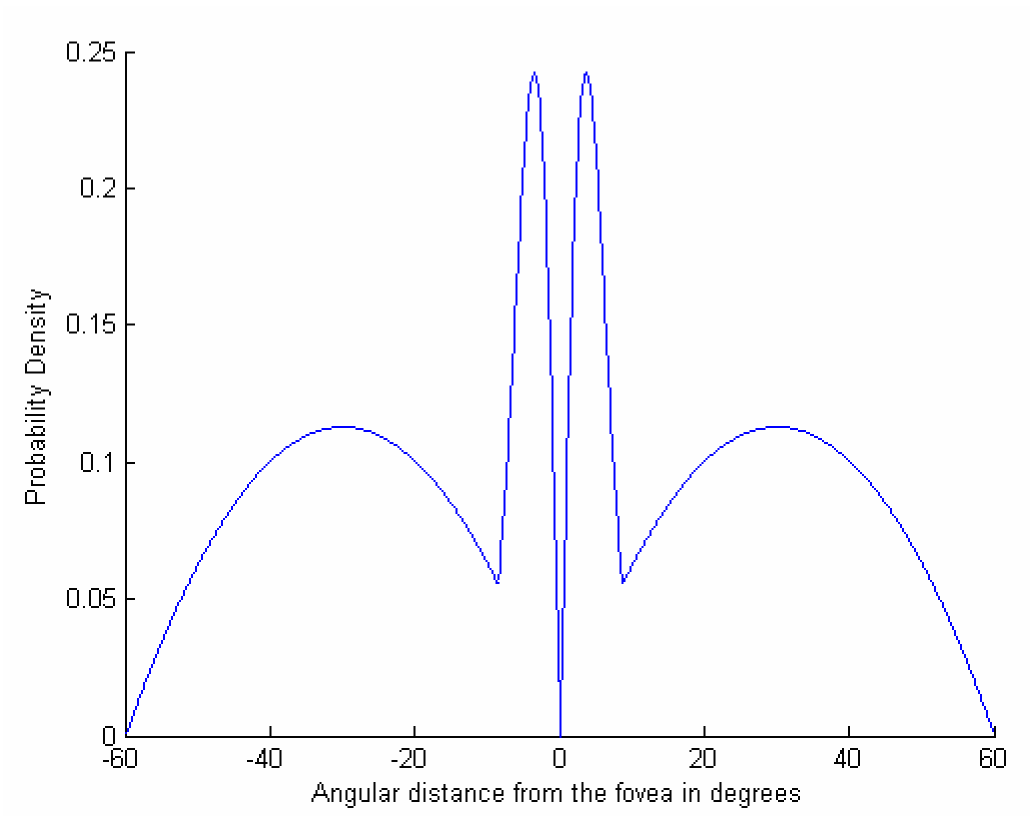


Fig. 4.4 Probability density of superposition of Gaussian and linear distribution

(multiplied by absolute value of the distance from the fovea)

This density considers that with a given value of the one dimensional density, the number of photoreceptors would increase quadratically with the distance from the fovea. By using the PDF from Fig. 4.4, we can easily generate distribution of photoreceptors with the one dimensional density as shown in Fig. 4.3.

Fig. 4.5 shows the corresponding cumulative distribution of the probability density in Fig.

4.3.

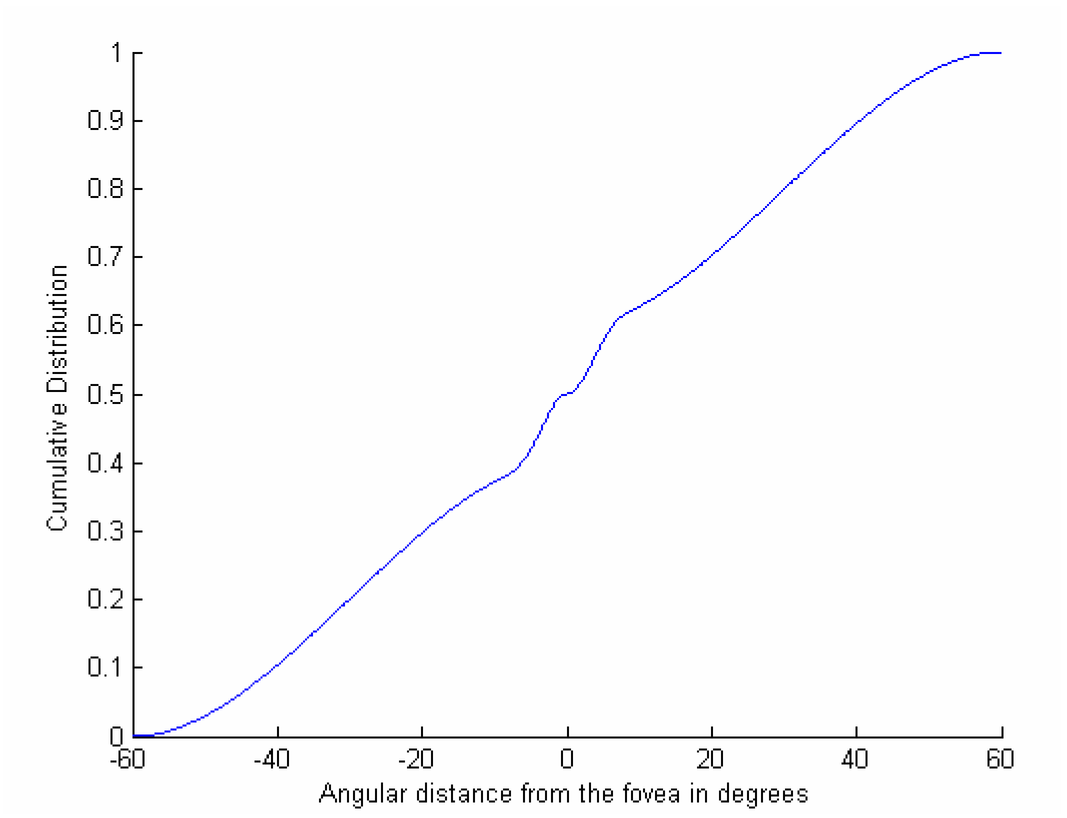


Fig. 4.5 Cumulative density of superposition of Gaussian and linear distribution (multiplied by absolute value of the distance from the fovea)

Using the obtained cumulative density function, we may generate a 2D distribution of cones and rods in the retina model. Fig. 4.6 illustrates the generation process of one photoreceptor of 2D distribution. To obtain the 2D distribution of photoreceptors that implements retina model 1, we first generate randomly uniform numbers r_u between 0-1, and then generate the retina sampling angular distances r_s from fovea by using the inverse function of the cumulative density function shown in Fig. 4.5:

$$r_s = \left| CDF^{-1}(r_u) \right|. \tag{4.1}$$

These distances r_s are combined with the randomly generated angles α between 0 - 2π, which represent the second angular coordinate (zenith angle) of each photoreceptor. As shown in Fig. 4.6, once r and α is determined, the position of the photoreceptor Q can be obtained, where $x_s = r_s \cos\alpha$, $y_s = r_s \sin\alpha$. (4.2)

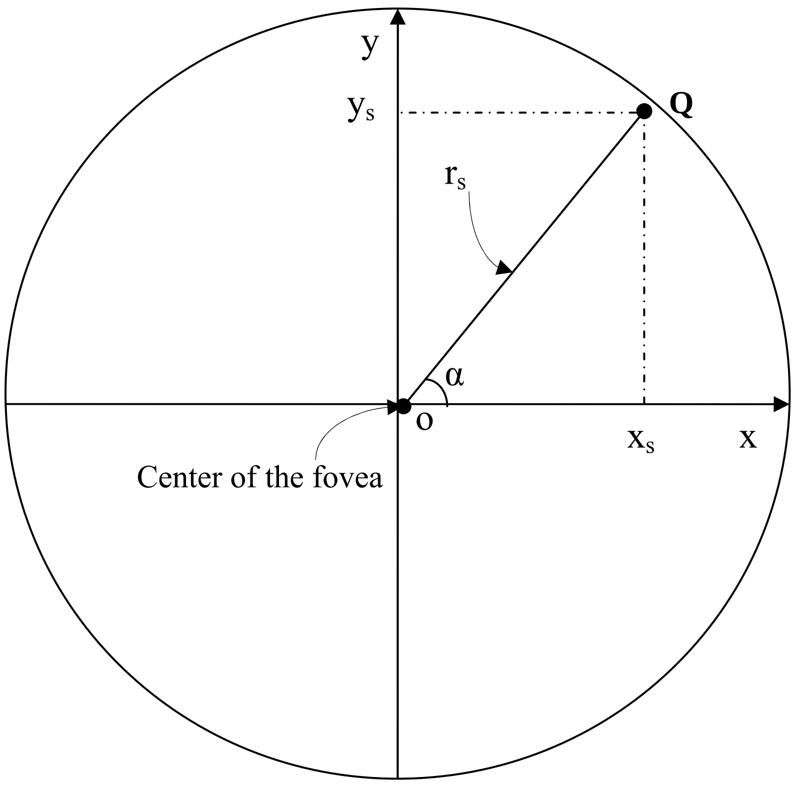


Fig. 4.6 Generation of 2D distribution of photoreceptors (model 1)

Fig. 4.7 shows an example of generated retina sampling based on model 1.

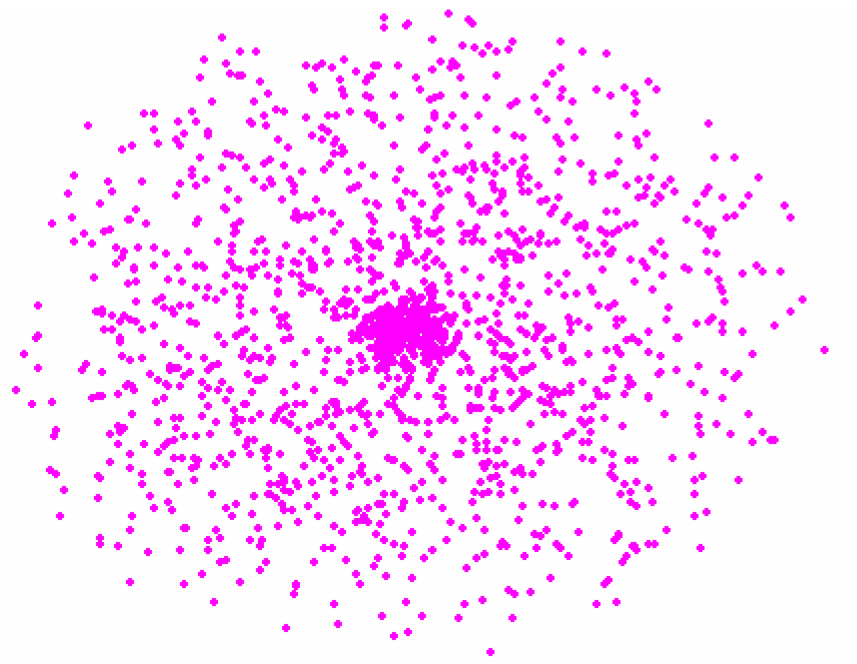


Fig. 4.7 Retina sampling distribution (model 1)

After the image is sampled by using the retina model shown in Fig. 4.7, we need to compact the sampled data in a uniform way which can be presented to the neural network. As shown in Fig. 4.7, there are many more samples in the center than in the peripheral area. In general, we need to first scale the 2D sampled data to a 2D uniform array, then move the overlaps to the nearest empty neighborhoods to achieve the maximum use of the neural network input space. The detailed compaction method is described below.

Retina Sample Compaction

The main purpose of retina sampling compaction is to fit all retina samples to a small matrix M_u . The size of M_u ($S_u \times S_u$) depends on the number of the generated photoreceptors and can be approximated by $r_s \times r_s$, where s is the number of samples (e.g., if the number of the generated photoreceptors is 50^2, then the size of the matrix is 50×50).

The compaction algorithm is as follows:

1). For each photoreceptor at (x_s, y_s), generate the corresponding uniform coordinates (x_u, y_u) in M_u, where

$$x_u = round[S_u(r_u - 0.5)\cos\alpha],\ y_u = round[S_u(r_u - 0.5)\sin\alpha], \tag{4.3}$$

where α is the zenith angle, and $r_u = CDF(r_s)$. (4.4)

Notice that r_u are the random values that were used to generate r_s and corresponding (x_s, y_s) in the first place, so they are already shown.

After obtaining the uniform coordinates, the photoreceptor at location (x_s, y_s) can be mapped into matrix M_u at location (x_u, y_u). Fig. 4.8 shows an example of a portion of M_u at intersection of rows $i-1$, i, $i+1$ and columns $k-1$, k, $k+1$. In fig. 4.8, black circles represent the photoreceptor mapped to a specific location. Cell A has 4 photoreceptors, and cells A1, A2, A3, A4, A5, A6, A7, A8 are its neighbor, with 2, 1, 1, 3, 0, 3, 0, and 0 photoreceptors respectively.

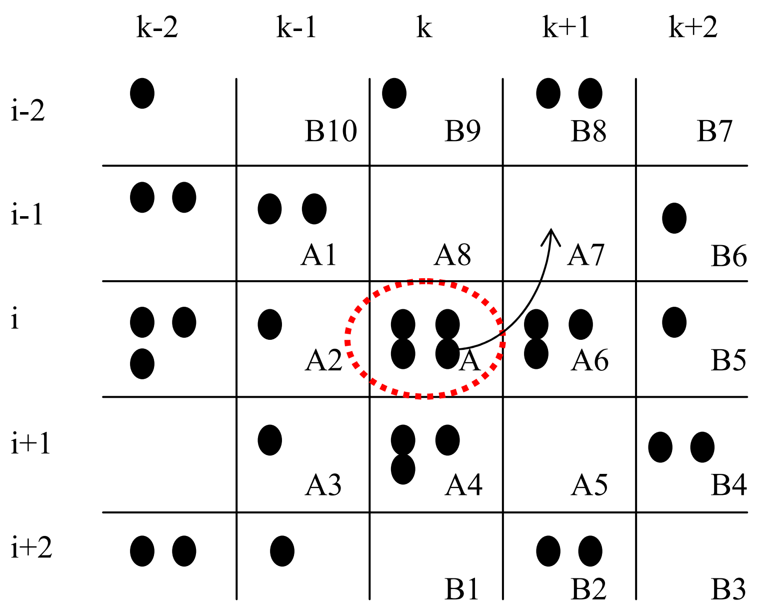


Fig. 4.8 Portion of a M_u

2).Calculate the number of overlaps $Num_{overlap}(x_u, y_u)$ and the demand $Dm(x_u, y_u)$ of each cell. The number of overlaps in a cell at location (x_u, y_u) is

$$Num_{overlap}(x_u, y_u) = Num_{perceptor}(x_u, y_u) - 1, \tag{4.5}$$

where $Num_{perceptor}(x_u, y_u)$ represents the number of photoreceptors in this cell.

For instance, in Fig. 4.8, there are 4 photoreceptors in cell A, thus the number of overlaps of cell A is 3.

Let $Em(x_u, y_u)$ denote the number of empty neighborhood cells around the cell (x_u, y_u). For instance, there are 3 empty neighborhood cells around cell A.

The demand of cell (x_u, y_u) is defined as:

$$Dm(x_u, y_u) = \frac{Num_{overlap}(x_u, y_u)}{Em(x_u, y_u)}, \tag{4.6}$$

where $Dm(x_u, y_u) = 0$ if $Num_{overlap}(x_u, y_u) = 0$ or $Em(x_u, y_u) = 0$.

Thus the demand of cell A is 1. The demand is used to prioritize the order of cells in the retina compaction. Cells with higher demand will be processed first.

3). Let us define the request of an empty cell (x_u, y_u) as the sum of demands of its neighbors.

$$Rq(x_u, y_u) = \sum_{i=1}^{8} Dm(x_{ui}, y_{ui}). \tag{4.7}$$

The request tells compaction algorithm which of the empty neighborhood cells can satisfy a demand with least possibility of nullifying other demands in its neighborhood. The higher the request level, the higher probability of demand nullification will be. A demand can be nullified if all empty neighborhood cells are occupied as a result of satisfying other demands. For example, empty neighbors of cell A have the following requests:

Let us assume that demands of cells B1, B2, B3, …, B10 are $Dm(B1) = 0$, $Dm(B2) = \frac{1}{4}$, $Dm(B3) = 0$, $Dm(B4) = \frac{1}{2}$, $Dm(B5) = 0$, $Dm(B6) = 0$, $Dm(B7) = 0$, $Dm(B8) = \frac{1}{3}$, $Dm(B9) = 0$, $Dm(B10) = 0$, respectively. In addition, $Dm(A1) = \frac{1}{2}$, $Dm(A2) = 0$, $Dm(A4) = 1$, $Dm(A6) = \frac{2}{3}$.

Then $Rq(A5)=\frac{1}{4}+\frac{1}{2}+1+1+\frac{2}{3}=\frac{41}{12}$,

$Rq(A7)=\frac{1}{3}+1+\frac{2}{3}=2$,

$Rq(A8)=\frac{1}{3}+1+\frac{2}{3}+\frac{1}{2}=\frac{5}{2}$.

4).Move one point in the cell which has the highest demand to its empty neighbor which has the lowest request level.

For instance, if cell A has the highest demand, then one of the photoreceptors in cell A will be moved to A7 as it has the minimum request.

After moving a point from A to one of its empty neighbors, we modify its demand, as well as demands of all the neighbors of its empty neighbor that satisfied A's demand (cell A7). In addition, we modify requests of all empty neighbors of A.

5).Repeat 1), 2) and 3) until $Dm=0$, which means no more photoreceptors shall be moved.

Fig. 4.9 shows the results of the compaction algorithm from retina model 1, where the small dot represents 0 overlap in one cell, the circle represents 1 overlap, the star represents 2 overlaps, the triangle represents 3 overlaps, and the square represents 4 overlaps. After the compaction algorithm, there are 274 empty cells filled by sampled points. In this retina sampling model 1, retina_model1.m is used in folder Retina_Sampling (see Appendix A).

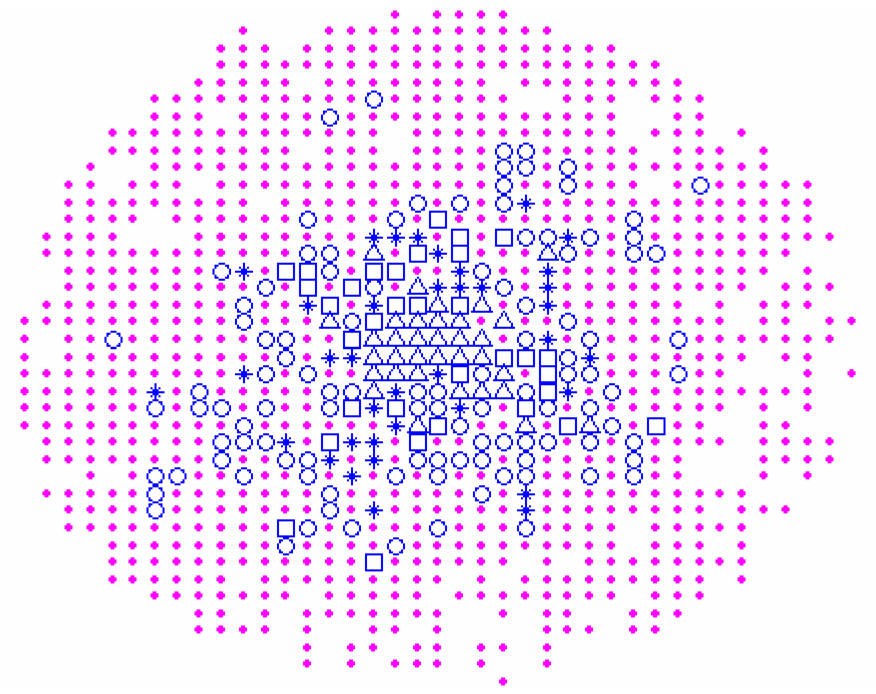


Fig. 4.9 Results of the compaction algorithm from retina model 1

Although the compaction algorithm satisfies all demands (if there are empty cells in neighborhood of cells with nonzero overlap), it leaves some cells with nonzero overlaps. Such cells demand is reduced to zero and effectively only one of its photoreceptors can enter the input data to the neural network. On the other hand, there are still empty cells in these locations where demands were zero. This leaves some inputs to the neural network unused. These deficiencies of the compaction algorithm are addressed in retina models 3 and 4.

4.2.2 Retina Model 2

Retina model 1 is based on theoretical approximation of experimented retina profile. It simplifies the experimentally obtained distribution with a symmetrical 2D density function. In order to improve our model, we developed retina model 2 that uses the experimental distribution shown in Fig. 4.1. The photoreceptors distribution we used is the summation of the rods and cones distribution from Fig. 4.1. Notice that the density of the photoreceptors is not symmetrical and covers larger portion of the visual field in the nasal direction. In this case, we have 2 CDF profiles for the nasal direction and temporal directions respectively as shown in Fig. 4.10.

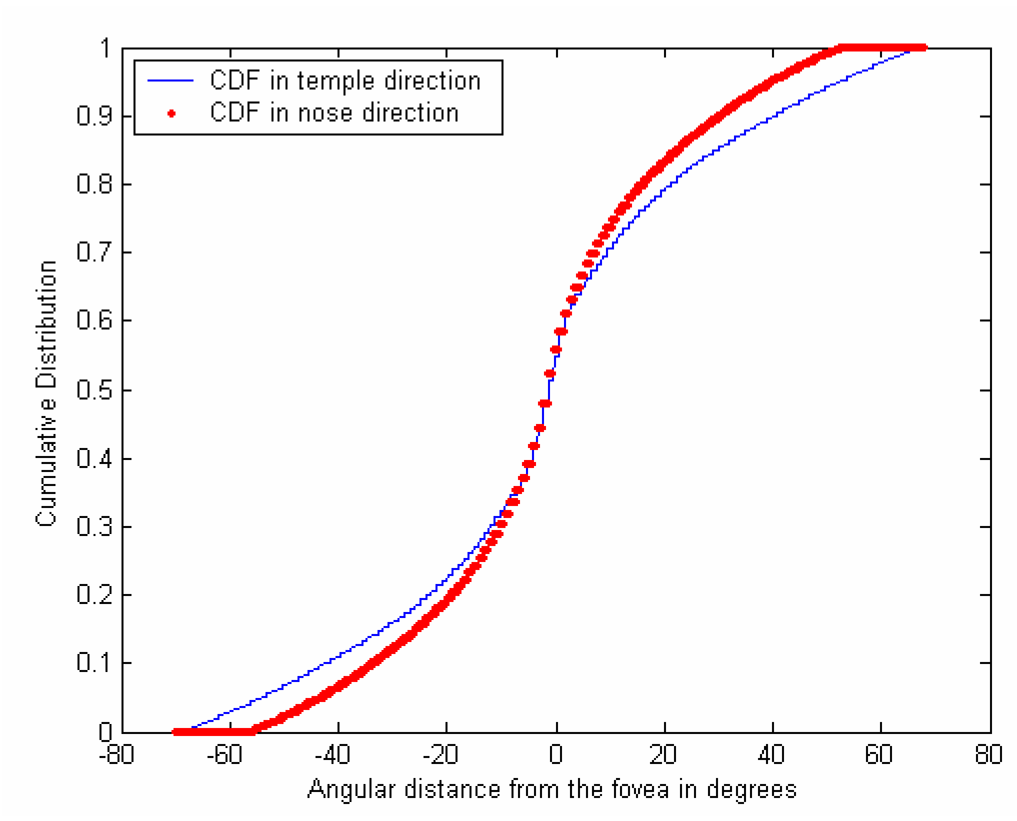


Fig. 4.10 Cumulative density at 0 and 180 degrees

We assume that the experimentally obtained distribution as shown in Fig. 4.1 also represents only one dimensional distribution of cones and rods density. We modified the cumulative probability density in Fig. 4.10 by multiplying it by the absolute value of the distance from the fovea. Fig. 4.11 shows the resulting probability density of the modified model.

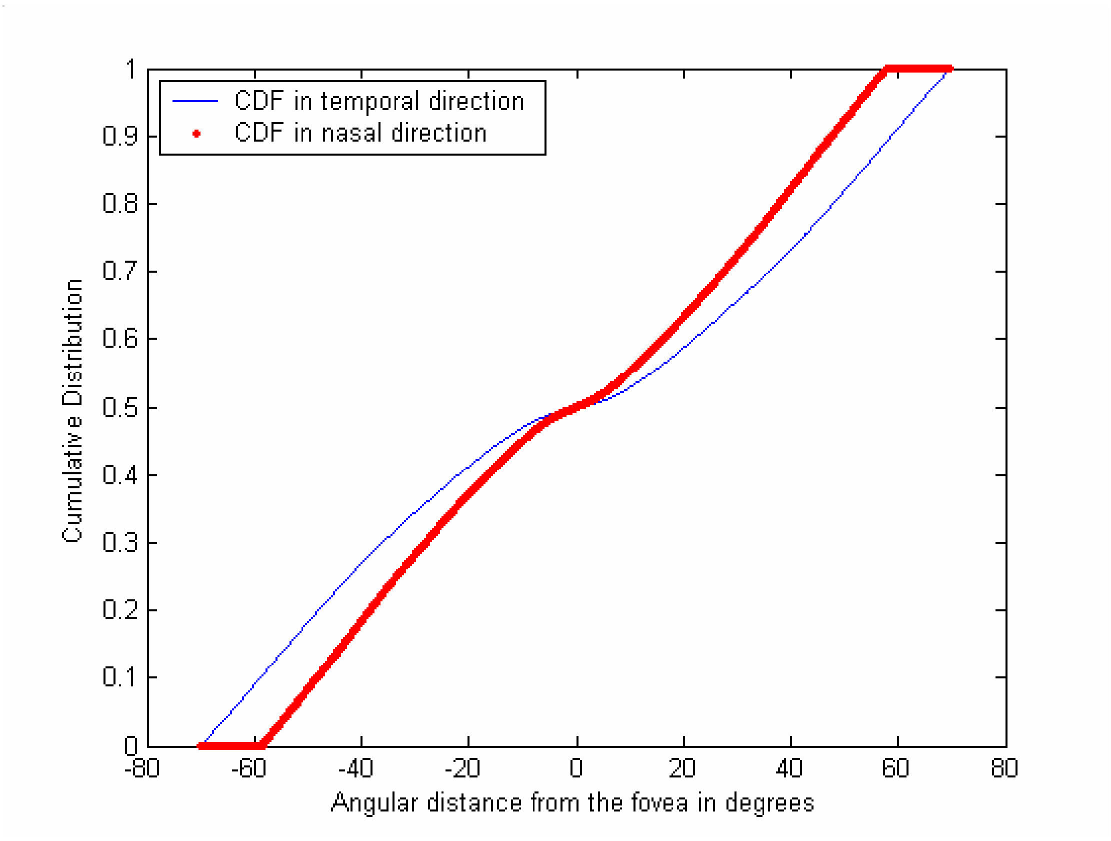


Fig. 4.11 Cumulative density at 0 and 180 degrees

(multiplied by absolute value of the distance from the fovea)

Fig. 4.12 shows an example of generating the retina sampling angular distance r_s from fovea. In general, the way of generating the retina sampling angular distance r_s is similar as described in retina model 1. However, in this model, the CDF profile varies at different directions. α represents the zenith angle of the photoreceptor, where α is between 0 - 2π . Suppose that the temporal direction is at $\alpha = 0$, and the nasal direction is at $\alpha = \pi$, with the CDF profile at nasal direction described by CDF1, and the CDF profile at temple direction described by CDF2. We have:

$$r_s = \left|CDF2^{-1}(r_u)\right| + \frac{\alpha}{2\pi}\left|CDF2^{-1}(r_u) - CDF1^{-1}(r_u)\right| \quad \text{when } 0 \le \alpha \le \pi,$$

$$r_s = \left|CDF2^{-1}(r_u)\right| + \frac{2\pi - \alpha}{2\pi}\left|CDF2^{-1}(r_u) - CDF1^{-1}(r_u)\right| \quad \text{when } \pi \le \alpha \le 2\pi. \tag{4.9}$$

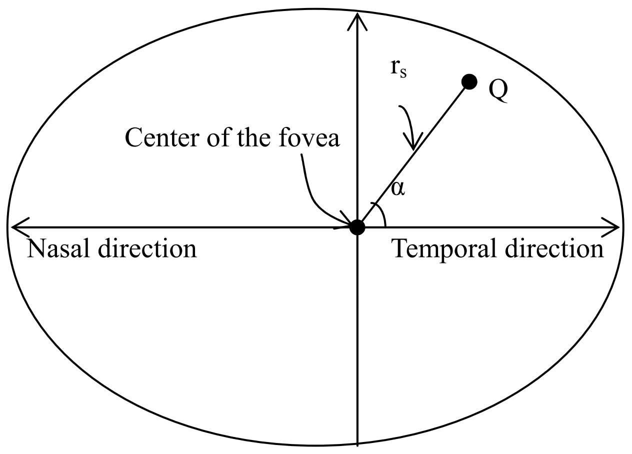


Fig. 4.12 Generation of 2D distribution of photoreceptors (model 2)

Fig. 4.13 shows the corresponding retina sampling by using the CDF shown in Fig. 4.11. Obviously, this is not the correct retina sampling of human retina since the intensity in the fovea is not high enough.

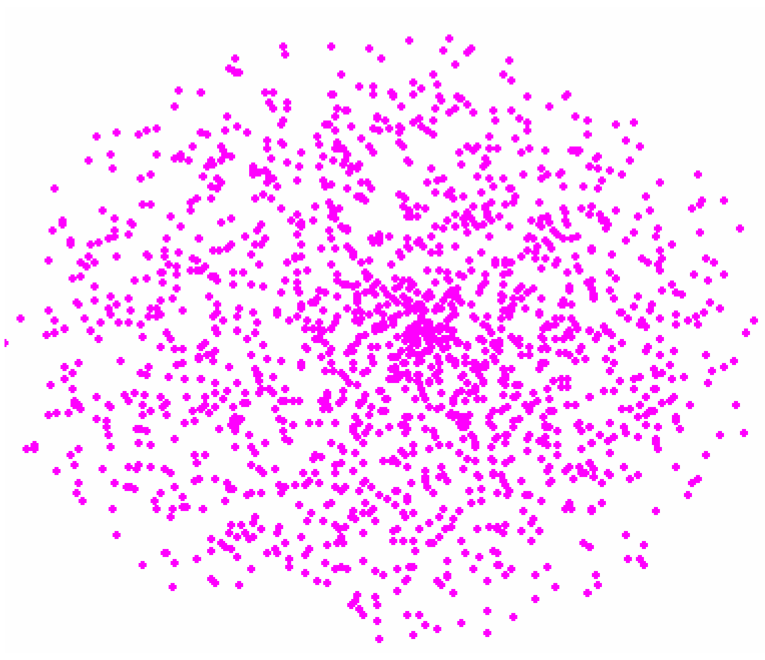


Fig. 4.13 Retina sampling distribution 1(model 2)

Therefore, the experimental experimentally obtained distribution in Fig. 4.1 should be already a linear projection from two-dimensional distribution of cones and rods. The CDF in Fig. 4.10 should be used. Fig. 4.14 shows the retina sampling of model 2 by using the CDF in Fig. 4.10. This model can be easily modified to handle separate distribution of rods and cones if needed in the neural network learning.

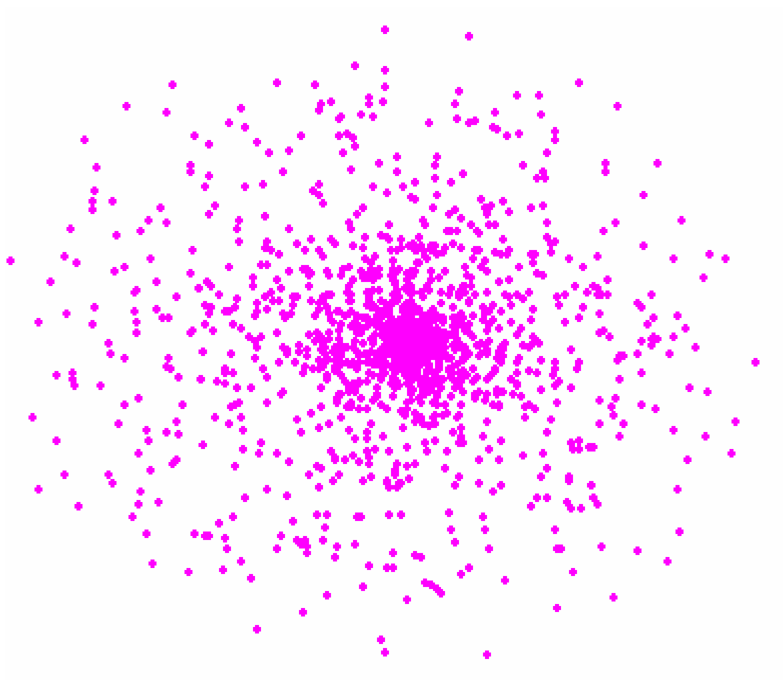


Fig. 4.14 Retina sampling distribution 2(model 2)

After the compaction algorithm, the retina model 2 in Fig. 4.14 represents inputs to the neural network as shown in Fig. 4.15. In this retina sampling model 2, retina_model2.m is used in folder Retina_Sampling (see Appendix A).

Fig. 4.15 shows the results of the compaction algorithm from retina model 1, where the small dot represents 0 overlap in one cell, the circle represents 1 overlap, the star represents 2 overlaps, the triangle represents 3 overlaps, and the square represents 4 overlaps. After the compaction algorithm, there are 249 empty cells filled by sampled points.

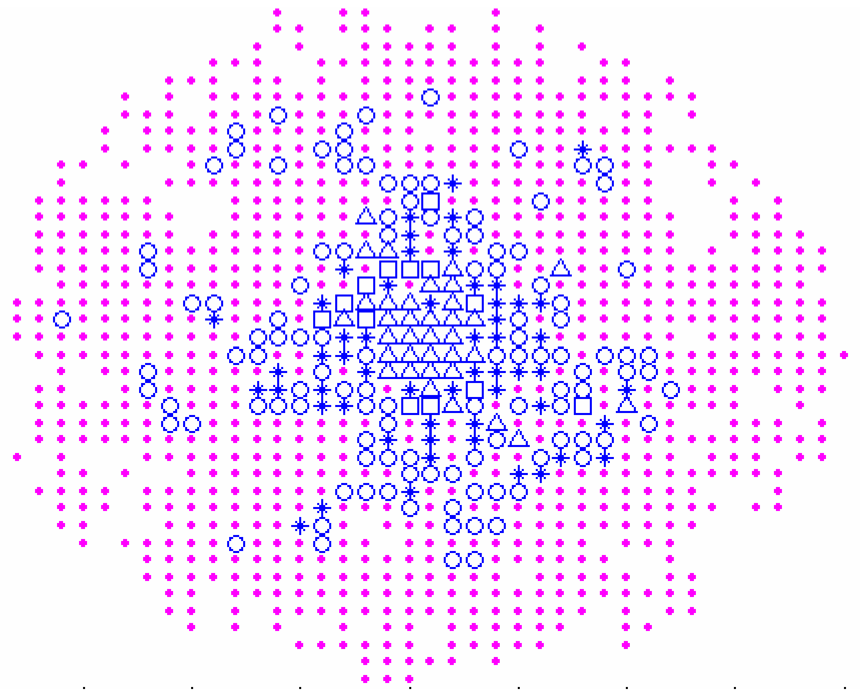


Fig. 4.15 Results of the compaction algorithm from retina model 2

As discussed above, both retina model 1 and 2 require the compaction algorithm to represent the sampled image to the neural network. A more direct approach which does not require the compaction step and guarantee 100% use of NN input space is presented in retina model 3 and 4.

4.2.3 Retina Model 3

In model 3, we first generate the uniform circle shown in Fig. 4.16. Each point in this uniform circle has an angle α ($0 \le \alpha \le 2\pi$) towards the center of the uniform circle. The distance of each point to the center is r_u, where $0 \le r_u \le 1$. By using the inverse of the CDF function generated in retina model 1, we can get the corresponding angular distance from the fovea from equation (4.1): $r_s = \left| CDF^{-1}(r_u) \right|$.

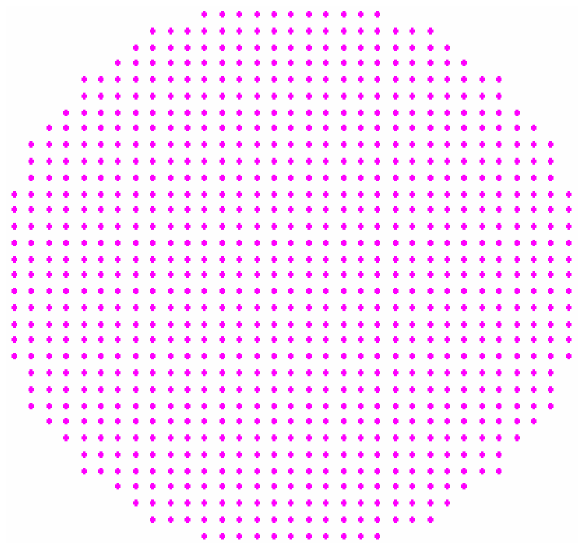


Fig. 4.16 Uniform circle

By locating the zenith angle α and angular distance r_s of each point in the uniform circle, we can get the retina sampling distribution shown in Fig. 4.17, where the uniform circle in Fig. 4.16 is its corresponding compaction matrix sent to the neural network. In this retina sampling model 3, retina_model3.m is used in folder Retina_Sampling (see Appendix A).

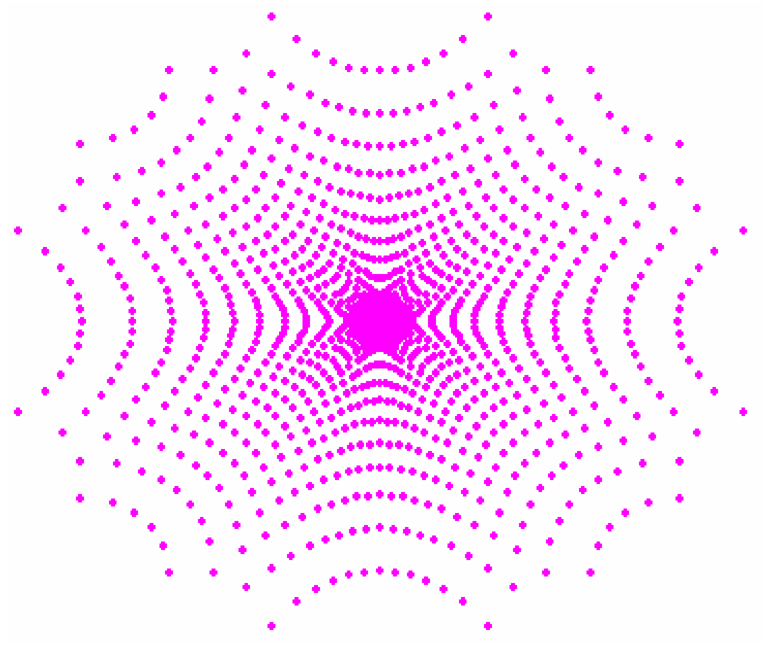


Fig. 4.17 Retina sampling distribution (model 3)

4.2.4 Retina Model 4

Since in Retina Model 2, the experimentally obtained distribution is already a projection, on Retinal Model 4, we need to modify the probability distribution by dividing it by the absolute value of the distance from the fovea. Fig. 4.18 shows the corresponding CDF.

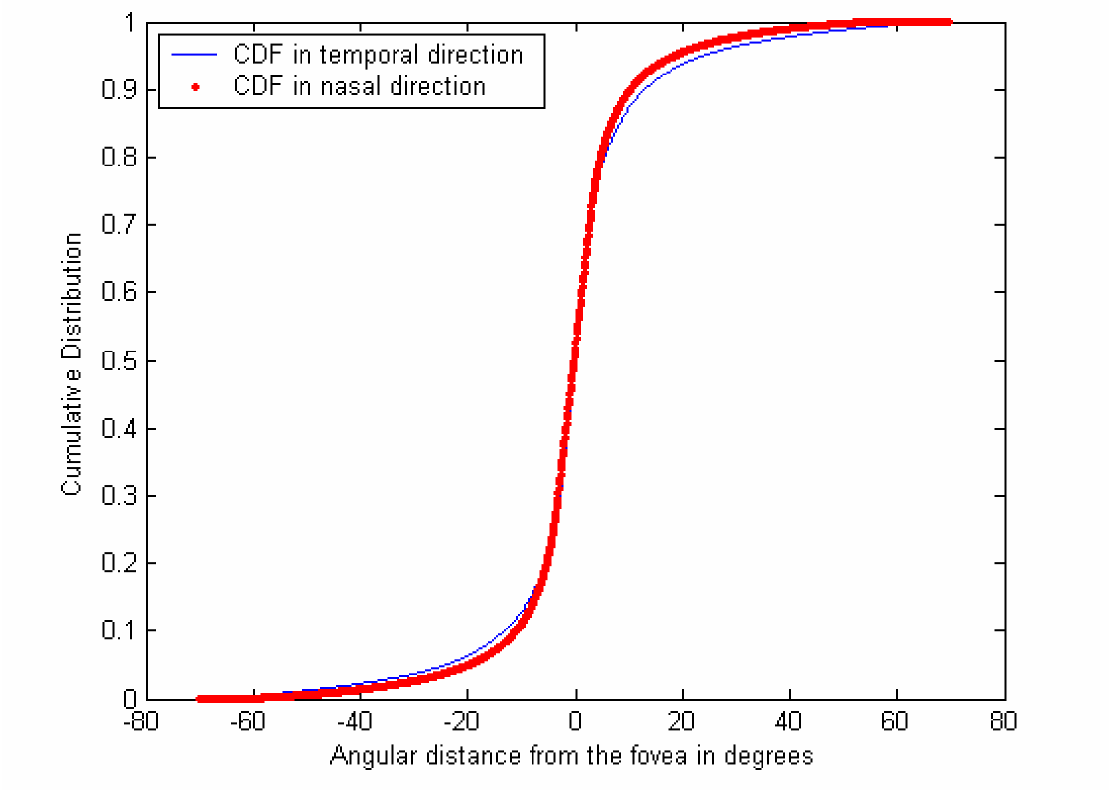


Fig. 4.18 Cumulative density at 0 and 180 degrees

(divided by absolute value of the distance from the fovea)

We map the uniform circle in Fig. 4.16 to the retina distribution by the inverse of the CDF function in Fig. 4.18. As discussed in retina model 2, each point in the uniform circle shown in Fig. 4.16 has an angle α $(0 \le \alpha \le 2\pi)$ towards the center of the uniform circle. The distance of each point to the center is r_u, where $0 \le r_u \le 1$. By using the inverse of two CDF functions generated in retina model 2, we can get the corresponding angular distance from the fovea as in equation (4.8):

$$r_s = \left|CDF2^{-1}(r_u)\right| + \frac{\alpha}{2\pi}\left|CDF2^{-1}(r_u) - CDF1^{-1}(r_u)\right| \qquad \text{when } 0 \le \alpha \le \pi,$$

$$r_s = \left|CDF2^{-1}(r_u)\right| + \frac{2\pi - \alpha}{2\pi}\left|CDF2^{-1}(r_u) - CDF1^{-1}(r_u)\right| \quad \text{when } \pi \le \alpha \le 2\pi.$$

Fig. 40 shows the retina sampling distribution of model 4, and its compaction matrix is the same as shown in Fig. 4.19.

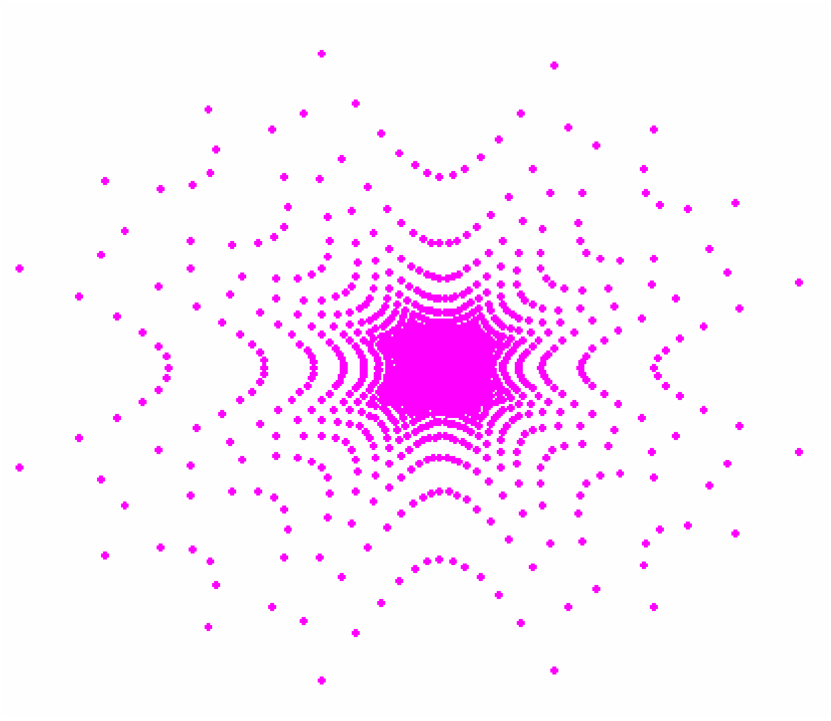


Fig. 4.19 Retina sampling distribution (model 4)

Next we will illustrate an effect of retina image sampling and compare data fed to the neural network by using various retina models. Fig. 4.20 shows a binary image with 1000×1000 pixels.

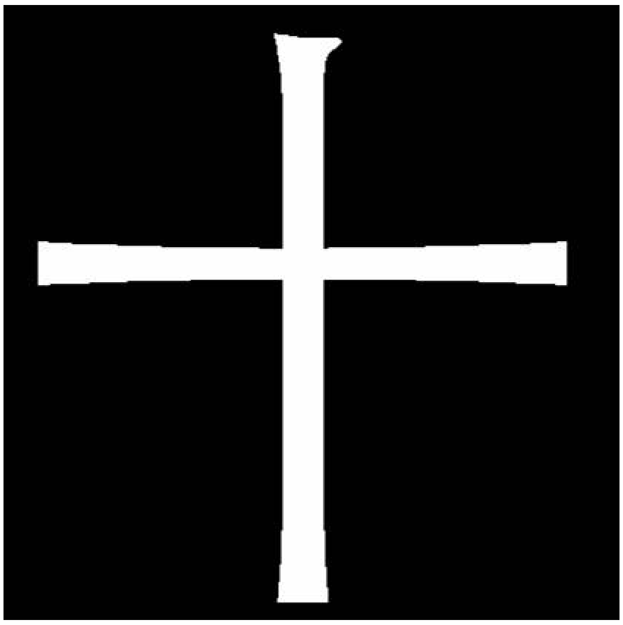
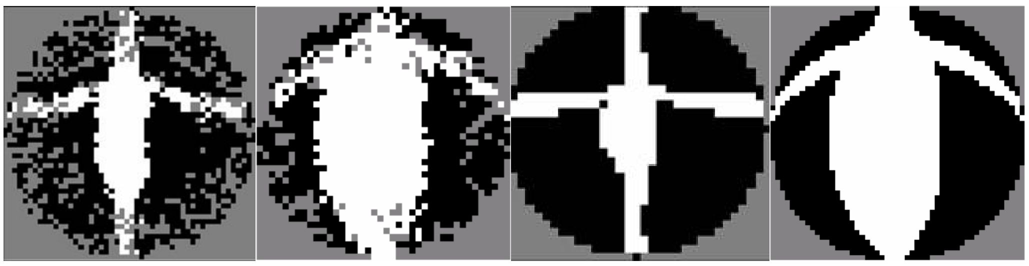


Fig. 4.20 Binary image (1000×1000)

Fig. 4.21 (a)-(d) show the sampled images send to the neural network after compaction by 4 retina sampling models respectively.



(a) Model1 (b) Model2 (c) Model3 (d) Model4

Fig. 4.21 Sampled images by 4 retina sampling models (50×50)

Obviously, retina model 4 achieves the best quality of the sampled images. From now on we will refer to this model as experimental uniform retina (EURETA) model. In this EURETA model (retina sampling model 4), retina_model4.m is used in folder Retina_Sampling (see

Appendix A).

Another example to illustrate the retina sampling presented in Fig. 4.22, shows a color image with 1000×1000 pixels.

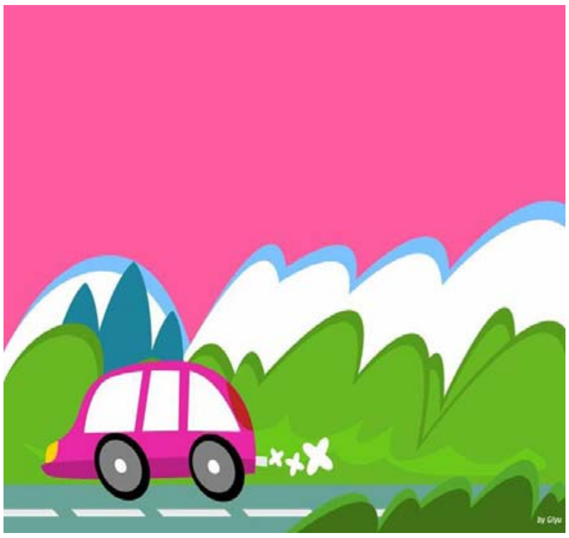


Fig. 4.22 Colorful image (1000×1000)

As shown in Fig. 4.2, d is the vertical distance from the object to the eye, and retina_d is the maximum vision range of the retina, where

$$retina\_d = d \cdot [\tan(70^o) + \tan(55^o)]. \tag{4.9}$$

By increasing d we can zoom out of the observed image sampling larger area and opposite, by decreasing d we can zoom in. In this way, EURETA can focus on whatever object or its feature that is interesting for the active vision system.

Fig. 4.23 (a), (b) and (c) show the sampled images with 41×41 pixels resolution from the original image (1000×1000 pixels) at different fovea locations with retina_d1=1000 pixels, retina_d2=1000 pixels, and retina_d3=680 pixels respectively.

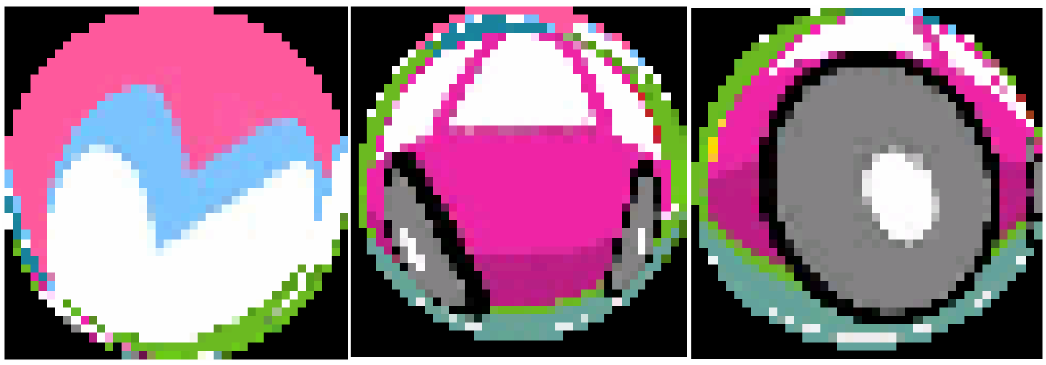

(a) (retina_d1) (b) (retina_d2) (c) (retina_d3)

Fig. 4.23 Sampled images by EURETA （50×50）

4.3 Retina Sampling with Ganglion Cells

4.3.1 Introduction

Ganglion cells are neurons located in the retina of the eye which receive visual information from photoreceptors via various intermediate cells such as bipolar cells, amacrine cells, and horizontal cells [Wikip]. Ganglion cells are the output neurons of the retina. Different from photoreceptors, which respond to light with graded changes in membrane potential, ganglion cells transmit information as trains of action potentials [Tes91]. Each ganglion cell responds to light directed to a specific area of the retina called receptive area. Ganglion cell receptive fields include the receptive center and the antagonistic surround. There are two basic kinds of ganglion cells: on-center cells and off-center cells as shown in Fig. 4.24. On-center cells are excited when stimulated in the center and inhibited when stimulated in the surround. Off-center cells are excited when stimulated in the surround while inhibited when stimulated in the center. In the figure, "+" stands for regions giving on responses, "-" for regions giving off responses [Tes91].

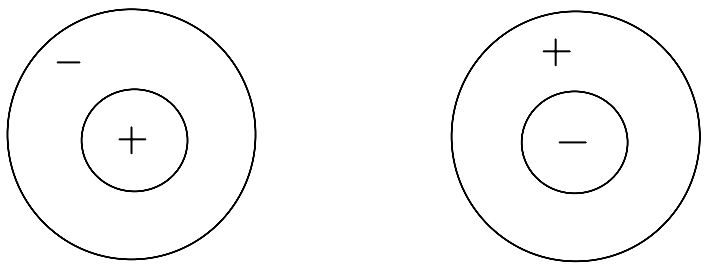


Fig. 4.24 On-center cell and off-center cell

These two classes of ganglion cells can be distinguished by their response to a small spot of light into the center of their receptive field. On-center ganglion cells fire when the light is directed applied to the center of their receptive fields and the light applied to the surround inhibits the on-center ganglion cells from firing. In an opposite way, off-center ganglion cells fire when the light is directed applied to the surround of their receptive field and inhibit when the light is applied to the center [Tes91].

Furthermore, ganglion cells that are found in humans and higher mammals seem to provide information to the visual system that is used to construct a picture of the visual world.

Ganglion cells seem to act as "feature detectors", providing "building blocks" from which the visual system can build up a picture of the world and the objects within it [Every].

In our biologically based active vision, we intend to model the ganglion cells in human retina to act as feature detectors. When combined with the retina sampling, our model can extract any interesting features from any high-resolution image and therefore achieve recognition that is more accurate.

4.3.2 Direct Ganglion Cells Implementation

In this section, we will discuss the use of ganglion cells in the retina sampling model to obtain feature extraction and achieve more accurate recognition. For simplicity, we will use two kinds of ganglion cells, one is spot detection cells, and the other is edge detection cells.

4.3.2.1 Spot Detection Cells

The spot detection positive (P) and negative (N) cells are shown in Fig. 4.25. Thus P cell corresponds to on-center cell and N cell corresponds to off-center cell. P cell has the excitatory inputs in the center represented by “+” and inhibitory inputs represented by “-” in the surround. N cell is opposite.

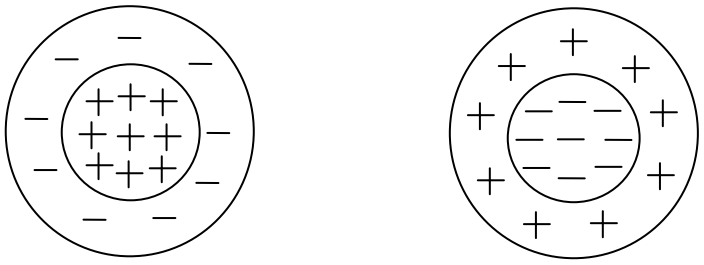


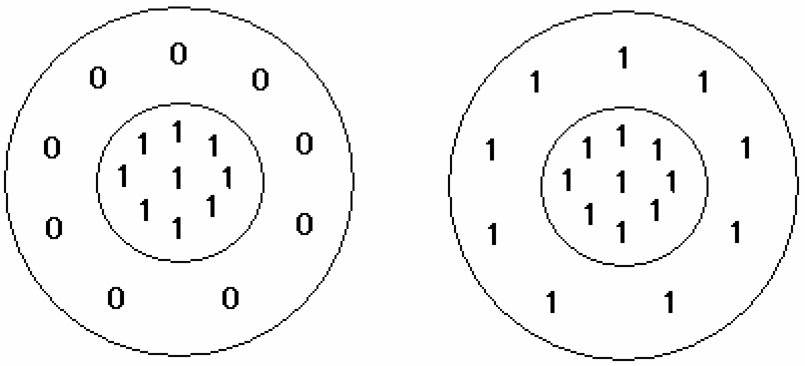
Fig. 4.25 P cell and N cell

The output of a spot detection cell is calculated by:

$$O1 = \sum_{i=1}^{n} S_i W_i \ . \tag{4.10}$$

where S_i is the signal value ranged from in 0 to 1, W_i is the weight, while the weight of the excitatory input is 1, and the weight of the inhibitory input is -1.

Fig. 4.26 shows an example of using the spot detection cell to detect the spot. For simplicity, we scale the image from 0-255 to 0-1. In fig. 4.26, “1” represents white pixel and “0” represents black pixel.



(a) (b)

Fig. 4.26 Example of spot detection

For a white spot image shown in Fig. 4.26(a), if we use the P cell as shown in Fig. 4.25, the output of the P cell is $O1 = \sum_{i=1}^{n} S_i W_i = 9$. If we use the same P cell for (b), we will get the output $O1 = \sum_{i=1}^{n} S_i W_i = 0$. Thus the white spot can be detected by P cell. By using similar analysis, we can demonstrate that a black spot can be detected by N cell.

P and N cells overlap each other exactly. In addition, they cover the entire visual field in a uniform overlapping pattern as shown in Fig. 4.27.

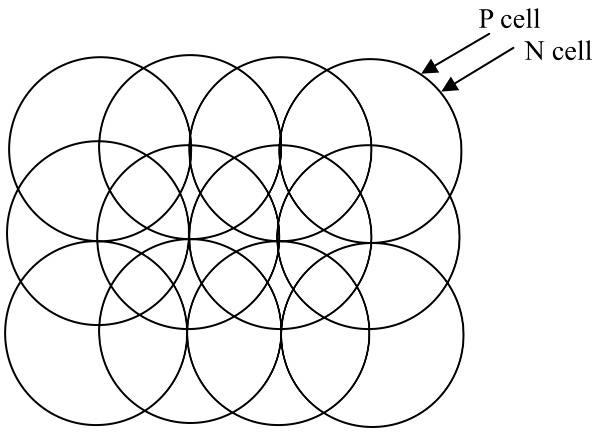


Fig. 4.27 Uniform overlapping pattern

Let us consider a 964×964 image "Lena" as shown in Fig. 4.28.

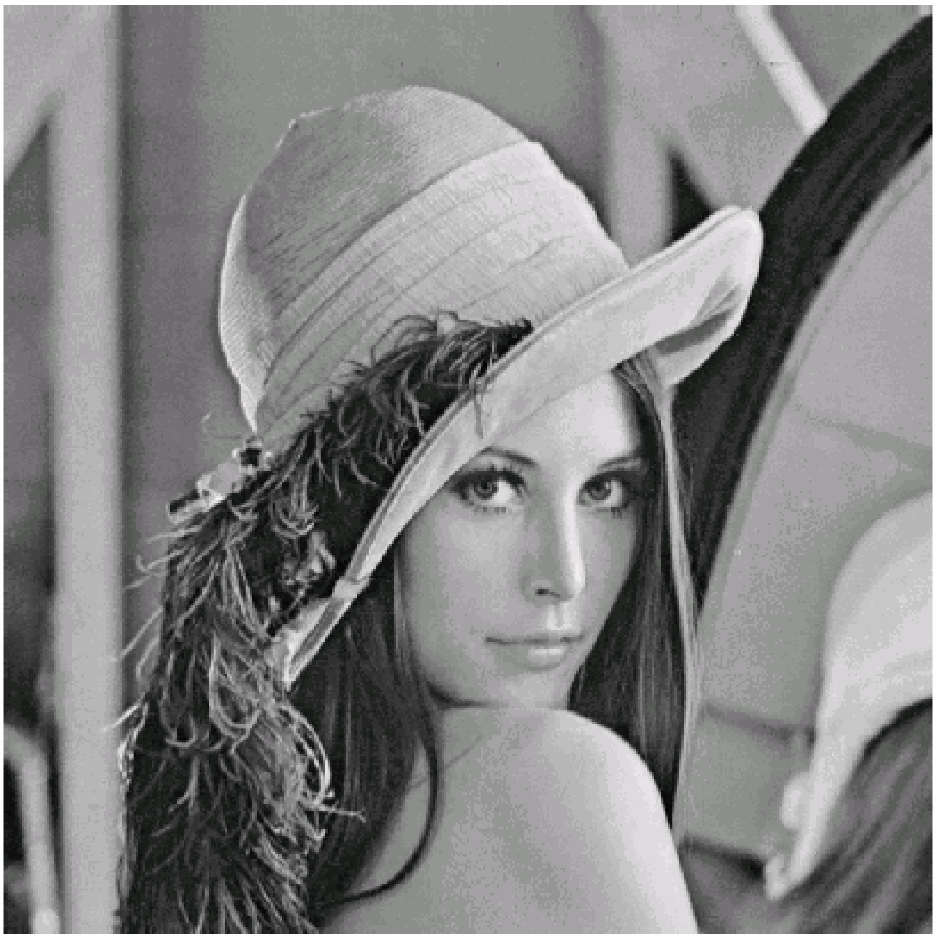


Fig. 4.28 Lena (964×964)

For simplicity, the following 8×8 P and N spot detection cells are used.

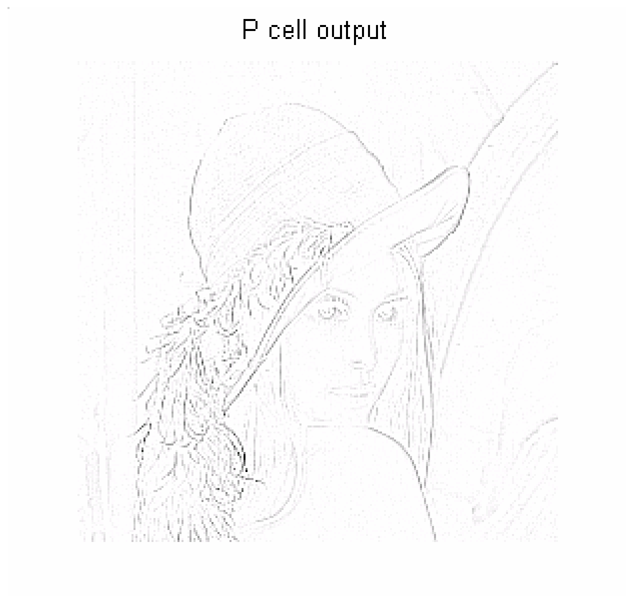
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	1	1	-1	-1
-1	1	1	1	1	1	1	-1
-1	1	1	1	1	1	1	-1
-1	1	1	1	1	1	1	-1
-1	1	1	1	1	1	1	-1
-1	-1	1	1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

1	1	1	1	1	1	1	1
1	1	-1	-1	-1	-1	1	1
1	-1	-1	-1	-1	-1	-1	1
1	-1	-1	-1	-1	-1	-1	1
1	-1	-1	-1	-1	-1	-1	1
1	-1	-1	-1	-1	-1	-1	1
1	1	-1	-1	-1	-1	1	1
1	1	1	1	1	1	1	1

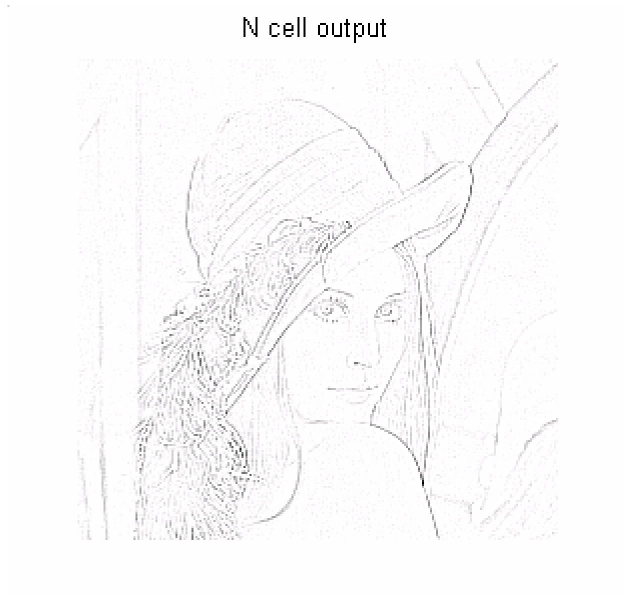
Fig. 4.29 Weight matrices of P cell and N cell

“-1” represents the weight for the inhibitory input and “+1” represents the weight for the excitatory input. The number of inhibitory and excitatory inputs is equal. These P and N cells overlap each other exactly. They cover the entire image “Lena” in Fig. 4.28 in a uniform overlapping pattern. As a result, the EURETA system needs 240×240 P cells and 240×240 N cells to cover the entire Lena image that has 964×964 pixels.

Fig. 4.30 shows the output of the P and N spot detection cells.



(a)



(b)

Fig. 4.30 P cell and N cell outputs (240×240)

Since the whole image is covered by those uniformly overlapping spot detection cells, the outputs of the overlapping spot detection cells actually show all the white and black edges of the image as illustrated in Fig. 4.30. In this spot detection, Main_One_Level_Ganglion.m is used in folder Ganglion Cells_8_8_new (see Appendix A).

4.3.2.2 Edge Detection Cells

Besides spot detection cells, we also have other feature detection cells like edge detection cells. Fig. 4.31 shows some examples of the weight matrices of positive feature detection cells, where (a) is the left edge detection cell, (b) is the upper edge detection cell, (c) is the rising diagonal edge detection cell. They still have the same size (8×8) as the spot detection cells.

1	1	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1

(a)

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

(b)

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	-1
1	1	1	1	1	1	-1	-1
1	1	1	1	1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1
1	1	1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1

(c)

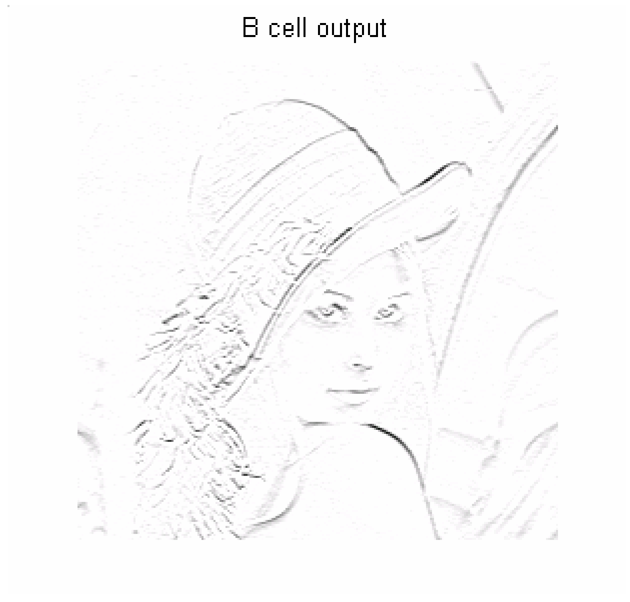
Fig. 4.31 Weight matrices of edge detection cells

Similarly, the output of a feature detection cell is calculated by (4.10):

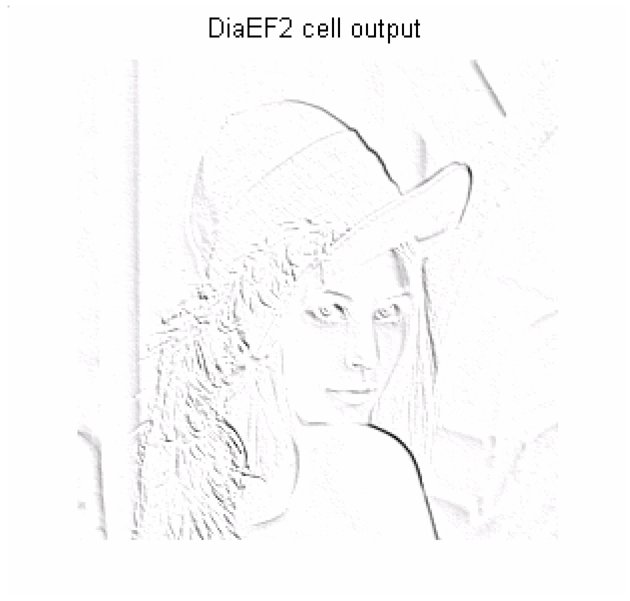
$$O1=\sum_{i=1}^{n} S_i W_i \,,$$

where "0" means it is not connected to the corresponding image input.

Fig. 4.32 shows the outputs of the edge detection cells, where (a) is the outputs of the bottom edge detection cells and (b) is the diagonal falling edge detection cells.



(a)



(b)

Fig. 4.32 Outputs of the edge detection cells (240×240)

Since the outputs of the detection cells are generated based on the original image Lena, now we consider mapping the outputs of the feature detection cells (240×240) back to the original image(964×964).

For instance, let us map the outputs of the bottom edge feature detection cells to the original image (964×964). The weight matrix of the bottom edge feature detection cell is shown in Fig. 4.33 (a). If the output of one bottom edge feature detection cell is positive, the signal values in the corresponding 8×8 feature detection cell in the 240×240 matrix will be obtained by:

$$Sig = O1_i \times FeatureMatrix\,. \qquad (4.11)$$

where $FeatureMatrix$ is the corresponding feature matrix. Feature matrix shows corresponding pixel intensity that matches given feature detection cell. For the bottom edge feature detection cell, its $FeatureMatrix$ is shown in Fig. 4.33 (b).

-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

(a)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

(b)

Fig. 4.33 Weight matrix and its corresponding feature

Fig. 4.34 shows the corresponding mapping of the bottom edge feature detection cells (964×964).

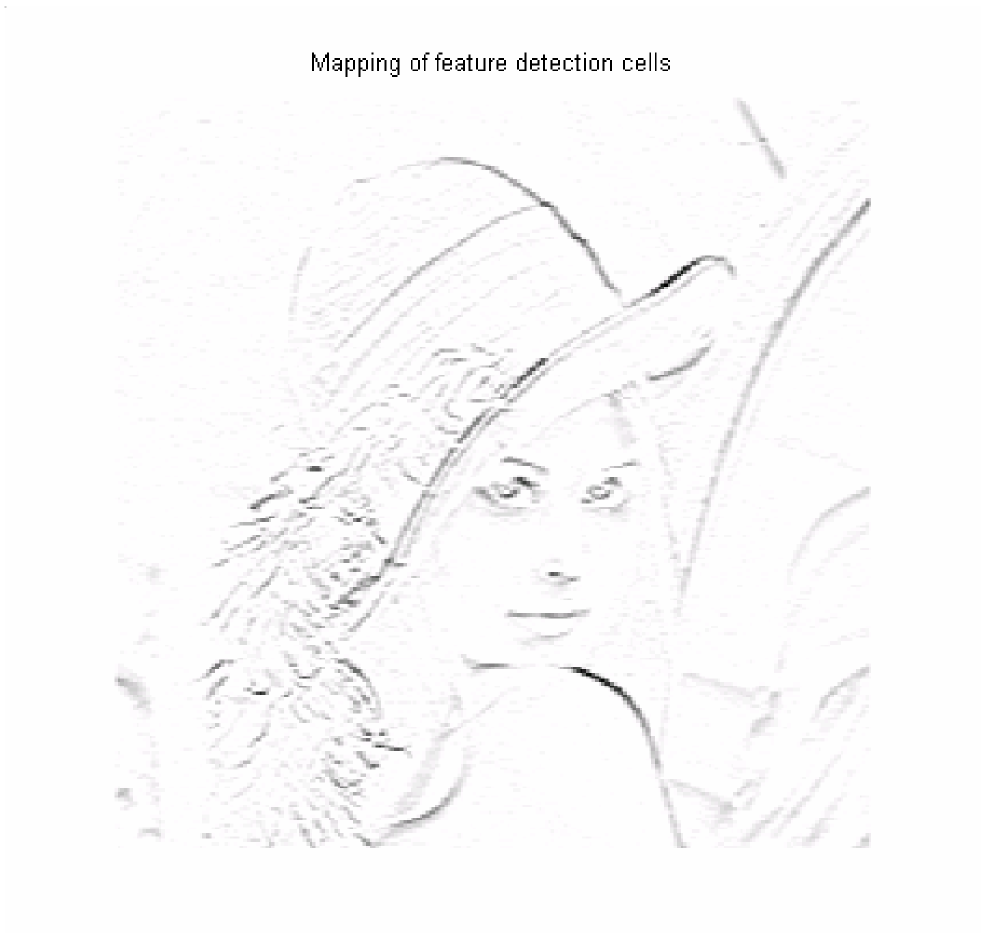


Fig. 4.34 Mapping of the bottom edge feature detection cells (964×964)

In a similar way, we can get the mapping of the negative spot detection cells as shown in Fig. 4.35.

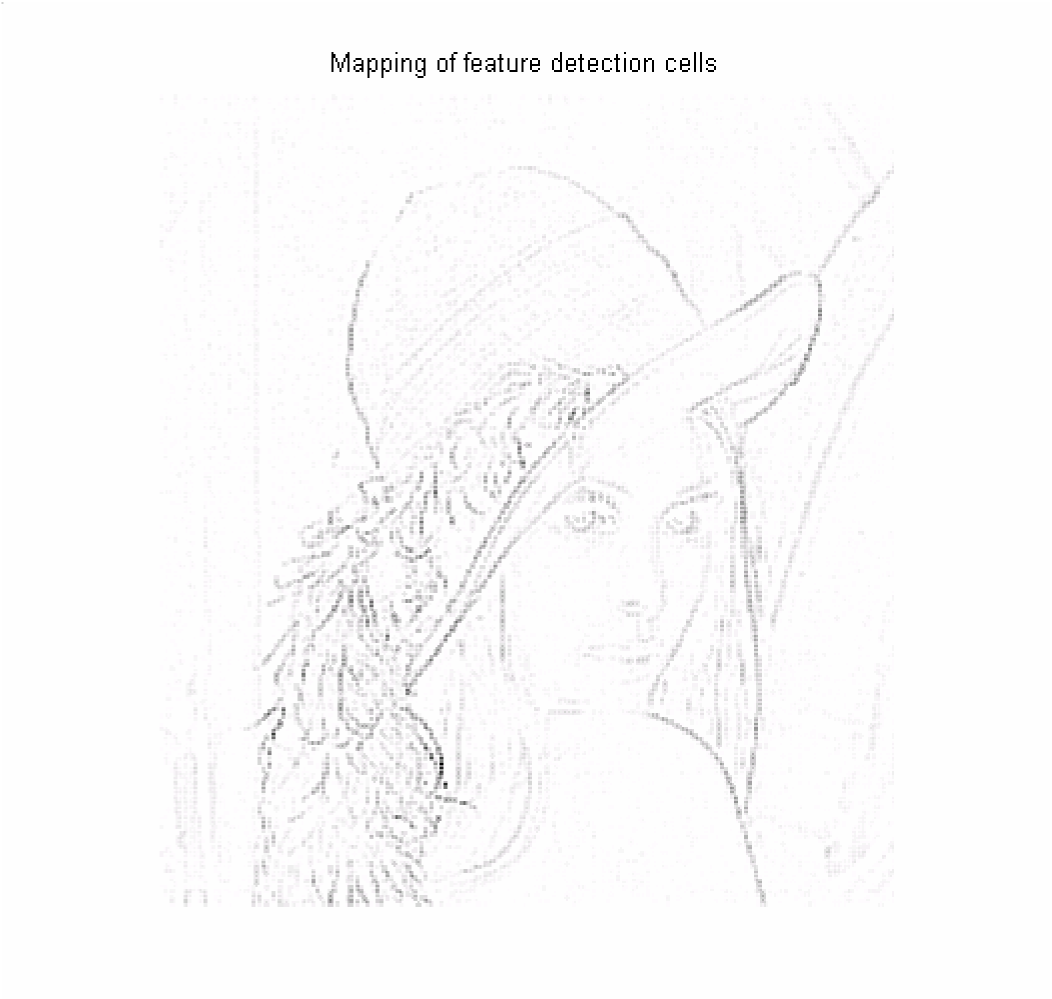


Fig. 4.35 Mapping of the negative spot detection cells (964×964)

Since we have different kinds of spot and edge detection cells, we can finally get the average mapping of all of these different features as shown in Fig. 4.36.

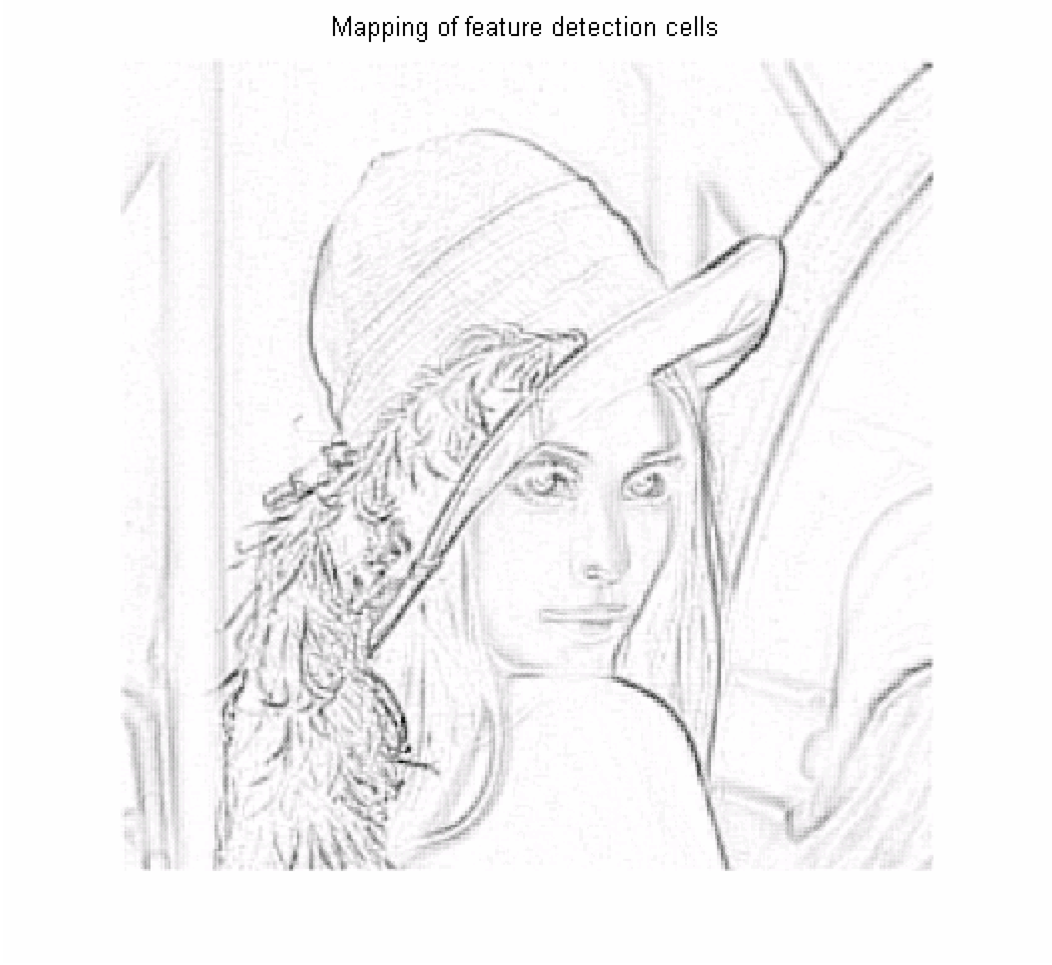


Fig. 4.36 Average mapping of different features (964×964)

In this edge detection, Main_One_Level_Ganglion.m is used in folder Ganglion Cells_8_8_new (see Appendix A).

4.3.2.3 Average Intensity Cells

The average intensity cells average their input and fire in proportion to this average. They are smaller than the spot detection cells, using roughly half of the spot detection cells inputs.

Fig. 4.37 shows the weight matrix of the average intensity cell, it only uses half of the spot detection cells inputs as shown in Fig. 4.29.

0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0

Fig. 4.37 Weight matrix of average intensity cell

The output of the average intensity cell is obtained by:

$$O1 = \frac{\sum_{i=1}^{n} S_i W_i}{m}, \tag{4.12}$$

where m is the number of the inputs with weighs are equal 1. In Fig. 4.37, $m = \frac{n}{2} = 32$.

Fig. 4.38 shows the outputs of the average intensity cells with 240×240 pixels.

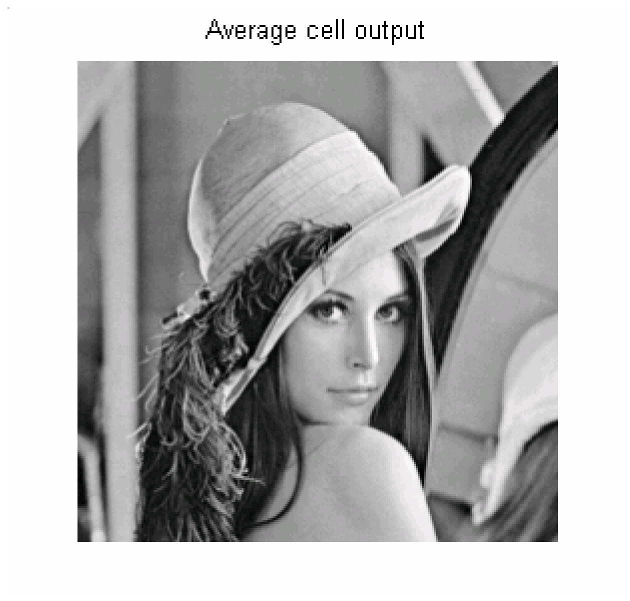


Fig. 4.38 Outputs of the average intensity cells (240×240)

Fig. 4.39 shows the mapping of the average intensity cells.

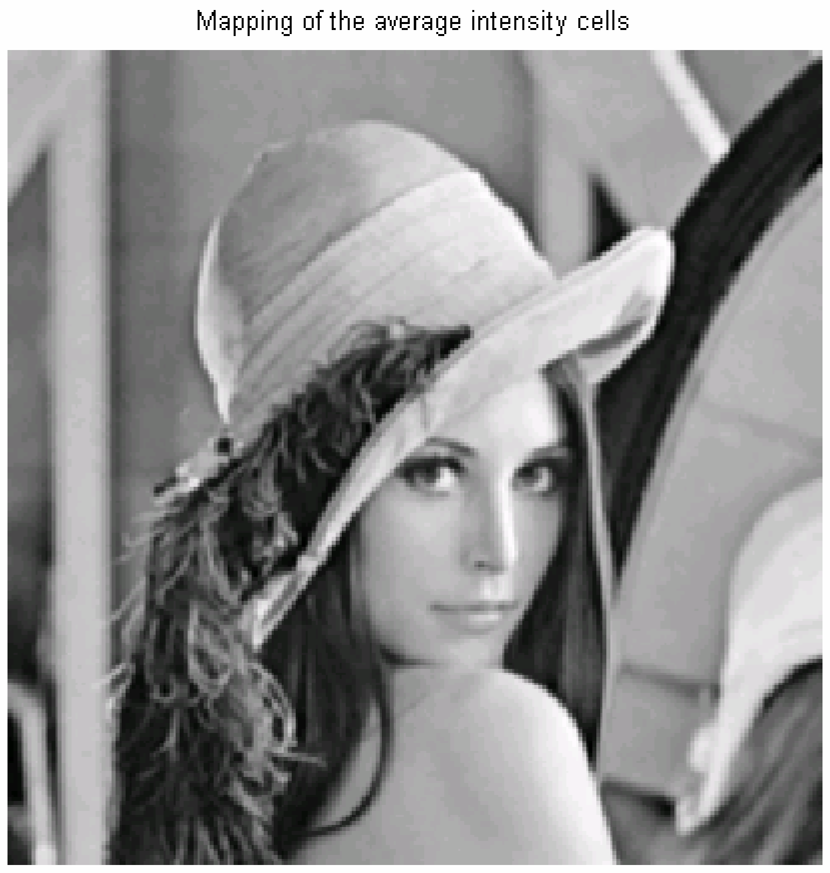


Fig. 4.39 Mapping of the average intensity cells (964×964)

Now we could average the mapping of average intensity cells (in Fig. 4.39), and the mapping of different features (in Fig. 4.36), as shown in Fig. 4.40. In Fig. 4.40, the edges are more distinct. In this average intensity cells implementation, Main_One_Level_Ganglion.m is used in folder Ganglion Cells_8_8_new (see Appendix A).

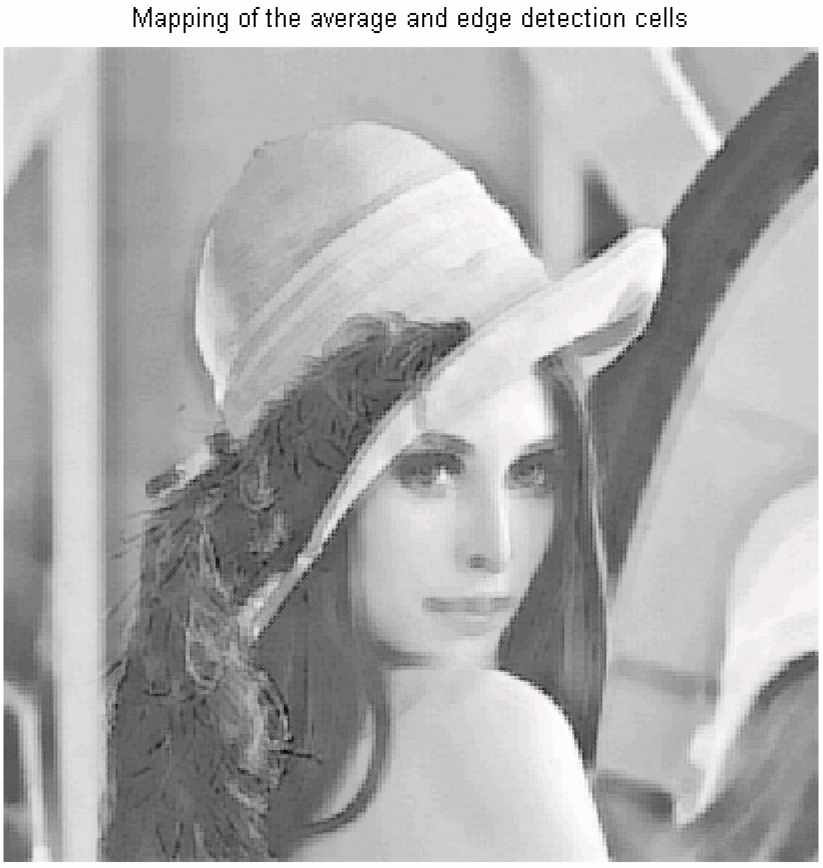


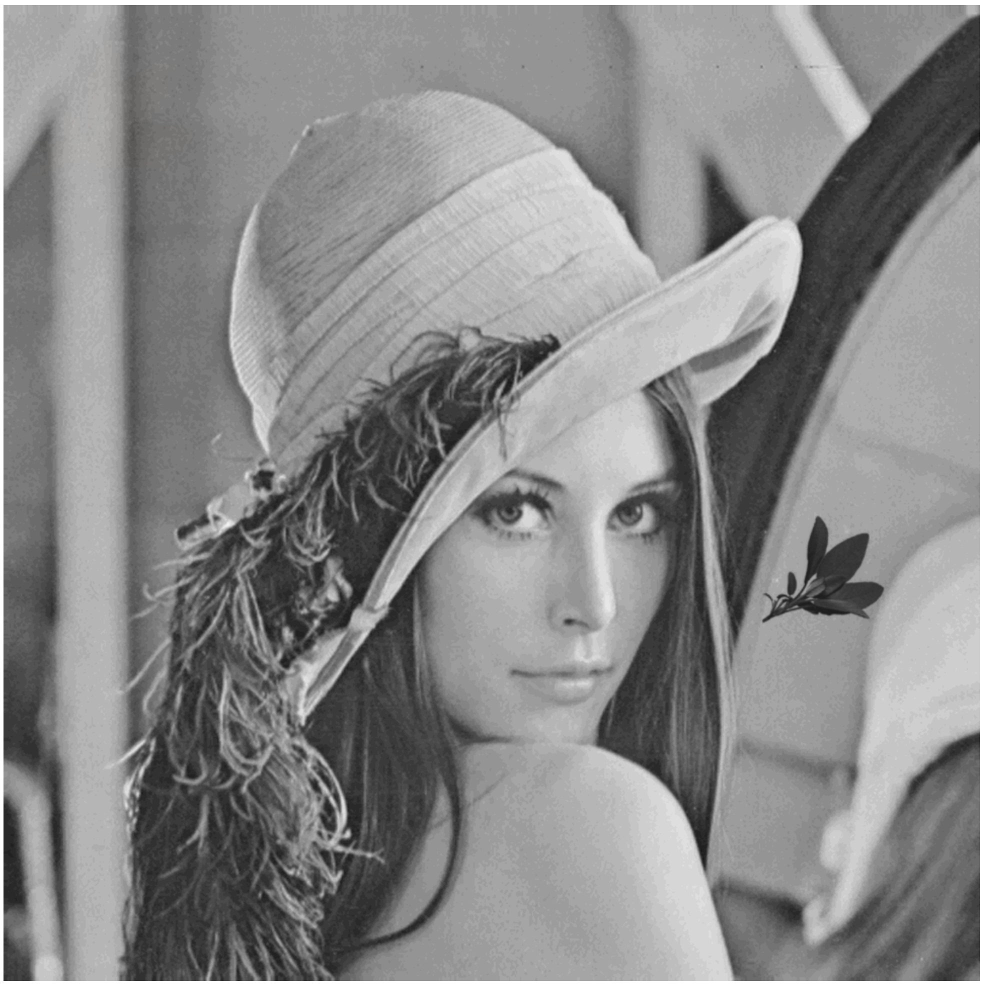
Fig. 4.40 Average of average cells mapping and edge detection mapping (964×964)

4.3.2.4 Motion Detection Cells

An important function of ganglion cells is motion detection. Motion can be detected by using dual neurons for all ganglion cells. To reduce the hardware effort, only the average intensity ganglion cells will be used. Fig. 4.41 shows the different frames at time t=T0, T0+1 and T0+2.

(a) Frame at time t=T0 (964×964)

(b) Frame at time t=T0+δ (964×964)



(c) Frame at time t=T0+2δ (964×964)

Fig. 4.41 Frames at time t=T0, T0+δ, T0+2δ

Dual cell B copies the average cell output and holds it for a time period δ.

For instance, at time t=T0+δ, the output of the dual cell B is:

$$B_{out}(T_0+\delta) = A_{out}(T_0)\,. \tag{4.13}$$

The motion detection cell MD fires in proportion to the absolute value of $B_{out}(T_0+\delta)$ and $A_{out}(T_0+\delta)$, that is:

$$MD = \left|B_{out}(T_0+\delta) - A_{out}(T_0+\delta)\right| = \left|A_{out}(T_0) - A_{out}(T_0+\delta)\right|. \tag{4.14}$$

Fig. 4.42 shows the result of motion detection cells at time t=T0+δ. In this motion detection, Main_One_Level_Ganglion_Motion_Detection.m is used in folder Ganglion Cells_8_8_new (see Appendix A).

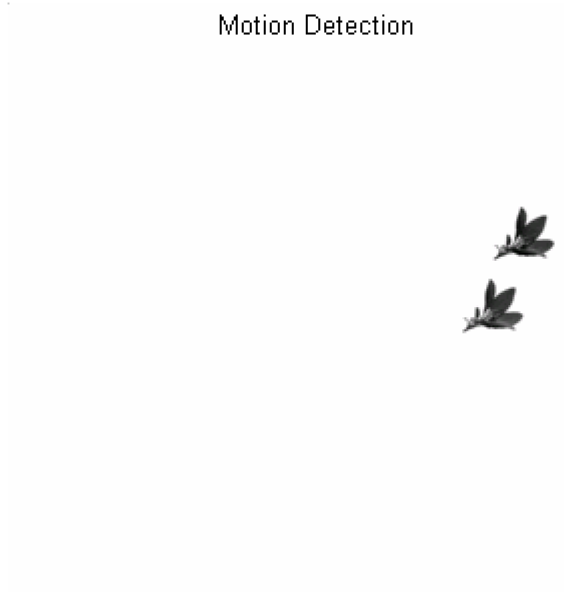


Fig. 4.42 Result of motion detection cells at time t=T0+δ (240×240)

Similarly, Fig. 4.43 shows the result of the motion detection cells at time t=T0+2δ.

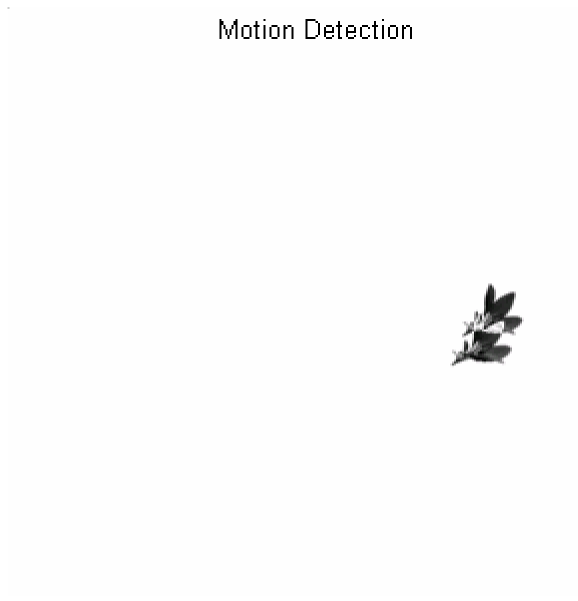


Fig. 4.43 Result of motion detection cells at time t=T0+2δ (240×240)

4.3.3 Ganglion Cells Implementation with Retina Sampling

In the previous section, we discussed about using the direct ganglion cells implementation to extract features. Now we will combine this ganglion cells implementation with the retina sampling. In the following ganglion cells implementation with retina sampling, Main_One_Level_Ganglion_With_Retina_Sampling.m is used in folder Ganglion Cells_8_8_new (see Appendix A).

Fig. 4.44 shows a sampled image with pixels 221×221 from the original image "Lena".

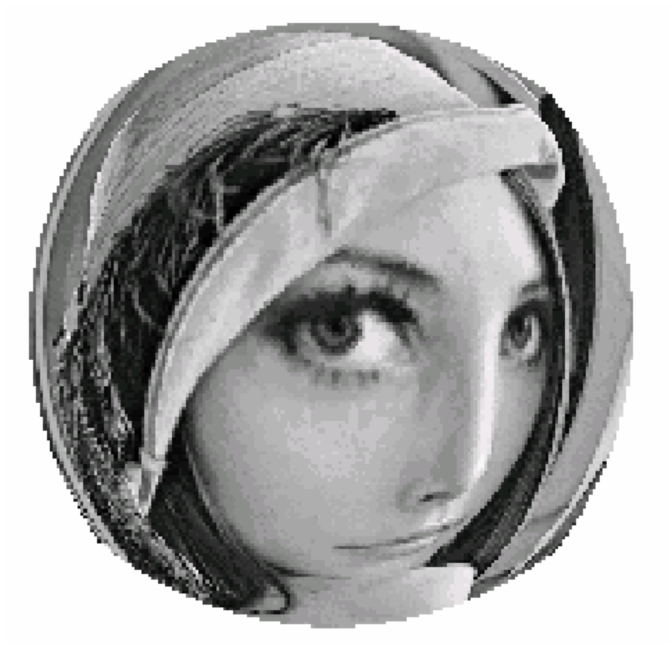


Fig. 4.44 Sampled image (221×221)

This sampled image will be first covered by those overlapping spot detection cells. The outputs of the overlapping P and N spot detection cells are shown in Fig. 4.45

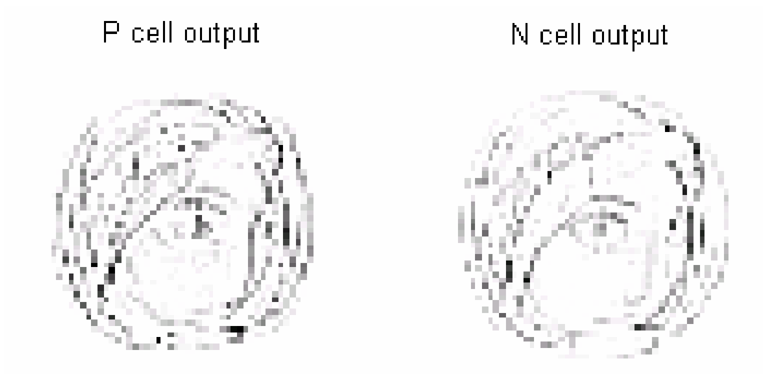
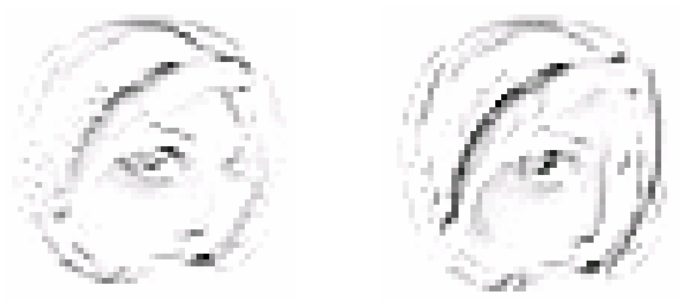


Fig. 4.45 P cell and N cell outputs (54×54)

Fig. 4.46 (a) and (b) show the outputs of the bottom edge and diagonal rising edge detection cells respectively.



(a) (b)

Fig. 4.46 Outputs of the edge detection cells (54×54)

The average mapping of these different spot and edge detection cells is shown in Fig. 4.47.

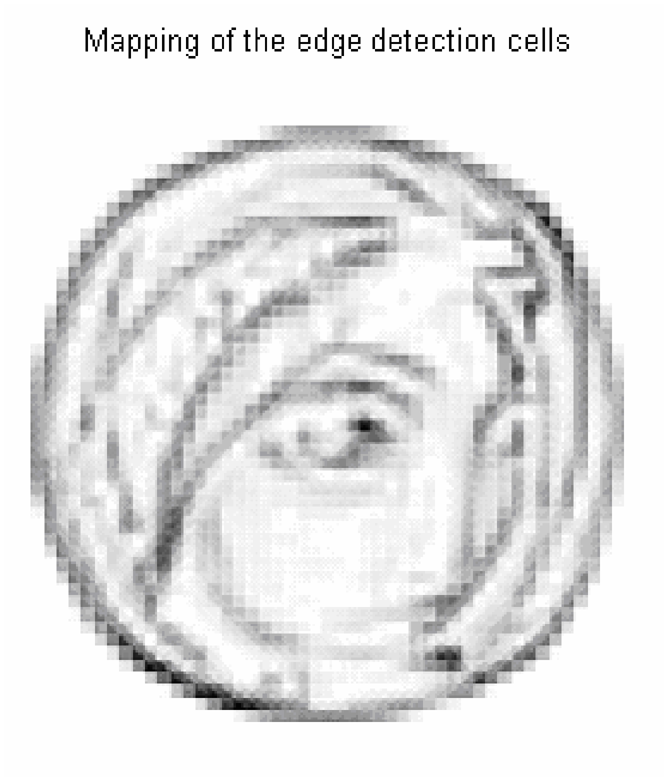


Fig. 4.47 Average mapping of different features (221×221)

Fig. 4.48 shows the mapping of the average intensity cells.

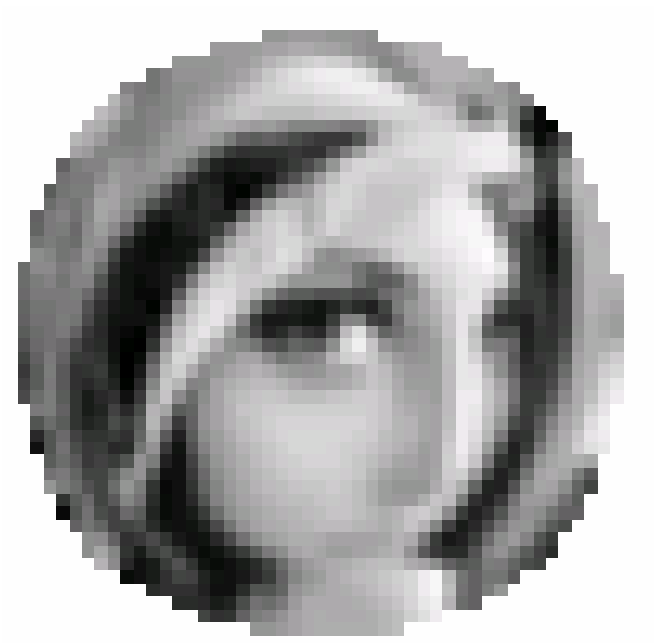


Fig. 4.48 Mapping of the average intensity cells (221×221)

Similarly, we could average the mapping of average intensity cells (in Fig. 4.48), and the mapping of different features (in Fig. 4.47), as shown in Fig. 4.49.

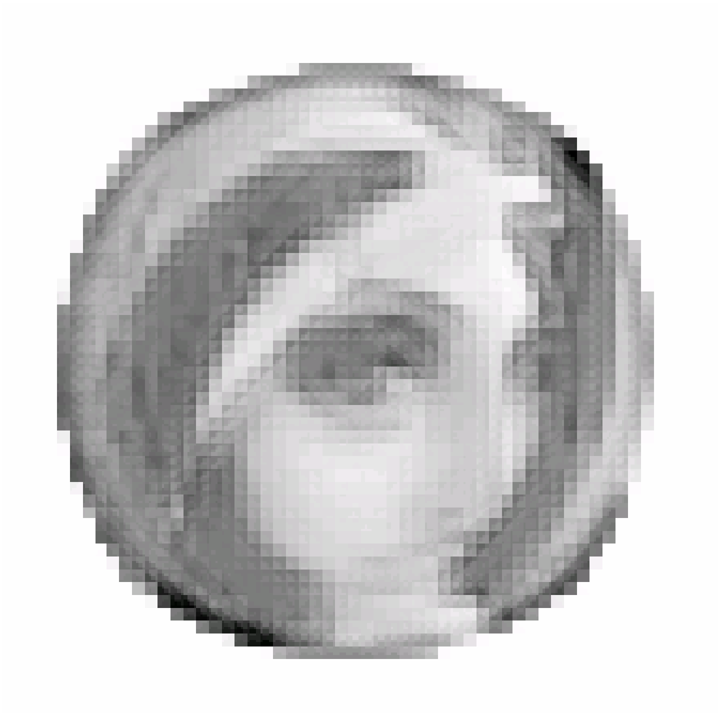
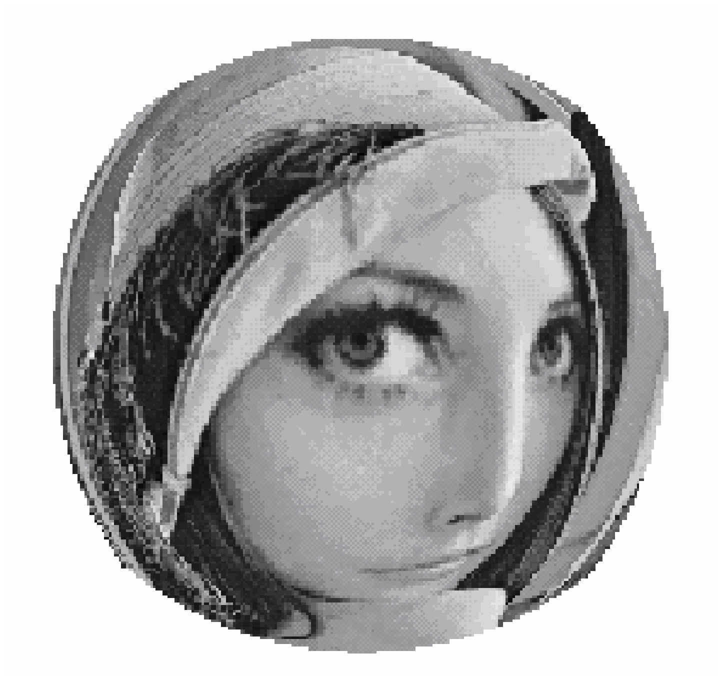


Fig. 4.49 Average of average cells mapping and edge detection mapping (221×221)

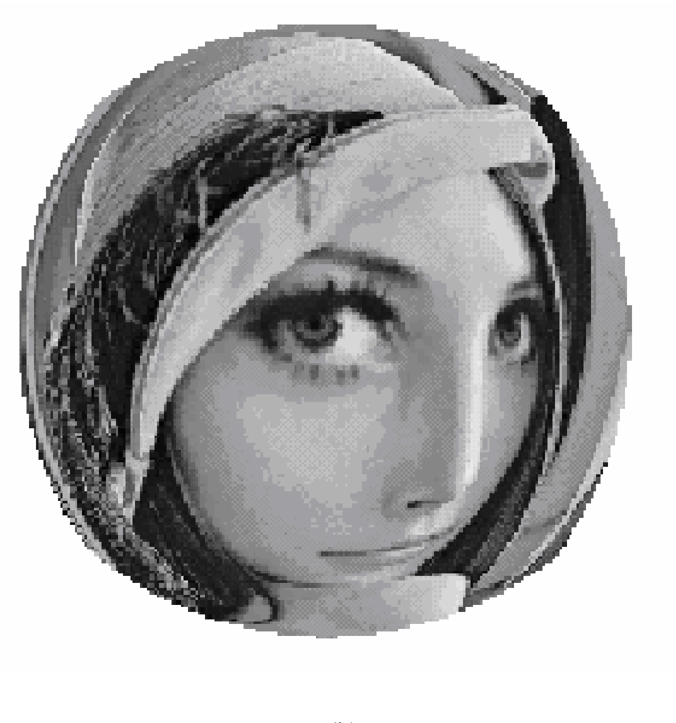
4.3.4 Retina Sampling with Micro-saccades

Our eyes perform small movements even while we look at a stationary visual scene. Micro-saccades are a kind of fixational eye movement. They are small, jerk-like, involuntary eye movements, similar to miniature versions of voluntary saccades [Answe]. They complement saccades providing visual enhancement of observed images through spatio-temporal associations and time domain filtering.

By implementing micro-saccades, we can get a set of sampled images. In this micro-saccades with retina sampling, Main_Micro_Saccade_Sampling.m is used in folder Ganglion Cells_8_8_new (see Appendix A). Fig. 4.50 shows two examples of the sampled images by micro-saccades.



（a）



(b)

Fig. 4.50 Examples of sampled images by micro-saccades (221×221)

Fig. 4.51 shows the average image of the sampled images by micro-saccades in Fig. 4.50. Because of the small eye movements, the average image looks slightly blurred.

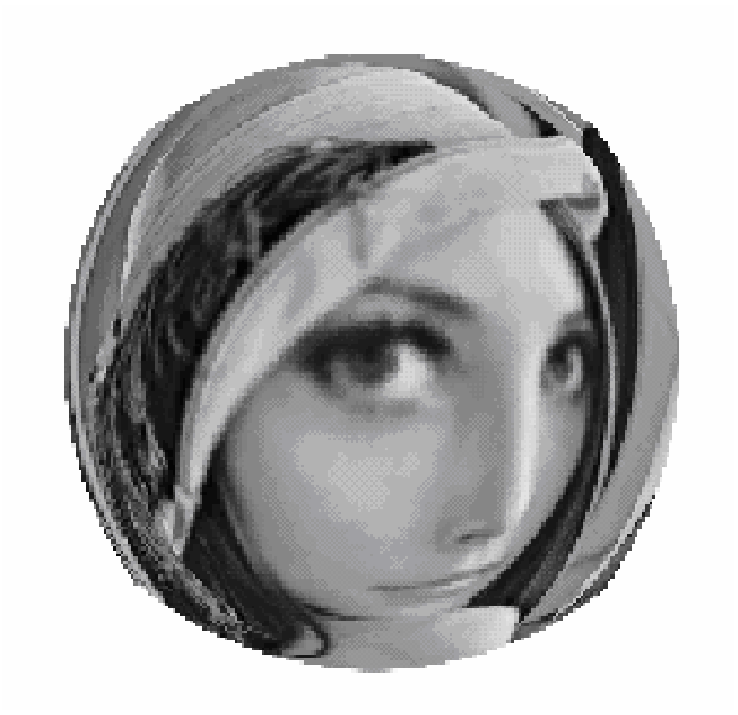


Fig. 4.51 Average sampled image by micro-saccades

For our active vision model, the sampled image will be sent to the neural network for recognition. However, what human perceived is not the direct sampled image from the retina, but the image that project the sampled image back to the original image plane.

Now let us consider projecting the sampled image back to the original image plane. In section 4.2.3 and 4.2.4, we discussed how to get the retina distribution by transforming the uniformly distributed samples within a unit circle. Take the sampled image (a) in Fig. 4.50 for example. Uniformly distributed samples in image (a) in Fig. 4.50 can be projected back to the original image. The size of the sampled image is 221×221 pixels, which is 5.26% of the original image size (964×964 pixels). In order to fill up empty spaces and recover the original image, we need to expand the signal value of one sampled pixel to its neighborhood. For instance, we could expand each sampled pixel to its $A_{\exp} \times A_{\exp}$ pixels neighborhood in the original image space. If there is an overlap between two neighborhood areas, we will take the average pixel intensity in this overlap. For $A_{\exp} = 5$, we will obtain the image as shown in Fig. 4.52.

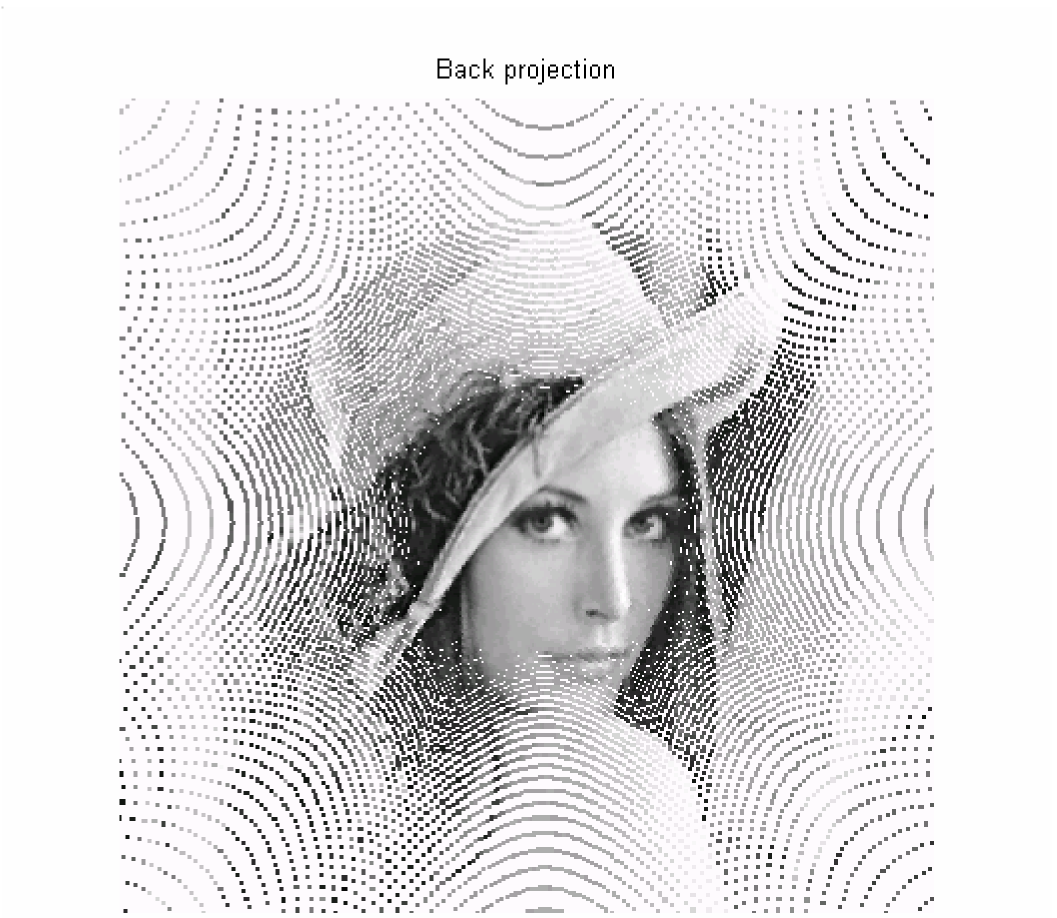


Fig. 4.52 Partially recovered image

Image in Fig. 4.52 shows that some areas in the original image space are not recovered. It is a result of non-uniform retina sampling.

Since the density of the photoreceptors decreases as the distance between photoreceptors and the fovea $dist$ increases, in order to recover the whole image, we need to increase the $A_{\exp}$ as the distance between photoreceptors and the fovea increases.

By experimentally obtained data, the size of the expand area $A_{\exp}$ is:

$$A_{\exp} = \frac{1}{17500} \times dist^{2.26} + 2\,. \tag{4.15}$$

By using this equation, we could recover the whole image as shown in Fig. 4.53.

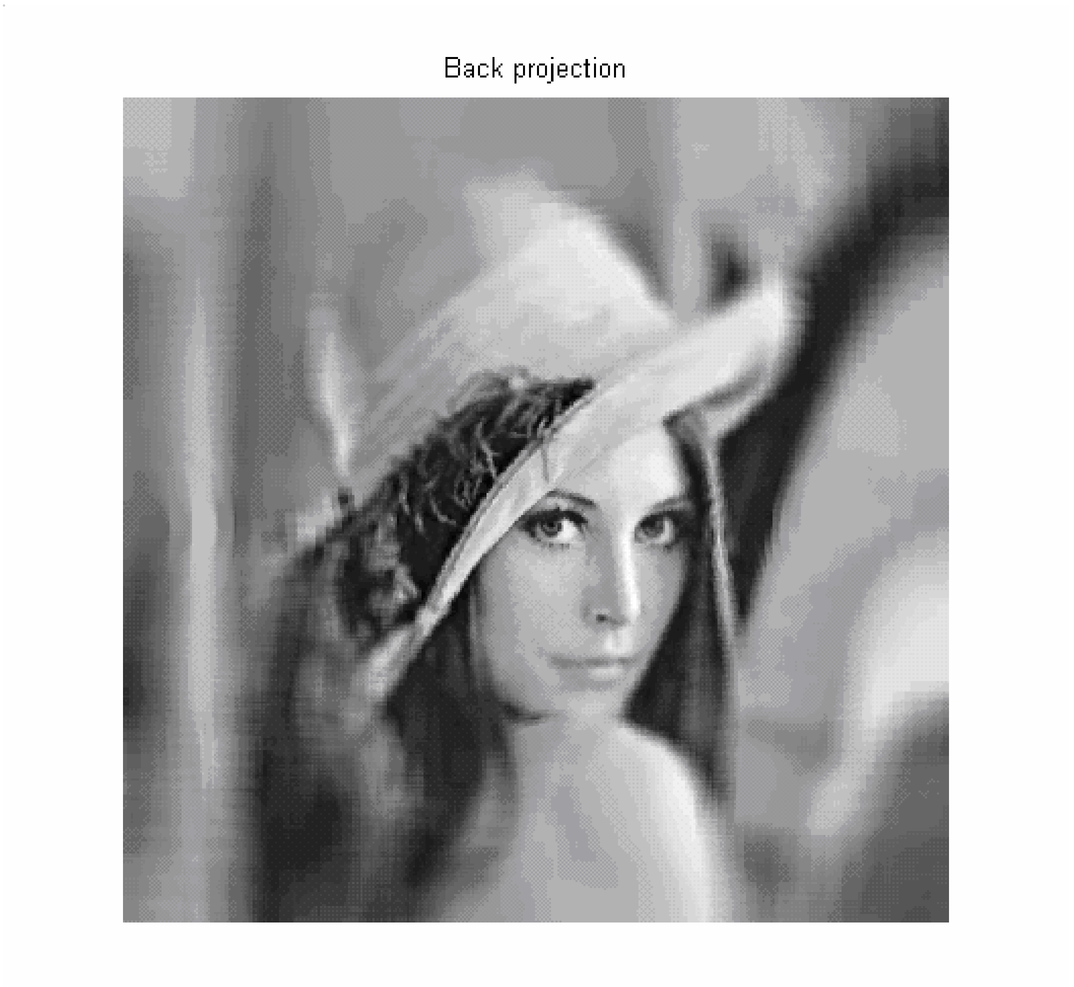


Fig. 4.53 Recovered image

To illustrate the affects of micro-saccades, we can project 20 sampled images generated by micro-saccades back to the original image and average them. In micro-saccades, the retina quickly and slightly moves across the image. These moves provide the filtering affect in neighborhood of a sampled data and effectively enhance perception. As a result, we could get more smooth-looking image as shown in Fig. 4.54. In addition, the recovered image looks more detailed and clear around the fovea area compared with the recovered image in Fig. 4.53.

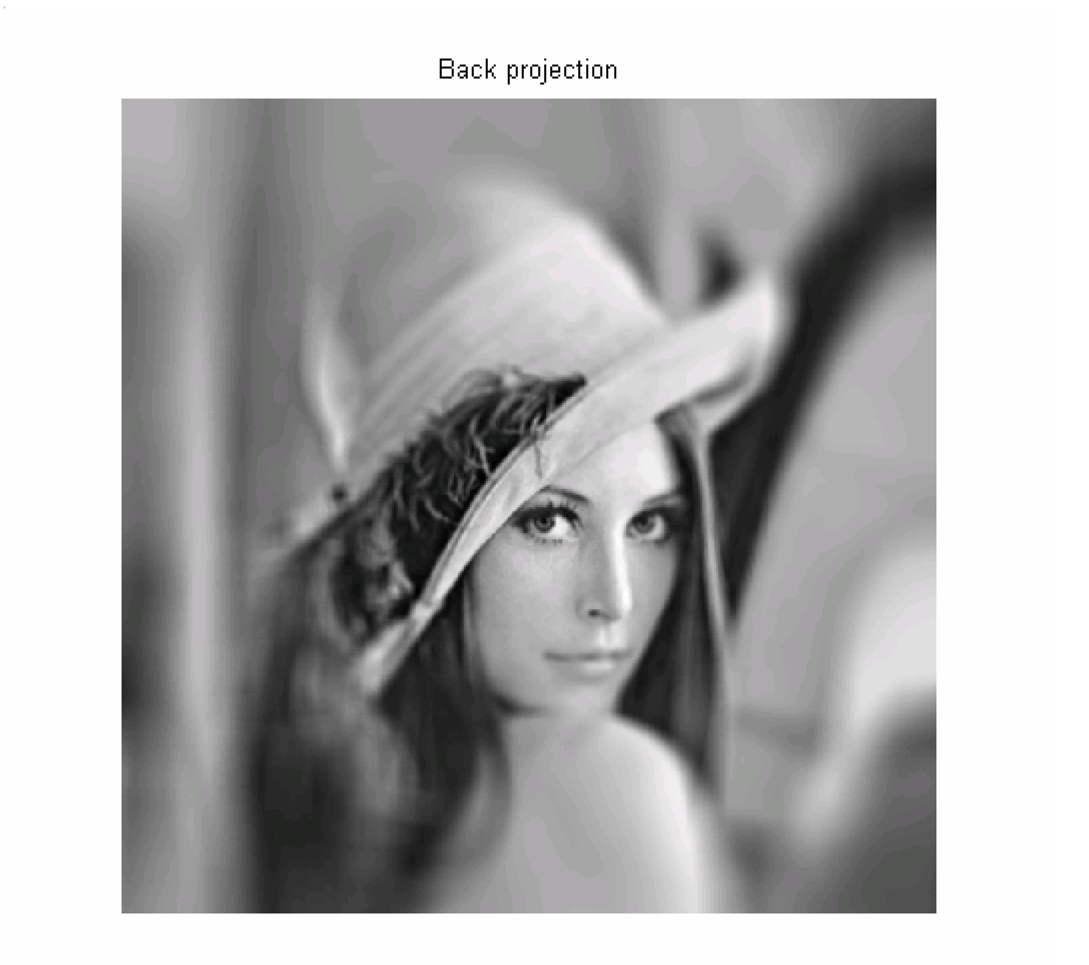


Fig. 4.54 Recovered image by micro-saccades

In section 4.3.2.4, we discussed direct motion detection implementation. Now let us consider combining retina sampling and micro-saccades with the motion detection.

Fig. 4.55 shows the average sampled image with 3 times micro-saccades of the image at time t=T0 in Fig. 4.41(a).

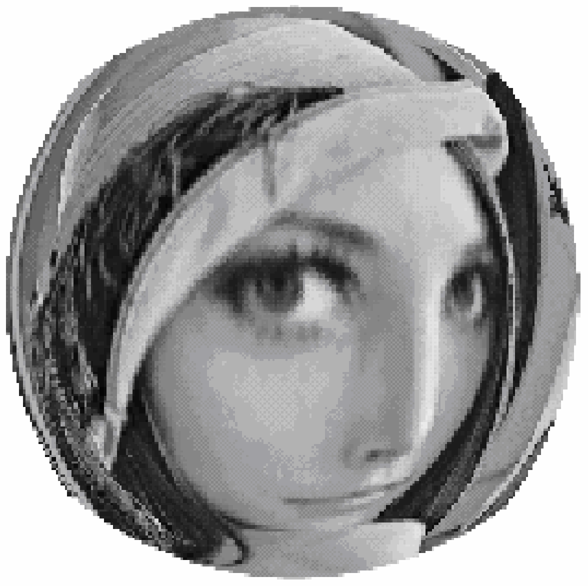


Fig. 4.55 Average sampled image by micro-saccades at time t=T0

Fig. 4.56 shows the corresponding recovered image of the average sampled image at time t=T0 in Fig. 4.55.

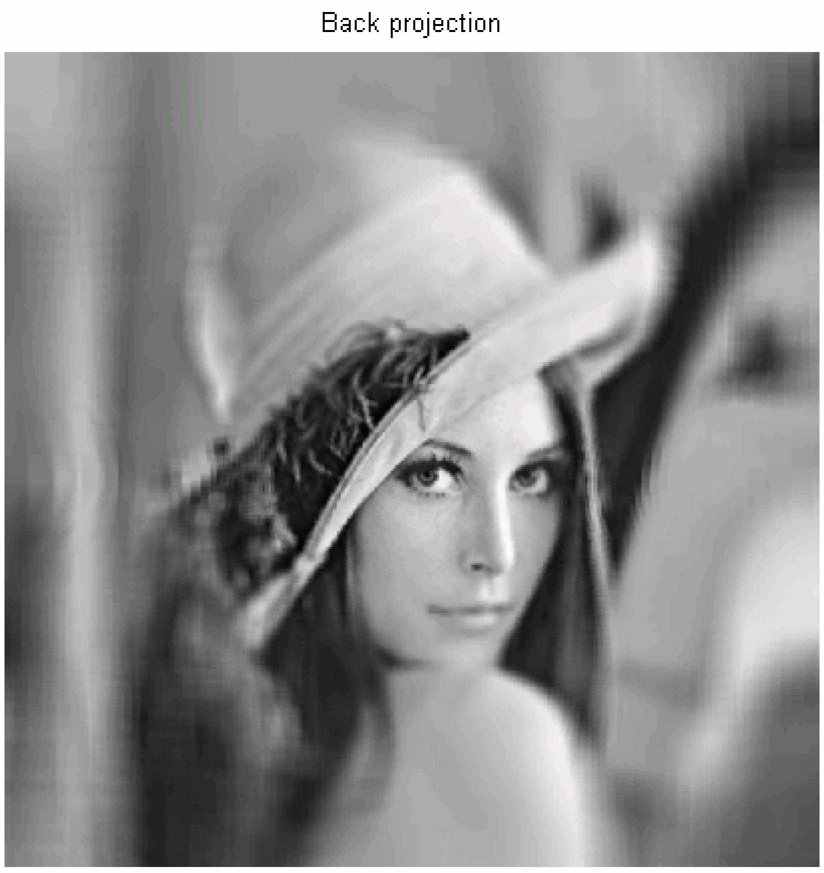


Fig. 4.56 Recovered image by micro-saccades at time t=T0

Similarly, Fig. 4.57 shows the average sampled image with 3 micro-saccades of the image at time t=T0+δ in Fig. 4.41(b).

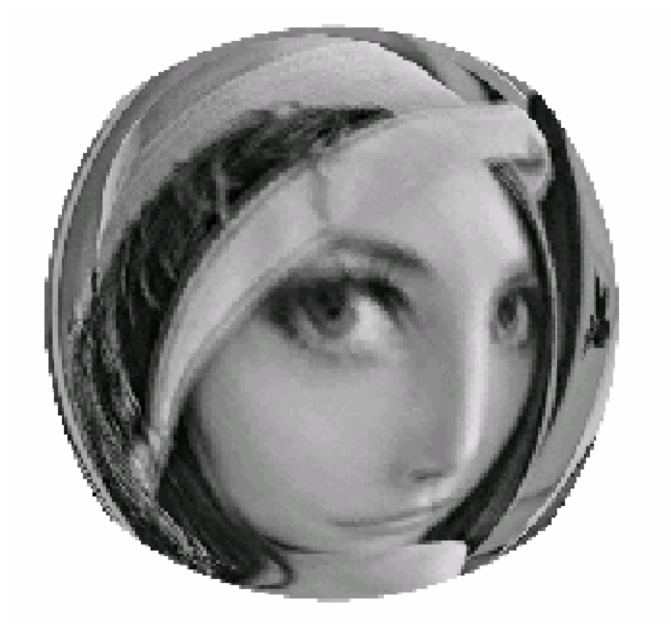


Fig. 4.57 Average sampled image by micro-saccades at time t=T0+δ

Fig. 4.58 shows the corresponding recovered image of the average sampled image at time t=T0+δ in Fig. 4.57.

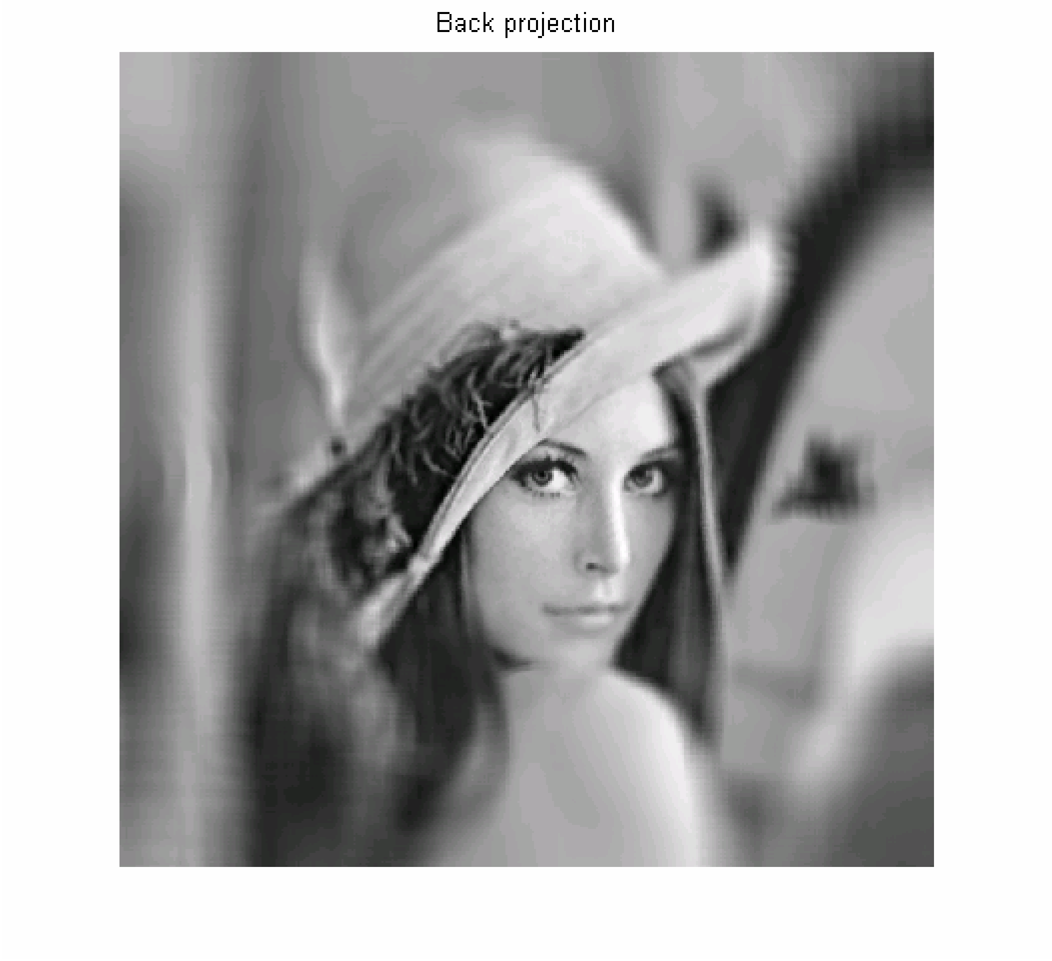


Fig. 4.58 Recovered image by micro-saccades at time t=T0+δ

By using the motion detection cells as described in section 4.3.2.4, we can show the result of the motion detection through MD cell processing of Fig. 4.56 and Fig. 4.58 as illustrated in Fig. 4.59.

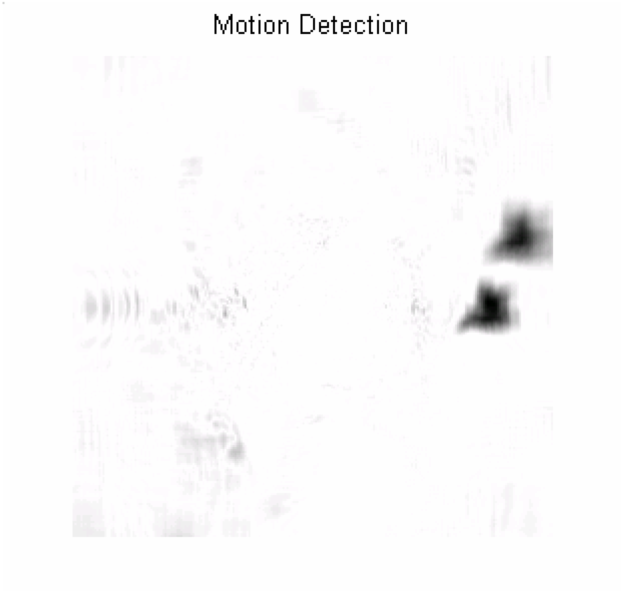


Fig. 4.59 Result of motion detection cells with 3 micro-saccades at time t=T0+δ

In Fig. 4.59, we can see that there is some noise in the background. If we increase the combined affect of micro-saccades to 10, we will get the result of motion detection cells as shown in Fig. 4.60. Thus, micro-saccades helped to reduce the background noise, while maintaining strong motion detection signals.

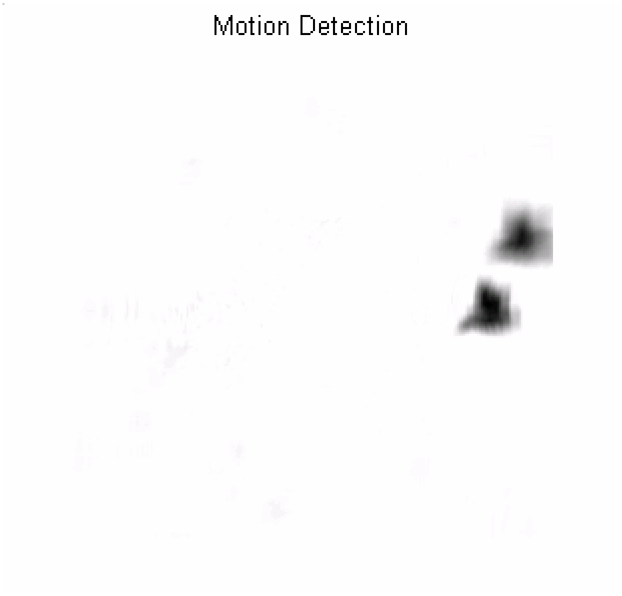


Fig. 4.60 Result of motion detection cells with 10 micro-saccades at time t=T0+δ

The mapping of the average intensity cells in Fig. 4.48 can also be projected back to the original image space. Fig. 4.61 shows the recovered image by back projected average intensity cells. Compared with the recovered image directly obtained from the retina sampling in Fig. 4.53, Fig. 4.61 has lower resolutions.

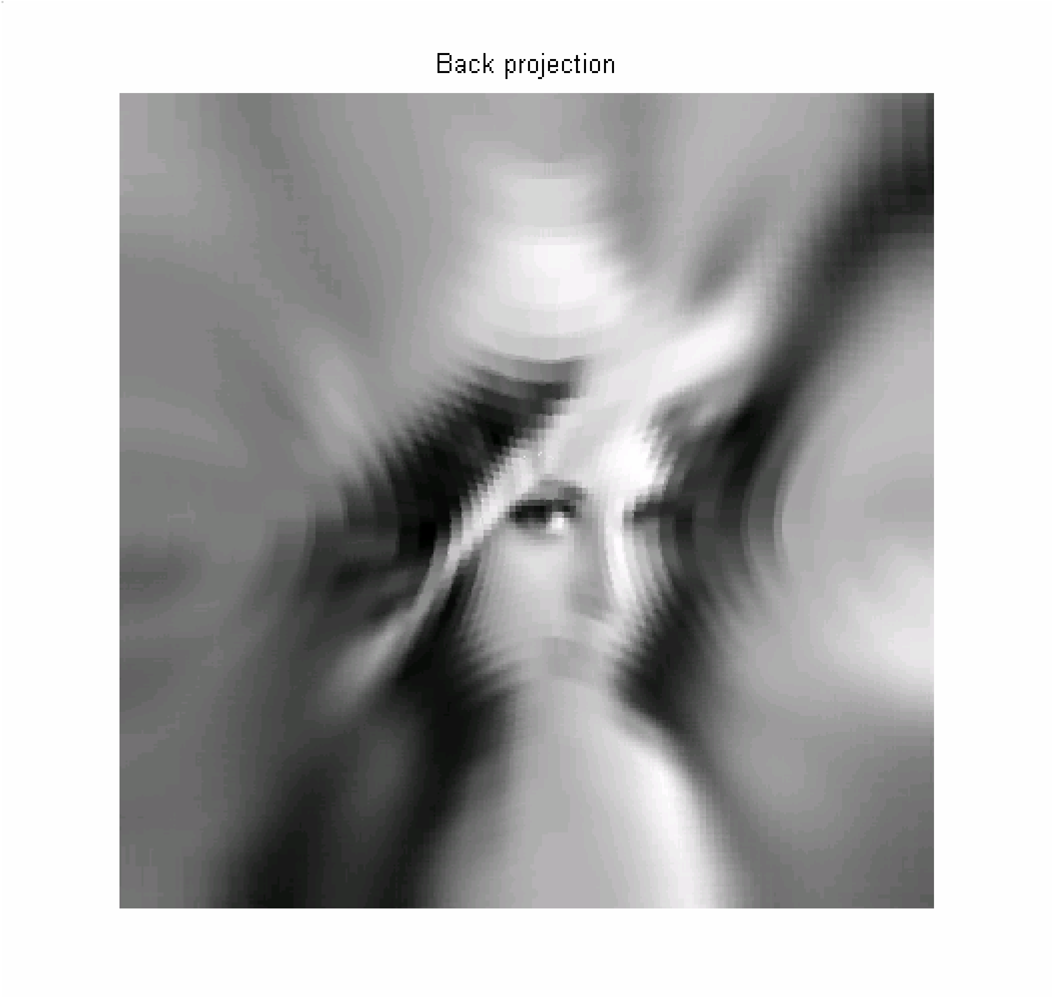


Fig. 4.61 Recovered image by average intensity cells

Fig. 4.62 shows the image recovered from sampling average intensity cells with 20 micro-saccades. As expected, micro-saccades made the result smoother than the recovered image in Fig. 4.61.

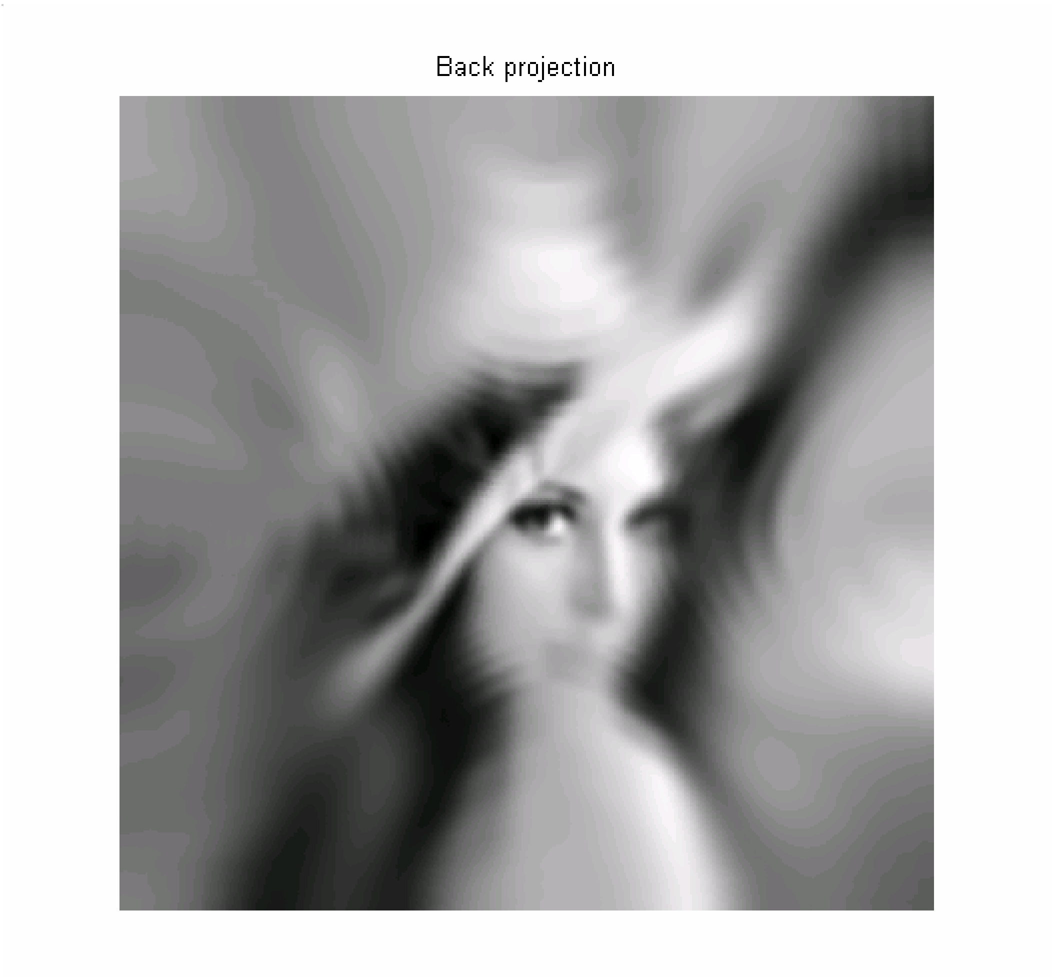


Fig. 4.62 Recovered image sampled by average intensity cells with micro-saccades

4.4 Saccade Movements Network

4.4.1 Introduction and Network Structure

A task of recognizing moving objects with continues observation and saccade movements becomes an interesting research topic. The saccade eye movement extracts spatial relations between different features in an object. The mechanism of saccade eye movement enables the neural network to analyze the whole pattern with relatively small number of samples received through photoreceptors enabling higher precision of the recognition task. During the saccade movements, a stationary object will shift on the retina in proportion to the magnitude and the direction of a saccade [Whi04]. In addition, the saccade movement may be useful for observation of many objects in the visual field. By following the object and its features, saccades make object projections on the retina more stable and thus make identification of moving objects easier.

In this section, we will use the EURETA model described in section 4.2.4 to sample the high resolution images. We describe an architecture and algorithm for saccade movements network (SMN). This network can identify the whole high resolution image using a low resolution retina after several accurate saccades.

The block structure of SMN is shown in Fig. 4.63. The SMN contains the retina sampling model and the modified SOWTAC network. The original high resolution images are first sent to the retina sampling model. After the retina sampling, the original image will be presented by a set of extracted features with low resolution saccade images. These sampled features are then sent to the modified SOWTAC network for further processing.

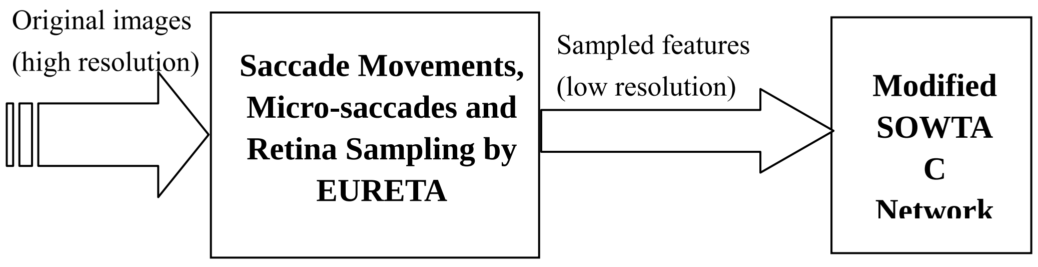


Fig. 4.63 Structure of Saccade Movements Network (SMN)

4.4.2 Retina Sampling, Saccade Movements and Micro-saccades

In this section, we will describe the saccade movements, micro-saccades and retina sampling by EURETA block of SMN from Fig. 4.63. Fig. 4.64 shows the original image with 320×240 pixels obtained by using face generation software [Faceg]. In this section, main_retina_model4.m is used for feature detection and extraction, micro-saccades, and retina sampling in folder RLS_WTA_Classificaton_Saccade\Feature_Detection_Micro_Saccade (see Appendix A).

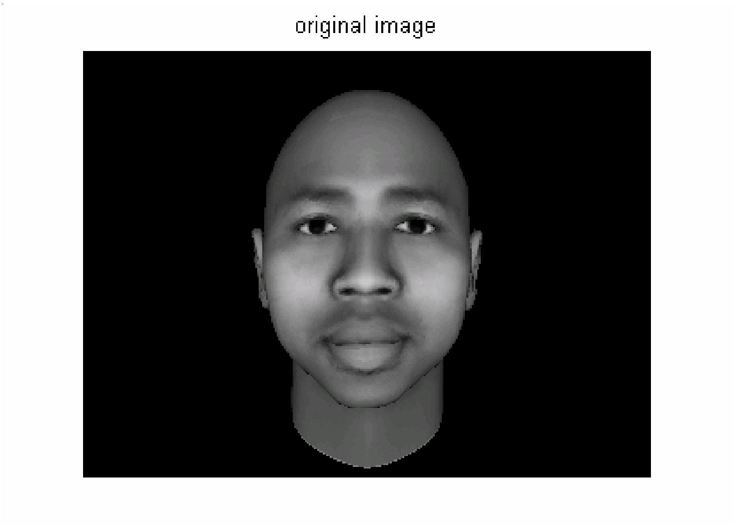


Fig. 4.64 Original Image (320×240)

Once the original image is received from the input, we need to do the image segmentation to find the interesting features of the image and perform feature extractions. We use edge detection and basic morphology tools [Mathw] for finding the interesting features of the image. The segmentation includes detecting the feature, dilating the image, filling interior gaps, removing connected object on border, and smoothing the feature. Fig. 4.65 shows the interesting features of the original image in Fig. 4.64. In Fig. 4.65, the stars represent the center of each interesting feature and the rectangles represent their boundaries.

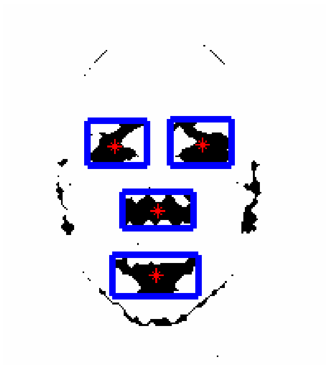
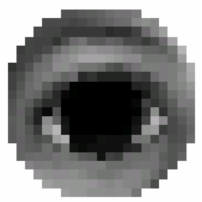
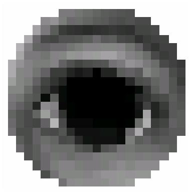
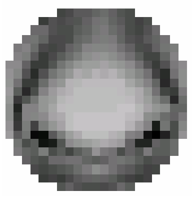
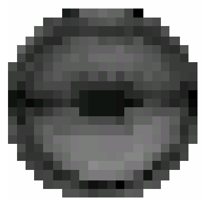
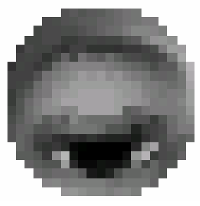
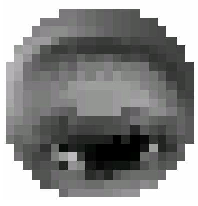
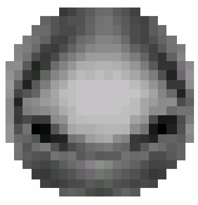
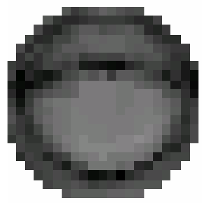


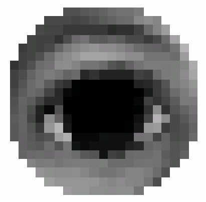
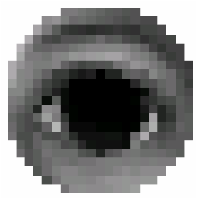
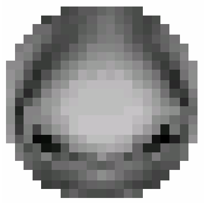
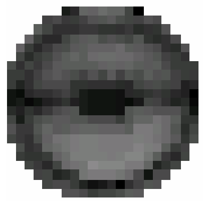
Fig. 4.65 Interesting features and boundaries

The retina sampling model EURETA will sample each interesting feature by placing its fovea towards the center of each interesting feature (called the focus point).

However, sometimes the focus point obtained from [Mathw] is not accurate. In order to improve the sampling quality, we will use reinforcement leaning optimization and micro-saccades. They will help the retina find the perfect sampling position. Initially, we stored some ideal samples as references for micro-saccades. By using the reinforcement leaning optimization algorithm from [Liu06], the fovea of the retina moves slightly each time until the sampled image is closest to the corresponding ideal sample. Table 4.1 shows the results of the image sampling by using optimization and micro-saccades. The first row shows the stored ideal samples. The second row shows the initial sampled images. The third row shows the final sampled images after micro-saccades.

Table 4.1 Image sampling by micro-saccade

	Left eye	Right eye	Nose	Mouth
Ideal images				
Initial sampled images				

After micro-saccades				

After the original image is presented by a set of extracted features with low resolution saccade images, these sampled features will be then sent to the modified SOWTAC network for further processing.

4.4.3 Modified SOWTAC Network

Fig. 4.66 shows the detailed hierarchical structure of the modified SOWTAC network. Layer A to layer C contains a set of parallel sub-networks SOWTAC as described in detail in Chapter 2. Layer D is the dual neuron layer for saccade merging. Layer E defines the classified categories.

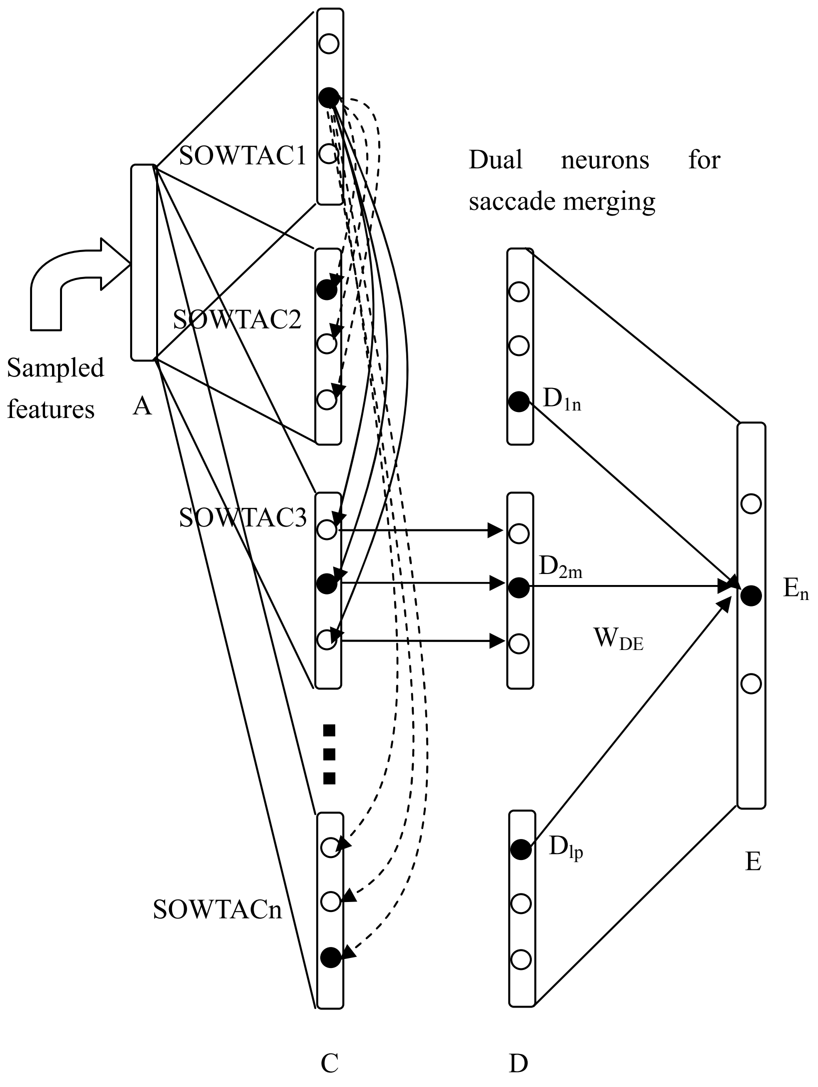


Fig. 4.66 Hierarchical structure of multiple SOWTAC network

After saccade movements and retina sampling by EURETA, we will obtain a set of different features, like eye, nose and mouth. The sampled features will be sent to layer A as inputs. SOWTAC1 to SOWTACn receive the same input from layer A and process it in parallel. Specifically, SOWTAC1 differentiates different features (that is to tell whether the current input feature is an eye, a nose, or a mouth); and SOWTAC2 to SOWTACn differentiate within a certain feature (e.g., to differentiate between different kinds of noses). For instance, SOWTAC2 differentiate a nose from all different kinds of noses, SOWTAC3 differentiate a mouth from all different kinds of mouths, SOWTACn differentiate an eye from all different kinds of eyes. Thus, the output neurons of SOWTAC1 represent categories of different features (eyes, nose, mouth), and the output neurons of SOWTAC2 to SOWTACn represent categories of a certain feature (e.g., which kind of nose).

The operations of SOWTAC1 to SOWTACn are the same as described in Chapter 2. During the supervised leaning, SOWTAC1 will be trained with eyes, noses and mouths. SOWTAC2 will be trained with different noses. SOWTAC3 will be trained with different mouths and SOWTACn will be trained with different eyes. In Fig. 4.66, the black dot represents the firing neuron and the black circle represents the quiet neuron. The output neurons of layer C in SOWTAC1 provide excitatory or inhibitory links to the output neurons of layer C in SOWTAC2 to SOWTACn. If SOWTAC1 identifies the current feature input is a mouth, then the neuron in layer C of SOWTAC1 representing the mouth category will fire. This firing neuron in SOWTAC1 will in turn provide excitatory link to the output neurons in SOWTAC3 which is specialized in differentiating different mouths. It will allow SOWTAC3 to copy its output neurons activity to the corresponding dual neurons in layer D. Meanwhile, the firing neuron in SOWTAC1 will provide inhibitory links to SOWTAC2 and SOWTACn, thus prevent SOWTAC2 and SOWTACn from copying their output neurons activity to the dual neurons in layer D.

Dual neuron layer D has several groups. The number of groups equals to the number of the output neurons of SOWTAC1. The number of dual neurons in each group equals to the number of the output neurons of SOWTAC2 to SOWTACn. Therefore, different groups in layer D can be regarded as different feature categories. Different neurons in each group represent the category of a certain feature.

Layer E defines different categories of the original images (which are combinations of different features). W_{DE} is the weight matrix from layer D to layer E, where

$$E = CW_{DE}. \tag{4.16}$$

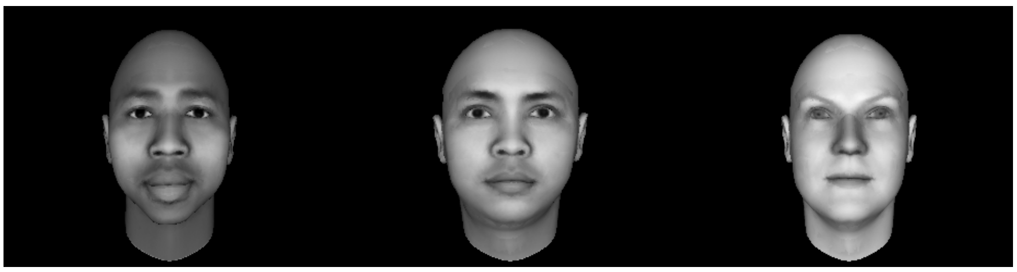
We perform the supervised learning between layer D and layer E to obtain the initial weight matrix from layer C to layer D as:

$$W_{DE0} = pinv(D) \times E. \tag{4.17}$$

During the unsupervised learning, a set of sampled features from retina sampling model will be sent to both SOWTAC1 to SOWTACn. Suppose that $D_{1n}, D_{2m}, \ldots, D_{Lp}$ are the firing dual neurons of layer D, which in turn triggers firing of the node E_n in layer E. Then the wining node E_n represents the classified category of the original image.

4.4.4 Simulation Results

In this SMN simulation, main_wta_output_setting_nonlinearmodel_cleanup_sacade.m is used in folder RLS_WTA_Classificaton_Saccade (see Appendix A). The SMN is tested in the following. Fig. 4.67 shows the training set used in the supervised learning.



Person A Person B Person C

Fig. 4.67 Training set

Fig. 4.68 shows the testing images used in the unsupervised training. As shown in Fig. 4.68, we can see that the position of each person is slightly different from the training set.

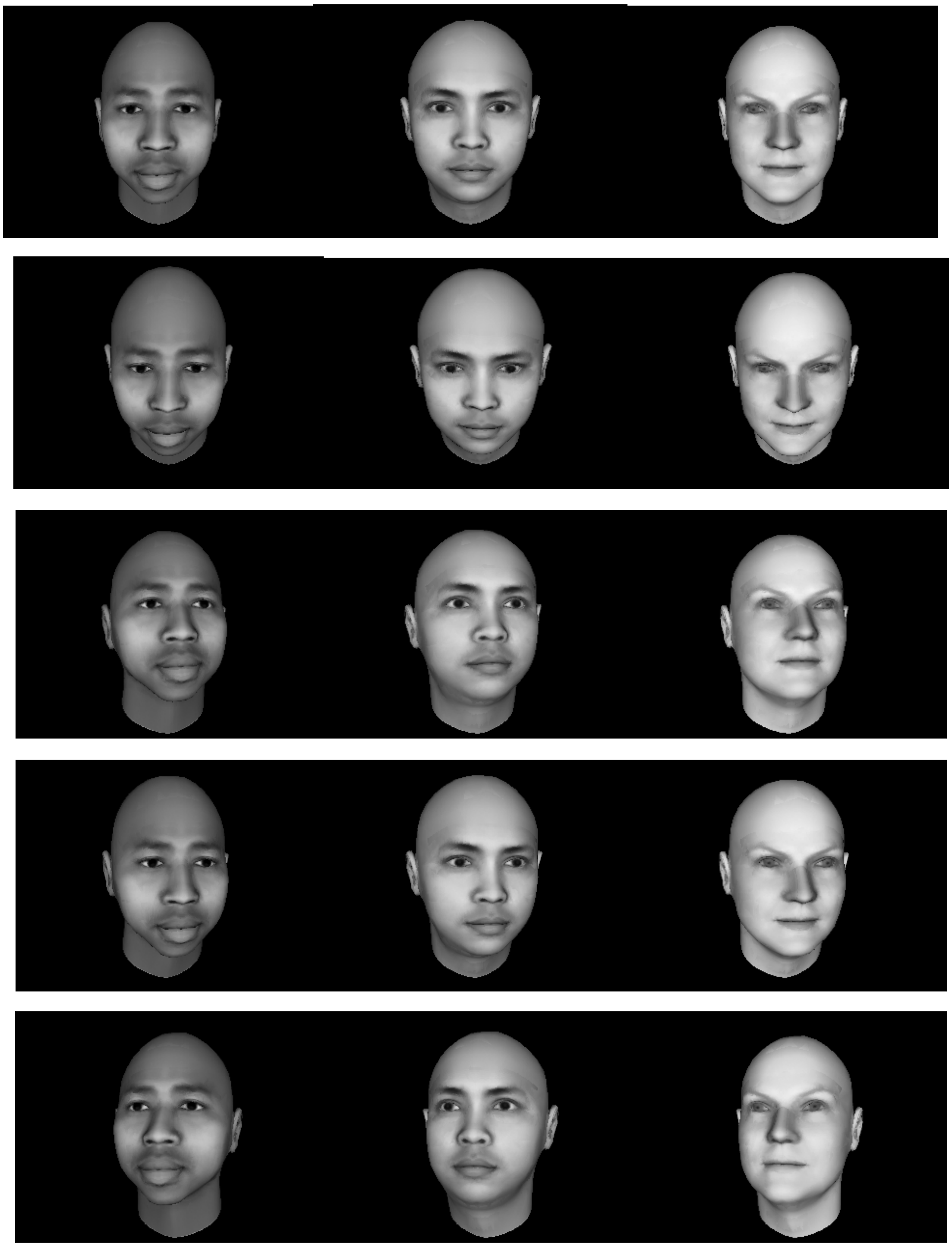


Fig. 4.68 Testing set

The testing set is sent to the SMN first for retina sampling and then to the modified SOWTAC network for recognition. Table 4.2 shows the activities of output neurons in layer E which defines the categories for different people. Node 1 is person A category, node 2 is person B category and node 3 is person C category. From table 4.2, we can see that the network correctly recognized each person in testing set.

Table 4.2 Activities of output neurons in layer E

Input to SMN	Node 1 (Person A)	Node 2 (Person B)	Node 3 (Person C)
Person A	1	0	0.00070700
Person A	1	0	0.00071837
Person A	1	0	0.00081445
Person A	1	0	0.00075313
Person A	1	0	0.00071274
Person B	0	1	0.00140860
Person B	0	1	0.00104000
Person B	0	1	0.00094245
Person B	0	1	0.00085233
Person B	0	1	0.00100880
Person C	0.0017494	0	1
Person C	0.0018738	0	1
Person C	0.0017975	0	1
Person C	0.0018909	0	1
Person C	0.0017892	0	1

5. Conclusions and Future Work

5.1 Conclusions

This thesis research mainly focuses on an innovative artificial vision model that uses self-organized, invariant representations, integrating continuous observation and saccade movements to achieve pattern recognition.

The neural network uses hierarchical feedback structures and winner-take-all scheme to build object representations, and self-organize invariant transformations. This network not only can classify different categories, but can also identify different views of the same object, even when a current view of the object cannot be uniquely recognized. If the network receives continuous input representing the same object over a period of time, the network anticipates that the input represents the same object by using feedback prediction signals. It maintains the same internal representation for the received input, unless there is a strong disagreement between the feedback predictions and received input signals.

This thesis also builds an artificial retina sampling model. Our retina model is built up by modeling the rods and cones distributions in human retina. In addition, the retina model models the ganglion cells to achieve spot detection and feature detection. When combined with the retina sampling, our model can extract any interesting feature from any high-resolution image, and therefore achieve more accurate recognition. By using the artificial retina sampling, our artificial vision model is capable of extracting spatial relations between different features in an object. The mechanism of saccade eye movement enables the neural network to analyze the whole pattern with relatively small number of samples received through photoreceptors enabling higher precision of the recognition task.

By using both supervised and unsupervised learning scheme, our saccade movement network can identify objects with high resolution. By repeating saccade movements and building their invariant temporal correlations at a sufficient detail level, an object can be perceived successfully.

Active vision is accomplished by a coordination of saccade movements, retina sampling, micro-saccade optimization and filtering, and spatio-temporal association between extracted features. It is this closed loop coordination of sensory vision mechanism with motor control of eyeball muscles, which makes the vision to be an active action of perception, recognition, understanding and expectation rather than staring, typical for classical image recognition.

5.2 Future Work

Although this thesis has achieved much as described above, there are still some future work that needs to be done.

This thesis has introduced a retina sampling model which is biologically based on rods, cones and ganglion cells. The retina sampling model can sample color images and detect motions. The neural network still needs to be improved to work in the real world, which contains color images and motions. In addition, this thesis mainly aims on image recognition. Similarly, speech recognition could be accomplished with similar hierarchical feedback network structures and spatio-temporal association combined with attention, speech recognition, understanding and expectation.

In the saccade movement network, the number of features is fixed. A more dynamic scheme needs to be implemented, which could introduce new features as they appear in the visual field or as new categories of objects are introduced.

Major effort in this thesis has been on software simulation. Hardware implementation of the active vision will be goal of future works.

References

[Ait03] M.J. Aitkenhead, A.J.S. McDonald. A Neural Network Face Recognition System. Engineering Applications of Artificial Intelligence 16, pp. 167-176, 2003.

[Answe] Answers.com. Answers Corporation. Available http://www.answers.com/topic/microsaccade

[Bra05] G. Bradski, A. Kaehler, V. Pisarevsky, V. Learning-Based Computer Vision with Intel's Open Source Computer Vision Library. Intel Technology Journal, May 2005. Available:http://developer.intel.com/technology/itj/2005/volume09issue02/art03_learning_vision/p01_abstract.htm

[Dor04] M. Dorr. Effects of Gaze-Contingent Stimuli on Eye Movements. Diploma Thesis, University of Lübeck, Germany, 2004.

[Every] Everything2. The Everything Development Company. Available: http://everything2.com/index.pl?node_id=244601

[Faceg] Facegen Modeller. Singular Inversions. Available: http://www.facegen.com/products.htm

[Gre99] G. Gancarz, S. Grossberg. A Neural Model of the Saccadic Eye Movement Control Explains Task-Specific Adaptation. Vision Research, Technical Report CAS/CNS-TR-98-024 39 3123 - 3143, 1999.

[Iva04] Y. Ivanov, B. Heisele, T. Serre. Using Component Features for Face Recognition. In Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition, pp. 421-426 2004.

[Kan73] T. Kanade. Picture Processing by Computer Complex and Recognition of Human Faces. Phd thesis, Kyoto University, 1973.

[Kar96] K. Sobottka and I. Pitas. Extraction of Facial Regions and Features Using Color and Shape Information. Proc. Int. Conf. Pattern Recognition, vol. 3, pp. 421-425, Aug. 1996.

[Lia00] R. Liao, S.Z. Li. Face Recognition Based on Multiple Facial Features. Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition, pp. 239-244, 2000.

[Lee02] R.S.T. Lee. Elastic Face Recognizer: Invariant Face Recognition Based on Elastic

Graph Matching Model. Int. Journal of Pattern Recognition and Artificial Intelligence, vol. 16, No. 4 pp. 463-479, 2002.

[Liu06] Y. Liu. A Novel Optimization Algorithm by Reinforcement. In preparation. Ohio University, 2006

[Mathw] Detecting a Cell Using Image Segmentation. Image Processing Toolbox, the Mathworks. Avaliable: http://www.mathworks.com/products/image/demos.html

[Pic03] M. Picardi and T. Jan. Recent Advances in Computer Vision. The Industrial Physicist, vol 9, no. 1, 2003.

[Riede] G. Riedel, Physiology of Human Cells, Available: http://www.aberdeen.ac.uk/sms/ugradteaching/course.php?ID=10

[Sim00] T. Sim, R. Sukthankar, M. Mullin, and S. Baluja. Memory-based Face Recognition for Visitor Identification. In Proc. of IEEE Int. Conf. on Automatic Face and Gesture Recognition, pp. 214-220, 2000.

[Tri00] M. M. Trivedi, K. Huang. Intelligent Environments and Active Camera Networks. IEEE Transactions on Systems, Man and Cybernetics, vol 2, pp. 804-809, Oct. 2000.

[Tes91] M. Tessier-Lavigne. Phototransduction and information processing in the retina. In E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors, Principles of neural science, chapter 28, pp. 400–419. Appleton and Lange, Norwalk, Connecticut, 1991.

[Tor04] L. Torres, Is There Any Hope for Face Recognition?, Proc. of the 5th International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS 2004, pp. 21-23, Lisboa, Portugal, April 2004.

[Whi04] R. L. White III and L. H. Snyder. A Neural Network Model of Flexible Spatial Updating, J Neurophysiology, no. 91: pp. 1608-1619, 2004.

[Wikip] Wikipedia, the free encyclopedia. Available: http://en.wikipedia.org/wiki/Ganglion_cell

[Zha03] W. Zhao, R. Chellappa, A. Rosenfeld, P.J. Phillips. Face Recognition: A Literature Survey, ACM Computing Surveys, 2003, pp. 399-458

Appendix A: Source Code Used in Thesis

The source code packages used in this thesis are available at:

http://www.ent.ohiou.edu/~lily

RLS_WTA_Classification, package for SOWTAC tool in Chapter 2.

RLS_WTA_Invariance, package for CON tool in Chapter 3.

Retina_Sampling, package for retina sampling models in Chapter 4.

Ganglion Cells_8_8_new, package for ganglion cells implementation in Chapter 4.

RLS_WTA_Classificaton_Saccade, package for SMN tool in Chapter 4.

Access to these packages is to be granted upon request. To use the source code package, please refer to the help document in each sub-directories.