Petroleum University of Technology

Ahwaz Faculty of Petroleum Engineering

Department of Electrical Engineering

Title of Thesis:

Application of Artificial Intelligence in Face Recognition

Supervisor:

Dr. Karim Salahshoor

By:

Moein Danesh Ara

9463103

Abstract

Face recognition is an interesting field for application of artificial intelligence. Three objectives can be considered in such a research study. These have to do with face localization, alignment and identification. Although becoming a mature field, many challenges still remain to be handled for face recognition. Pose and illumination variations, face similarities within a family as well as possible difficulties to find enough training samples (which also need to have high enough resolution) for each subject are some of these challenges. From obtaining the raw data to providing them to a face recognition algorithm some processes are obligatory such as locating the face in a camera image and aligning it to prevent suffering from noise due to variances of pose and illumination.

In this research, different approaches for face recognition will be studied. An efficient approach is selected to design and implement in MATLAB software package. The obtained results in the research will be developed in the MATLAB and presented after analysis of the outcomes.

Acknowledgment

The author wishes to express appreciation to Dr. Karim Salahshoor professor of Petroleum University of technology Ahwaz Faculty for his assistance to complete project.

Table of Contents

List of Figures

Chapter 1

Introduction

Today we are living in a highly technological environment. We use devices which are getting smarter every day. Artificial intelligence has become one of the important technological aspects of today's high-tech world. Face recognition, as a biometric authentication technique, is an important application field of artificial intelligence. Its main advantage is that, unlike other biometric techniques such as finger print, iris and speaker recognition, it does not require the applicant to spend time in the personal data acquisition process. For instance, facial recognition software, which is deployed in a public area where many different people pass by, can recognize faces of passers in a crowd and can help identifying a criminal. Its main disadvantage is the sensitivity to illumination variances, poses and occlusions which occur in unstructured environments.

Detection In realistic conditions, faces of people are mixed with other faces as well as with other objects in camera images. For a face recognition application to function automatically, faces should be detected and localized first to be useful for the recognition application. Face detection, a special case of object detection, uses a search algorithm whose goal is finding the location of a face in an image. To do this, several samples (sub-images) are cropped from the source image and analyzed by a binary classifier that decides whether an image patch contains a face. After this, the sub-images which contain a face will be returned as detected faces.

There are two kinds of recognition problems: face verification and face detection. While in the former, the goal is to find whether two input faces belong to the same person, in the latter given an input image contains several faces the purpose is to identify faces correctly.

Chapter 2

Artificial Intelligence Structure

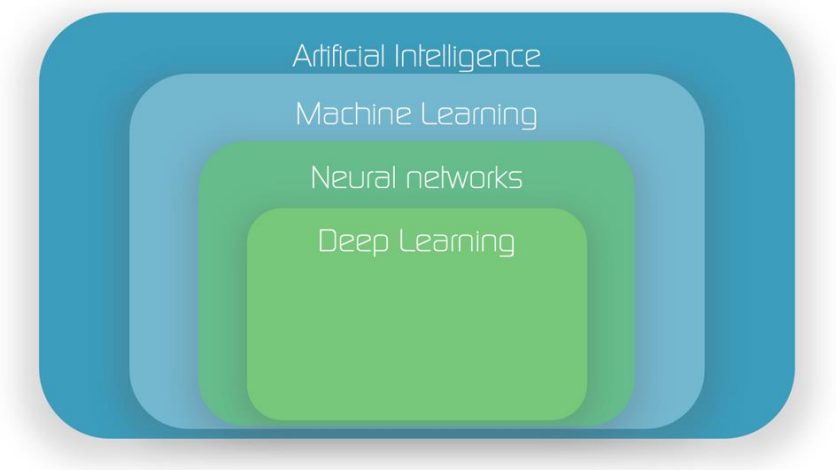

Figure 2. 1 Artificial Intelligence Structure

2.1 Turing Test

The Turing test developed by Alan Turing (Computer scientist) in 1950. He proposed that "Turing test is used to determine whether or not computer(machine) can think intelligently like human"?

Imagine a game of three players having two humans and one computer, an interrogator (as human) is isolated from other two players. The interrogator job is to try and figure out which one is human and which one is computer by asking questions from both of them. To make the things harder computer is trying to make the interrogator guess wrongly. In other words, computer would try to indistinguishable from human as much as possible.

If interrogator wouldn't be able to distinguish the answers provided by both human and computer then the computer passes the test and machine(computer) is considered as intelligent as human. In other words, a computer would be considered intelligent if it's conversation couldn't be easily distinguished from a human. The

whole conversation would be limited to a text-only channel such as a computer keyboard and screen.

He also proposed that by the year 2000 a computer "would be able to play the imitation game so well that an average interrogator will not have more than a 70-percent chance of making the right identification (machine or human) after five minutes of questioning." No computer has come close to this standard.

2.2 Artificial Intelligence

Artificial intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans. Some of the activities computers with artificial intelligence are designed for include:

- Speech recognition
- Learning
- Planning
- Problem solving

Artificial intelligence is a branch of computer science that aims to create intelligent machines. It has become an essential part of the technology industry.

Research associated with artificial intelligence is highly technical and specialized. The core problems of artificial intelligence include programming computers for certain traits such as:

- Knowledge
- Reasoning
- Problem solving

- Perception
- Learning
- Planning
- Ability to manipulate and move objects

Knowledge engineering is a core part of AI research. Machines can often act and react like humans only if they have abundant information relating to the world. Artificial intelligence must have access to objects, categories, properties and relations between all of them to implement knowledge engineering. Initiating common sense, reasoning and problem-solving power in machines is a difficult and tedious task.

Machine learning is also a core part of AI. Learning without any kind of supervision requires an ability to identify patterns in streams of inputs, whereas learning with adequate supervision involves classification and numerical regressions. Classification determines the category an object belongs to and regression deals with obtaining a set of numerical input or output examples, thereby discovering functions enabling the generation of suitable outputs from respective inputs. Mathematical analysis of machine learning algorithms and their performance is a well-defined branch of theoretical computer science often referred to as computational learning theory.

Machine perception deals with the capability to use sensory inputs to deduce the different aspects of the world, while computer vision is the power to analyze visual inputs with a few sub-problems such as facial, object and gesture recognition.

Robotics is also a major field related to AI. Robots require intelligence to handle tasks such as object manipulation and navigation, along with sub-problems of localization, motion planning and mapping.

2.3 Machine Learning

Machine learning is an artificial intelligence (AI) discipline geared toward the technological development of human knowledge. Machine learning allows computers to handle new situations via analysis, self-training, observation and experience.

Machine learning facilitates the continuous advancement of computing through exposure to new scenarios, testing and adaptation, while employing pattern and trend detection for improved decisions in subsequent (though not identical) situations.

Machine learning is often confused with data mining and knowledge discovery in databases (KDD), which share a similar methodology.

Tom M. Mitchell, a machine learning pioneer and Carnegie Mellon University (CMU) professor, predicted the evolution and synergy of human and machine learning. Today's Facebook News Feed is a perfect example. The News Feed is programmed to display user friend content. If a user frequently tags or writes on the wall of a particular friend, the News Feed changes its behavior to display more content from that friend.

Other machine learning applications include syntactic pattern recognition, natural language processing, search engines, computer vision and machine perception.

It's difficult to replicate human intuition in a machine, primarily because human beings often learn and execute decisions unconsciously.

Like children, machines require an extended training period when developing broad algorithms geared toward the dictation of future behavior. Training techniques include rote learning, parameter adjustment, macro-operators, chunking, explanation-based learning, clustering, mistake correction, case recording, multiple model management, back propagation, reinforcement learning and genetic algorithms.

2.4 Artificial Neuron Network (ANN)

An artificial neuron network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on that input and output.

ANNs are considered nonlinear statistical data modeling tools where the complex relationships between inputs and outputs are modeled or patterns are found.

ANN is also known as a neural network.

An ANN has several advantages but one of the most recognized of these is the fact that it can actually learn from observing data sets. In this way, ANN is used as a random function approximation tool. These types of tools help estimate the most cost-effective and ideal methods for arriving at solutions while defining computing functions or distributions. ANN takes data samples rather than entire data sets to arrive at solutions, which saves both time and money. ANNs are

considered fairly simple mathematical models to enhance existing data analysis technologies.

ANNs have three layers that are interconnected. The first layer consists of input neurons. Those neurons send data on to the second layer, which in turn sends the output neurons to the third layer.

Training an artificial neural network involves choosing from allowed models for which there are several associated algorithms.

2.5 Deep Learning

Deep learning is a collection of algorithms used in machine learning, used to model high-level abstractions in data through the use of model architectures, which are composed of multiple nonlinear transformations. It is part of a broad family of methods used for machine learning that are based on learning representations of data.

Deep learning is a specific approach used for building and training neural networks, which are considered highly promising decision-making nodes. An algorithm is considered to be deep if the input data is passed through a series of nonlinearities or nonlinear transformations before it becomes output. In contrast, most modern machine learning algorithms are considered "shallow" because the input can only go only a few levels of subroutine calling.

Deep learning removes the manual identification of features in data and, instead, relies on whatever training process it has in order to discover the useful patterns in the input examples. This makes training the neural network easier and faster, and it can yield a better result that advances the field of artificial intelligence.

Chapter 3

Different Methods and approaches to face detection

Finding the location and size of a face in an image can enable a face recognition application to extract features from a face corresponding to different entities. Furthermore, it allows to align different faces, so that more accurate recognition results can be obtained.

Face alignment is an important requirement for a successful face recognition application. A human face in an image can be in a variety of scales, positions and poses. Without any alignment of the face entities in an image, recognition performance is very limited.

Face detection is a sub-field of object detection in images. The approaches can be classified into three fundamental methods: Shape-based models, feature based models and appearance-based models.

Shape-based models depend on a geometrical model of the face and use this model to decide whether an image patch contains a face. It extracts contour properties of the image patch and compares these to the model using a similarity measure. a separability filter is used for feature extraction and the Hough transform is used for model fitting. The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. Some researchers focus on color images in order to exploit skin

color of faces. So, a color conversion algorithm is applied to the image containing a face so that the separation of skin color from the background becomes easier. After the conversion the face is detected by means of a face mask calculation. Shape-based face detection models may be suitable for real-time face-tracking applications. However, they are sensitive to different rotation angles and image quality. Moreover, for obtaining more precise results, these models use more parameters to model the shape and this results in an extensive engineering effort and the application is computationally more demanding.

Feature-based methods focus on finding local features related to the face. features of the face are extracted with the SURF algorithm Then these features are given to a support vector machine (SVM) to locate face. Special linear and non-linear filters constructed from Gabor wavelets are used to detect the face and it corner features. Then these features are further filtered to remove false features from the detected feature set. A voting mechanism is finally applied to compute the most accurate location of the face.

Appearance-based models make a model from face images by using the photometric appearance of the faces. Since no specific a priori information related to faces is used, a sufficient number of training data to learn the parameters for face detection is needed. For the purpose of eliminating noise and reducing dimensionality, feature extraction and normalization operations to training data are usually applied. As for

feature extraction techniques, principal component analysis (PCA), and edge detection methods are some of the techniques being used. After all these operations the output is given to a classifier for training. As classifiers, adaptive boosting, neural networks and SVMs have been used. In a paper patches of example face images are processed by principal component analysis (PCA) to reduce the dimensionality and make a model face for classifying unseen image patches if they contain a face. In another paper, a Gabor filter is used as a feature extractor and an SVM is used as a classifier. To make the final detector more robust to rotations, the face images are populated by rotation, translation and mirroring operations before giving them to a Gabor filter to be processed and then finally the output of it is fed into the SVM to train the classifier. The biggest advantage of the appearance-based methods is that they are applicable to all kinds of different objects, because they are based on machine learning algorithms to learn the model from training data. Therefore, they also often require almost no a priori knowledge and less engineering effort. A disadvantage is that they may need a lot of labelled data to learn a very good performing model.

3.1 Classifier and Feature Extraction Methods

3.1.1 Support Vector Machine

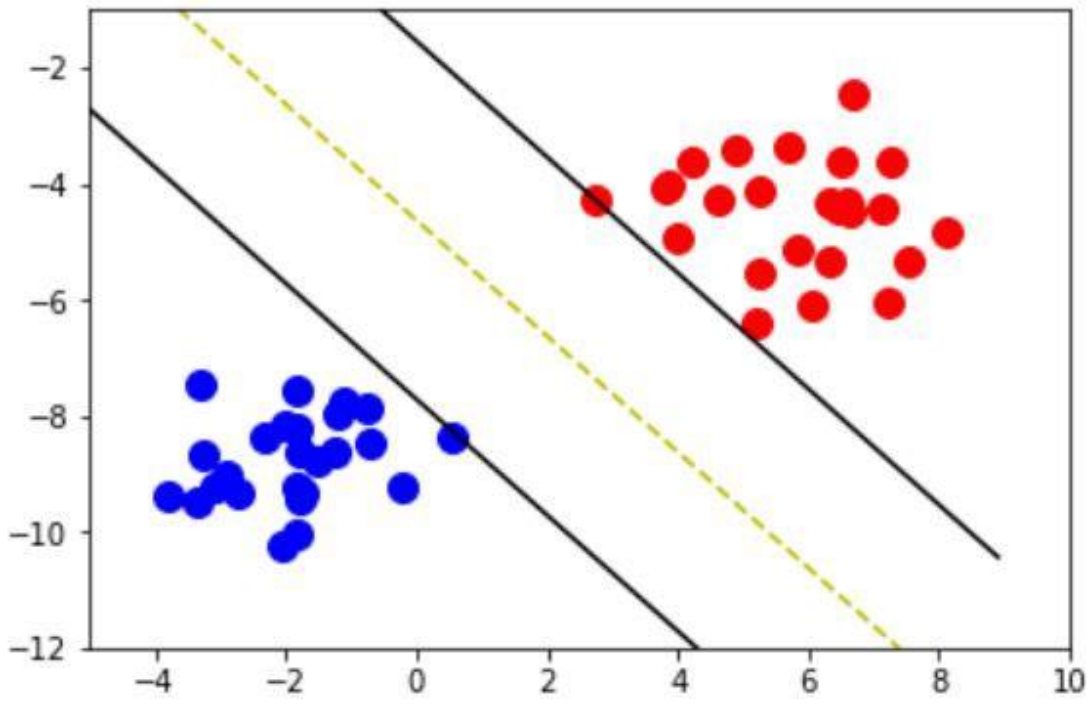

Figure 3. 1 High level view of SVM

The support vector machine (SVM), invented by Vapnik and co-workers, is a machine learning algorithm which is very useful for two-class pattern recognition problems. The SVM algorithm assumes that the maximum margin between two classes makes the best separation. Although originally developed as a linear classifier, an SVM can be used with non-linear kernels to produce a non-linear classifier. We will shortly describe the SVM. Let D be a training dataset,

$D = \{(x_i, y_i), 1 < i < n\}$

where $x_i \in R^p$ are input vectors and $y_i \in \{1, -1\}$ are binary labels.

Given an input vector $x_i$the linear SVM outputs the following class output o_i:

$$o_i = g(x_i) = sign(w^T x_i + b)$$

where w is the weight vector and b is the bias. To compute the weight vector w and the bias b, the SVM minimizes the cost function:

$$J(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi i$$

subject to constraints:

$$w^T x_i + b \geq +1 - \xi i \; for \; y_i = +1$$

and

$$w^T x_i + b \leq +1 - \xi i \; for \; y_i = -1$$

where C weighs the training error and $\xi i \geq 0$ are slack variables.

One possible disadvantage of this soft margin method is that it increases the number of support vectors and therefore it increases the chance of overfitting. A recent algorithm proposes a solution to this, called separable case approximation, which achieves the right separation with a decreased number of support vectors without using soft margins.

Non-linear Case. Although linear separation is faster and less complex than non-linear models, it is not suitable for all kinds of data. Because of this problem, the non-linear SVM model was proposed by Boser, Guyon, and Vapnik.

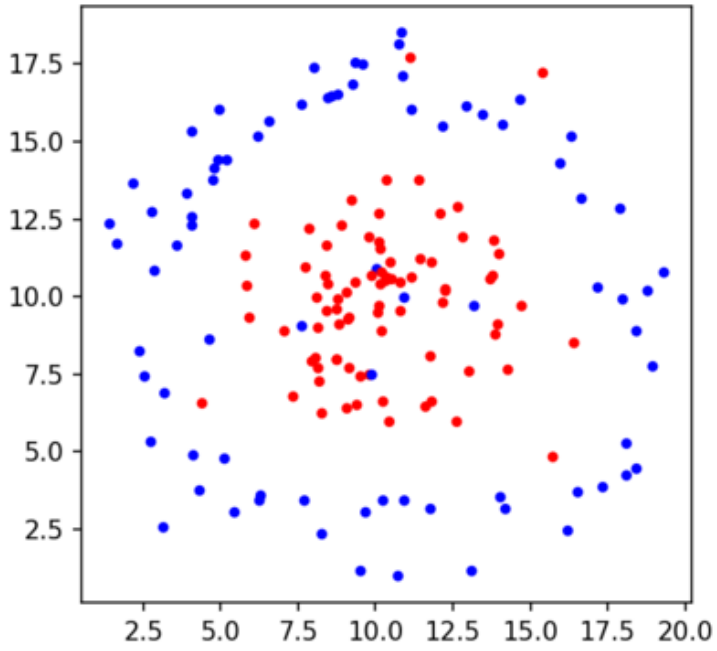

Figure 3. 2 Non-Linearly separable data

In this case, the dot product between two input vectors that leads to a linear classifier, is replaced with a non-linear kernel function that allows to separate non-linearly separable data. Many kernel functions have been proposed. The most often used kernel functions are the radial basis function (RBF):

$$RBF: K(x,y) = e^{(-\gamma||x-y||)^2}, \gamma > 0,$$

and the polynomial kernel:

$$POLY: K(x,y) = (x^T y + c)^d$$

Recently, to make benefit of discrimination capabilities of both kernels, a combination of these kernels given above is proposed, where the kernel formula becomes:

$$POLY - RBF: K(x,y) = (e^{(-\gamma||x-y||)^2} + c)^d$$

When using a kernel function, the decision function becomes:

$$o_i = g(x_i) = sign(\sum_{j=1}^{n} K(x_i, x_j)w_j + b)$$

where the weight w_j for an example $x_j$is given by $\alpha_j y_j$. Here α_j is computed by optimizing the dual objective problem of the SVM.

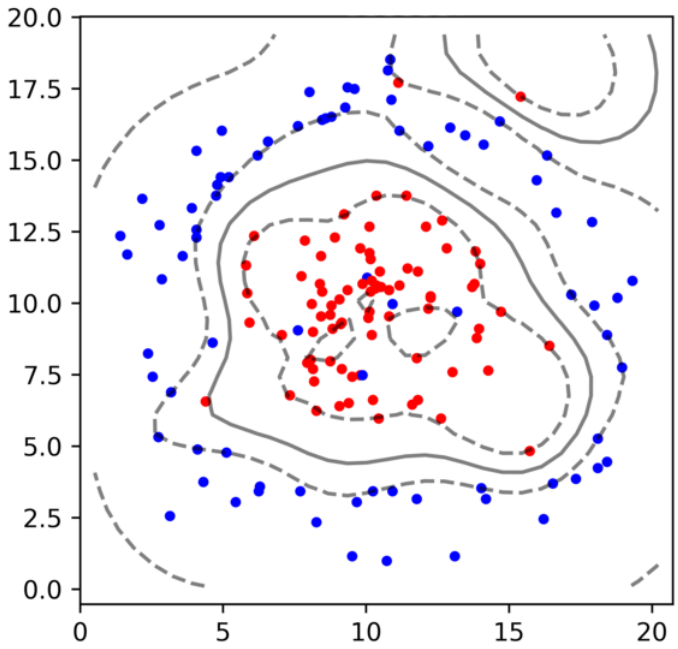

Figure 3. 3 SVM with RBF Kernel

3.1.2 Restricted Boltzmann Machine

An RBM is an energy-based neural network model used for suppression of noise and reducing the dimensionality of the input data. It is composed of two layers: an input layer and a hidden layer, which are connected to each other through (symmetric) weighted connections. This structure is called a bipartite graph. There are many possible implementation methods of these layers depending on the structure of the data to be modeled. While the two layers can be implemented with the same layer

type, different activation functions in different layers can also be used. The binary stochastic layer is the most prevalent implementation.

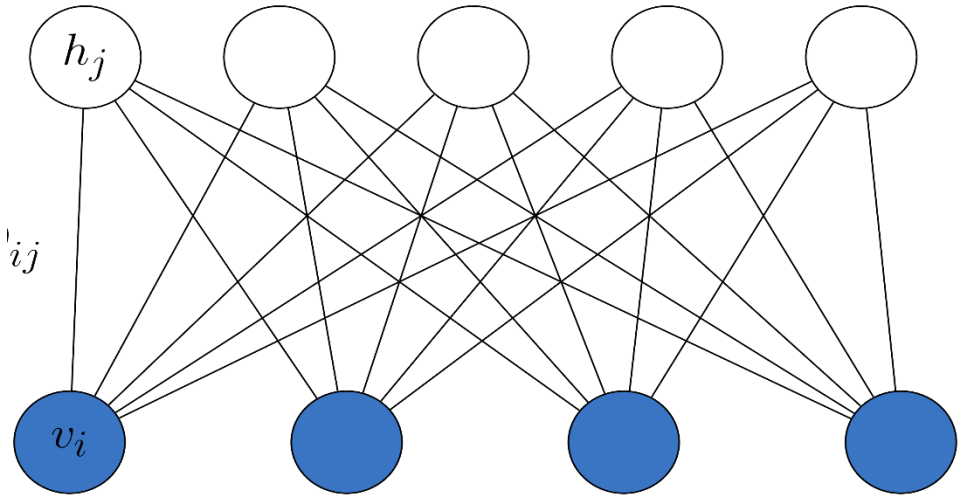

Figure 3. 4 An RBM with 5 hidden and 4 visual (or input) units

The mathematical description of the RBM is briefly given below.

Let v_i be the value of input unit i and h_j be the activity value of hidden unit j that models the input data and $\widehat{v_i}, \widehat{h_j}$ are reconstructed input and hidden values. h_j is computed from the input vector by:

$$h_j = f(b_j + \sum_i v_i w_{ij})$$

$\widehat{v_i}$ and $\widehat{h_j}$ are computed as:

$$\widehat{v_i} = f(a_j + \sum_i h_i w_{ji}) \text{ , } \widehat{h_j} = f(b_j + \sum_i \widehat{v_i} w_{ij})$$

where $f(\cdot)$ is the activation function, a_j is the bias for input unit j, b_j is the bias value for hidden unit j and w_{ij}'s are weights connecting input and hidden units. For the linear function $f(x) = x$ and for the logistic function $f(x) = \frac{1}{1+\exp(-x)}$.

To build a model using RBMs, the weight vector w is to be optimized. The most often used method to find the best weight vector, proposed by Hinton (2002), is the contrastive divergence algorithm. In this algorithm, the weight vector w is optimized according to the following update rule:

$$\Delta w_{ij} = \eta(\langle v_i h_j \rangle - \langle \hat{v}_i \hat{h}_j \rangle)$$

where η is the learning rate, $\hat{v}$ are reconstructed values of the input data and $\hat{h}$ are reconstructed values of hidden units. The angle brackets denote the expected value of any v_i, h_j pair, which are computed using a batch of training examples. Biases are updated by:

$$\Delta a_i = \eta(\langle v_i \rangle - \langle \hat{v}_i \rangle),\ \Delta b_j = \eta(\langle h_i \rangle - \langle \hat{h}_i \rangle)$$

After the optimization process, values of h_j are computed with the RBM given the input vector and then given to a classifier as a feature vector.

3.1.3 Difference-of-Gaussians Filter

The difference-of-Gaussians (DoG) filter is an edge detection algorithm that detects edges by subtraction of one blurred version of an original image from another, which is a less blurred version of the original. Let f be the image matrix, and let G_1 and G_2 be the first and second Gaussian functions, which produce Gaussian matrices for convolving the image. The Gaussian function is:

$$G_i(x, y) = \frac{1}{2\pi {\sigma_i}^2}\, e^{-\frac{x^2+y^2}{2{\sigma_i}^2}}, i \in \{1, 2\}$$

The Gaussian blurred images are:

$$O_i = (f \otimes G_i(x, y)), i \in \{1,2\}$$

Where $\otimes$ is a convolution operation. Finally, the final output image is computed by: $O = O_2 - O_1$.

Blurring an image using a Gaussian convolution kernel suppresses spatial information with high-frequency properties. Subtracting one blurred image from the other helps keeping spatial frequencies that are preserved in the two blurred images. So, the DoG can be considered a low band-pass filter which discards all except some significant spatial frequencies that are present in the original image.

Figure 3. 5 Sample frog image

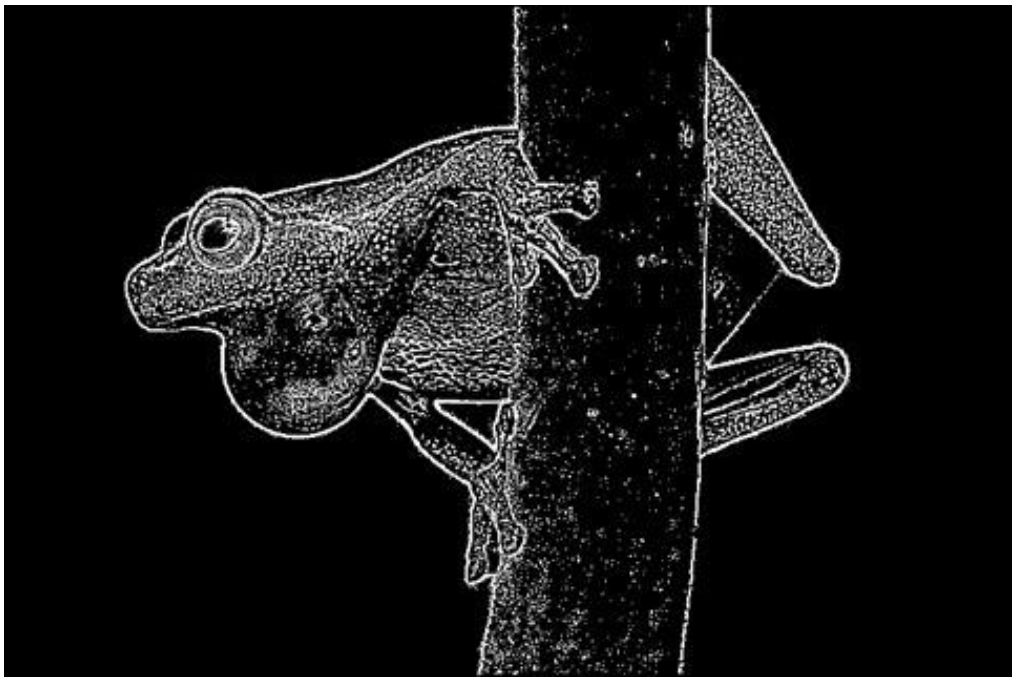

Figure 3. 6 DoG filter (Kernel 5×5, Weight1 2.7, Weight2 0.1)

3.1.4 Gabor Wavelets

A Gabor wavelet is a filter used for edge detection operations. It is a convolution product of a sinusoidal plane wave and a Gaussian function. The mathematical definition of the filter is given below:

$$g_{real}(x, y; \lambda, \theta, \phi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma_i^2}} \cos(2\pi \frac{x'}{\lambda} + \varphi)$$

$$g_{imaginary}(x, y; \lambda, \theta, \phi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma_i^2}} \sin(2\pi \frac{x'}{\lambda} + \varphi)$$

where $x' = xcos\theta + ysin\theta \; and \; y' = xsin\theta + ycos\theta.$

As can be seen above, it has five parameters to affect the response of the filter. Here, λ represents the wavelength of the sinusoidal wave function, θ represents the

orientation, ϕ is the phase offset, σ is the standard deviation of the Gaussian function and γ is the spatial aspect ratio which determines the ellipticity of the Gabor function.

In fact, the Gabor functions are Gaussian shaped band-pass filters, with dyadic treatment of the radial spatial frequency range and multiple orientations, which represent an appropriate choice for tasks requiring simultaneous measurement in both space and frequency domains. Gabor filters act very similar to mammalian visual cortical cells so they extract features from different orientation and different scales. Each Gabor filter has a real and an imaginary component that are stored in M×M masks, called R_{pq} and I_{pq} respectively, where p = 1 . . . S, denotes the scale, and q = 1 . . . L, denotes the orientation.

Suppose we have an image like:

Figure 3. 7 Sample car image

And then we calculate Gabor features at 5 scales (S = 5) and 8 orientations (L = 8). The mask size for example can be set to 27*27.

Gabor wavelets created using two control parameters are like:

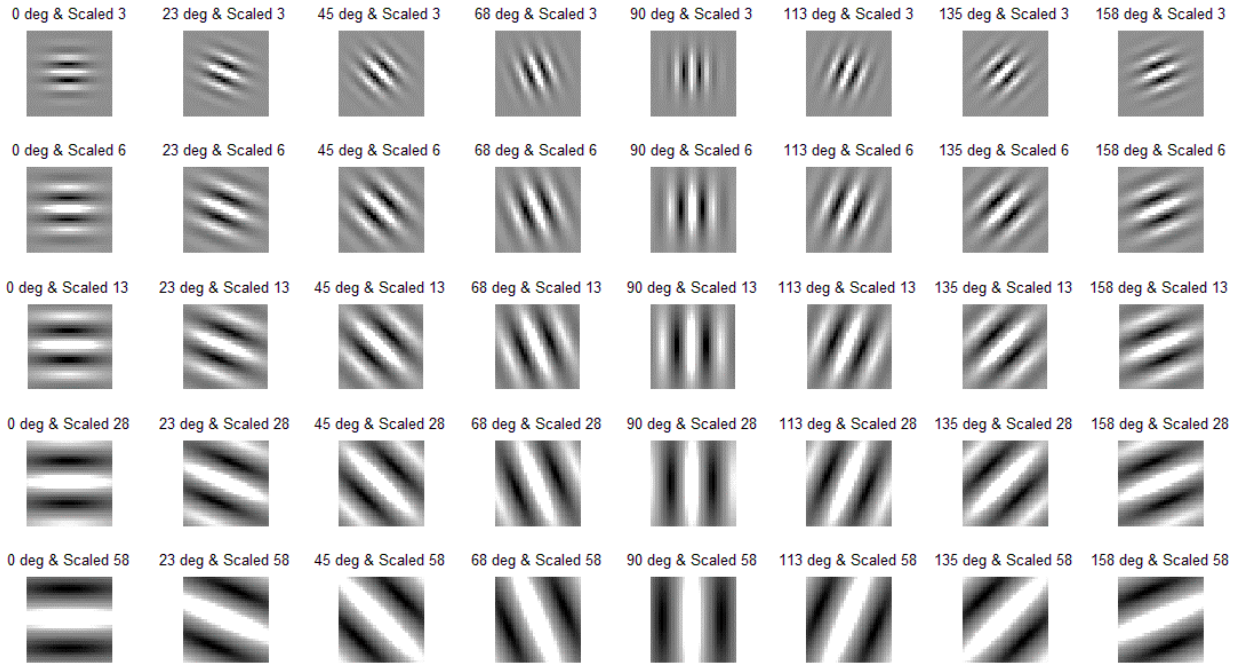

Figure 3. 8 Gabor wavelets created using two control parameters

Now it is need to convolve each filter with the region of interest of the image to get $8 \times 5 = 40$ different representation (response matrices) of the same image. Each filter region or image gives us a feature vector. The feature vector may consist of: average amplitude, maximum, minimum, range, local energy and etc. For instance: Local Energy obtained with summing up the squared value of each matrix value from a response matrix and Mean Amplitude obtained with sum of absolute values of each matrix value from a response matrix.

Thus, at the end we will get two matrices for real and imaginary parts that will be [1x40] each. Also, we can append one matrix to the other to create a [1x80] feature matrix for One image and thus create a [nx80] vector for n images for further training purpose.

After convolution we have 40 different representation like:

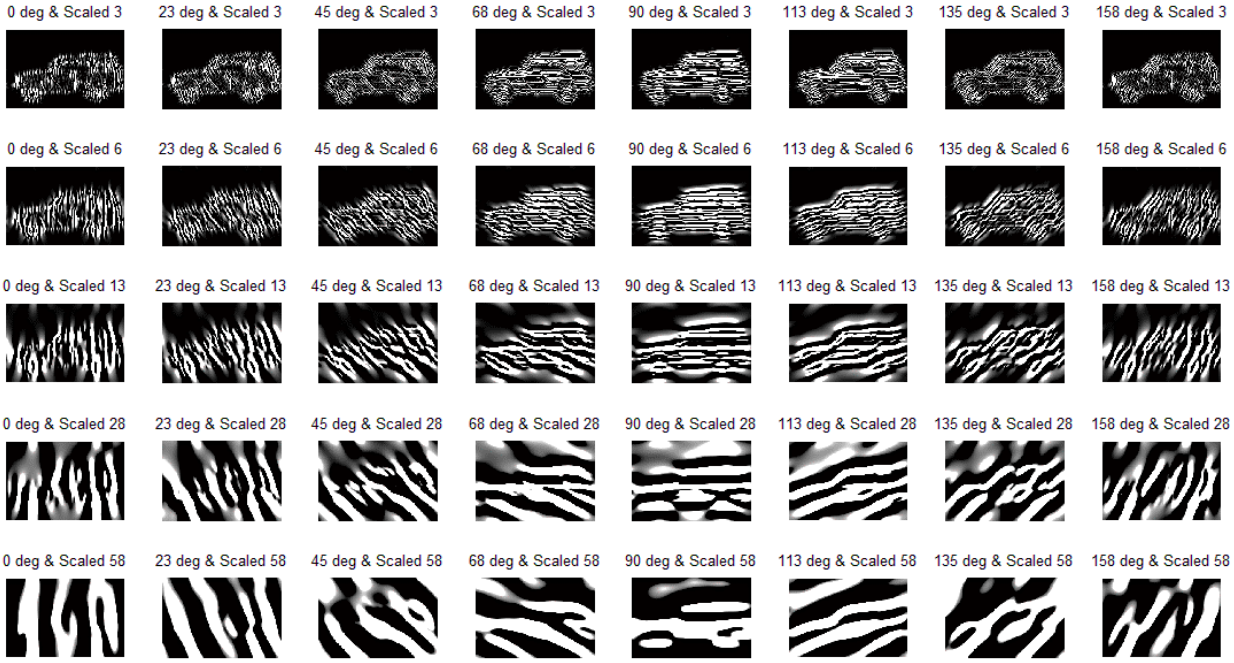

Figure 3. 9 convolution result of sample car image with Gabor wavelets

Chapter 4

The Face Recognition System

In this chapter our purpose is to train a machine using MATLAB which recognize faces in an imported custom image.

The training method is divided into three parts: Collecting necessary training data and make image dataset, creating feature vectors using a feature extraction method, and supervised training using an SVM for making a model to discriminate between face and non-face regions.

Figure 4. 1 Examples of face images used for training

Figure 4. 2 Examples of non-face images used for training

4.1 Image Dataset

The image dataset was constructed manually at the beginning of the project, by collecting images containing a human face from the Internet as positives and other samples as negatives in these images. Computing power limitations forced us to choose small images, the face and non-face images are in the size of 18 pixels in width and 27 pixels in height.

The final training dataset constructed contains 69 face images and 59 non-face images.

4.2 Training the SVM with Feature Vectors

In this project we used Gabor filters as feature extraction method. The output of the feature extraction process is used to train the SVM classifier. The radial basis function kernel is used as the kernel of the SVM.

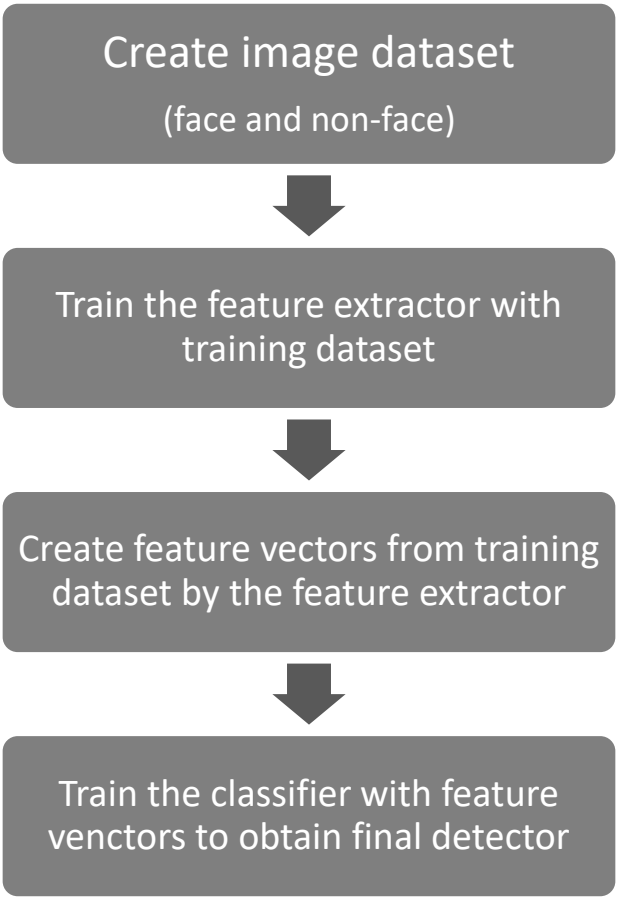

Figure 4. 3 Block diagram of the training algorithm

MATLAB code of program added to appendix.

4.3 Testing the face recognition system

After training phase, we should test the system. We collected group of images for test phase from internet. When we import custom images, program apply feature extraction method on it and then specify some yellow points on image which could contain faces. Then it applies trained SVM on image and change yellow points to red for non-face parts and yellow to green for face parts points. Every single point that gets recognized as part of a face and its color changes to green, eight points

surrounding it are nominated as possible face parts. The program will check these points too.

Test 01:

Figure 4. 4 Program Test Image 01

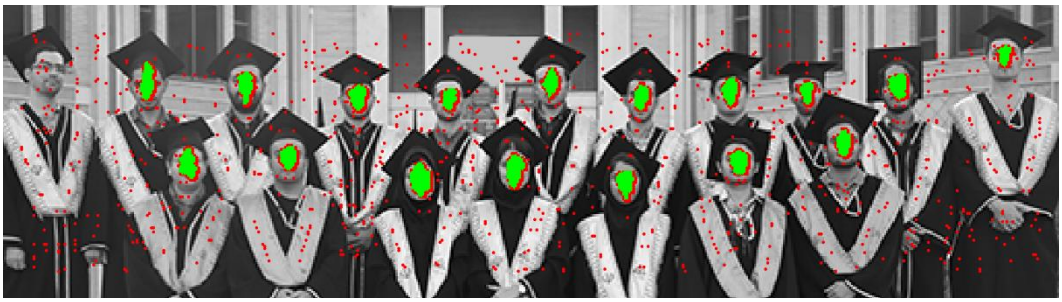

Figure 4. 5 Result of Program Test 01

In this test, we have an unrecognized face. This result may be obtained for the following reasons: alignment of his face, low number of training images of faces and non-faces, Gabor filter configuration and etc.

Test 02:

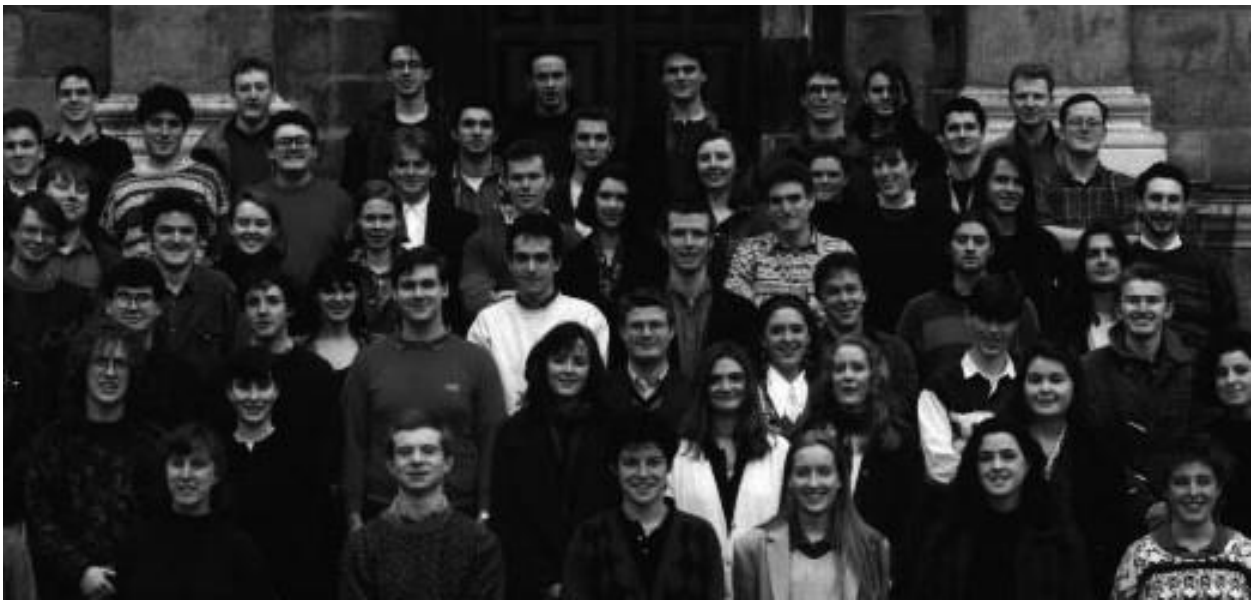

Figure 4. 6 Program Test Image 02

Figure 4. 7 Result of Program Test 02

In this test, all faces are recognized but some non-face parts determined as face parts. configuration of Gabor feature extraction or low number of images for training phase, may cause of this case.

Test 03:

Figure 4. 8 Program Test Image 03

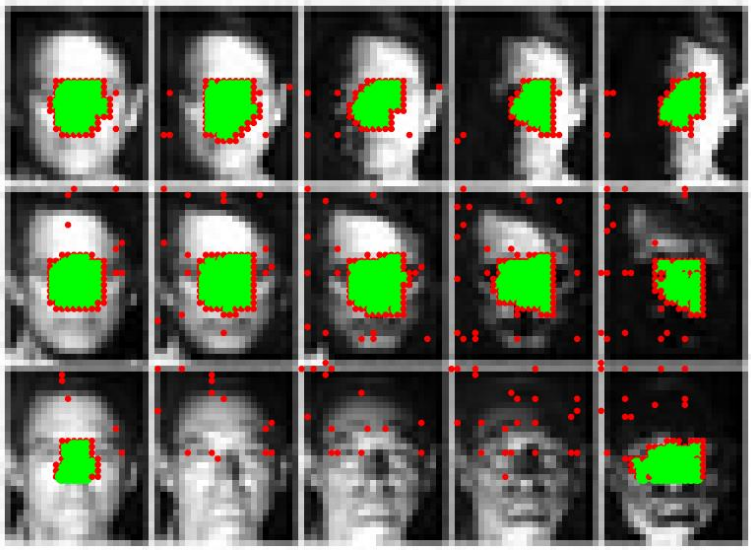

Figure 4. 9 Result of Program Test 03

In this test we have three different faces in five different light condition. First two faces are recognized in all conditions but the last one is unrecognized in three of them. configuration of Gabor feature extraction or low number of images for training phase, may cause of this case.

Chapter 5

Conclusion and Recommendation

In this face-recognition project we used Gabor filter as feature extraction method and SVM as classifier. We used 69 face images and 59 non-face images as image dataset to train our machine. Limitation on processing power forces us to use a low number image dataset in the size of 18 pixels in width and 23 pixels in height. We have test results of a machine with these parameters in chapter four. After analyzing the results, we conclude that the machine is about 88% accurate in face-recognition. Increasing number of training images would result in more accurate machines. the training images must be in different alignments and different light conditions to achieve higher accuracy.

We can test and compare results of machines with different configuration, use different feature extraction methods and other classification methods to find the best way to create a face-recognition machine and make it more accurate.

More accurate machines need more processing power and take more time to train, so we should make a tradeoff to create the optimal machine with our limitations and conditions.

Appendix

In this appendix we have MATLAB functions and scripts used in this project.

main.m file:

```
clear all;
close all;
clc;
warning off;

if ~exist('gabor.mat','file')
    fprintf ('Creating Gabor Filters ...');
    create_gabor;
end
if exist('imgdb.mat','file')
    load imgdb;
else
    IMGDB = loadimages;
end
if exist('net.mat','file')
    load net;
end
while (1==1)
    choice=menu('Face Detection',...
                'Create Dataset',...
                'Create SVM',...
                'Test on Photos',...
                'Exit');
    if (choice == 1)
        IMGDB = loadimages;
    end
    if (choice == 2)
        net = trainnet(IMGDB);
    end
    if (choice == 3)
        pause(0.01);
        [file_name, file_path] = uigetfile ('*.jpg');
        if file_path ~= 0
            im = imread ([file_path,file_name]);
            try
                im = rgb2gray(im);
            end
            tic
            imscan (net,im);
            toc
        end
    end
    if (choice == 4)
        clear all;
        clc;
        close all;
        return;
    end
end
```

Appendix. 1 main.m file

create_gabor.m file:

```
close all;
clear all;
clc;

G = cell(5,8);
for s = 1:5
    for j = 1:8
        G{s,j}=zeros(32,32);
    end
end
for s = 1:5
    for j = 1:8
        G{s,9-j} = gabor([32 32],(s-1),j-1,pi,sqrt(2),pi);
    end
end

figure;
for s = 1:5
    for j = 1:8
        subplot(5,8,(s-1)*8+j);
        imshow(real(G{s,j}),[]);
    end
end

for s = 1:5
    for j = 1:8
        G{s,j}=fft2(G{s,j});
    end
end
save gabor G
```

Appendix. 2 create_gabor.m

gabor.m file:

```
function Psi = gabor (w,nu,mu,Kmax,f,sig)
m = w(1);
n = w(2);
K = Kmax/f^nu * exp(1i*mu*pi/8);
Kreal = real(K);
Kimag = imag(K);
NK = Kreal^2+Kimag^2;
Psi = zeros(m,n);
for x = 1:m
    for y = 1:n
        Z = [x-m/2;y-n/2];
        Psi(x,y) = (sig^(-2))*exp((-.5)*NK*(Z(1)^2+Z(2)^2)/(sig^2))*...
                   (exp(1i*[Kreal Kimag]*Z)-exp(-(sig^2)/2));
    end
end
```

Appendix. 3 gabor.m

loadimages.m file:

```
function IMGDB = loadimages
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
face_folder = 'face/';  %LOCATION OF FACE IMAGES
non_face_folder = 'non-face/'; %LOCATION OF NON-FACE IMAGES
file_ext = '.png';
out_max = 1;  % DESIRED OUTPUT FOR DETECTING A FACE
out_min = 0; % DESIRED OUTPUT FOR NOT DETECTING A FACE
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
if exist('imgdb.mat','file')
    load imgdb;
else
    IMGDB = cell (3,[]);
end
fprintf ('Loading Faces ');
folder_content = dir ([face_folder,'*',file_ext]);
nface = size (folder_content,1);
for k=1:nface
    string = [face_folder,folder_content(k,1).name];
    image = imread(string);
    [m, n] = size(image);
    if (m~=27 || n~=18)
        continue;
    end
    f=0;
    for i=1:length(IMGDB)
        if strcmp(IMGDB{1,i},string)
            f=1;
        end
    end
    if f==1
        continue;
    end
    fprintf ('.');
    IM {1} = im2vec (image);    % ORIGINAL FACE IMAGE
    IM {2} = im2vec (fliplr(image));    % MIRROR OF THE FACE
    IM {3} = im2vec (circshift(image,1));
    IM {4} = im2vec (circshift(image,-1));
    IM {5} = im2vec (circshift(image,[0 1]));
    IM {6} = im2vec (circshift(image,[0 -1]));
    IM {7} = im2vec (circshift(fliplr(image),1));
    IM {8} = im2vec (circshift(fliplr(image),-1));
    IM {9} = im2vec (circshift(fliplr(image),[0 1]));
    IM {10} = im2vec (circshift(fliplr(image),[0 -1]));
    for i=1:10
        IMGDB {1,end+1}= string;
        IMGDB {2,end} = out_max;
        IMGDB (3,end) = {IM{i}};
    end
end
fprintf ('\nLoading non-faces \n');
folder_content = dir ([non_face_folder,'*',file_ext]);
nnface = size (folder_content,1);
for k=1:nnface
    string = [non_face_folder,folder_content(k,1).name];
    image = imread(string);
    [m, n] = size(image);
    if (m~=27 || n~=18)
        continue;
    end
    f=0;
    for i=1:length(IMGDB)
        if strcmp(IMGDB{1,i},string)
            f=1;
        end
    end
    if f==1
        continue;
    end
    fprintf ('.');
    IM {1} = im2vec (image);
    IM {2} = im2vec (fliplr(image));
    IM {3} = im2vec (flipud(image));
    IM {4} = im2vec (flipud(fliplr(image)));
    for i=1:4
        IMGDB {1,end+1}= string;
        IMGDB {2,end} = out_min;
        IMGDB (3,end) = {IM{i}};
    end
end
fprintf('Done.\n');
save imgdb IMGDB;
```

Appendix. 4 loadimages.m

trainnet.m file:

```
function NET = trainnet(IMGDB)

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
options = optimset('maxiter',100000);
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fprintf('Creating & training the machine ->\n');

T = cell2mat(IMGDB(2,:));
P = cell2mat(IMGDB(3,:));
net = svmtrain(P',T','Kernel_Function','linear','Polyorder',2,'quadprog_opts',options);
fprintf('Number of Support Vectors: %d\n',size(net.SupportVectors,1));
classes = svmclassify(net,P');
fprintf('done. %d \n');
save net net
NET = net;
```

Appendix. 5 trainnet.m

im2vec.m file:

```
function IMVECTOR = im2vec (W27x18)

load gabor;
W27x18 = adapthisteq(W27x18,'Numtiles',[8 3]);
Features135x144 = cell(5,8);
for s = 1:5
    for j = 1:8
        Features135x144{s,j} = abs(ifft2(G{s,j}.*fft2(double(W27x18),32,32),27,18));
    end
end
Features45x48 = cell2mat(Features135x144);
IMVECTOR = reshape (Features45x48,[19440 1]);
```

Appendix. 6 im2vec.m

imagescan.m file:

```
function imscan (net,im)
close all
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% PARAMETERS
SCAN_FOLDER = 'imscan/';
UT_FOLDER = 'imscan/under-thresh/';
TEMPLATE1 = 'template1.png';
TEMPLATE2 = 'template2.png';
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
warning off;
delete ([UT_FOLDER,'*.*']);
delete ([SCAN_FOLDER,'*.*']);
[m, n]=size(im);
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% First Section
C1 = mminmax(double(im));
C2 = mminmax(double(imread (TEMPLATE1)));
C3 = mminmax(double(imread (TEMPLATE2)));
Corr_1 = double(conv2 (C1,C2,'same'));
Corr_2 = double(conv2 (C1,C3,'same'));
Cell.state = int8(imregionalmax(Corr_1) | imregionalmax(Corr_2));
Cell.state(1:13,:)=-1;
Cell.state(end-13:end,:)=-1;
Cell.state(:,1:9)=-1;
Cell.state(:,end-9:end)=-1;
Cell.net = ones(m,n)*-1;
[LUTm, LUTn]= find(Cell.state == 1);
imshow(im);
hold on
plot(LUTn,LUTm,'.y');pause(0.01);
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Second Section
xo = [-1,-1,-1, 0, 0,+1,+1,+1]; %offset
yo = [-1, 0,+1,-1,+1,-1, 0,+1]; %offset
while (1==1)
    [y, x] = find(Cell.state==1,1);
    if isempty(y)
        break;
    end
    imcut = im(y-13:y+13,x-9:x+8);
    Cell.state(y,x) = -1;
    Cell.net(y,x) = svmclassify(net,im2vec(imcut)');
    if (Cell.net(y,x) == 1)
        plot(x,y,'.g','MarkerSize',15);pause(0.01);
        for k=1:8
            nx = x + xo(k);
            ny = y + yo(k);
            if (Cell.state(ny,nx) ~= -1)
                Cell.state(ny,nx) = 1;
            end
        end
    else
        plot(x,y,'.r');pause(0.01);
    end
end

function output = mminmax(input)

max_ = max(max(input));
min_ = min(min(input));

output = ((input-min )/(max -min ) - 0.5) * 2;
```

Appendix. 7 imagescan.m

References

1. Freund, Y. and Schapire, R. E. (1999). A short introduction to boosting.
2. Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey.
3. Cootes, T. F., Edwards, G. J., and Taylor, C. J. (1998). Active Appearance Models.
4. Castrillón-Santana, M., Déniz-Suárez, O., Antón-Canalís, L., and Lorenzo Navarro, J. (2008a). Face and Facial Feature Detection Evaluation.
5. Freund, Y. and Schapire, R. E. (1995). A decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.
6. Vapnik, V. (1998). Statistical Learning Theory. Wiley.
7. Cristianini, N. and Shawe-Taylor, J. (2010). An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.
8. Geebelen, D., Suykens, J., and Vandewalle, J. (2012). Reducing the Number of Support Vectors of SVM Classifiers Using the Smoothed Separable Case Approximation.
9. Joachims, T. (1999). Making large-Scale SVM Learning Practical.
10. Boser, B. E., Guyon, I., and Vapnik, V. (1992). A Training Algorithm for Optimal Margin Classifiers.