

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

Evaluation Of Appearance Based
Methods For Facial Expression
Recognition.

By
Abhiram Ganesh.

Supervisors:

Prof. dr. Henk Corporaal (TU/e, NL)

Prof. dr. J.J. Lukkien (TU/e, NL)

Prof. dr. Manahora M. Pai (MU, India)

dr. Anteneh. A. Abbo (Philips)

Ir. Vincent Jeanne (Philips)

Eindhoven, August 2008

Acknowledgements

I am greatly indebted to Professor Dr. Henk Corporaal, Professor Dr. J.J. Lukkien, Dr. Anteneh. A. Abbo (Philips) and Vincent Jeanne (Philips), Professor Dr. Manohara Pai M.M. who have provided guidance me, advice and the outlook that were needed for my development as a master student. I would also like to thank TU/e, Manipal University and Philips (Eindhoven) for giving me this opportunity to do my master program and the thesis.

I consider my self lucky to have worked with such great persons whose knowledge and experience in the field of computer vision has benefited me in my work and has helped me grow in understanding the field and made my approach towards the field more broader. I was also encouraged and supported to continue and perform good work by my friends at the university and at work and for this I am really grateful to them. Last but not the least I would like to thanks my parents for their love, encouragement and support. I will always respect and love them.

Abstract

Analyzing the valence of emotion in human face images although is a trivial task for humans, the same process involves a lot of constraints for computer programs. Some of these constraints are due to the noise introduced by digital capturing devices such as cameras, while others include illumination variations and pose variations in the captured images. A number of approaches have been suggested [1],[2],[3] for detecting the presence of human faces in images which is also the first step in the process of analyzing the valence of emotions in the detected face images. The most important problem here is one that involves a way to represent most important characteristics that go into defining the face image. These important characteristics are defined as the feature vectors of the image. Most of these approaches present a good representation of face images and few have adopted a learning procedure to classify these features into face and non face classifications [1],[4]. Most of these approaches classify a single image and use the information extracted from single image (feature extraction) for classification purposes. Similar kind of work has been done in the area of speech recognition. A major difference in the approaches in two areas is that speech recognition includes temporal changes in the classification of feature vectors. One such significant approach is the one involving Hidden Markov Models.

In the current work we investigate the couple of appearance based approaches namely the Local Binary patterns [2], and the optical flow motion estimation [5] as a part of obtaining the feature vectors. This process is followed by process of clustering the feature vectors for which we use self organizing maps [6]. We then model a Markov model to identify single emotion changes in a video sequence. The emotion changes are typically of the form neutral emotion to six other classes of emotion viz. happy, sad, angry, disgusted surprise and fear. We apply the feature vectors obtained using both the afore-mentioned methods to train a Hidden Markov Model and then use this trained model for further emotion detection.

List of Tables

Table 8.1 Neutral to Angry success rates. The number of hidden states is 20.	45
Table 8.2 Neutral to Disgust success rates. The number of hidden states is 20.	45
Table 8.3 Neutral to Fear success rates. The number of hidden states is 20.	46
Table 8.4 Neutral to Happy success rates. The number of hidden states is 20.	46
Table 8.5 Neutral to Sad success rates. The number of hidden states is 20.	46
Table 8.6 Neutral to Surprise success rates. The number of hidden states is 20.	46
Table 8.7 Neutral to Angry success rates with combined centroids.	47
Table 8.8 Neutral to Disgust success rates with combined centroids.	47
Table 8.9 Neutral to Fear success rates with combined centroids.	47
Table 8.10 Neutral to Happy success rates with combined centroids.	47
Table 8.11 Neutral to Sad success rates with combined centroids.	47
Table 8.12 Neutral to Surprise success rates with combined centroids.	48
Table 8.13 Results with 25 Hidden States, 50 Observable States.	48
Table 8.14 Results with 15 Hidden States, 30 Observable States.	49
Table 8.15 Results with 10 Hidden States, 20 Observable States.	49

List of Figures

Figure 2.1 Training Phase in General Emotion Recognition Process	6
Figure 2.2 Testing Phase in General Emotion Recognition Process.....	8
Figure 4.1 LBP Operator	11
Figure 4.2 Computed neighbourhood of pixels with different radius R and different sizes of neighbourhood N	12
Figure 4.3 Structural features that can be detected using Uniform LBP.....	15
Figure 4.4 Face image portioning and vector computation.....	16
Figure 4.5 Weight assignment for different regions	17
Figure 5.1 Training phase in the proposed emotion recognition approach	21
Figure 5.2 Testing phase in the proposed emotion recognition approach.....	22
Figure 6.1 Aperture Problem: Motion direction orthogonal to intensity gradient	25
Figure 6.2 Partial derivative estimation in Horn-Schunck approach	27
Figure 6.3 Optical Flow estimation using Horn-Schunck approach.....	29
Figure 6.4 Generating regions wise histogram of gradients.	30
Figure 6.5 General structure needed for SOM.....	31
Figure: 7.1 Example of Hidden Markov Model.....	37
Figure 7.2 Training with HMMs.....	39
Figure 7.3 Example of the structure of the HMMs to be trained.	40
Figure 7.4 Testing with HMMS.....	41
Figure 7.5 Combined centroid computation	42
Figure 7.6 Feature vector to symbol mapping in alternate approach.....	42
Figure 7.7 Training HMMs in the alternate approach.....	43
Figure 7.8 Feature extraction and mapping to symbol for a sample/test image sequence.....	43
Figure 7.9 Testing with HMMs in alternate approach	44

Abbreviations

PCA	Principal Component Analysis
LDA	Linear Discriminant Analysis
LBP	Local Binary Operator
OFE	Optical Flow Estimation
HOG	Histogram of Gradients
HMM	Hidden Markov Model
SOM	Self Organizing Maps

Table of Contents

Acknowledgements	i
Abstract	ii
List of Tables	iii
List of Figures	iv
Abbreviations	v
1. Introduction	3
1.1. General	3
1.2. Related works	4
2. Emotion Recognition Process	6
2.1. Training Phase	6
2.2. Testing	7
3. Feature Extraction Methods	9
3.1. Principal Component Analysis	9
3.2. Linear Discriminant Analysis	10
4. Local Binary Patterns	11
4.1. Description of LBP Operator	11
4.2. Uniform Local Binary Patterns	14
4.3. Emotion Recognition using LBP	15
4.3.1. Feature Extraction using LBP	16
4.3.2. Testing with LBP	17
5. Proposed Emotion Recognition Approach	20
5.1. Training Phase	20
5.2. Testing Phase	21
6. Feature Extraction in Proposed Approach	23
6.1. Optical Flow Estimation	23
6.1.1. Horn and Schunck's Method	23
6.1.1.1. Assumptions	24
6.1.1.2. Constraints	24
6.1.1.2.1. Constant Brightness Constraint	24
6.1.1.2.2. Smoothness Constraint	26
6.1.1.3. Estimation of partial derivatives and Laplacians	26
6.1.1.3.1. Partial Derivatives	26
6.1.1.3.2. Laplacians	28
6.1.1.4. Iterative Solution	28
6.1.1.5. Experimental setup	29
6.2. Histogram of Gradients	29
6.3. Self Organizing Maps	31
6.3.1. Inputs and structures used in SOMs	32
6.3.2. The Learning Algorithm	32
6.3.3. Computing the best matching weight column and neighbourhood	33
6.3.4. Adjusting the weights	34
6.3.5. Centroid computation from Weights and mapping to symbols	35
6.3.6. Advantages	35
6.3.7. Disadvantages	35
7. Hidden Markov Model	36

7.1.	Introduction	36
7.2.	The three basic problems.....	37
7.3.	Experimental Setup.....	38
7.4.	Training with HMM.....	38
7.5.	Testing with HMM	40
7.6.	Alternative approach with HMMs	41
8.	Results, Suggestions and Conclusion	45
8.1.	Results.....	45
8.2.	Suggestions.....	49
8.3.	Conclusion.....	50
9.	References.....	51

1. Introduction

1.1. General

Human faces provide cues about the emotional state of an individual and play an important role in human to human nonverbal communication. This is possible but not limited through facial expressions. The nonverbal communication through expressions is of great significance in normal day to day interaction among humans as it compliments verbal communication. Facial expressions form a powerful source of information in nonverbal communication such as emotional state of individual [7] and can provide an indication of intent [8] or help in regulating behaviours [9]. On the other hand the interaction between computers and humans is limited to speech, textual and to some extent visual. To utilize the information provided by facial expressions and to improve the communication between humans and computers, it is critical to develop a computer vision system that is reliable, correct and efficiently bridges the communication gap between computers and humans.

The main focus in computer vision approaches that deal with facial expression analysis is on classifying emotions into a very small set [10] [11] [12] [13] [14] [15]. The earliest work done using this approach is one of Darwin's [16]. A more recent work based on the same approach is the one by Ekman[7]. According to his work emotions can be classified into six basic emotions viz. anger, disgust, fear, happy/Joy, sadness and surprise. On the other hand a manual system was developed by psychologists called the Facial Action Coding System (FACS) which uses discrete movements of the face (called Action units or AUs) to code facial expressions. A single AU or a combination of AUs can represent various human emotions. Although humans can produce many different facial expressions with varying complexity, intensities and with different intended meaning, as initial steps the focus should be on developing a computer vision system that can at least discriminate between the aforementioned six basic emotions.

1.2. Related works

A large number of researchers in the field of computer vision have developed different techniques in order to make it possible to automatically recognize facial expression. We will now briefly review some of these major paradigms.

Principal component analysis (PCA) has been used for several purposes including face recognition [3], expression recognition of forehead regions and brows [17], object recognition with varying poses and illumination [18]. The method has also been used along with optical flow estimation for mouth motion and smile recognition [11] and lip-reading [19].

The approach with optical flow and PCA for lip reading [19] uses template matching to minimize the sum of squared differences between the flow curve of sample word and the flow curve of word templates. The approach with optical flow and PCA for mouth motion and smile recognition [11] uses a similar approach. The idea in this case is to find the similarities between the parameters of linear combinations in PCA of the sample and training image sequences.

Kobayashi and Hara [20][21][22] developed an approach to recognize six basic facial expressions, mixed facial expressions and their intensities. The approach was developed using three sets of artificial neural networks. Sixty facial feature points were selected manually that formed the inputs to the neural networks. For the six basic facial expressions the approach scored a recognition rate 88.7%. The tests were performed on fifteen subjects. For the mixed facial expression the approach scored a recognition rate of 70%. The tests in this case were performed using ten subjects.

Another approach by Sejnowski, Barlett, Golomb, Viola, Larsen, Hager, and Ekman [17] proposed an approach that involved three methods and artificial neural networks. The aim of the approach was to recognize upper facial expression. The three methods were PCA of difference images, optical flow along with correlation coefficients and high gradient component (wrinkle detection). The best result from their approach was reported to have a recognition rate of 91%. The tests involved 80 image sequences of 20 subjects totalling up to 400 images.

Another approach was developed by Lien, Kanade, Zlochower, Cohn, and Li [23] which used two different methods to recognize facial action units or a combination of facial action units using Hidden Markov Models. The two methods were facial feature point tracking and pixel-wise flow tracking along with PCA. The approach was

devised to recognize upper and lower face expressions. The recognition rates were 93% for lower face expressions and 91% for upper face expressions.

All of the mentioned approaches are related to the current work. They also provide important references to the current work. In the current work we evaluate the use of some appearance based approaches viz. Local Binary Patterns (LBP) and Optical Flow Estimation (OFE) for the purpose of expression recognition. We evaluate the approach that employs LBP and is suitable to be implemented on embedded platforms. The optical flow estimation is coupled with histogram generation of oriented gradients, Self Organizing Maps (SOM) and Hidden Markov Models (HMM). The key difference between our approach and the one by Lien, Kanade, Zlochow, Cohn, and Li is the combination of dense optical flow estimation, region wise gradient histograms, SOM and HMMS.

2. Emotion Recognition Process

In general the steps involved in an emotion recognition process can be broadly classified into two phase viz. training and testing. The purpose of the training phase is to use a huge collection of images from an image database and compute different models that reflect the properties of different classes of emotions. These properties are values that represent key features of the image and are also called the feature vector of the image. The purpose of testing phase is to test the classification produced by the models computed in the training phase.

2.1. Training Phase

The general steps involved in the training phase are shown in figure 2.1 below. The input to the training phase is a collection of images showing human faces cropped to show only the face region. These images are also called as face images. These face images are then passed through a feature extraction step. In the feature extraction step key attributes of the images are computed and stored as a vector called the feature vector. These feature vectors define or represent the most important properties observed in the face image such as an edge or point. There are two advantages of this step. The first advantage is that the size of the data is reduced from the entire image to only a few selected important features. The second advantage is that this selection of features gives more structured information than just basic pixel values of the images. Thus the feature vectors can be considered as the minimal set that is adequate to represent the face image.

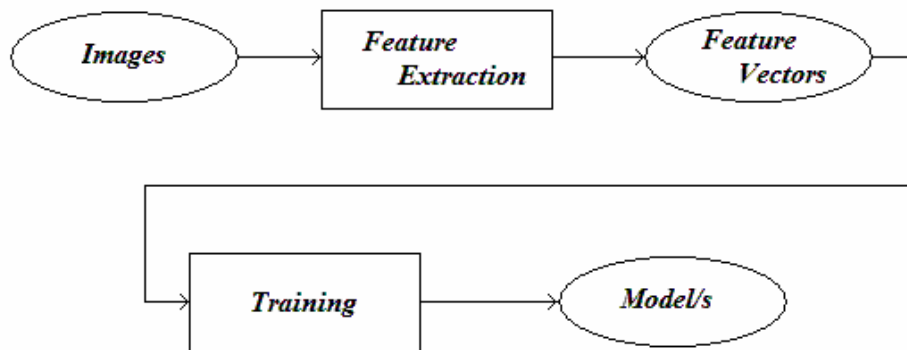


Figure 2.1 Training Phase in General Emotion Recognition Process

The next step in this phase is the training step. In the training step the key idea is to build a model that closely represents the values of the feature vectors for a given class of face images. The training can be done through a variety of learning algorithms including but not limited to boosting algorithms [4], algorithms based on support vector machines or algorithm based on neural networks. The training could be done on face images showing a selected class of emotions or the entire set of emotions. If the training is done for a selected class of emotions then a model is built for each class of emotions. Hence in the case of six basic emotions mentioned earlier a separate model will be built for each of six emotions. The input images for a particular model will be only the images that show the corresponding emotion. If the training is desired for building a single model for the entire set of emotions then the entire set of face images is used for the training phase.

2.2. Testing

The general steps involved in the testing phase are shown in figure 2.2 below. The inputs to this phase are the models that were built during the training phase and the test images for which the emotions are to be recognized. Here again only the face region is used as rest of the image do not contribute information about the expression/emotion. In a typical real time scenario the input image would be the detected face image from an earlier face detection phase. The first step here again would be a feature extraction phase where the key features from the face image are extracted. The extraction method must be same as the one used in the training phase. The output of this step is the feature vector of the face image that would then be subjected to a testing step. In the testing step the feature vector is tested against the models built during the training phase. The output of the testing step is a score that indicates the emotion that is detected by the model. This score is usually in the form of distance or probability and it defines which model was best suited for the feature vector extracted in the previous step. In the testing step there are two possible ways that can be employed. The first possibility is in the case when one model was built per class of emotion. Here the feature vector is tested against all the models and their scores then define which model was the most suited one. The second possibility is the case when only one model was built for the entire set and a single score defines the possible emotion detected.

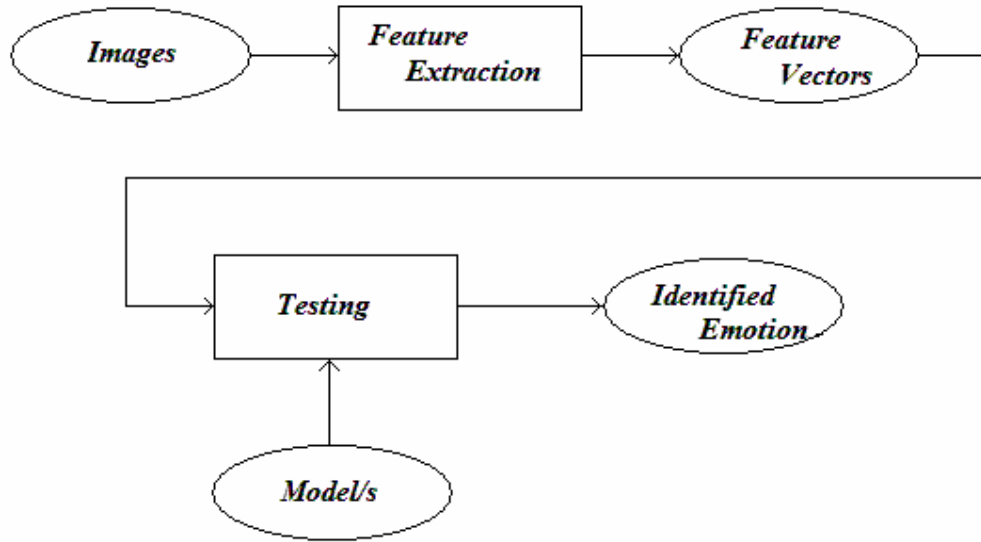


Figure 2.2 Testing Phase in General Emotion Recognition Process

During testing a sample image the approach can correctly classify it as the correct emotion expressed. In another case the approach can also wrongly classifies a sample image as the correct emotion expressed. These cases constitute false positives. It could also be the case that the approach wrongly classifies a sample image as incorrect emotion expressed. These cases constitute false negatives.

3. Feature Extraction Methods

Over the last couple of years a number of methods were developed for face recognition systems including Principal Component Analysis(PCA)[3], Linear Discriminant Analysis(LDA)[26], Local Binary Patterns(LBP)[2] and Optical Flow[5]. The same methods can be used for the feature extraction steps for the emotion recognition process as well. In this chapter we will briefly describe the first two methods. The next chapters will have a more detailed description of the LBP and Optical Flow methods.

3.1. Principal Component Analysis

Principal component analysis or PCA is a statistical method and is one of the most widely used results from linear algebra. The method is generally used in a scenario which involves high dimensionality of data and there is a desire to obtain the most important information from this data. Since this scenario closely resembles the case with feature extraction step involved in face detection/emotion recognition systems, a number of face detection/recognition and emotion recognition approaches have employed this method for the feature extraction step. The key idea in PCA is to transform the high dimensional data to a new coordinate system of lower dimensions while still preserving the most important information. This is done by computing a covariance matrix and a set of values called the eigenvalues and eigenvectors from the original data. The covariance matrix consists of values that measure the degree of linear relationship between the dimensions (variance). The dimensionality can be reduced by retaining only those characteristics of the data which contribute most to its variance. The key idea is that upon projection on to the new coordinate system the first principal component lies on the first coordinates the second principal component lies on the second coordinate and so on. Thus by selecting only a few eigenvectors (principal components) from the data a reduced data set is obtained while still retaining the most important information.

In the case of face recognition/detection the dimensions of data can be considered as the number pixels. The covariance matrix and the eigenvectors of this covariance matrix is then calculated. The reduction in dimensions is done by selecting only the most important eigenvectors and discarding the rest. The projection on to the new

coordinate system reduces the dimensionality of the input image as well as retains the key features. Also the features that correspond to lesser important characteristics of input image such as background which are not relevant to expression or face recognition/detection analysis are filtered out. This makes the measure of the analysis more reliable and the reduced dimensionality reduces the cost of analysis.

3.2. Linear Discriminant Analysis

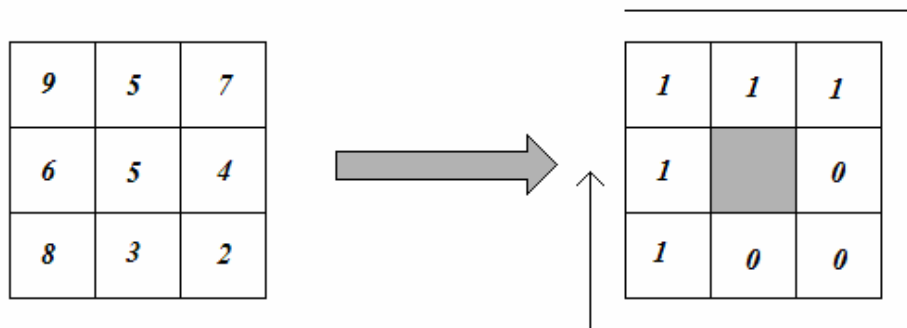
Linear discriminant analysis or LDA is a statistical method that is used to classify a set of objects into groups. The grouping is done by observing a set of features that describe the object. The process is similar to principal component analysis in that both approaches work on establishing a linear relationship between the dimensions of the data. The key difference between the two approaches is that LDA uses the linear relationship to model the differences into classes of objects, where as PCA does not take into account any differences in the linear relationship. In general LDA tries to find relationship that maximizes the differences in the dimensions of image data from different classes and the differences between the images of the same class are minimized. The key idea in LDA is to perform a linear transformation on the data to produce a lower dimensional set of features. The linear transformation is such that these lower dimensional set of features describe the group they belong to in the best possible way. This is possible since LDA tries to reduce the differences among the set of feature within the same group and increases the differences between set of features from different groups. In the most ideal case LDA tries to find a projection that completely separates one group from the other. A new object can then be classified by using a linear combination of a set of features obtained from the object and assign the object to a group that produces the highest conditional probability for the given object. In face recognition system we can use a huge collection of face images of different subjects and form classes of set of features. LDA will try and minimize the differences in the set of features for the face images of the same person and increase the differences among set of features from face images of different persons. This will also be the case in the case of classifying a new face image among different groups. This is possible since the reduced set of features from the new face image after linear transformation would have fewer differences with the set features of the correct class and more differences with the set of features of a different class.

4. Local Binary Patterns

Local Binary Patterns or LBP was introduced by Ojala et al [24] and can be used to describe the texture and shape of digital image. With local Binary patterns it is possible to determine some information about the neighbourhood of a particular pixel. This is because an LBP code is produced by comparing the intensity value of a pixel with those of its neighbours. In this section we will describe Local Binary Patterns in detail and along with the capability to describe texture and shape of an image.

4.1. Description of LBP Operator

The LBP operator for a pixel was originally designed to work with a neighbourhood of eight neighbouring pixels. The operator is simple and just involves comparison of the intensity value of the central pixel with the intensity values of the neighbouring pixels. If the intensity value of the neighbouring is greater than or equal to the central pixel then a '1' assigned for the corresponding bit value in an eight bit binary code for the central pixel. If on the other hand if the intensity value of the neighbouring pixel is less than the central pixel then a '0' is assigned for the corresponding bit value in an eight bit binary code for the central pixel. An example of the operator functionality is shown in figure 4.1 shown below.



LBP code: 11101100

Decimal: 236

*(a) Intensity values
of nine pixels*

*(b) corresponding LBP assignment
for central pixel*

Figure 4.1 LBP Operator

The local binary patterns can be extended to neighbourhood of various sizes other than eight neighbouring pixels. In this case the pixels that are included for comparison with the central pixel are calculated by a circle in the following way.

Let N represents the number of pixels involved in the comparison. Let the central pixel be represented by C and have coordinates (x_c, y_c) . Let R represent the distance between the central pixel and each of the N neighbouring pixels. Then the coordinates of N neighbouring pixels are computed by the following formula.

$$\begin{aligned} x &= x_c + R \cos(2\pi n / N) \\ y &= y_c + R \sin(2\pi n / N) \end{aligned} \quad (4.1)$$

The figure 4.2 below shows the computed pixels for LBP calculation with varying radius(R) sizes and number of pixels (N) involved. Since according to the formula above certain coordinates may not fall exactly on a single pixel it is suggested to use a bilinear interpolation of neighbouring pixel intensity values for the comparison. For a pixel p the corresponding LBP notation used is $LBP_{N,R}(p)$.

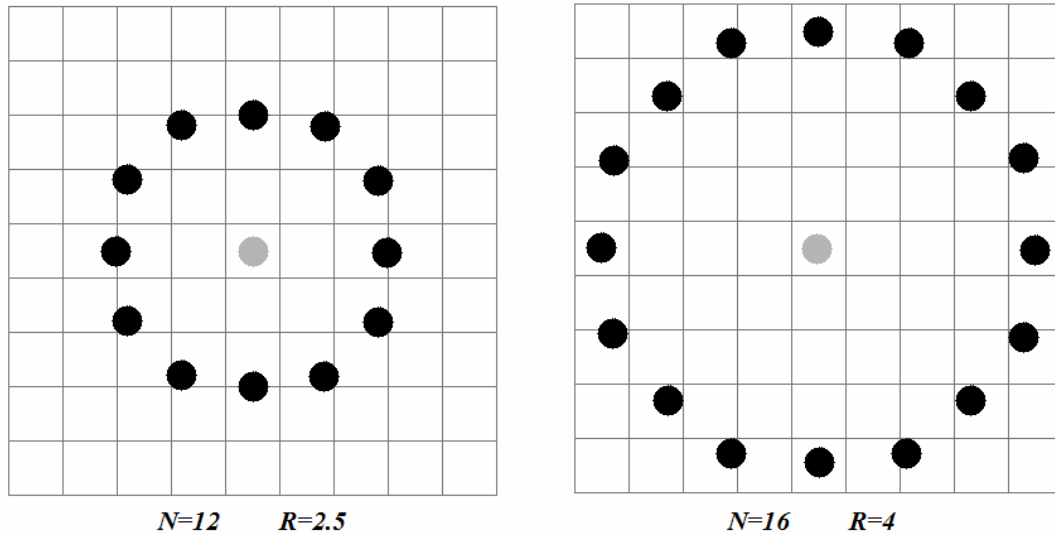


Figure 4.2 Computed neighbourhood of pixels with different radius R and different sizes of neighbourhood N

The LBP operation can be then summarized as follows. If I_C represents the intensity value of the central pixel and if the intensity values of the N neighbouring pixels is

given by I_n for $n = 0, 1, \dots, N$ then the texture T in the local neighbourhood of C can be represented as:

$$T = t(I_C, I_o, I_1, \dots, I_{N-1}) \quad (4.2)$$

This texture can also be represented in an alternative way. For the alternative way the intensity value of the central pixel C must be subtracted from the pixel values of the computed neighbourhood values. In this alternative approach the texture T in the local neighbourhood of C can then be represented as a combination of the intensity value of the central pixel i.e. I_C and the differences of intensity values of the neighbouring pixels with respect to the central pixel C .

$$T = t(I_C, I_o - I_C, I_1 - I_C, \dots, I_{N-1} - I_C) \quad (4.3)$$

According to Ojala et al. the texture component I_C represents the overall luminance of the pixel at C and is not related to the local texture of the neighbourhood. Thus this component does not provide much useful information to texture analysis. Thus even after dropping this component from the texture description T much of the texture information is still retained by the following expression.

$$T \approx t(I_o - I_C, I_1 - I_C, \dots, I_{N-1} - I_C) \quad (4.4)$$

Although the above description of texture is invariant against shifts in the intensity values it is still not invariant against scaling of intensity values. To achieve invariance against such scaling only the signs of the differences are considered for texture description and not the actual differences. Thus the new texture description can be given as follows

$$T \approx t(s(I_o - I_C), s(I_1 - I_C), \dots, s(I_{N-1} - I_C)) \quad (4.5)$$

Where,

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (4.6)$$

Thus for pixels whose intensity value is greater than or equal to the intensity value of the central pixel the sign assigned would be 1 else a 0 would be assigned as the sign. As a last step for the LBP calculation of the central pixel C a binomial weight is assigned to the sign. The binomial weights are summed to produce the LBP code for the central pixel C as follows.

$$LBP_{N,R}(C) = \sum_{n=0}^{N-1} s(I_n - I_C) \cdot 2^n \quad (4.7)$$

The LBP operator in figure 3 can be calculated using the above formula where $N=8$ and $R=1$. The only difference with respect to other LBP code calculation is that the intensity values of neighbouring pixels must be interpolated before the calculation in case these neighbours do not fall at the center of a pixel. LBP calculated by the above formula can produce a large number of patterns and thus do not help in reducing the size of data. In order to efficiently use local binary patterns for image processing the use of uniform local binary patterns is suggested. The details of uniform local binary patterns are given in the subsection below.

4.2. Uniform Local Binary Patterns

A local binary pattern is called uniform if and only if the pattern contains two or less bitwise transitions from a 0 to 1 or vice versa. In other words the local binary pattern should contain zero or two transitions. A single transition is not possible since the patterns are considered in a circular fashion. There are only two patterns that have zero transitions i.e. the pattern with all zeros and the pattern with all ones. If the LBP code was generated using N number of neighbours then there are in all $N(N-1)$ patterns with two transitions. The uniform local binary patterns with a radius of R and neighbourhood of N pixels the notation used is $LBP_{N,R}^{u2}$.

There are two advantages in using uniform local binary patterns, the first being a reduction in the range of possible patterns. For a uniform local binary pattern of bit-length of eight bits the possible number of patterns is $8(8-1) + 2 = 58$ patterns. For non-uniform local binary patterns of the same bit-length the possible number of patterns is $2^8 = 256$ patterns.

The second advantage of using uniform local binary patterns is that these patterns can still discriminate between various structural features present in the data even with reduced dimensionality. The structural features that can be detected by uniform local binary patterns are spot, spot/flat, line end, edge and corner. These patterns are shown in figure 4.3 below.

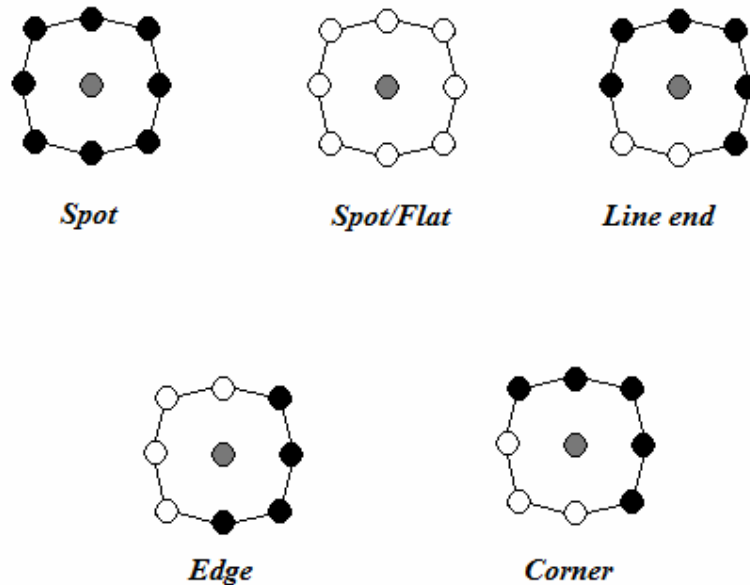


Figure 4.3 Structural features that can be detected using Uniform LBP

4.3. Emotion Recognition using LBP

The following subsection describes the procedure that was used for recognizing emotion expressed by a face image using Uniform Local Binary Patterns. The key idea is to extract a set of feature vectors using LBP as the operator. These feature vectors are then trained to obtain a model for each of the six basic emotion class mentioned in earlier section. To evaluate the emotion expressed by a sample face image the same LBP operator is used to extract a feature vector. This feature vector extracted from the sample face image is then tested against the models built for the six basic emotions in the training phase.

4.3.1. Feature Extraction using LBP.

The entire face image is partitioned into $7 \times 6 = 42$ regions as shown in figure 4.4 below. This partitioning is well suited for emotion recognition as different regions show different features of a human face. The dimensions of the face image are kept constant (108x147) for all the images. Then the $LBP_{8,2}^{u,2}$ code is computed for every pixel of the face image. After the LBP code is computed these codes are then grouped region-wise into 59 different bins. Of these bins $8(8-1) = 56$ bins represent values for uniform LBP with two transitions. Two other bins represent values for uniform LBP of the type all zeros and all ones. The remaining non-uniform LBP are all clubbed into one bin. Once these bins are calculated they are concatenated column wise as shown in the figure. Thus the final vector consists of $59 \times 42 = 2478$ rows and single column. This vector forms the feature vector of given image.

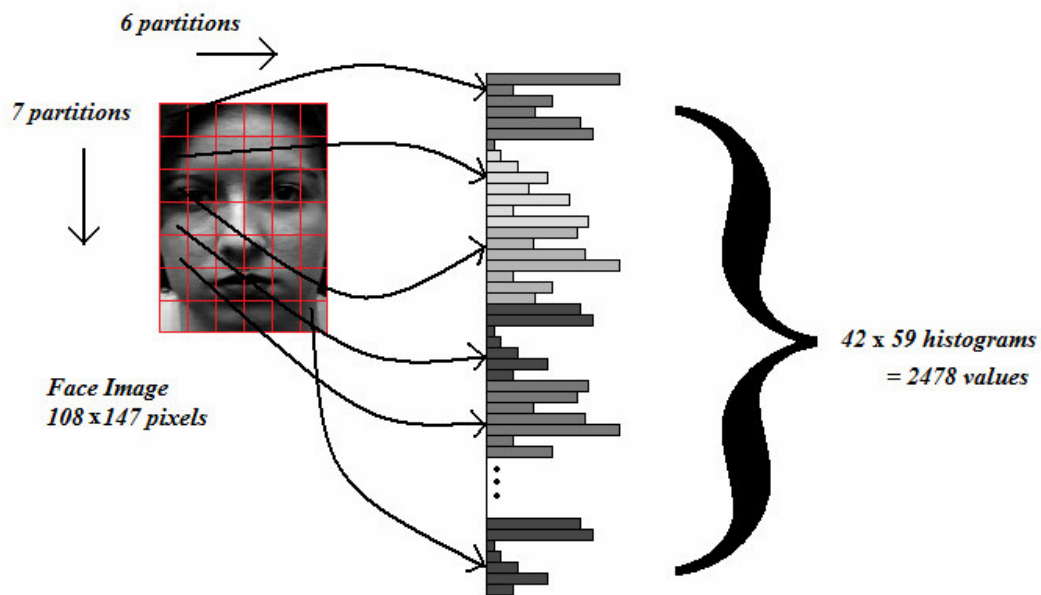


Figure 4.4 Face image partitioning and vector computation

Further observations lead to the fact that all regions do not contribute equally to the emotion expressed by a face image. Hence there is a need to multiply the histogram bins of each region with suitable corresponding weights. The assignment of weights to corresponding regions is as shown by figure 4.5.

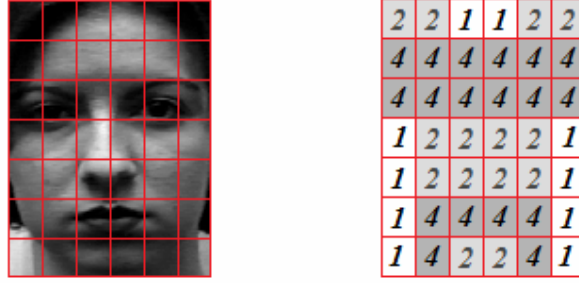


Figure 4.5 Weight assignment for different regions

4.3.2. Testing with LBP

In the case of a sample image for which the emotion expressed by it is to be determined then the first step would be the feature extraction step. This step is done in the same way as explained above. The feature vector thus obtained is then tested against each of the six model vectors computed during the training phase. There are a number of measures that can be used to determine which model vector closely resembles the feature vector computed from the sample image. These measures are usually in the form of distances and the model vector that produces minimum distance with respect to the sample feature vector determines the emotion expressed. Some of these measures are listed below.

Let S represents the sample feature vector and M represents one model vector. As mentioned above there are 42 regions and let N be the number of pixels in the neighbourhood. Thus there are $N(N-1)+3$ bins per region.

Histogram Intersection

$$D(S, M) = \sum_{j=1}^{42} \left(\sum_{i=1}^{N(N-1)+3} \min(S_{i,j}, M_{i,j}) \right) \quad (4.8)$$

Log-likelihood Statistics

$$L(S, M) = \sum_{j=1}^{42} \left(- \sum_{i=1}^{N(N-1)+3} S_{i,j} \log M_{i,j} \right) \quad (4.9)$$

Chi Squared Statistics

$$\chi^2(S, M) = \sum_{j=1}^{42} \left(\sum_{i=1}^{N(N-1)+3} \frac{(S_{i,j} - M_{i,j})^2}{S_{i,j} + M_{i,j}} \right) \quad (4.10)$$

Chi Squared Statistics with weights

$$\chi_w^2(S, M) = \sum_{j=1}^{42} w_j \left(\sum_{i=1}^{N(N-1)+3} \frac{(S_{i,j} - M_{i,j})^2}{S_{i,j} + M_{i,j}} \right) \quad (4.11)$$

Consider two sample feature vectors S1 and S2 and M1 which represents model vector. The values of these vectors are shown in the table below:

S1	S2	M1
5	9	2
60	80	50

In the case of histogram intersection always the minimum value is selected for computation of the distance measure. So in the case of the samples given above always m1 is chosen as the minimum value and this leads to S1 and S2 having the same distance. In the case of chi-square statistics the square of the absolute differences between M1 and the sample vectors are chosen as the distance measure. We can see from the example that S1 and S2 will have different distance measures as compared to histogram intersection. Thus we can say that chi-square statistics is much more capable of differentiating different vectors.

In the case of log-likelihood statistics we always take the log of the model vector and multiply with the corresponding sample vector value. Consider the following values for S1, S2 and M1.

S1	S2	M1
5	15	10
40	60	50

Thus in the case log-likelihood statistics although both the vectors are very similar to the model vector they will be classified as different because their distance measures

will be biased by the value of the sample vector. For instance although 5 and 15 are at a distance of 5 from 10, since they are multiplied to log 10 their distance measures would be -5 and -15 respectively. Again chi-square statistics takes into account the square of the absolute difference and will classify these two vectors to be similar. This is not the case in the case of log-likelihood statistics.

All the three measures involve different levels of complexity in their calculation. The choice of the method could be based on the complexity which is very important constraint while implementing these methods on an embedded platform.

The last measure includes information about the weights of regions and hence is best suited for the approach described above. The value computed is a measure of similarity and hence the smaller the value the more similar are the two vectors i.e. the sample feature vector and the model vectors.

5. Proposed Emotion Recognition Approach

The previous sections describe an approach to recognize emotions expressed in a single face image. In the case of real life experiences expressions on human faces changes as a result of changes in the human's environment. These changes in expression are a direct result of the changes in emotion state of the human. Also these changes take place over time. In other words the change in human emotion changes with time and is a gradual process. The approach mentioned earlier can not be directly applied to take into account the temporal aspect of this change. For this reason the following approach is suggested which is described in detail in the following sections. The approach takes into account the temporal changes in a sequence of image sequences.

5.1. Training Phase

The Training phase of the proposed approach is shown in figure 5.1. The first major change is that here we take as input a set of sequences of images instead of a single image. The image sequences start with a face image showing no expression. The final image in the sequence is a face image showing one of the six basic emotions. The intermediate images show the change from no expression to final basic emotion. The image sequences are then subject to the feature extraction step to get a set of feature vectors in the form of symbols. The computation of the symbols is done in a number of steps including calculation of optical flows, histogram of gradients and clustering of feature vectors into groups. The centroids of these groups are then defined as a mean of the vectors belonging to these groups. The centroids form a codebook which is used to map the original histogram of gradients into symbols. These symbols are then used to train a Hidden Markov Model (HMM) to reflect a particular class of emotions.

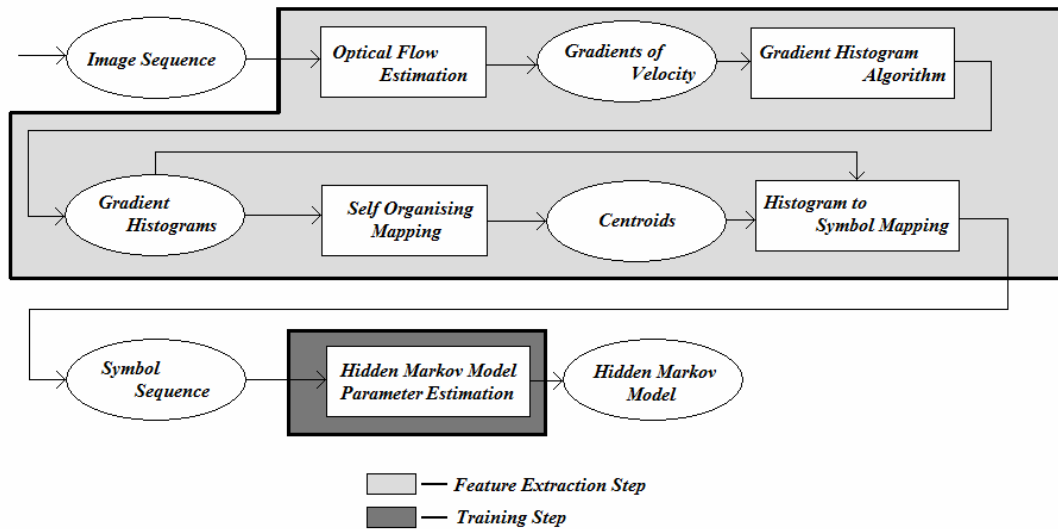


Figure 5.1 Training phase in the proposed emotion recognition approach

At the end of the training phase a HMM is trained for each class of emotion. Hence the training process is performed once for each class of emotions with the training set consisting of only image sequences showing the same final basic emotion.

5.2. Testing Phase

The Testing phase of the proposed approach is shown in figure 5.2. The feature vectors are extracted in almost the same way as the training phase, except that the centroids or codebook is not computed here. Instead the codebook computed in the training phase is used for mapping histogram of gradients to symbols. The most general measure used in both training and testing is the least Euclidean distance. The feature vector (sequence of symbols) along with the HMM models of each emotion class (trained in the training phase) is then passed as inputs to a Forward Backward procedure (FB). The FB procedure computes for a HMM how well the HMM matches a given sequence of symbols. The scores are represented as a log of probabilities and the greater this value the better is the match between the sequences of symbols and HMM and hence the emotion transition for which the corresponding HMM was trained for.

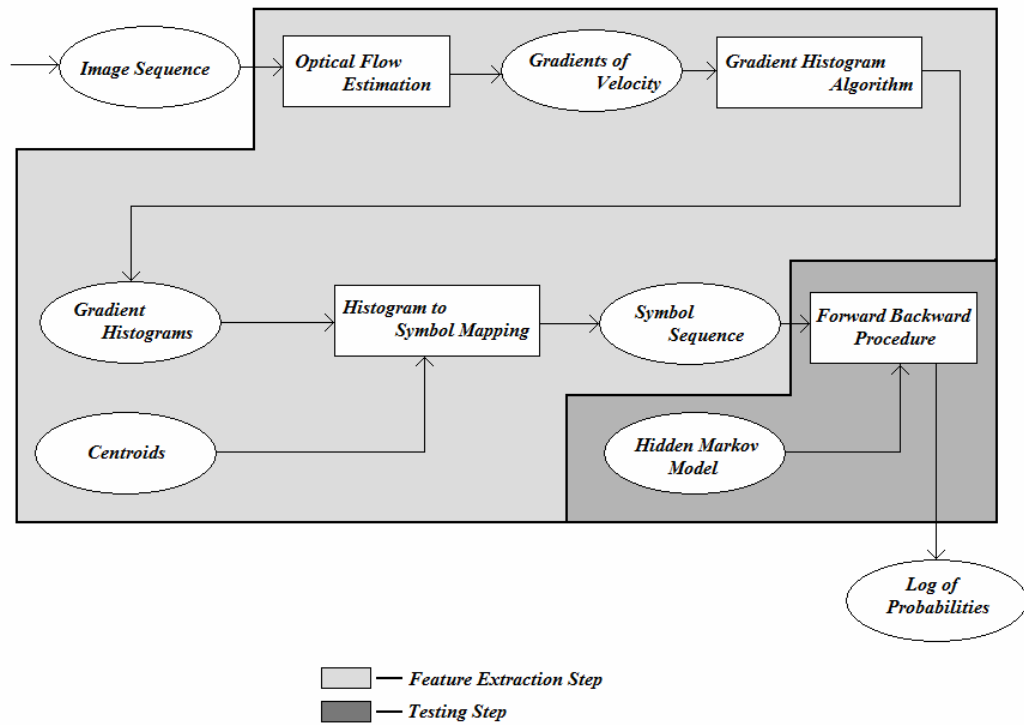


Figure 5.2 Testing phase in the proposed emotion recognition approach

6. Feature Extraction in Proposed Approach

The steps involved in the feature extraction step include a number of sub-steps including Optical flow estimation using Horn and Schunck's Algorithm[5], calculation of Histogram of Gradients (HoG) and clustering/codebook generation using Self Organizing Maps [6] (SOMs). Each of these sub-steps is described in detail in the following sub sections.

6.1. Optical Flow Estimation

In simple words optical flow can be explained as the movement of an object with respect to the observer's field of view. For example consider a stationary object and a moving observer. As the observer moves closer to the object the object appears to get bigger in the observer's field of view and as the observer moves away from the object the object appears to get smaller in the observer's field of view. At the same time the edges of the object tend to move outwards as the observer moves closer to the object and vice versa. The reason for the movements is because the observer projects the object onto a two dimensional plane. The movements of the object have apparent velocity with respect to other objects in the observer's field of view. The estimation of optical flow attempts to compute the direction and magnitude of these velocities.

6.1.1. Horn and Schunck's Method

According to Horn and Schunck objects in an image appear to move with the changes in observer's view. This change in observer's view can be produced by movement of the observer or the objects in the view. The approach proposed by Horn and Schunck tries to estimate this apparent movement of brightness pattern or optical flow of objects in consecutive images. The approach involves some difficulties in estimation of the optical flow at different pixels of image independently of their neighbouring pixels. The reason for the difficulty is the fact that the apparent velocity at each pixel has two components where as the changes in the brightness patterns have only one component. These difficulties are dealt with by introducing certain constraints. The constraints and assumptions are mentioned in the following subsections.

6.1.1.1. Assumptions

A number of assumptions are made initially in order to make the problem of calculating optical flow simple. One of the assumptions is that the objects in the images have a flat surface to eliminate variations in intensities due to shading. To eliminate the negative influences of illumination it is assumed that the image is produced under constant and uniform illumination. The third assumption is that the reflectance of the objects in the image varies smoothly. An additional assumption is that the reflectances of the objects in the image do not have discontinuities. The assumptions on reflectance help the computation as it makes it possible that image intensity is differentiable.

6.1.1.2. Constraints

A number of constraints were introduced in the computation of optical flow from a sequence of images. These constraints are explained in the subsections below.

6.1.1.2.1. Constant Brightness Constraint

The first constraint that was introduced in the computation was that intensity patterns for the same point in consecutive images over time remains unchanged. If E represents the intensity of point in an image then the constant brightness constraint can be expressed mathematically as follows

$$\frac{dE}{dt} = 0 \quad (6.1)$$

The equation can be expanded using chain rule of differentiation as follows

$$\frac{\partial E}{\partial x} \frac{dx}{dt} + \frac{\partial E}{\partial y} \frac{dy}{dt} + \frac{\partial E}{\partial t} = 0 \quad (6.2)$$

Now, by assigning

$$u = \frac{dx}{dt}, v = \frac{dy}{dt} \text{ and } E_x = \frac{\partial E}{\partial x}, E_y = \frac{\partial E}{\partial y} \text{ and } E_t = \frac{\partial E}{\partial t} \quad (6.3)$$

The expression can be rewritten as

$$E_x u + E_y v = -E_t \quad (6.4)$$

Or in other words,

$$\begin{bmatrix} E_x & E_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -E_t \quad (6.5)$$

From the above equation it can be seen that the component of motion that lies in the same direction as that of intensity gradient (E_x, E_y) can be given by

$$(E_x, E_y) = -\frac{E_t}{\sqrt{E_x^2 + E_y^2}}$$

(6.6)

There is however a problem called the aperture problem due to which the component of movement cannot be calculated if it lies at right angles to the intensity gradient. This problem is best demonstrated with an example shown in the figure 6.1 below.

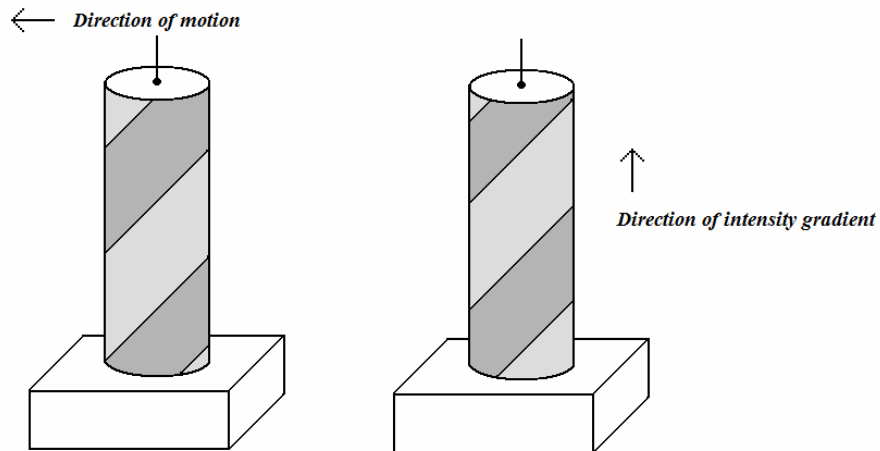


Figure 6.1 Aperture Problem: Motion direction orthogonal to intensity gradient

The figure shows a pole with two different colour patterns drawn spirally along the surface at an inclined angle. If the pole is rotated horizontally although the motion is horizontal the successive image shows an upward movement of the pattern. This leads to the direction of intensity gradient and the direction of motion to lie at right angle to each other. Thus it is not possible to estimate the component of motion in the horizontal motion locally without introducing additional constraints.

6.1.1.2.2. Smoothness Constraint

Most objects in the images commonly exhibit rigid motion. This leads to an assumption that points belonging to the same objects have similar velocities. This is also true for velocity component of the motion in the direction of intensity gradient. This assumption results in a conclusion that the velocity component in the direction of intensity gradient varies smoothly at almost all points except in areas involving occlusion. The smoothness of optical flow field can be measured by sum of squares laplacians of u and v given by,

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad (6.7)$$

$$\text{and} \quad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \quad (6.8)$$

6.1.1.3. Estimation of partial derivatives and Laplacians

For the computation of optical flow and the minimization, the partial derivatives of intensity patterns and laplacians of x and y components of the flow must be estimated. Horn and Schunck provided some equations that are described in the following subsections.

6.1.1.3.1. Partial Derivatives

Consider a set of images showing the same objects at slightly different time intervals. The method described by Horn and Schunck to estimate the partial derivatives E_x , E_y and E_t involves finding the first differences of intensity values of four neighbouring

points in two such consecutive image frames. The intensity values of these four points from the two images form a cube which is shown in figure 6.2.

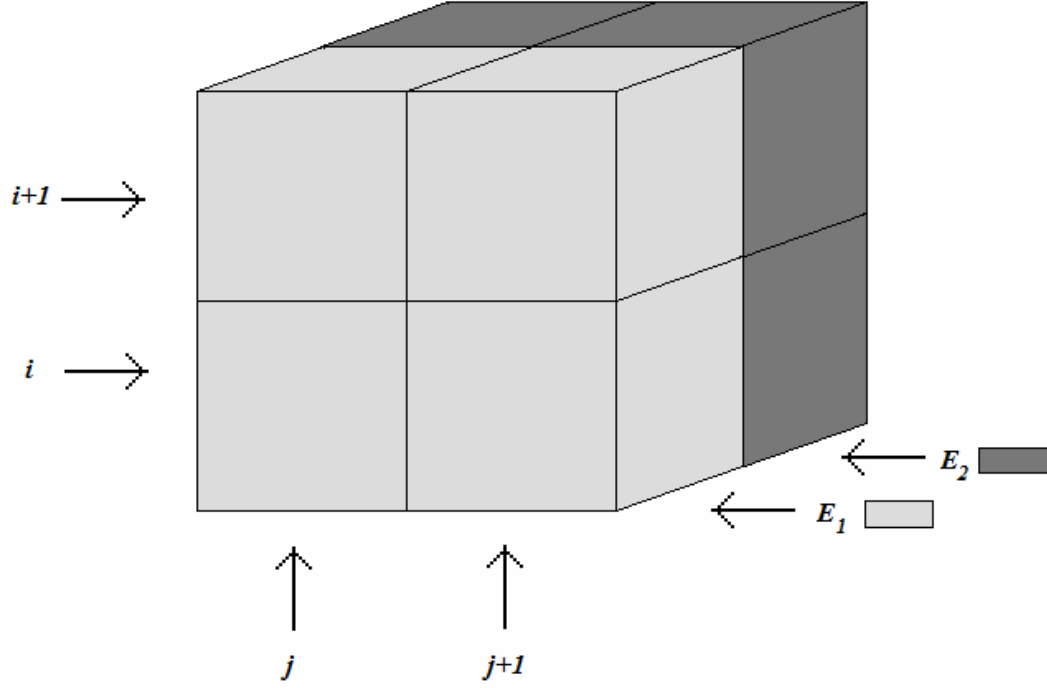


Figure 6.2 Partial derivative estimation in Horn-Schunck approach

Suppose that intensity values of the points in these consecutive values are represented by E_1 and E_2 for first and second image frames respectively. The partial derivatives E_x , E_y and E_t are computed by the sum of the average of the differences of corresponding intensity values as given below

$$E_x = \frac{1}{4}(E_{2,i+1,j+1} - E_{2,i+1,j}) + \frac{1}{4}(E_{2,i,j+1} - E_{2,i,j}) + \frac{1}{4}(E_{1,i+1,j+1} - E_{1,i+1,j}) + \frac{1}{4}(E_{1,i,j+1} - E_{1,i,j}) \quad (6.9)$$

$$E_y = \frac{1}{4}(E_{2,i+1,j} - E_{2,i,j}) + \frac{1}{4}(E_{2,i+1,j+1} - E_{2,i,j+1}) + \frac{1}{4}(E_{1,i+1,j} - E_{1,i,j}) + \frac{1}{4}(E_{1,i+1,j+1} - E_{1,i,j+1}) \quad (6.10)$$

$$E_t = \frac{1}{4}(E_{2,i+1,j} - E_{1,i+1,j}) + \frac{1}{4}(E_{2,i,j} - E_{1,i,j}) + \frac{1}{4}(E_{2,i+1,j+1} - E_{1,i+1,j+1}) + \frac{1}{4}(E_{2,i,j+1} - E_{1,i,j+1}) \quad (6.11)$$

6.1.1.3.2. Laplacians

The laplacians of x and y components of the optical flow i.e. were estimated using the following approximation by Horn and Schunck.

$$\nabla^2 u \approx k(\bar{u}_{i,j} - u_{i,j}) \quad (6.12)$$

$$\text{and } \nabla^2 v \approx k(\bar{v}_{i,j} - v_{i,j}) \quad (6.13)$$

where the value of k is taken as 3 since the averages for the derivative were computed according to the cube shown in figure 6.2. \bar{u} and \bar{v} are weighted average of the velocity (u and v respectively) components at neighbouring points and the weights are assigned as follows.

$$\bar{u}_{i,j} = \frac{1}{6}(u_{i-1,j} + u_{i,j+1} + u_{i+1,j} + u_{i,j-1}) + \frac{1}{12}(u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} + u_{i+1,j-1}) \quad (6.14)$$

$$\bar{v}_{i,j} = \frac{1}{6}(v_{i-1,j} + v_{i,j+1} + v_{i+1,j} + v_{i,j-1}) + \frac{1}{12}(v_{i-1,j-1} + v_{i-1,j+1} + v_{i+1,j+1} + v_{i+1,j-1}) \quad (6.15)$$

6.1.1.4. Iterative Solution

Since finding a solution to the constrained problem by directly solving the simultaneous equations is computationally expensive Horn and Schunck suggested an iterative approach. The velocities u and v are estimated iteratively by using the already estimated derivatives and the velocities computed in the previous iterative step. Let the velocities estimated in the n th step be denoted by \bar{u}^n and \bar{v}^n , then the velocities estimated in the $n+1$ th step is adjusted by the following equations,

$$\bar{u}^{n+1} = \bar{u}^n - E_x \left(\frac{E_x \bar{u}^n + E_y \bar{v}^n + E_t}{\alpha^2 + E_x^2 + E_y^2} \right) \quad (6.16)$$

$$\text{and } \bar{v}^{n+1} = \bar{v}^n - E_x \left(\frac{E_x \bar{u}^n + E_y \bar{v}^n + E_t}{\alpha^2 + E_x^2 + E_y^2} \right) \quad (6.17)$$

6.1.1.5. Experimental setup

For the purpose of emotion recognition a sequence of images were used which were captured at slightly increasing time instances. The sequences represented the change in expression from neutral face image to one of the six basic emotions. The images were cropped to include only the face region of the image and were resized to dimension of 108x147 pixels. The optical flow was calculated for two consecutive images. Thus if the sequence had n images then number of optical flows computed would be n-1. The figure 6.3 below shows as an example some images along with the computed optical flows. The image sequences were taken from Cohn-Kanade database.

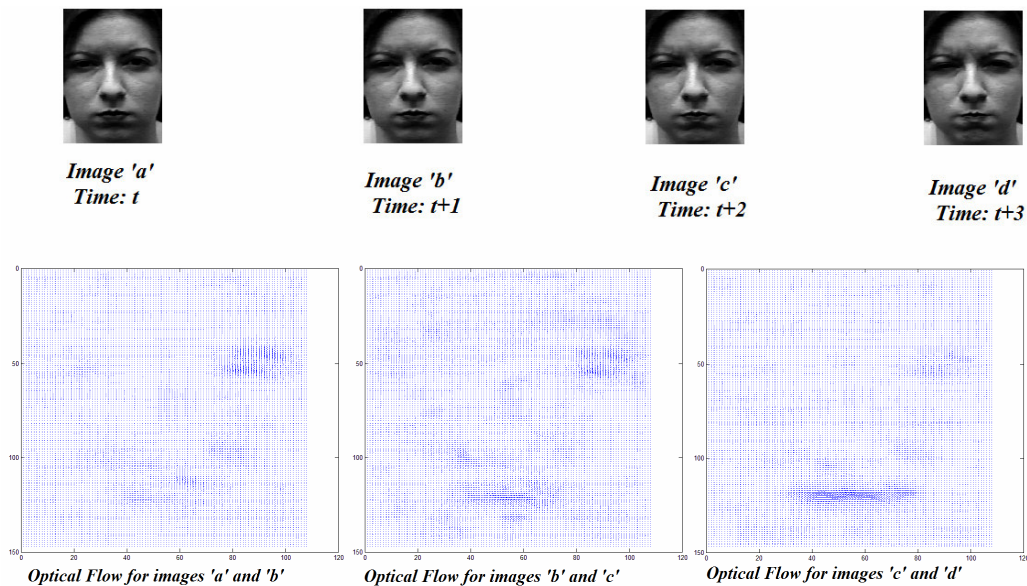


Figure 6.3 Optical Flow estimation using Horn-Schunck approach

The Figure 6.3 shows the optical flow between images a and b, images b and c and images c and d. As you can see the first optical flow shows that there are changes in the face image due to movements in the subjects left eye region and some movements in the persons left cheek and the persons lip regions. Similar changes can be seen between images b and c and between images c and d.

6.2. Histogram of Gradients

The next sub-step in the feature extraction step is the generation of histogram of gradients. There are two reasons for this step the first being the reduction of size of the data and second is due to the division of the image into regions. Since different

regions of the face provide information about transition with varying levels there is a need to assign corresponding weights to these regions. Since histogram gives collective information about specific features in the region the division of image into region is performed before the histogram generation.

There is a difference in the generation of the histogram of gradients from that of generation of a regular histogram. The histogram of gradients uses for computation two dimensions than just a single dimension as in the case of a regular histogram. As seen before the estimated motion velocities computed in the previous sub-step has two components(x and y). These components can also be expressed as a combination of direction or orientation and magnitude. In the case where the histograms are generated using the motion velocities the images are divided into same regions as discussed earlier in feature extraction using LBP. Then for each of these regions a histogram is built with the orientation as the bins and the bin count is increased by the magnitude of the velocities with the same orientation. An example of such a generation is shown in figure 6.4.

In this case we way we consider a 4x4 matrix of optical flow and we divide this into four regions of size 2x2. We then generate a histogram of gradients for every region. For each region (in the experimental setup- we consider and angle range of 30 degrees as the range of the bin for histogram generation.) we consider the orientation of the optical flow per pixel to select the bin, where as the contents of the bin are incremented by the magnitude of the optical flow.

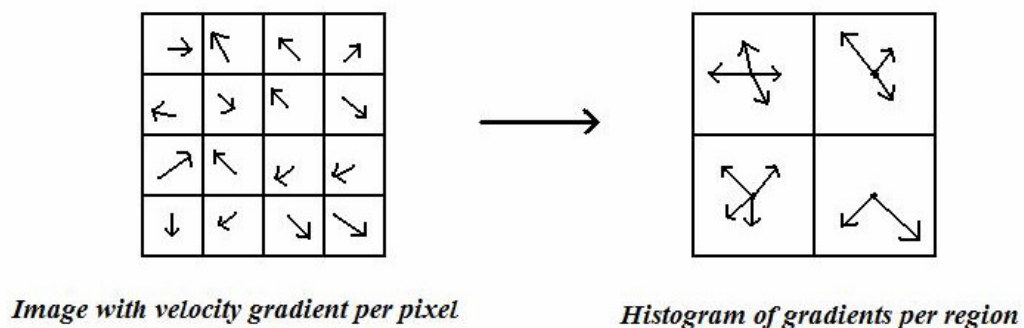


Figure 6.4 Generating regions wise histogram of gradients.

The histograms are then concatenated in the same way as in the case of feature extraction using LBP. These vectors are the input to the next sub-step in the feature extraction step.

6.3. Self Organizing Maps

Self organizing maps or SOMs was invented by Teuvo Kohonen [6] and serve as a data visualization technique. There is also another purpose for which the technique can be used in which case the technique is used to reduce the dimensions of the data. In this case the reduction is due to clustering of similar data into groups. In the case of feature extraction step the technique is used for clustering purpose. The concatenated histograms computed in the previous sub-step are used as input to this step. These histograms are then clustered into groups and the centroid of these groups then used as representative vectors for each group. In the last step the histograms are then mapped to this reduced space and thus the data reduction takes place.

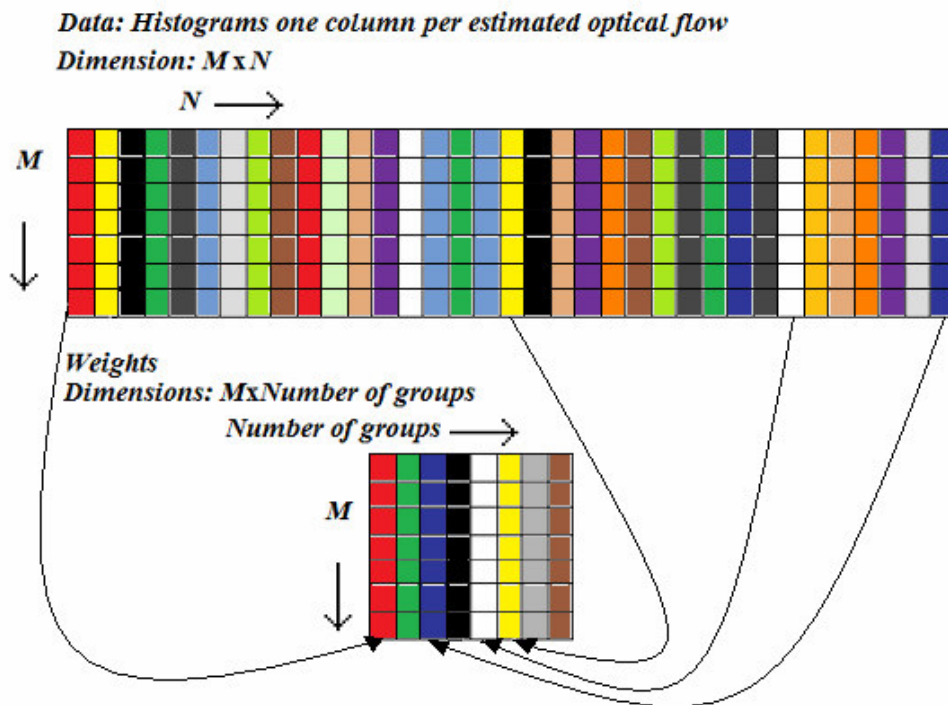


Figure 6.5 General structure needed for SOM

6.3.1. Inputs and structures used in SOMs

The structures needed for self organizing maps are better explained using figure 6.5. The data for the SOM step is a set of Histograms of gradients (HOGs) computed for a set of image sequences showing transitions from neutral to specific emotion. Each column of the data represents the HOG computed for two consecutive images. There is another input to the SOM which is the weight vectors. The weight vectors must satisfy certain properties. The height dimension must be the same as the height dimension of the data. On the other hand the width of the weight vector must be at least as big as the desired number of clusters. The weights are initialized as a random set of numbers. The figure depicts the case of 1-dimensional SOM.

6.3.2. The Learning Algorithm

The columns of the weight vector represent the clusters onto which each of the input data will finally be mapped into. The solution to the problem of clustering is iterative. During one such iteration the algorithm takes one column from the data and finds the column from weight vector that closely represents the input vector. The closest weight vector is then adjusted along with the neighbouring weight vectors so that the match is optimized. This happens once for each of the input column during each such iteration. The algorithm proceeds with the optimization and finally settles into a map with stable clusters. The algorithm is shown below.

Learning Algorithm:

1. *Choose sufficient number of groups.*
2. *Initialize the weight vector with random numbers.*
3. *A column from data is chosen at random.*
4. *For the chosen column the best matching weight column is found.*
5. *A neighbourhood of the best matching weight column is computed.*
6. *All the weight columns in the neighbourhood of the best matching weight column along with the best matching weight column are adjusted to more closely represent the input data.*
7. *Steps through 3 to 6 are repeated for chosen number of times or till the weight columns are no longer affected.*

The weight columns in the neighbourhood are adjusted such that the columns closer to the best matching weight column are changed more to reflect the input data and the ones farther are changed less.

6.3.3. Computing the best matching weight column and neighbourhood

To compute the best matching weight column the Euclidean distance between the each weight vector and the data column is computed. The one with the least distance is chosen as the best matching weight vector. Suppose V indicates the input data and W indicates the current weight column then the Euclidean distance say D between the two vectors is given by,

$$D = \sqrt{\sum_{i=1}^M (V_i - W_i)^2} \quad (6.18)$$

where M is the height of the column.

The size of the neighbourhood of the best matching weight column must decrease after each iteration. Also the weight columns in the neighbourhood must be adjusted corresponding to their distance from the best matching weight columns. For these we use an exponential decay function for calculating the neighbourhood as well as for adjusting the weights during one such iteration.

For calculating the neighbourhood of the best matching weight column a measure of distance is used. All the weight columns which are within this distance from the best matching weight column belong to the neighbourhood. The distance is reduced after every iteration using an exponential decay function and is of the form,

$$newDist = oldDist * \exp\left(-\frac{t}{\lambda}\right) \quad (6.19)$$

Where $newDist$ is the new distance for the current iteration step, $oldDist$ is the distance from the previous step; t is the iteration number and λ is a time constant. The value of λ is computed as follows

$$\lambda = \frac{\text{Number of iterations}}{\log(\text{oldDist})} \quad (6.20)$$

6.3.4. Adjusting the weights

Once the neighbourhood is calculated the next step is to adjust the weights. There are some considerations that are needed to be addressed. The first consideration is that the weight column must be adjusted corresponding to its distance from the best matching weight column. The second consideration is that the adjustment must also decrease over time so that the algorithm converges. The first consideration is addressed by using an exponential function which is dependent on the distance of the weight column from the best matching weight column and also dependent on the current size of neighbourhood. If $dist$ denotes the distance of a weight column within the neighbourhood from the best matching weight column then the function is calculated as,

$$\Theta(t) = \exp\left(-\frac{dist^2}{2 * newDist}\right) \quad (6.21)$$

The second consideration is addressed by using reducing the rate of adjustment (learning rate) using a decay function similar to the one shown earlier for neighbourhood calculation.

$$newLearningRate = oldLearningRate * \exp\left(-\frac{t}{\lambda}\right) \quad (6.22)$$

The weights are finally adjusted using the following equation

$$W(t+1) = W(t) + \Theta(t) * newLearningRate * (V(t) - W(t)) \quad (6.23)$$

where $V(t)$ represents the input data for which the adjustment is taking place and the $W(t)$ represents the weights that are being adjusted at iteration $t+1$.

6.3.5. Centroid computation from Weights and mapping to symbols

After the weights are adjusted the next step is to map columns of the input histogram into one of the clusters. The centroids are computed by averaging all the input data that are mapped to the same cluster. The next step in the feature extraction step is to map the histogram of gradients to symbols. The easiest way to map is to map the histograms to symbols is to output the index of the centroid to which the histogram maps using minimum Euclidean distance as a measure.

6.3.6. Advantages

Self organizing map is a good technique for visualization as the clustering of similar data gives an insight into the nature of relationship between different dimensions. One way of visualization can be done by assigning different colours to the data and then rearranging the original data based on the weights. Another important advantage of using SOMs is that the clustering is not easily influenced by existence of spike values in the data if the number of groups chosen for the learning purpose is sufficiently big. If the number of clusters to be formed is sufficiently big then a spike in data does not get clubbed with one of the clusters. Thus the maps represent a more reliable clustering.

6.3.7. Disadvantages

One of most significant disadvantages of using SOMs is that the method is computationally expensive. As the size of data increases the size of neighborhood should also be increased to get more stable clustering. But this also exponentially increases the number of distances to be calculated. Also SOMs can not be generated if some data has a missing dimension. Thus the method is limited to be used by approaches where the all data have the same dimensions. The third disadvantage is that every run of the algorithm can produce a different clustering. This is dependent on the number of iterations and the size of the number of clusters.

7. Hidden Markov Model

Hidden Markov models [25] are statistical models and are used in a wide number of applications involving temporal pattern recognition such as speech recognition, gesture recognition and handwriting recognition. The following subsections introduce in short the general methods used in such systems along with the adaptation done in order to use HMMs for the purpose of emotion recognition.

7.1. Introduction

Hidden Markov Models are used in scenarios where the system to be developed can be thought as a Markov process. In regular Markov model the states are directly visible to the observer. In a hidden Markov model the states are not visible to the observer, but instead a set of states are visible to the observer that influence the invisible states. The main difference in the case of Hidden Markov Models and a Markov Model is that in the case of a Markov Model the states are directly visible to the observer and there is only one probability distribution defining the transition from one state to the other. But in the case of Hidden Markov Model apart from the transition probabilities there is also a probability distribution that defines at which state what observations are possible and to what extent. Thus a hidden Markov Model is doubly stochastic. In the case of Hidden Markov Model it is also possible to estimate the transition probabilities and output probabilities based on the observation sequences. Thus even if these probabilities are initially unknown they can be estimated based on the observations. Given a sequence of observations the key idea then is to determine the parameters of the Markov process by using this sequence of observations. The model thus developed can be used for further analysis such as pattern recognition. Figure 7.1 shows as an example a Hidden Markov Model. In this sense the Hidden Markov Models (HMMs) are better suited for the purpose of recognition as the symbols generated in the previous steps form the observation sequence for the probability estimations (Training). The HMMs thus trained can be further used for recognizing the expression in a sample image sequence.

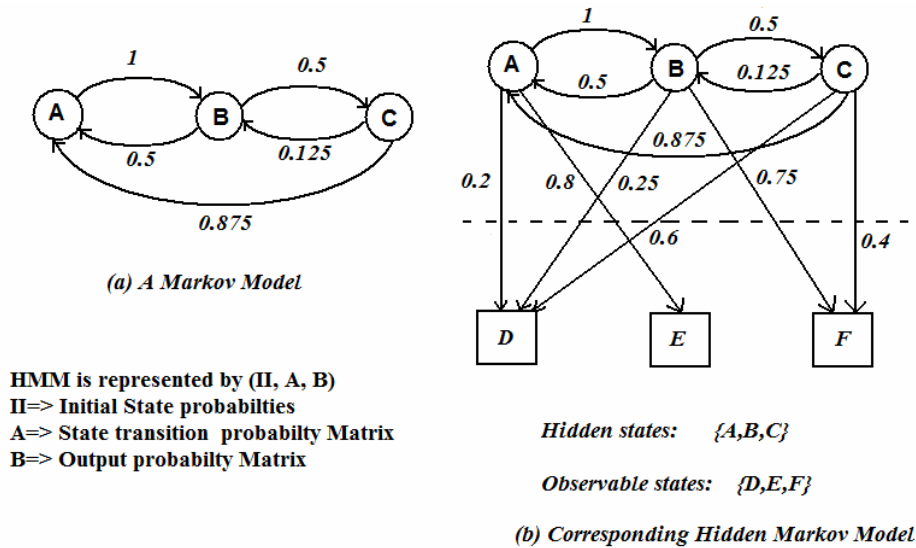


Figure: 7.1 Example of Hidden Markov Model

Figure 7.1 (a) shows the Markov model of a system. The arrows represent the probability with which the system can move from any state to any other state and these are called as transition probability. Figure 7.1 (b) shows a corresponding Hidden Markov model. The states above the dashed line are hidden. The states shown in boxes are observable states or possible observations/outputs. An arrow from a hidden state to an observable state shows the probability of the hidden state to produce or generate the observation and these are called as output probabilities. As shown in the figure a Hidden Markov model is represented using three parameters $\lambda = (\Pi, A, B)$. Π is used to denote the initial state probabilities. In the case of Figure 7.1 if the system starts at state A then the initial state probabilities is defined as $[1 \ 0 \ 0]$ for hidden states $[A \ B \ C]$. For an HMM with N hidden states A is an $N \times N$ matrix showing the probability distribution for transition. For an HMM with N hidden states and M observable states B is an $N \times M$ matrix showing the probability distribution for observation generation. Both A and B have to be estimated.

7.2. The three basic problems

Associated with HMMs are three basic problems:

1. Given the parameters of the HMM and an observation sequence, compute the probability that the observation sequence was generated by the HMM and

- calculate the probabilities of the hidden state. The problem is solved by using an algorithm called as the forward-backward algorithm.
2. Given the parameters of the HMM and an observation sequence, find the most probable sequence of hidden states that might have generated the observation sequence. The problem is solved using the Viterbi algorithm.
 3. Given a set of sequences of observations, adjust the parameters of the HMM to reflect the observations. The problem is solved using Baum-Welch algorithm.

7.3. Experimental Setup

For the purpose of emotion detection we build a HMM for each of the six basic emotions. For this purpose we use the Cohn-Kanade database and the Baum-Welch algorithm. The training set comprises of six different set of image sequences each of which show the expression corresponding to the emotion for which the HMM is built. The database consist of 33 image sequences of neutral to angry expression, 38 image sequences of neutral to disgust expression, 50 image sequences of neutral to fear expression, 97 image sequences of neutral to happy expression, 71 image sequences of neutral to sad expression and 76 image sequences of neutral to surprise expression. The average number of images per sequence is roughly about 21.

7.4. Training with HMM

The figure 7.2 shows the steps in general used to train the HMMs and are applicable to all the expressions. The first step is to use the sequences of images that belong to same set (showing same expression at the end) and apply the Horn-Schunck algorithm to obtain the optical flow. This optical flow is calculated for all the pixels in the images. The optical flow thus computed is then passed through the histogram generation step to obtain column wise vector. The vectors after being computed for the entire set are then passed through the SOM algorithm to get a set of centroids. The centroids are then used to map the optical flows to sequences. All the sub-steps till here form the part of the feature extraction step. The symbols along with an initial estimate of the transition probability matrix and initial estimate of output probability matrix are then passed to the Baum-Welch algorithm. The algorithm will estimate the parameters of the HMM, viz. the transition and output probability matrices.

The key factors that may have an effect on the final estimation of model parameters are the number of hidden states, the number of observable symbols and the initial estimates of transition and output probability matrix. There are no good or bad rules for deciding the number of hidden states and the number of observable sequences. Most approaches suggest a trial and error method for these values. According to[] the initial estimates of transition probability matrix could be based on either random or uniform distribution.

Another important factor for transition probability matrix is that the use of zeros may lead to an unstable estimation as these zeros will never change and hence may not reflect the feature vectors used for training. On the other hand the initial estimation for output probability matrix should incorporate intelligent guesses for better accuracy. This may involve manual inspection of the symbols generated.

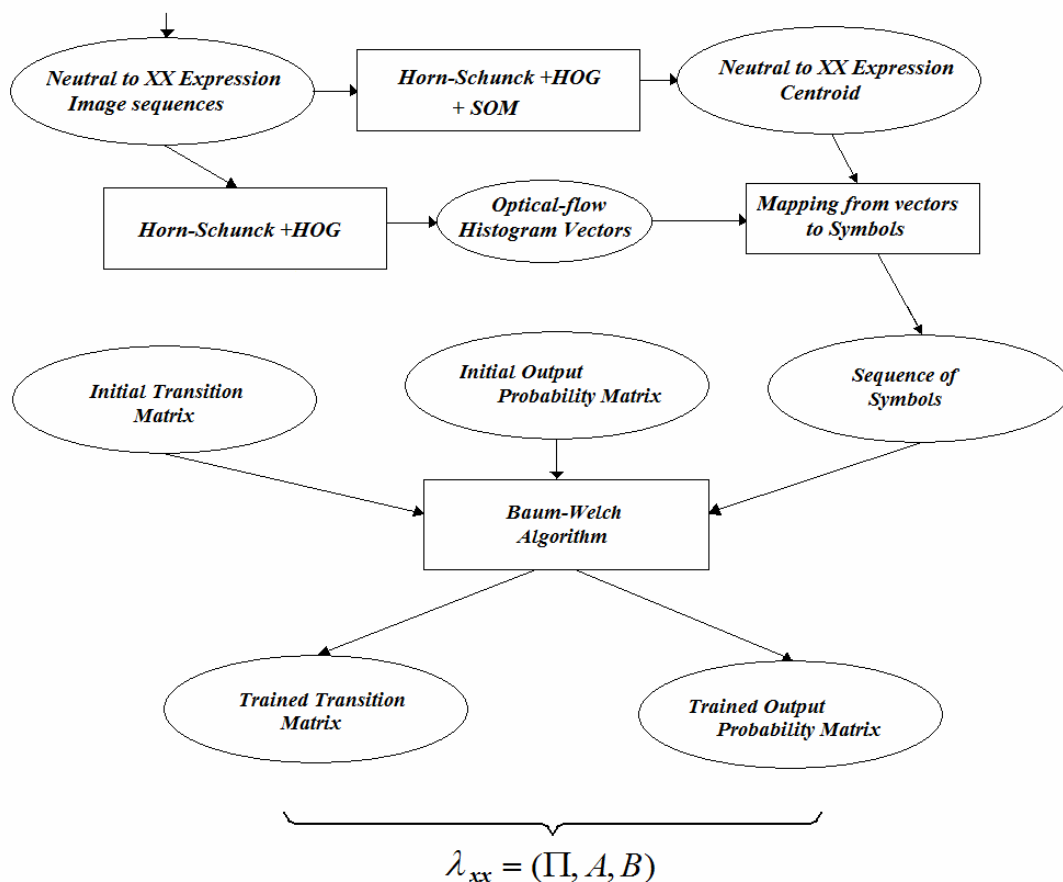


Figure 7.2 Training with HMMs

For the purpose of emotion recognition the transition probability matrix was chosen such that each hidden state could either remain in the same state or could to the immediate next state with equal probability. The last hidden state how ever has a transition probability of 1. The structure is shown in figure 7.3 below. Each value of the output probability matrix was initialized to be equal to the ratio of a single hidden state to the number of observable states. A number of experiments were carried out by varying the number of hidden states, the number of observable states and the number of centroids. Details of the results of these experiments are shown in the result section.

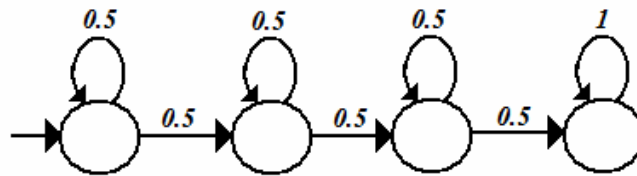


Figure 7.3 Example of the structure of the HMMs to be trained.

For more reliable results of training and testing the entire set of sequence of images for an expression is divided into ten parts (of complete sequences); one part is separated out for testing and the other nine parts are used for training. The process is repeated ten times over different combinations of the parts. The process is in general called n-cross validation and justifies the performance of the model over the entire database. Also as an addition step the histogram of regions were multiplied with weights as shown in figure 4.5.

7.5. Testing with HMM

The testing phase aims at evaluating the emotion HMMs in their capability to recognize the correct emotions. One of the inputs to this phase is a set of sample test images. The one-tenth of the database that was not used for training now forms the sequences of images that will be tested against the models. The optical flow of each such sequence is computed and then the mapping to symbols is done using the centroids of each set of emotions calculated during the training phase. Thus for each sequence of images there are six sequences of symbols. These sequences of symbols are tested against the corresponding HMMs. Here the testing is done using the Forward-Backward algorithm, which tests to see how much each HMMs is probable of generating the sequence. The maximum probability thus indicates the most

probable HMM that can generate the sequence. Since this HMM was built using the image sequence for a particular emotion transition the new sequence most probably expresses this emotion transition. The Steps involved in testing phase with HMMs is shown figure 7.4.

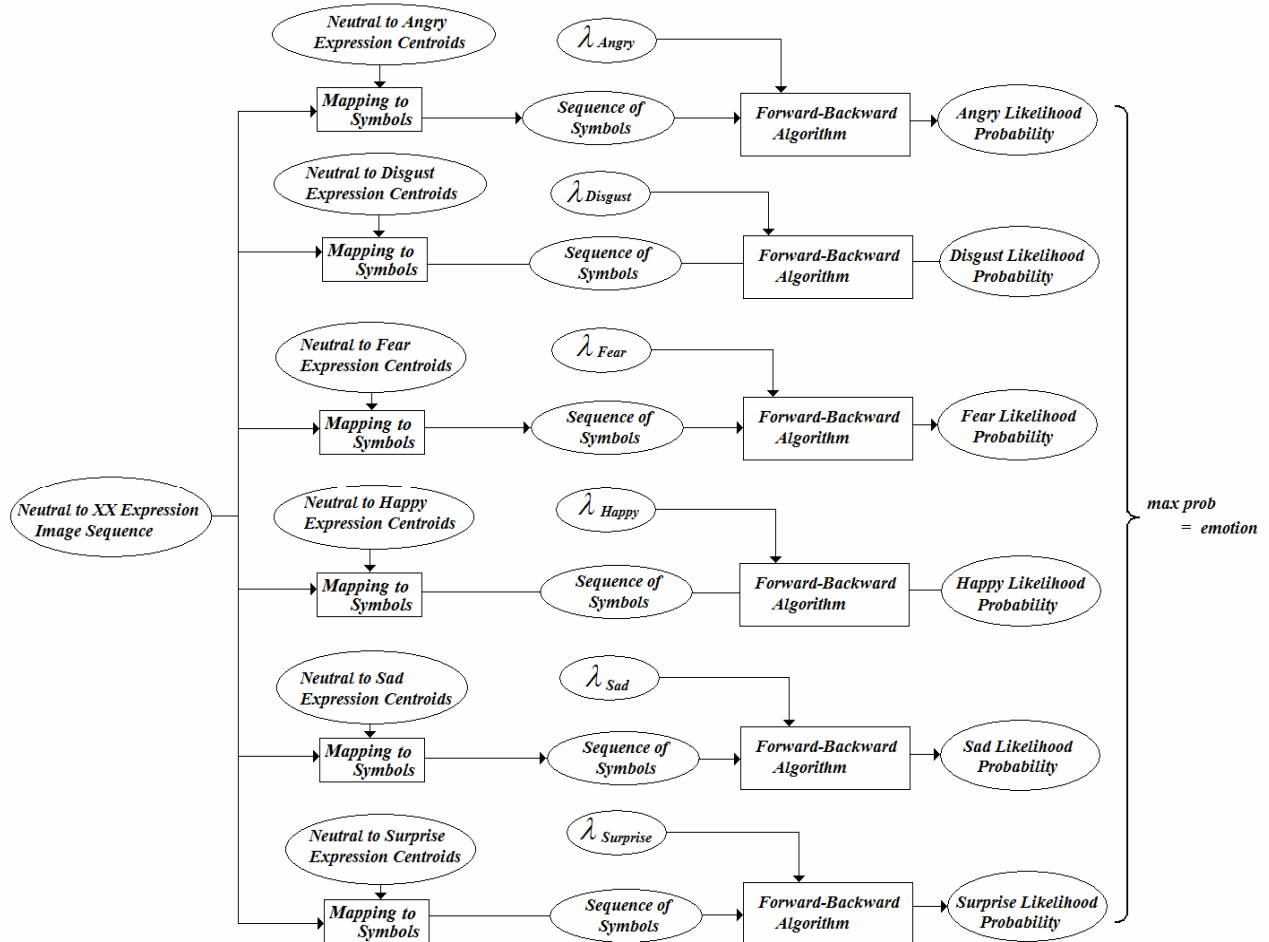


Figure 7.4 Testing with HMMS

7.6. Alternative approach with HMMs

The main idea behind the alternative approach was to have a common code book for mapping the extracted features to symbols. The first few steps remain the same i.e. the codebooks/centroids are generated for the individual sets of image sequences per emotion class. The codebooks thus generated are then passed through the SOM algorithm again to get a common codebook (Combined Centroids). The next step is

then mapping the feature vectors to symbols using the combined centroids. The steps for creating the combined centroids are shown in figure 7.5 below. The steps for mapping the feature vectors to symbols are shown in figure 7.6 below.

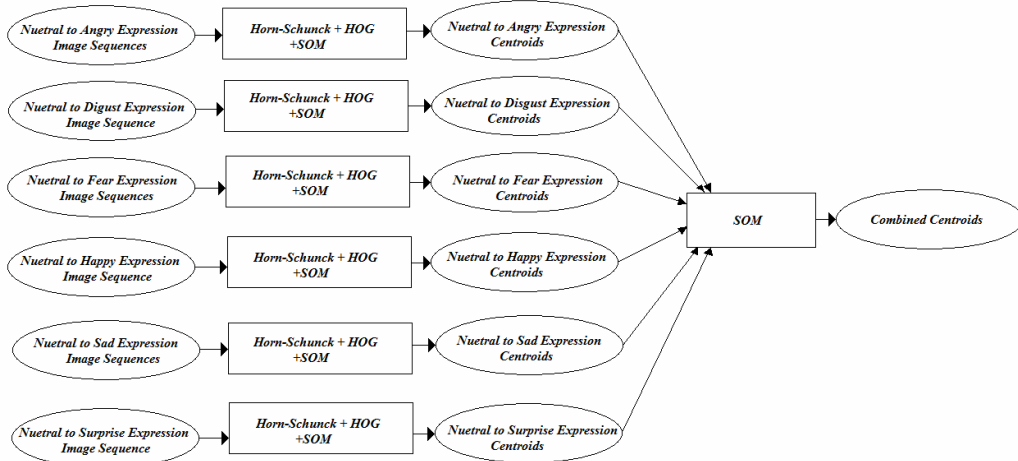


Figure 7.5 Combined centroid computation

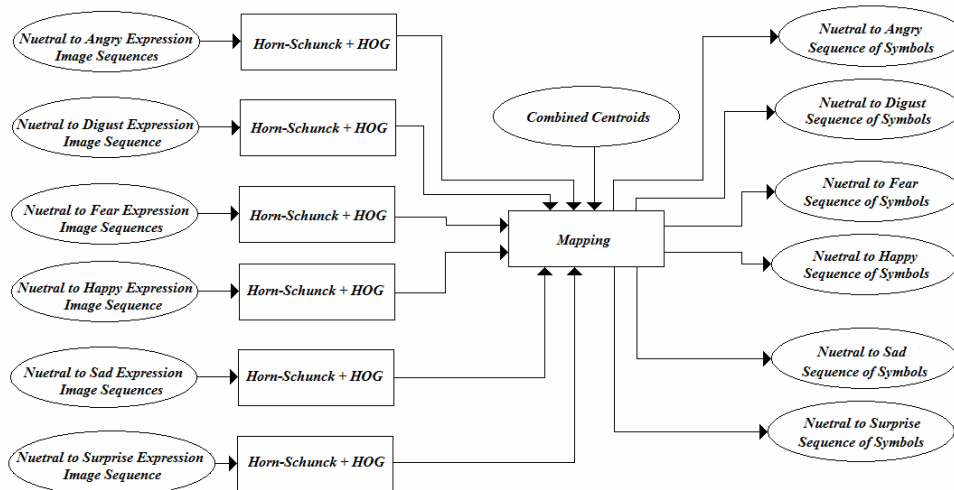


Figure 7.6 Feature vector to symbol mapping in alternate approach

The models are built the same way as in the previous case; the only difference being that the symbols are built from the combined centroids. The figure 7.7 below shows the steps involved in training models.

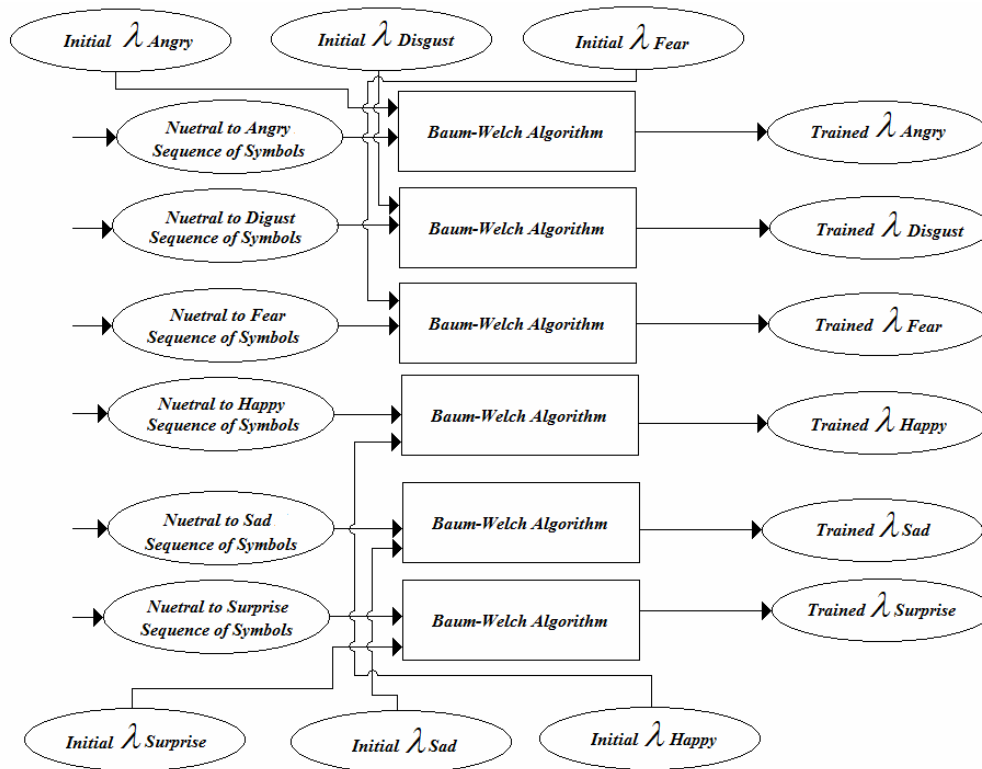


Figure 7.7 Training HMMs in the alternate approach

For testing a sample image the steps change as shown in the figure 7.8 below. The Feature vectors are first computed in the same way. This is followed by mapping the feature vectors to a set of symbols using the combined centroids.

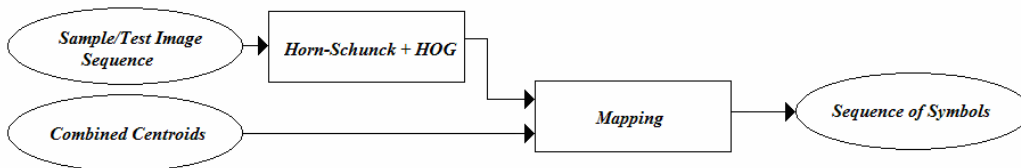


Figure 7.8 Feature extraction and mapping to symbol for a sample/test image sequence

The symbols are then evaluated against each of the HMMs as shown in figure 7.8. The likelihood probabilities are considered the same way as explained earlier.

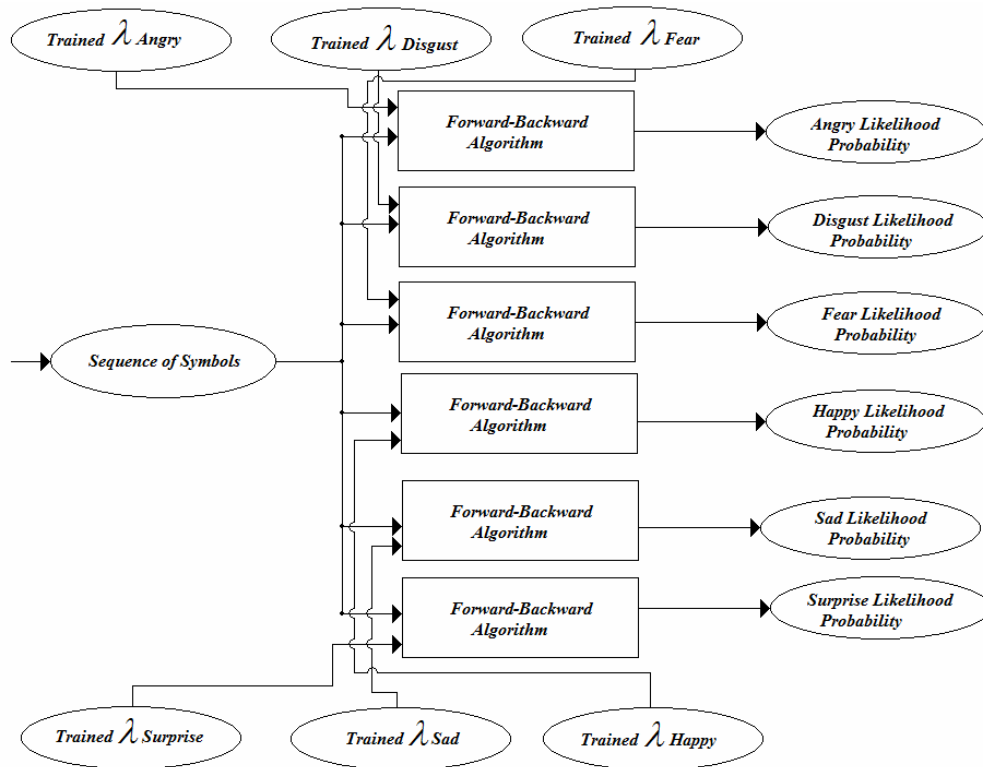


Figure 7.9 Testing with HMMs in alternate approach

8. Results, Suggestions and Conclusion

The following chapter lists some of the major results that were produced during the experiments along with some suggestions that could help in overcoming some of the shortcomings of the current approaches and also states the conclusions of the work.

8.1. Results

There were a number of experiments carried out by varying the number of groups, the number of states and the way the centroids were calculated. The tables from 8.1 to 8.6 show the results of HMMs trained with 20 of hidden states and the observable states equal to the number of centroids. The approach used here is the one where the centroids were not combined. The first columns in the tables indicate the number of observable states. The first column shows the number of groups for which the HMM were trained. The remaining columns indicate the number of image sequences that were correctly identified for each round. Here the entire database was divided into groups of 10. For each round one group was used for testing the HMMs and the remaining 9 groups were used for training the HMMs. The partition for testing and training were changed for each round.

Table 8.1 Neutral to Angry success rates. The number of hidden states is 20.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0	0	0	0	0	0	0	0	0.3333	0
30	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0.3333	0	0.3333
100	0	0.3333	0	0.3333	0	0.3333	0	0	0	0.6666
200	0.3333	0.6667	1	0.3333	0.3333	1	0.3333	0.3333	0.3333	1

Table 8.2 Neutral to Disgust success rates. The number of hidden states is 20.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.3333	0	0	0	0	0	0	0	0	0
30	0	0	0	0.3333	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0.3333	0	0
200	0.3333	0	0	0.3333	0	0	0	0	0	0

Table 8.3 Neutral to Fear success rates. The number of hidden states is 20.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.2	0.2	0.4	0.6	0.4	0.6	0.6	0.8	0.2	0.8
30	0.2	0.4	0.4	0.2	0.4	0.2	0.6	0.2	0.6	0.6
50	0	0.2	0	0.2	0	0.4	0.4	0.4	0.2	0.4
100	0.2	0.2	0.2	0.2	0	0	0	0.4	0	0.4
200	0.2	0.2	0	0	0	0	0.2	0	0	0.2

Table 8.4 Neutral to Happy success rates. The number of hidden states is 20.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.6667	0	0.3333	0.5556	0.3333	0.3333	0.3333	0.2222	0.4444	0.2222
30	0.3333	0.2222	0.4444	0.5556	0.2222	0.4444	0.3333	0.3333	0.3333	0.6666
50	0.1111	0.3333	0.5556	0.4444	0.2222	0.4444	0.3333	0.4444	0.4444	0.2222
100	0.6667	0.2222	0.5556	0.7778	0.3333	0.4444	0.2222	0.4444	0.5555	0.5556
200	0.2222	0.1111	0.1111	0.6667	0.2222	0.4444	0.4444	0.2222	0.1111	0.2222

Table 8.5 Neutral to Sad success rates. The number of hidden states is 20.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.2857	0	0.1429	0	0	0.1429	0	0	0.1429	0
30	0.2857	0	0.1429	0.1429	0.2857	0	0	0.2857	0.4286	0.1429
50	0.4286	0.1429	0	0	0.4286	0.5714	0.1429	0	0.2857	0.2857
100	0.1429	0.4286	0.2857	0.1429	0.2857	0.2857	0	0.2857	0.2857	0
200	0	0	0.1429	0	0	0	0	0	0	0

Table 8.6 Neutral to Surprise success rates. The number of hidden states is 20.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.8571	0.4286	0.2857	0.8571	0.4286	0.4286	0.4286	0.5714	0.4286	0.4286
30	0.8571	0.4286	0.1429	0.5714	0.4286	0.1429	0.4286	0.5714	0.4286	0.5714
50	0.2857	0.7143	0.1429	0.2857	0.1429	0.2857	0.5714	0.1429	0.2857	0.2857
100	0.7143	0.7143	0.4286	0.7143	0	0.4286	0.2857	0.4286	0.7143	0.2857
200	0.1429	0.1429	0	0.1429	0.1429	0.1429	0.4286	0.1429	0.4286	0.1429

The tables 8.7 to 8.12 show the results that were generated using the second approach (combined centroids). The second approach seems to produce slightly better results than the first approach. Hence further experiments were performed using this approach.

Here again the rows and columns of the tables have the same meaning as described for the previous set of results.

Table 8.7 Neutral to Angry success rates with combined centroids.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.6667	0.6667	0.3333	1	0.6667	0.6667	0.6667	0.6667	0.3333	0
30	0.3333	0.6667	0.6667	1	0	0.6667	0.3333	0.6667	0	0
50	1	1	0.6667	0.6667	0	1	0.6667	0.6667	0.3333	0
100	1	1	1	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	1
200	1	0.6667	1	1	0.6667	1	0.6667	0.6667	0.6667	1

Table 8.8 Neutral to Disgust success rates with combined centroids.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0	0	0	0	0	0.6667	0	0	0	0
30	0	0	0.3333	0	0	0	0	0	0	0.3333
50	0.3333	0.3333	0.6667	0	0	0	0	0	0	0
100	0	0	0.3333	0	0	0	0.3333	0	0.3333	0
200	0	0	0	0	0	0	0	0	0	0

Table 8.9 Neutral to Fear success rates with combined centroids.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.4	0.6	0.4	0.6	0.6	0.6	0.6	0.8	0.6	0.8
30	0.6	0.2	0.4	0.2	0.8	0.4	0.8	0.6	0.2	0.6
50	0.6	0	0.2	0.4	0.4	0.4	0	0.6	0.8	0.4
100	0.4	0.2	0	0	0.4	0.4	0.8	0.8	0.2	0
200	0.2	0	0.2	0	0.2	0	0	0	0.2	0.2

Table 8.10 Neutral to Happy success rates with combined centroids.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.8889	0.5556	0.7778	0.8889	0.7778	0.7778	0.8889	0.7778	0.8889	1
30	0.8889	0.6667	0.8889	0.7778	0.5556	0.8889	0.8889	0.7778	0.5556	0.8889
50	0.8889	0.7778	0.7778	0.7778	0.6667	1	0.7778	0.7778	0.6667	0.7778
100	0.6667	0.8889	0.6667	0.8889	0.7778	0.6667	0.6667	0.5556	0.8889	0.4444
200	0.4444	0.3333	0.2222	0.5556	0.4444	0.2222	0.5556	0.4444	0.5556	0.3333

Table 8.11 Neutral to Sad success rates with combined centroids.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
---------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	---------

20	0.2857	0	0	0	0.4286	0.4286	0.1429	0	0.1429	0.2857
30	0.2857	0	0	0.1429	0.4286	0.5714	0.2857	0.1429	0.5714	0.2857
50	0	0.1429	0	0.8571	0.2857	0.1429	0.2857	0.4286	0.2857	0.4286
100	0.1429	0	0	0.1429	0	0.1429	0	0.1429	0	0
200	0	0	0	0	0	0	0	0	0	0

Table 8.12 Neutral to Surprise success rates with combined centroids.

No. of groups	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
20	0.4286	0.7143	0.5714	0.8571	0.7143	0.4286	0.8571	0.7143	0.8571	0.7143
30	0.4286	0.5714	0.7143	1	0.7143	0.7142	0.5714	0.8571	0.7143	0.7143
50	0.7143	0.7143	0.5714	0.8571	0.5714	0.4285	0.8571	0.7143	0.5714	0.4286
100	0.5714	0.5714	0.4286	0.7143	0.5714	0.8571	0.8571	0.5714	0.8571	0.5714
200	0.4286	0.4286	0.4286	0.4286	0.4286	0.426	0.4286	0.2857	0.4286	0.4286

During the initial experiments that were done it was observed that with the current structure of the HMMs the best results were produced when the number of hidden states were equal to half the number of observable states. The following results were produced using this information. Since the HMMs with 50, 30 and 20 observable states performed better than most of the other HMMs the experiments were carried out using these parameters. Tables 8.13 to 8.15 show the results of these experiments. In these tables the first column indicates each of the six basic expressions for which the success rates (correct recognition) were computed. The remaining columns have same definitions for rounds. The last column shows the average success rates over all the rounds. The last column in last row indicates the average performance of the entire system to correctly recognise the expression in the image sequences over the entire database.

Table 8.13 Results with 25 Hidden States, 50 Observable States

Expression	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10	Avg.
AN	0.6667	0.3333	0.3333	0.6667	0.6667	0.6667	0.6667	0.3333	0.3333	0.3333	0.5
DI	0.6667	0.6667	0.6667	1	0	0.6667	0.6667	0.3333	1	0.6667	0.6333
FE	0.5714	0.8571	0.7143	0.5714	0.7143	0.5714	0.5714	0.4286	0.7143	0.2857	0.6
HA	0.8889	0.7778	0.7778	0.8889	0.6667	1	0.7778	0.6667	0.8889	1	0.8333
SA	0.5714	0.8571	0.7143	0.5714	0.7143	0.5714	0.5714	0.4286	0.7143	0.2857	0.6
SU	0.7143	0.8571	0.5714	0.8571	0.7143	0.5714	0.8571	1	0.7143	0.2857	0.7143
Avg.											0.6468

Table 8.14 Results with 15 Hidden States, 30 Observable States

Expression	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10	Avg.
AN	0.6667	0.6667	0.6667	0.6667	0	0.6667	0.3333	1	0	0.3333	0.5
DI	0.6667	0.3333	0.3333	0.6667	0	0.3333	0	0.6667	0.3333	0	0.3333
FE	1	0.7143	0.7143	0.7143	0.5714	0.8571	0.5714	0.7143	0.2857	0.5714	0.6714
HA	0.8889	0.8889	1	1	1	0.7778	1	0.6667	0.7778	1	0.9
SA	1	0.7143	0.7143	0.7143	0.5714	0.8571	0.5714	0.7143	0.2857	0.5714	0.6714
SU	0.5714	0.7143	1	0.7143	0.8571	0.8571	0.7143	1	0.5714	0.7143	0.7714
Avg.											0.6413

Table 8.15 Results with 10 Hidden States, 20 Observable States

Expression	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10	Avg.
AN	1	0.6667	0.3333	0.6667	0.6667	1	0.6667	1	0.3333	0.3333	0.6667
DI	1	0.6667	0	1	0.3333	0.3333	0.3333	0.6667	0	0	0.4333
FE	0.7143	0.7143	0.4286	0.8571	0.4286	0.5714	0.5714	0.5714	0.4286	0.8571	0.6143
HA	1	0.6667	0.6667	1	0.6667	0.8889	0.7778	0.7778	0.7778	0.8889	0.8111
SA	0.7143	0.7143	0.4286	0.8571	0.4286	0.5714	0.5714	0.5714	0.4286	0.8571	0.6143
SU	0.8571	0.8571	0.5714	0.8571	0.8571	0.8571	0.7143	0.7143	0.8571	0.7143	0.7857
Avg.											0.6542

Some additional experiments were performed using 20 observable states and by varying the number of hidden states but the results did not improve. As compared to some of the state-of-the-art approaches this approach still lacks in success rates (state-of-the-art approaches have success rates of 92-95%). To improve the quality of results some modifications to the existing approach was needed. Some of these modifications were suggested and are explained in the section 8.2.

8.2. Suggestions

Some important observations were made during the experiments. Firstly the number of image sequences varied significantly. The least number of images in a sequence was as low as 9 and the most number of images in another sequence was as high as 43. Although the HMMs trained over these sequences may be more robust, since the number of such sequences also varied these sequences could prove to be insufficient to train the HMMs. As an alternative a fixed number of sequences could be used for training for further evaluation of the approach. Secondly the Optical flow was always calculated for two consecutive images in a sequence, instead a new approach would be to calculate the optical by keeping the first image of the sequence as the reference

image. This would give the optical flow of corresponding points in the image with respect to the first image. This approach could be used in finding the expression intensity and not just the classification of the expression. Thirdly the experiments with HMMs were made using feature vectors extracted by applying optical flow. There are other ways of representing the face images including PCA, LDA and LBP. These feature extraction techniques could be used along with HMMs to evaluate the use of HMMs for the purpose of emotion recognition. For the incorporating the temporal information into the approach the difference of feature vectors could be used.

8.3. Conclusion

The report explains two different approaches for facial emotion recognition. The first approach using LBP was partially implemented on Xetal2 processor. The LBP calculation for face image as well as region wise histogram generation was implemented on Xetal2. This work was done as a part of the internship work at Philips. The second approach involving Hidden Markov Models was evaluated using Matlab. The approach shows an overall recognition of 65% for transition from neutral to six basic emotion expressions. To improve the recognition rate some suggestions were made which involved the feature extraction being made with respect to the first image in the sequence. The approach as yet is not capable of evaluating emotion intensities. Some suggestions were made so that the system is made capable of evaluating emotion intensities. The initial feature extraction procedure plays an important role when being implemented on an embedded device. The approach could also be used along with a simple feature extraction technique so as to make it possible to be implemented on an embedded device.

9. References

1. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, Kauai, HI, December 2001.
2. T. Ahonen, A. Hadid, and M. Pietikäinen, "Face Recognition with Local Binary Patterns," *Proc. Eighth European Conf. Computer Vision*, pp. 469-481, 2004.
3. M. Turk and A. Pentland, "Face recognition using eigenfaces," Proc. IEEE Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, 1991.
4. Xiangsheng Huang , Stan Z. Li , Yangsheng Wang, Jensen-Shannon Boosting Learning for Object Recognition, Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2, p.144-149, June 20-26, 2005
5. B. Horn and B. Schunck. Determining optical flow. In Artificial Intelligence, volume 17, pages 185--204, 1981. 2
6. Kohonen, T. Self Organizing Maps; Springer Series in Information Sciences, Springer: Espoo, Finland, 1994.
7. Ekman, P., "Facial Expression and Emotion," American Psychologist, Vol. 48, pp. 384-392, 1993.
8. Fridlund, A.J., Human Facial Expression: An Evolutionary View, Academic Press, San Diego, CA, 1994.
9. Cohn J.F., and Elmore, M., "Effect of Contingent Changes in Mothers' Affective Expression on the Organization of Behavior in 3-Month-Old Infants," *Infant Behavior and Development*, Vol. 11, pp. 493-505, 1988.
10. Black, M.J., and Yacoob, Y., "Tracking and Recognizing Facial Expressions in Image Sequences, Using Local Parameterized Models of Image Motion," University of Maryland, Technical Report CS-TR-3401, January 1995.
11. Black, M.J., Yacoob, Y., Jepson, A.D., and Fleet, D.J., "Learning Parameterized Models of Image Motion," IEEE Conference on Computer Vision and Pattern Recognition, pp. 561-567, Puerto Rico, June 1997.
12. Essa, I.A., "Analysis, Interpretation and Synthesis of Facial Expressions," MIT Media Laboratory, Perceptual Computing Technical Report 303, Ph.D. dissertation, February 1995.

13. Mase, K., "Recognition of Facial Expression from Optical Flow," Institute of Electronics, Information and Communication Engineers Transactions, Vol. E74, pp. 3474-3483, 1991.
14. Rosenblum, M., Yacoob, Y., and Davis, L.S., "Human Emotion Recognition from Motion Using a Radial Basis Function Network Architecture," University of Maryland, Technical Report CS-TR-3304, June 1994.
15. Yamato, J., Ohya, J., and ISHII, K., "Recognizing Human Action in Time-Sequential Images Using Hidden Markov Model," IEEE International Conference on Computer Vision, pp. 379-385, 1992.
16. Darwin, C., *The Expression of Emotions in Man and Animals*, John Murray, 1872, reprinted by University of Chicago press, 1965.
17. Bartlett, M.S., Viola, P.A., Sejnowski, T.J., Golomb, B.A., Larsen, J., Hager, J.C., and Ekman, P., "Classifying Facial Action," *Advances in Neural Information Processing Systems* 8, pp. 823-829, MIT Press, Cambridge, MA, 1996.
18. Murase, H., and Nayar, S.K., "Visual Learning and Recognition of 3-D Objects from Appearance," *International Journal of Computer Vision*, Vol. 14, No. 1, pp. 5-24, 1995.
19. Mase, K., and Pentland, A., "Automatic Lip-reading by Optical-Flow Analysis," *Systems and Computers in Japan*, Vol. 22, No. 6, pp. 67-76, 1991.
20. Kobayashi, H., and Hara, F., "The Recognition of Basic Facial Expressions by Neural Network," *Proceedings of International Joint Conference on Neural Network*, pp. 460-466, 1991.
21. Kobayashi, H., and Hara, F., "Recognition of Six Basic Facial Expressions and their Strength by Neural Network," *IEEE International Workshop on Robot and Human Communication*, pp. 381-386, September 1992.
22. Kobayashi, H., and Hara, F., "Recognition of Mixed Facial Expressions by Neural Network," *IEEE International Workshop on Robot and Human Communication*, pp. 387-391, September 1992.
23. James J. Lien, Takeo Kanade, Adena J. Zlochow, Jeffrey F. Cohn, and Ching-Chung Li, "Automatically Recognizing Facial Expressions in the Spatio-Temporal Domain.", *Workshop on Perceptual User Interfaces*, pp. 94-97, Banff, Alberta, Canada, October 19-21, 1997.
24. T. Ojala, M. Pietikainen and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition* vol. 29, 1996.
25. Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77 (2), 257-286.

26. S. Balakrishnama and A. Ganapathiraju. Linear discriminant analysis - a brief tutorial. Institute for Signal and Information Processing, 1998.