Automatic recognition of facial expressions in linedrawings

Jiří Filip, Luděk Holoubek

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Autumn stay at Delft University of Technology
Faculty of Information, Technology and Systems
Department of Mediamatica/Knowledge Based Systems

November 29, 2000

Contents

Acknowledgements

This work was carried out thanks to grant of SOCRATES/ERASMUS students interchange program between Faculty of Electrical Engineering at Czech Technical University in Prague and Faculty of Information, Technology and Systems at Delft University of Technology. We'd like to devote our acknowledgement to Prof.Ing. Václav Hlaváč CSc. (Department of cybernetics at CTU Prague) and drs.dr. Leon J.M. Rothkrantz (Department of Mediamatica/Knowledge Based Systems at TU Delft) for realising of our stay in Delft. We'd like also to thank to drs. dr. Leon J.M. Rothkrantz for valuable and helpful advices during our work and for careful check-up of this report.

Abstract

In present world communication between people play very important role. Computers are important part of this communication chain. To enable to make communication more effective is useful to use also other sources of information than typing or speech. In this work we concerned on studying possibilities of non-verbal communication especially by face expression. We used in our work for simplification linedraws instead of photos. From this images we obtained information about basic facial features and for this data we performed statistical operations. Finally we trained Neural Network and Bayesian classifier for evaluation of unknown data.

Chapter 1

Introduction

Many people over the world are using their computer for communication. There are many ways how to communicate; by electronic mail, by chat, by phone, by videophone, meeting in a virtual space, etc. Communication by electronic mail isn't very dynamic, phone and videophone needs very fast connection. This last method has some other disadvantages. Very popular is interactive communication like chat or meeting in a virtual space. These methods don't need fast connection and have no redundancy in transfer of data. But the disadvantage is, that people aren't able to write everything they want by keyboard so fast they want. Non-verbal communication is useful for control of communication. Very useful can be a non-verbal keyboard with special buttons to express some emotions, moods, opinion, etc. Very popular are smiles, small character-coded faces pushed at the end of sentence. For example a message sent through chat: We can go at the party :-). To express emotions connected with special buttons some system can use picture and send it to another system. The other system need to interpret what emotion is displayed in the received picture (picture can be users photo or line drawing). Our work is focused at the problem of the interpretation of facial expression from the picture or linedraw picture. Every facial expression is characterised by a specific contour of the eyes, mouth and eyebrows. For interpreting, recognition of face expression, classifying to the six basic classes(happiness, sadness, anger, fear, surprise and disgust), we used and developed some tools. We developed a tool for automatic extraction of characteristic points from line draw pictures. And we classified this data by a feedforward Neural Network and Bayesian classifier. Our data, the linedrawing pictures were generated by an already developed applet. We have had good results for artificial data generated by the applet. We used PCA to get better knowledge about our data.

The face expressions are connected with emotions which are an inseparable part of human communication - non-verbal communication.

Chapter 2

Theory

The problem to solve is recognising emotions of people. Our system is designed for the representation of facial expressions from indirect data. It means that we have no contact with someone who creates this expression. We have only a picture from a frontal-view. Next discuss will be only about how to recognise facial expression from the frontal-view. First we must define, which expressions we will have to interpret. There are many studies [3],[4] about the most common expressions. Ekman used six basic emotions: happiness, sadness, surprise, fear, anger and disgust [3]. In his work are these basic emotions universally and uniquely described. This specification is very often used by many researchers in this area [2], [4]. There is Johansson's point-light display experiment certifyied by Bassili [11], and later by Bruce [12]: People were able to recognise a facial expression from a few points(white marks) of the face transfered through a monitor. The points belong to the contours of the eyebrows, eyes, mouth, nose and chin, areas with many information about the presented expression. This points are called characteristic points. There are many face models to define the position of characteristic points in the face. We decided to use the model of Kobayashi (*Figure 2.1*).

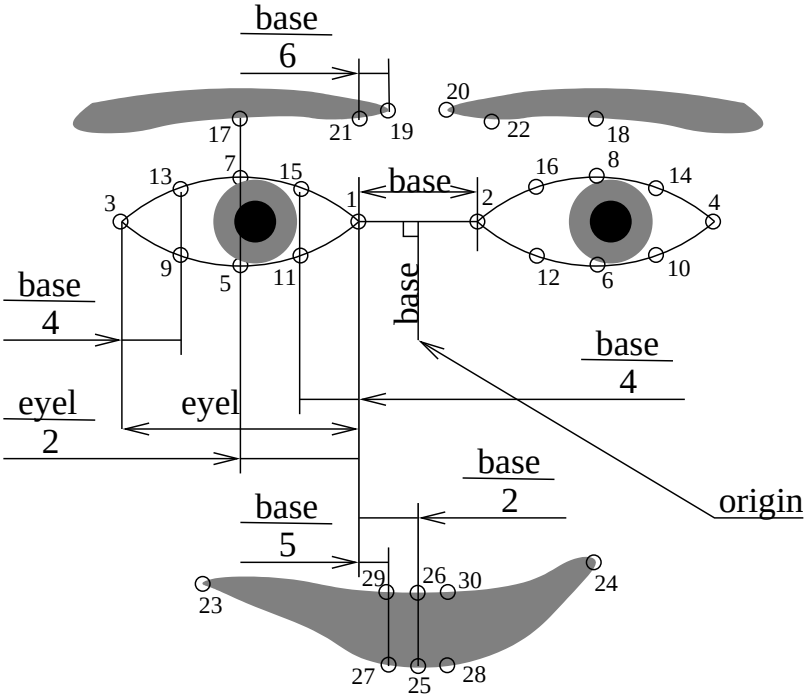


Figure 2.1: Face Model of Kobayashi with characteristic points

The model contains 30 points. But you need 60 coordinates to describe these points, you can expect big redundancy here. To reduce the ambiguity of the data we computed 21 distances between some characteristic points according to the model of Kobayashi [2]. How to obtain 21 data from 60

data is showed in *Appendix D*. Kobayashis model is defined for the human face and similar faces. It is not possible to use this model to describe some cartoons pictures. See *figure 2.2*. Rules for the used model of Kobayashi (2.1):

$$\frac{EYE}{2} > \frac{BASE}{4} \quad , \quad \frac{MOUTH}{2} > \frac{BASE}{5} \tag{2.1}$$

If these equations are not satisfied, we have to adapt the parameters to get Kobayashi-like models.

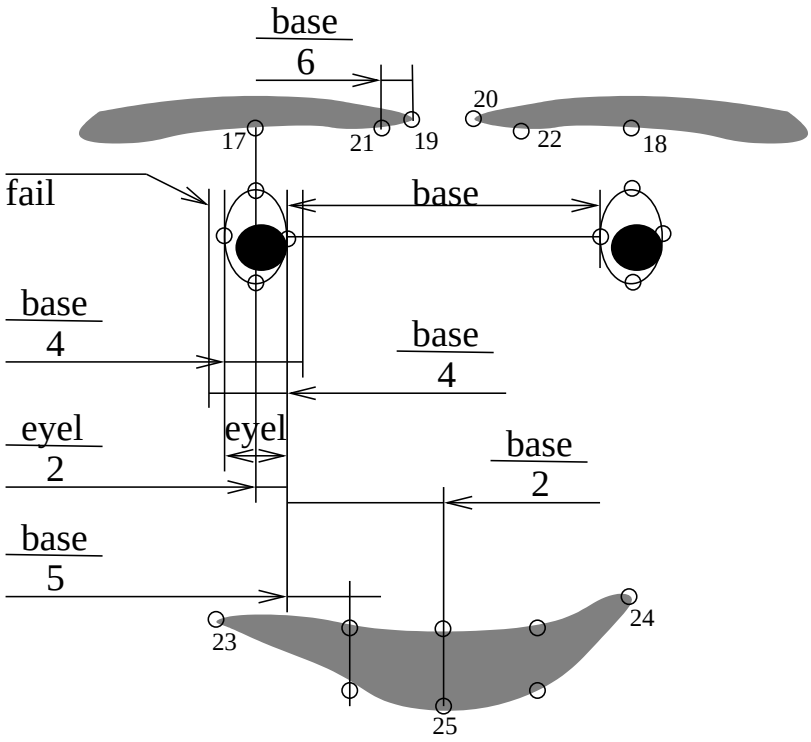


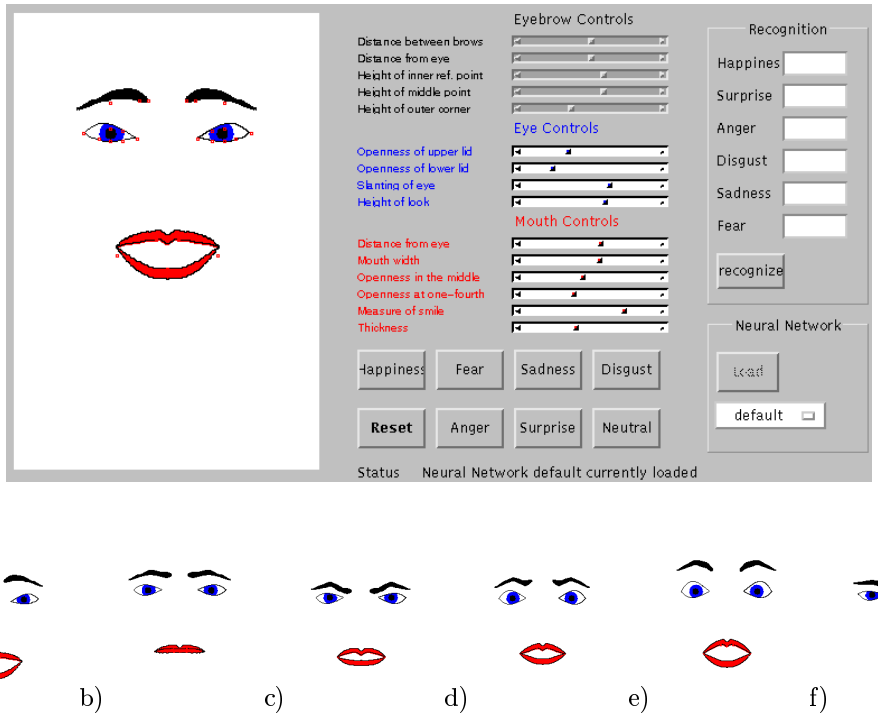
Figure 2.2: Face model of Kobayashi - showing the constraints of the model

Chapter 3

Tools

During our work we used the following tools:

1. For generating images of linedrawings for our system we used a Java applet. The applet was made by the Group of Knowledge Base Systems at TUDelft. We had use a frame grabber to transfer the pictures to the graphics format we needed. There is a possibility to change facial expressions very easy by sliders in the applet (*Figure 3.1*). It isn't a problem to produce almost any facial expression we need.



a) b) c) d) e) f)

Figure 3.1: Snapshot of a Java applet Face linedrawings generator and some obtained emotions: a) happiness b) sadness c) anger d) fear e) surprise f) disgust

2. To convert an image, loading, face cutting, face settlement, face feature extracting and finding of characteristic points as you can see on *Figure 4.1*, we made in C++ under UNIX. These tools are designed like an executable file. It has an input and an output file. There are many switches to change its behaviour here. An input file is an image and an output file is a set of characteristic points. We have used output format compatible with application for Neural Networks. We worked with standard libraries and with *QT 1.44* libraries for graphic visualisation.
   For more complex mathematical matrix computation (SVD, PCA) we used a mathematical

library MatClass 1.0d written in C++. This package can be downloaded from the internet page University of Manchester (ftp://ftp.mcc.ac.uk/pub/matclass/pc/).

3. As a last step we applied Stuttgart Neural Network Simulator v 4.1. The SNNS application was developed at the University of Stuttgart, Institute for Parallel and Distributed High Performance Systems and available as public domain software on internet (http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/). With this application it is very easy to design a neural network, its training, testing and visualisation. It has a very helpful graphic user interface.

4. For 3D plotting of our PCA results of our program and for implementation of Bayesian classifier we used MATLAB 5.2 (under the Windows NT) and Octave version 2.0.16 (under SunOS 5.7).

5. We used program SPSS Professional Statistics 6.1 - version for Sun Solaris OS 5.7 to test our PCA results and to perform more complex statistical operations (Varimax rotation, etc.).

Chapter 4

Implementation of a facial expression recognition system

The goal of our work was to make an automatic system for facial expression recognition. To solve this problem we split it up in two parts. The first part consists of extracting the characteristic points from a face linedrawing according to the model developed by Kobayashi [2]. The second task is to recognise emotions from the distances between characteristic points. For solving this task we used Neural Networks and Bayesian classifier. We divided the first part of our work to more consequently performed operations (*Figure 4.1*). Each of this operations is described in sections below.

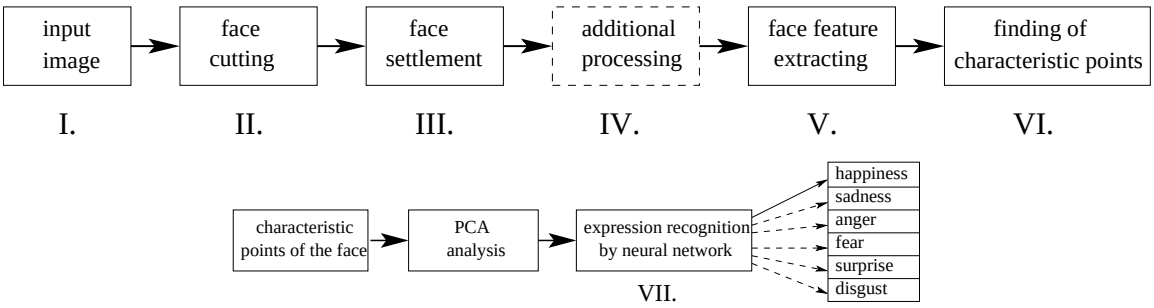


Figure 4.1: Flowchart of image analysis

4.1 Input image

The input of our program is an image in GIF or BMP format of arbitrary resolution and colour palette. For our purposes we suppose that the face linedrawings contains both *eyes* with *eyebrows* and *mouth* which are not stuck together. We also assume that the image is free of noise and the only dark points are placed in face features.

In fact for testing we used two different kinds of images :

- images generated by the Java applet which was described in Chapter 3 (*Appendix B*)
- images drawn by hand (*Appendix C*)

Because input pictures aren't photos but only linedrawings of eyes and mouth, our program automatically converts them into black&white images, for easier postprocessing.

4.2 Face cutting

Since the proportion from the face features in an input image (eyes with eyebrows and mouth) can be varied it is necessary for following efficient work to cut only important part of the image which

contains these important features. This we performed in a very easy way. We moved from all four sides of the image to opposite sides and in each direction we set a new end of the image in a position where we found some dark pixels. As an optional parameter of this function it is possible to set some neighbourhood which will enclose the cut image. Procedure of cutting of the face from the image is displayed in *Figure 4.2*.

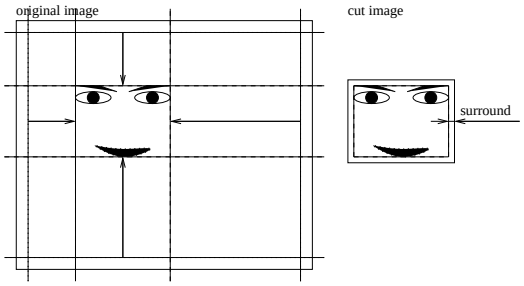


Figure 4.2: Procedure of cutting of the face from the image

4.3 Face settlement

Now we already have an image, which is only an area which contains both eyes with eyebrows and mouth. The next problem which may occur is that eyes are not aligned to the horizontal direction of the image, but rotated. In this case it is needed to perform back rotation. The question is now how to get the value of the angle by which the image is rotated. There is many possible ways how to solve this problem. One of them is to use some simple edge detector to find edges in the image. Then for every row and column we computed the amount of black pixels which are derived from edges (histogram). After this we will gradually rotate the picture for some value (in our case 5 degree - optimal for speed and program functionality) around the centre of the image and try to find in this histograms , which are represented by amounts of black pixels in rows and columns, some characteristic features which representing the begin and end of eye, mouth etc. The goal is to find the rotation (angle) in which will be these characteristic peaks the biggest. We tried this method but we found that results were ambiguous for our linedrawings. Thus we performed little simplification which consist in watching of histograms in both directions through rotation. The image is properly rotated if the vertical histogram contains the lowest number of items while the horizontal histogram is the most symmetric according centre of the image *Figure 4.3*. This method works well in almost all cases of rotated images.

Figure 4.3: Two examples of using edge detection for recognising the angle of rotation of an image

Another possible approach which we also tried is rotating of axis leading through center of the image and observing of similarity between both sides of this axis (*Figure 4.4*).

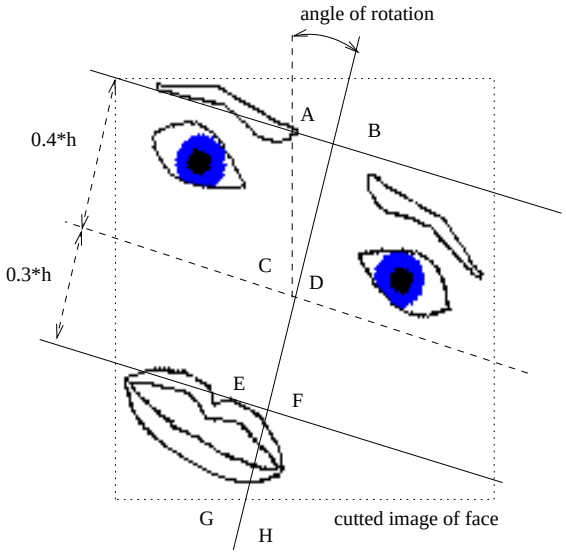


Figure 4.4: Finding of angle of image rotation by using similarity between areas A-B, C-D, E-F, G-H in the image

We divided observed area to eight parts A..H and we computed similarity between A and B, C and D, E and F, G and H by *equation 4.1.*

$$J = \left| \frac{n_A}{n_A + n_B} - \frac{n_B}{n_A + n_B} \right| + \left| \frac{n_C}{n_C + n_D} - \frac{n_D}{n_C + n_D} \right| + \left| \frac{n_E}{n_E + n_F} - \frac{n_F}{n_E + n_F} \right| + \left| \frac{n_G}{n_G + n_H} - \frac{n_H}{n_G + n_H} \right| \quad (4.1)$$

Border between areas A,B and C,D was set to 0,4 of height of image and border between areas E,F and G,H was set to 0,3 of height of image. We obtained this values by testing. Where $n_A, n_B, n_C, n_D, n_E, n_F, n_G, n_H$ are amounts of black pixels in areas A,B..,H. We rotated this axis in picture and choosed as appropriate the angle where had this function minimum. This method was less successful than method based on observing of histograms. Hence we finally used first mentioned method.

4.4 Additional processing

By images which are drawn by hand or by rotated images there may occur some holes in the edges (edges discontinuity). This problem we tried to eliminate using a morphological operation *closing*, which consists gradually of dilatation and erosion by structure elements 3x3 . A problem may occur when the resolution of an input image is too low. In this case it is necessary to disable this morphology block or at least to choose a smaller structure element. Hence is this block in *Figure 4.1* depicted by dashed line - it is optional block.

4.5 Face features extracting

The goal of this function is to find areas which contains eyes with brows and mouth. The outer coordinates (closed to the edges of the image) of these areas were easy to find by moving from the edges of the image to the centre from all four direction while we didn't find some black pixels. The worse problem was finding inner coordinates of this areas. Therefore we have used an averaging mask of certain width which moves for example from the top till the bottom of the image. In the place where the average of mask is the lowest, there is a border between eyes and mouth. Then we could use the same method (histograms along axis) for separating only the eyes in a horizontal direction. Thus the output of this function are coordinates of three area containing left eye with left eyebrow, right eye with right eyebrow and mouth.

4.6 Finding of the characteristic points

This operation we can divide into several steps(*Figure 4.5*):

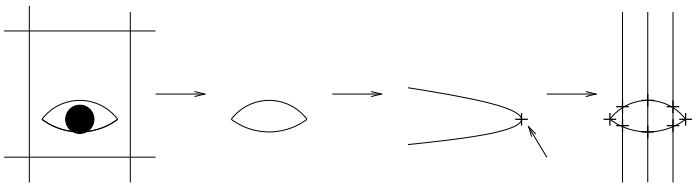


Figure 4.5: Steps in procedure to find the characteristic points

1. Contour following by class "edgeagent", class "curve"
2. Localising eye corners in a robust way
3. Extract characteristic points from a class curve
4. Processing of data, transformation

4.6.1 Contours following by a class edgeagent, class curve

To extract edges we have used a method which is contained in the class edgeagent. First it is necessary to set which picture we are using now, after this we have to set a starting point, direction(possible directions are 0-West, 1, 2-North, 3, 4-East, 5, 6-South, 7)(*Figure 4.6*) and environment(4-default or 8). Starting points are computed from the coordinates returned by Face features extracting(*Figure 4.7* left).

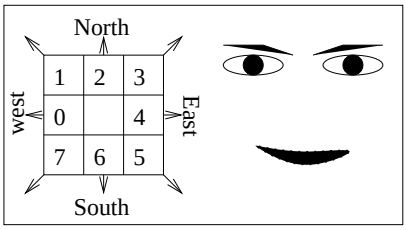


Figure 4.6: Explaining of used directions

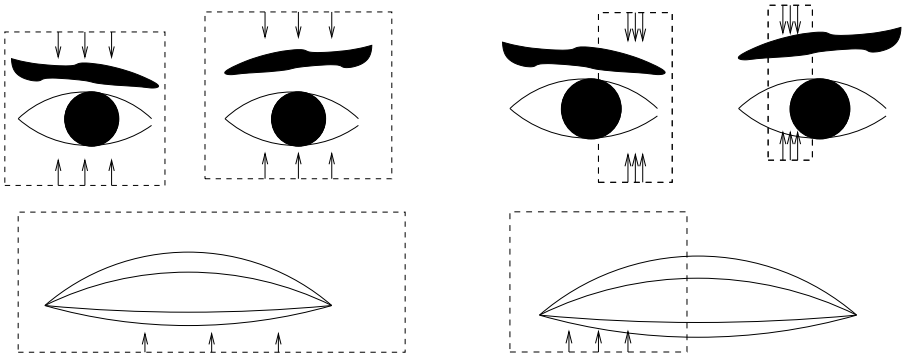


Figure 4.7: Starting points of Edgeagents

Startup points:

- for the mouth it is the middle point of the bottom rectangular area side of mouth-box, direction is set to North (2),
- for the eye it is the middle point of the bottom rectangular side of the eye-eyebrow-box, direction is set to North (2),
- for the eyebrow it is the middle point of the top rectangular side of the eye-eyebrow-box, direction is set to South (6).

After setting the startup points the method is started. We will call this method edgeagent. The edgeagent goes from the starting point, pixel by pixel, in the direction which was set. In each step the edgeagent controls its environment (*Figure 4.8* a)), it means eight pixels. If all of the eight points are background, then the edgeagent continues in direction. In case some of the eight points is foreground, this phase is interrupted. The next phase is following an edge we found(*Figure 4.8* b). A method for following the edge is described in *Table 4.1*:

Table 4.1: Edgeagent's rules

	Edgeagent's rules
1	if the actual point is the same as the first point on edge then STOP
2	if count of steps is greater than maxCurveLength then STOP
3	only directions for following the edge are ALLOWED (*Figure* 4.8 c)
4	recompute founded directions from 8-environment to 4-environment (*figure 4.9* d)
5	going back is PROHIBITED (*Figure 4.9* e)
6	going through one pixel hole("strait") is PROHIBITED (*Figure 4.9* f)
7	follow recomputed direction and SAVE points in environment into class curve
8	SAVE "strait" points into class curve too
9	if saving points are already in, don't push them into class curve

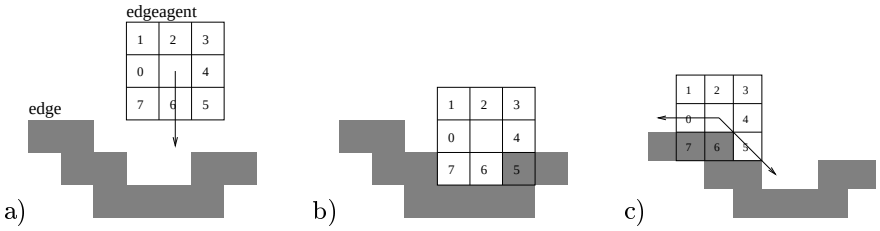


Figure 4.8: a)Looking for an edge, b)an edge was found, c)following the edge.

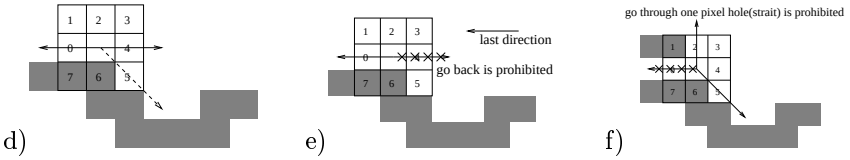


Figure 4.9: d)Recomputing from 8-environment to 4-environment, e)restriction to go back, f) restriction to go through hole.

By applying this rules from (*Table 4.1*), we will get a class curve, which represents the edge of the mouth, eyebrows and eyes. In case, that some input picture isn't typical or some part (for example eye) is missed, there is constraints here. First the maximal allowed length of the founded edge and the maximal allowed length from the starting point to an edge. This insurance you can see as second rule above. Into class curve are for example *Figure 4.9* f) put surround points 0, 1, 6, 7. Everything is computed in 8-environment, but directions are recomputed to 4-environment. Because there were causes here, where edgeagent always failed(never ending loop), when we used eight environment(*Figure 4.10*). This method has some advantages. If the boxes for mouth and eyes are computed badly, the edgeagent will find the correct edges(*Figure 4.7* right).

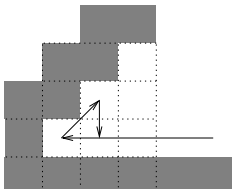


Figure 4.10: Example how the edgeagent can fail

4.6.2 Looking for eye corners in a robust way

How to find the corner of the eye? The most easy way is to get from the class curve max(x) value and after that get the corresponding y-points. And from the y-points to take for example median(y-points). But as you can see on *Figure 4.11*, this method is able to be a source of mistakes. We compute the main dimension "BASE" from the distance between left and right corner of the eye. And after this we divide all distances by the "BASE". In this way we get "normalised" coordinated of distances. That is why we need a very robust method without mistakes.

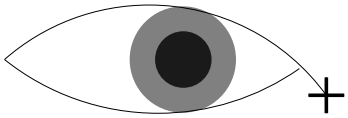


Figure 4.11: Example how easy method is able to fail

Therefore we have used a more robust method, fitting points by an adequate curve. The curve is a parabola. We start our fitting process with the first third of the eye. The best solution we found as a result with least squared error. As equation we used a parabola: $y = ax^2 + bx + c$. We have decided to use this equation for easier implementation. We must rotate approaching points because with this equation it is impossible to produce a parabola only in one direction. Parameters of the parabola are the result of computing (*equation 4.2*):

$$XA = Y, \qquad X^T XA = X^T Y, \qquad A = (X^T X)^{-1} X^T Y \tag{4.2}$$

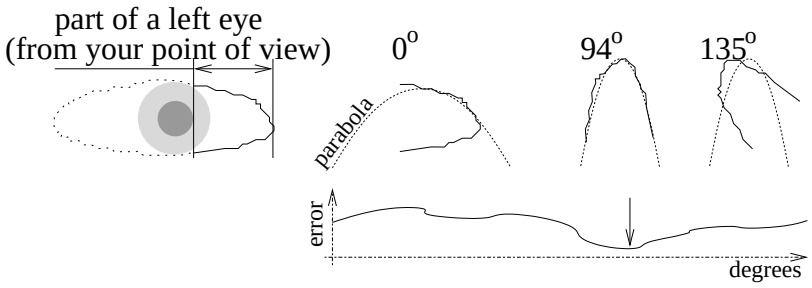


Figure 4.12: Method for fitting of a parabola

This computing fits the parabola (*Figure* 4.12) with least squared error. For each rotation we compute squares error between couples of the original point and a point of the parabola. Our solution are parameters of the parabola with least squared computed error (*figure 4.12*). The eye corner is found as a global extreme point of the founded parabola and its position is rotated back. From these both correct eye corners we compute the distance "BASE".

4.6.3 Extract characteristic points from the class curve

The position of the characteristic points is depicted in *figure 2.1.* Extracting points from classes curve is very easy. For example extracting of point 5 (*equation 4.3*).

$$x = point[1].x + \frac{eye}{2}, \qquad y = max(lefteye.getY(x)) \tag{4.3}$$

In this way we extract all the points. From the coordinates of the characteristic points 1., 2. and BASE we computed a new origin of coordinates(*Figure 2.1* centre of face). All points are recomputed into these new coordinates.

4.6.4 Processing of data, transformation

We have to be independent of the size of the picture, therefore all coordinates of the characteristic points are divided by BASE. This set of coordinates is saved to an output file. After that the coordinates of the characteristic points are transformed. From 30 x-coordinates and 30 y-coordinates to 21 distances. This transformation you can find in *Appendix D.* This new set of distances is saved to another file. Both files have a structure which is compatible with SNNS application.

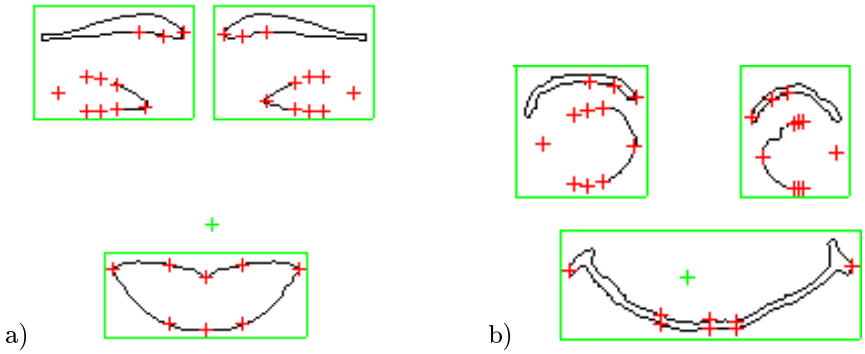


Figure 4.13: Results of the point extractor for a) an applet image b) a handmade image

4.7 Principal Component Analysis

As input data to NN we can use 60 data vector, which corresponds with 30 coordinates of the characteristic points, or we can use recomputed data set represented by 21 data vector [2]. But in this kind of representation of input data is still plenty of redundant information, which is independent of the expressed emotion. This redundancy can lead to confusion of the Neural Network and to avoid worse results. Hence we need to extract from our input data the most significant information, it means to reduce input data set into smaller one but with preservation of the most important features.

For dealing with this problem we found useful a standard method called Principal Component Analysis (PCA). Basic assumption is that for using PCA we are expecting input data with normal distribution.

4.7.1 Standard PCA computed from covariance matrix

This method is based on the analysis of the eigen values of the covariance matrix, which is created from the input data matrix.

Creation of covariance matrix The number of rows in our input data matrix corresponds with the size of the input data vector and the number of columns corresponds with the amount of patterns. Thus every column vector X_i represents data extracted from one image. From this matrix we computed mean values of all the rows and got a mean column vector. By subtraction of this vector from all column vectors in the input matrix we got the variance matrix B *equation 4.4.*

$$B = [X_1 - \widehat{X}, X_2 - \widehat{X}, ...X_n - \widehat{X}] \tag{4.4}$$

where $(X_1, X_2, ...X_n)$ are input vectors and $\widehat{X}$ is the already mentioned mean vector computed from the rows. The pattern covariance matrix we got by *equation 4.5.*

$$S = \frac{1}{N-1} B \cdot B^T \tag{4.5}$$

where N is the number of patterns (linedrawings).

Reducing the input data On the covariance matrix we performed Singular Value Decomposition (SVD) and received eigen values and eigenvectors of this matrix. Since the covariance matrix is symmetric we received eigenvectors directly as column vectors of the matrix S. After this we save only the eigen vectors which corresponded with bigger eigenvalues than our threshold. Amount of these selected eigenvectors is also the number of saved Principal Components. At the end we need to recompute our input data by matrix consists only of these saved eigenvectors by *equation 4.6.*

$$U = P^T \cdot X \tag{4.6}$$

where U is the new data matrix and P is mentioned set of the saved eigenvectors.

Representing of PCA results We created a short program in C++, which is able to reduce the amount of input data and left only the most important data depending on their variance. As input file is used standard SNNS file and computed output is also written to SNNS file. It is also possible to plot the chosen Principal Components in 2D or 3D. If we plot only two Principal Components in this way, we are able very easy to analyse if input data, it means the emotions(facial expressions), are separable or not.

4.7.2 Verifying PCA results by professional tools

For this task we used a standard statistical tool SPSS 6.1. Unfortunately in this tool is used as input to PCA correlation matrix instead of covariance matrix as in our case.

The computation and the results from our application and SPSS are quite similar. The only difference is that we have transformed our data to a **covariance** matrix instead of **correlation** matrix used in SPSS. **Correlation** matrix is computed like **covariance** matrix, but correlation between features X_i and X_j is normalised by dividing by their own covariance σ_i and σ_j. It means, that the solution with **covariance** matrix is more sensitive to variance of features and the solution with **correlation** matrix is more sensitive to correlation between features. The new axis after recomputing have direction following the maximums of the covariance or the correlation. We have had 88.9% of the variance in the first three components, SPSS had 80.0% of correlation in the first three components.

For computing of correlation matrix of our input data we need to compute for every couple of feature vectors X_i, X_j the correlation coefficient r_{ij}. This we can get by *equation 4.7.*

$$r_{ij} = \frac{cov(X_i, X_j)}{\sigma_i \sigma_j} \tag{4.7}$$

where

$$\sigma_i{}^2 = \frac{1}{N-1} \sum_{k=1^N} (X_{ki} - \mu_{X_i})^2 \qquad \sigma_j{}^2 = \frac{1}{N-1} \sum_{k=1^N} (X_{kj} - \mu_{X_j})^2 \tag{4.8}$$

where $cov(X_i, X_j)$ is matrix multiplying of feature vectors, μ_{X_j} is mean value vector (for all patterns) and N is amount of patterns in the input data.

Another question is how to interpret our reduced data. From the coefficients loadings table we can make some conclusions, but very often the loading of all features is confusing and it is not possible to interpret axis(Principal Components). For a reduced space it is possible to use many methods for "axis rotation" to obtain more clear insight in the observed data. After rotation, the result and the amount of the data is the same, but axis(Principal Components) have better directions. After rotation of coeficients loading from features are much more easy to interpret the meaning of each principal component. In our case we used the method **Varimax rotation** in SPSS. In our tests are presented results of PCA before Varimax rotation and after. For comparison are there also displayed results of PCA using only the covariance matrix.

4.8 Expression recognition by Neural Networks

Because developing complex tools for modelling Neural Networks wasn't our goal, we used a standard Neural Network tool (SNNS). We used the PCA method for data reduction, as was mentioned in the last section. After this reduction we obtained data which contained only three of the most significant features. The created Neural Network in SNNS also had only three inputs. A basic question was the amount of hidden neurons which we should use. In paper [7] the network with the best results had a structure 21x6x6. It means a hidden layer with six neurons. We created our network also in this way and performed some test for different amounts of hidden neurons. At the end we found as the best structure 3x6x6. Using more hidden layers than one wasn't not useful.

4.9 Expression recognition by Bayesian classifier

When we analysed the result of PCA for the training data, we found that this data are almost separable. Hence we decided to use also some simpler classifier than the Neural Network. For this task we used standard Bayesian classifier (described by *equation 4.9*).

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{N/2}\sqrt{\det(C_i)}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^T C_i^{-1}(\mathbf{x}-\mu_\mathrm{i})\right]. \tag{4.9}$$

where ω_i is one of the six classes (emotions) we have classified into, N is the amount of classes, C_i is the covariance matrix for class i of input data, x is input pattern and μ_i mean vector for class i. We classified unknown pattern to class which had the biggest probability $p(\mathbf{x}|\omega_i)$. We used this method for comparison with Neural Network and our results are in next chapters.

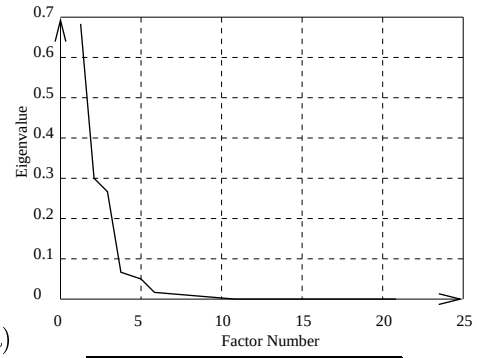
Chapter 5

Test1 - data generated by the applet

5.1 PCA of applet data

We performed PCA for our data generated by the applet. After computation of the sizes of the eigenvalues we concluded that most of the information is saved in the first three Principal components. The graph of sorted eigen values from the biggest to the smallest value (*Figure 5.1a*) confirms our conclusion, that first three components are enough for description of all data set. From the *Table 5.1b* is possible to see loading of the biggest eigenvalues. Hence for our final tests we have used a reduced input space from the 21-dimensional to 3-dimensions by PCA.

a)



b)

No.	eigen value	percentage of eigen value
1	0,68	48,7
2	0,30	21,6
3	0,26	18,6
4	0,07	4,8
5	0,05	3,4
..	..	..

Figure 5.1: Results for PCA, a)sorted eigenvalues b)table of eigenvalues

We used two kind of PCA: first computed from covariance matrix and second computed from correlation matrix. In the second case we perform also Varimax rotation for more clear representation of results. It means that we have three results for all three tests. In the *Table 5.1* is displayed the most significant loading of all Principal Components for all mentioned cases (the letter x means smaller significance and letter X means bigger significance if the observed feature). In the *Table 5.2* are the exact values of all computed loading. The loading to the biggest Principal Components from the face features is displayed in *Table 5.3* and *Figure 5.2*.

Plotting of data after data reduction to three dimensions in 2D and 3D is depicted in *Figures A.2 and A.1*.

Table 5.1: Representation of loading of the principal components

	Feature	PCA covar. matrix			PCA correl. matrix			PCA correl. +Varimax		
		PC1	PC2	PC3	PC1	PC2	PC3	PC1	PC2	PC3
1	as BASE, but between out.corners				x			**X**		
2	openness of eyes - middle	x	x		**X**			**X**		
3	openness of eyes, up.part - in.corners	**X**			**X**			**X**		
4	openness of eyes, low.part - in.corners		**X**			**X**			**X**	
5	openness of eyes, out.part of eyes	**X**	**X**		**X**			**X**		
6	openness of eyes, in.part of eyes	**X**	x		**X**			**X**		
7	openness of eyes, low.parts - in.corners, out.part		**X**			**X**			**X**	
8	openness of eyes, up.parts - in.corners, out.part	**X**			**X**			**X**		
9	openness of eyes, low.parts - in.corners, in.part		x			**X**			**X**	
10	openness of eyes, up.parts - in.corners, in.part	**X**			**X**			**X**		
11	between eyebrows									
12	between eyebrows and eyes in mid.parts	**X**	x	x				**X**		
13	between in.corners of eye and in.ends of eyebrows	**X**		x	**X**			**X**		
14	middle of eyebrow - second in.end of eyebrow									
15	second in.end of eyebrow - int.corners of eye	**X**	x	x	x			**X**		
16	between mouth corners			x		**X**				
17	openness of mouth in middle	x		**X**			**X**			**X**
18	mouth corners against upper middle of mouth		x	x		**X**				
19	mouth corners - low.second mid.points of mouth	**X**	**X**	**X**						
20	mouth corners - up.second mid.points of mouth		**X**	**X**		**X**				
21	openness of mouth, second mid.points	**X**		**X**			**X**			**X**

Table 5.2: Exact loading of the principal components

Feature No.	PCA from covariance matrix			PCA from correlation matrix			PCA from corr. matrix with varimax rotation		
	PC1	PC2	PC3	PC1	PC2	PC3	PC1	PC2	PC3
1	-0.05	0.01	-0.02	0.76	-0.13	-0.15	**0.74**	0.25	0.04
2	-0.19	0.14	-0.05	0.96	0.18	-0.16	**0.78**	0.58	0.19
3	-0.29	-0.03	-0.08	0.90	-0.31	0.07	**0.93**	0.06	0.21
4	-0.09	0.30	-0.02	0.54	0.71	-0.38	0.18	**0.95**	0.06
5	-0.38	0.28	-0.09	0.95	0.19	-0.15	**0.77**	0.58	0.20
6	-0.31	0.23	-0.08	0.95	0.20	-0.16	**0.77**	0.59	0.19
7	-0.09	0.32	-0.02	0.54	0.72	-0.37	0.18	**0.95**	0.07
8	-0.29	-0.04	-0.07	0.89	-0.33	0.10	**0.93**	0.03	0.22
9	-0.06	0.25	-0.00	0.48	0.74	-0.37	0.12	**0.95**	0.07
10	-0.25	-0.01	-0.07	0.91	-0.28	0.05	**0.93**	0.09	0.20
11	-0.09	0.02	-0.04	0.59	-0.10	-0.01	0.57	0.15	0.12
12	-0.24	-0.15	-0.19	0.66	-0.50	-0.07	**0.81**	-0.13	-0.06
13	-0.31	-0.09	-0.24	0.82	-0.45	-0.14	**0.94**	0.00	-0.06
14	-0.02	0.03	0.04	0.23	0.19	0.23	0.09	0.14	0.33
15	-0.37	-0.18	-0.25	0.77	-0.50	-0.06	**0.92**	-0.09	-0.02
16	-0.02	-0.07	-0.10	0.15	-0.63	-0.43	0.46	-0.27	-0.56
17	-0.16	-0.02	0.32	0.49	0.18	0.80	0.28	-0.02	**0.91**
18	-0.02	0.19	0.14	0.25	0.81	0.31	-0.18	0.61	0.63
19	-0.30	-0.50	0.48	0.36	-0.39	0.72	0.43	-0.50	0.60
20	-0.05	0.48	0.36	0.24	0.82	0.27	-0.20	0.64	0.60
21	-0.23	-0.01	0.56	0.47	0.22	0.80	0.24	0.00	**0.92**

Table 5.3: Representation of features corresponding with components

Component	Feature
PCA from covariance matrix	
1	eye - eyebrow, its distances, openness of eyes , position of mouth corners
2	openness of eyes, position of mouth corners
3	openness of mouth, position of mouth corners
PCA from correlation matrix	
1	eye - eyebrow, its distances, openness of eyes
2	openness of eyes, position of mouth corners
3	openness of mouth
PCA from correlation matrix + Varimax rotation	
1	eye - eyebrow, its distances, openness of eyes
2	openness of eyes
3	openness of mouth

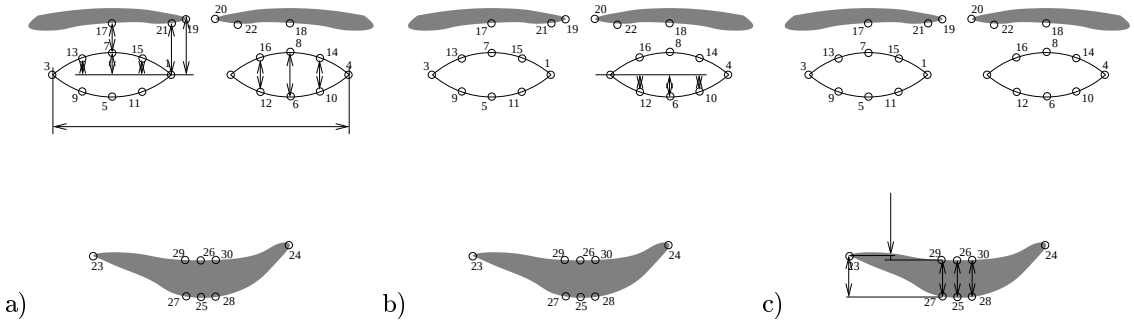


Figure 5.2: The connection between PCs (from correlation matrix after Varimax rotation) and features of the face a)First component, b)second component and c)third component.

5.2 Results of the Neural Network

In this case we have performed condition of size of training and test set. Our results were near 100% successful when we have used network structure 3x6x6 neurons as you can seen in *Table 5.4* - where 135/136 means 135 successfully recognised patterns from 136 patterns set. We have to note, that these results are valid only for faces generated by the applet. Our network and error during the learning process are depicted in *Figure 5.3*.

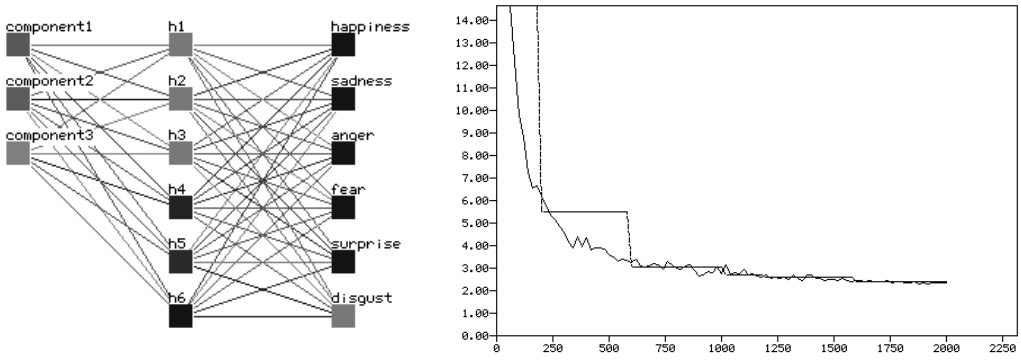


Figure 5.3: Architecture of the feedforward Neural Network and error during training

Table 5.4: Results of Neural Network

Happiness	**Sadness**	**Anger**	**Fear**	**Surprise**	**Disgust**
Pictures: 16 for each emotion: (10 typical + 6 stronger) = 96 After linear combination: ((16*16-16)*6+96) = 816 Training set: 816/2=408, Test set: 816/2=408					
136/136	136/136	135/136	136/136	136/136	136/136
Success: 99.9%					

5.3 Results of the Bayesian classifier

After the data space reduction by PCA(three factors) we used for classification Bayesian classifier. It wasn't possible to use for a training the data generated by the linear combination(120 patterns for each emotion), because of zero or very small determinant of a covariance matrix. We used the data generated by the linear combination as a testing set. We used for the training 16 pictures for each emotion. For testing we used the same data and the data generated by linear combination. Results are in the *Table 5.5*. Advantages of Bayesian classifier are is the very fast training and good generalisation.

Table 5.5: The result of Bayes for the artificial data

Happiness	**Sadness**	**Anger**	**Fear**	**Surprise**	**Disgust**
6x16 training patterns, test set is the same					
16/16	16/16	15/16	16/16	15/16	14/16
Success: 95.8%					
6x16 training patterns, test set 6x120 linear combination					
120/120	120/120	119/120	120/120	120/120	116/120
Success: 99.3%					
6x8 training patterns, test pattern 6x8 the other half					
8/8	4/8	5/8	7/8	5/8	6/8
Success: 72.9%					

Chapter 6

Test2 - handmade linedrawings

In this test part we used handmade linedrawings, which were scanned and saved like image files. We performed this test to check the reliability of our Characteristic point extractor and also to test the generalisation properties of our trained Neural Network. To obtain many different faces we asked 10 students to draw six basic emotions in an appropriate way (eyes, eyebrows and mouth). After this we had 60 images of the face (*Appendix C*).

We found that for people it isn't easy to express all emotions significantly by drawing. Students were usually able to draw faces corresponding with happiness and sadness, but when they had to draw fear or disgust they simply failed. We performed Principal Component Analysis for this data and found that all categories are overlapping and are absolute inseparable (*Figure A.3*). Hence we decided to use the data only for testing and not for training of the Neural Network.

6.1 Results of the Neural Network

We chose from this handdrawings of the face only these which corresponded with desired emotion and these we used for testing. It is also necessary to mention that the Neural Network was trained for data generated by the applet (21, 3-component data set). It means that we trained for almost real faces, but test drawings looked sometimes more like cartoons. Hence we got face recognition success rate only about *40%*. This fact can be caused by almost inseparable test data. Therefore we used also 21 features data (without reduction) to use all possible information presented in the data. Complete results are in *Table 6.1*.

Table 6.1: The result of NN for the handmade data

Happiness	**Sadness**	**Anger**	**Fear**	**Surprise**
6x16+6x120 applet training patterns - 3 features the test set is handmade				
3/11	6/10	0/8	2/6	1/5
Success: 30.0%				
6x16+6x120 applet training patterns - 21 features the test set is handmade				
8/11	2/10	2/8	2/6	3/5
Success: 42.5%				

6.2 Results of the Bayesian classifier

Results of Bayesian classifier for handmade data (*Table 6.2*) was a little bit better than results of Neural Network.

Table 6.2: The result of Bayes for the handmade data

Happiness	**Sadness**	**Anger**	**Fear**	**Surprise**
6x16 applet training patterns, test set is handmade				
6/11	4/10	1/8	6/6	1/5
Success: 45.0%				
40 handmade training patterns, test set is the same				
6/11	2/10	6/8	6/6	3/5
Success: 57.5%				

Conclusions

In this paper we introduced some problems occurring during automatic recognition of emotion from linedrawings of the face. Two kinds of images were used : faces generated by applet and handdrawings of face. We extracted from an input image the characteristic points of the face and this data was analysed and reduced by Principal Component Analysis. The reduced data were used for training of a Neural Network and Bayesian classifier. We concluded that both classifiers were very successful during testing of faces generated by applet (nearly 100%). When we tried to use classifiers for evaluating of handmade faces, they often failed. Our recognition success rate was about 50%. From PCA it is possible to see that it could be caused by inseparability of the handmade data. We have noticed that results of both used classifiers were nearly the same. Therefore we can recommend for ordinal use Bayesian classifier for its very easy implementation and fast performance.

Bibliography

[1] M. Pantic & L.J.M. Rothkrantz, "Automatic Recognition of Facial Expression and Human Emotions", ASCI'97 Proceedings of the third annual conference of the Advanced School for Computing and Imaging, Heijen, The Netherlands, June 2 - 4, 1997

[2] H. Kobayashi, F. Hara, "Recognition of Six Basic Facial Expressions and Their Strength by Neural Network", IEEE International Workshop on Robot and Human Communication 4/92. pages 381-386.

[3] P. Ekman, W. V. Friesen, "Unmasking the face", Prentice Hall, New Jersey, 1975.

[4] P. Ekman "Emotion in Human Face", Cambridge University Press, Cambridge, 1982.

[5] M. Sonka, V. Hlavac, R. Boyle, "Image Processing, Understanding, and Machine Vision", 2nd edition, PWS Boston, 1999. (1st edition Chapman&Hall, London 1993), pp. 770.

[6] D. C. Lay, "Linear Algebra and Its Applications", University of Maryland, Addison-Wesley Publishing Company, 1994

[7] T. Svoboda, M. Friedl, "Recognition of Facial Expressions by Neural Network ", TU Delft, 1994

[8] J. Brunner, "Recognition of Emotions from Facial Expressions by Neural Network", TU Delft, 1994

[9] G. Lencse, A. Varga, "Automatic Generation of Facial Expressions", TU Delft, 1994

[10] A. Nijholt, D.Heylen and K.Jokinen (eds.), "Learning to Behave, Proceedings of the seventeenth Twente Workshop on Language Technology", 18th, 19th, 20th October 2000, Enschede, The Netherlands

[11] J.N. Bassili, "Facial motion in the perception of faces and of emotional expression, Journal of Experimental Psychology: Human Perception and Performance 4", 373-379, 1978

[12] V. Bruce, "Recognising Faces" Lawrence Erlbaum. Hove, East Sussex, 1986

Appendix A

PCA results plotting

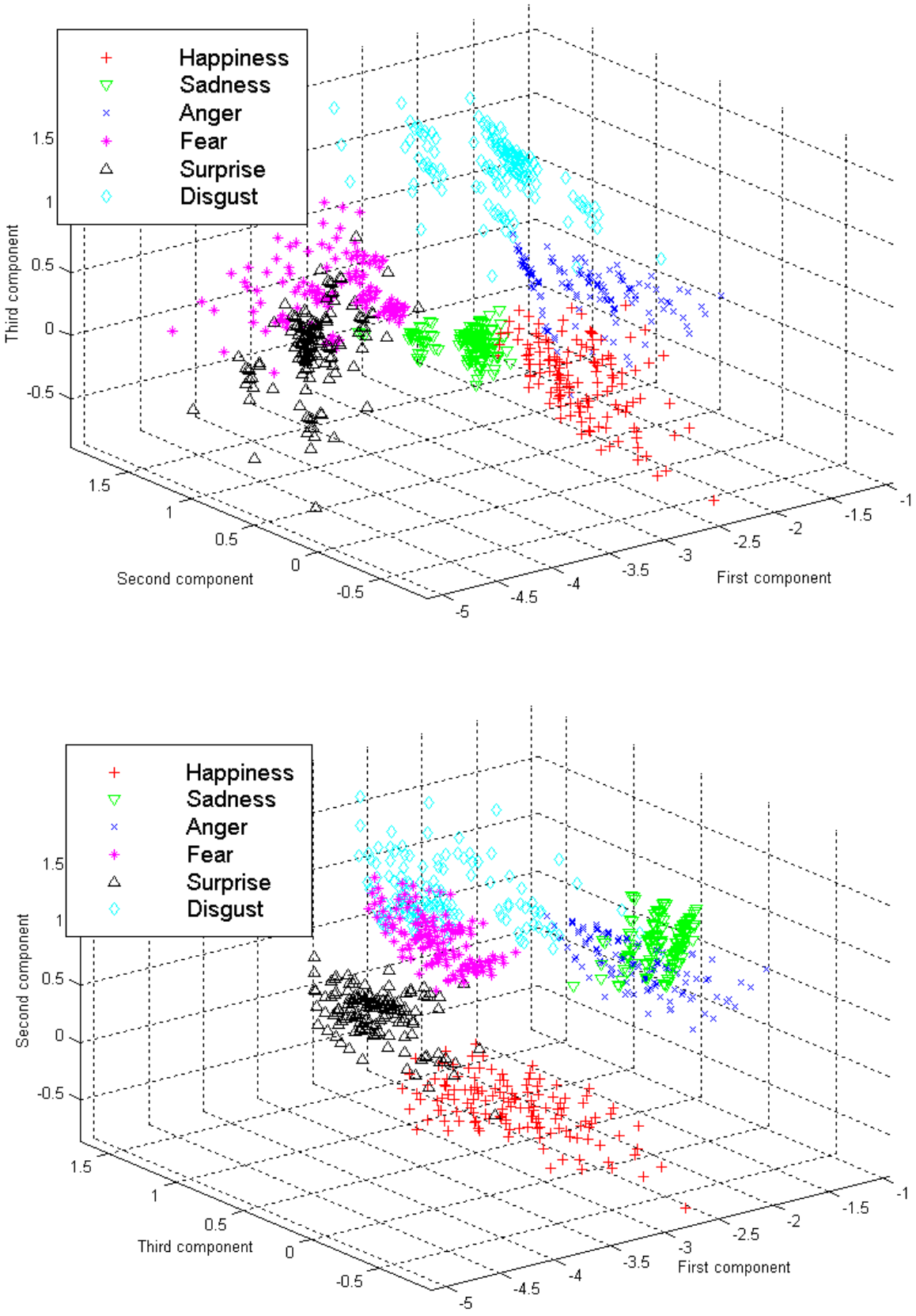


Figure A.1: 3D plotting of results of PCA for applet data from two different points of view

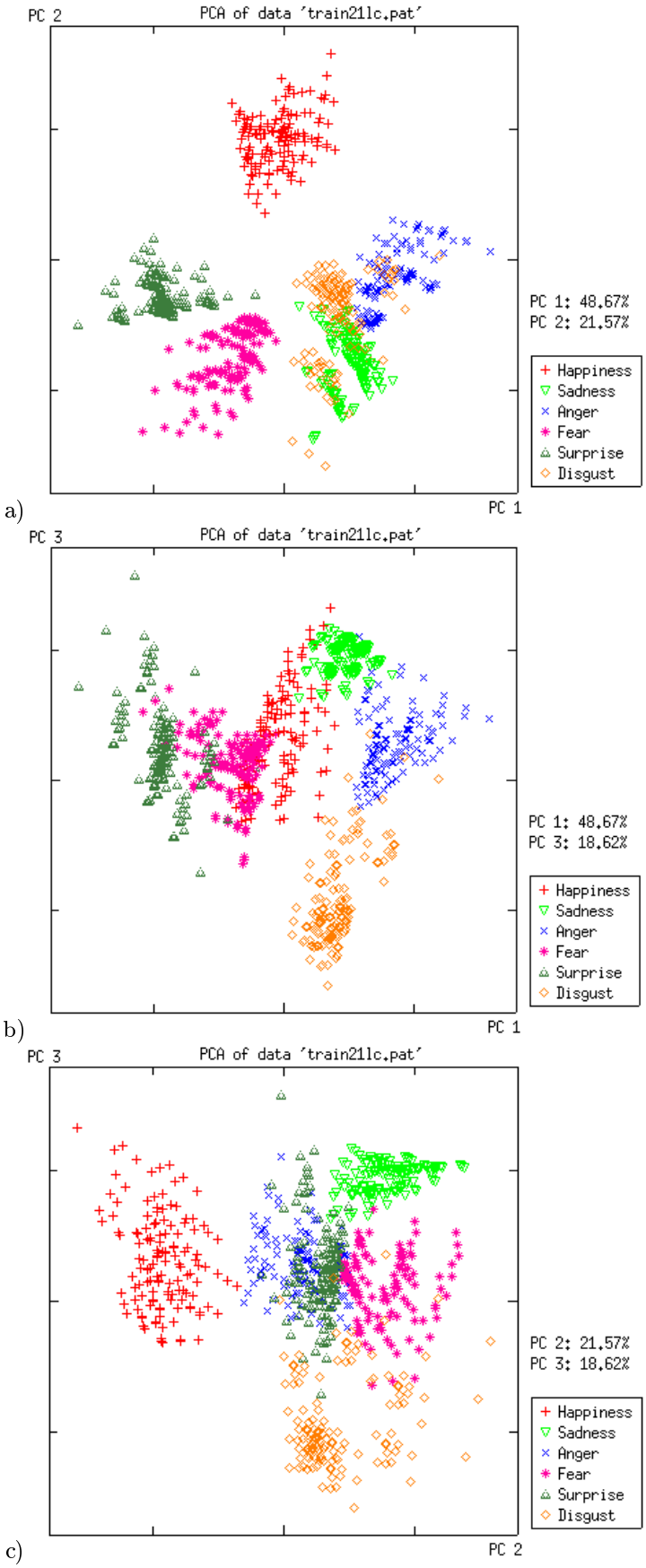


Figure A.2: 2D plotting of results of PCA for applet data a) components 1&2 b) components 1&3 c) components 2&3

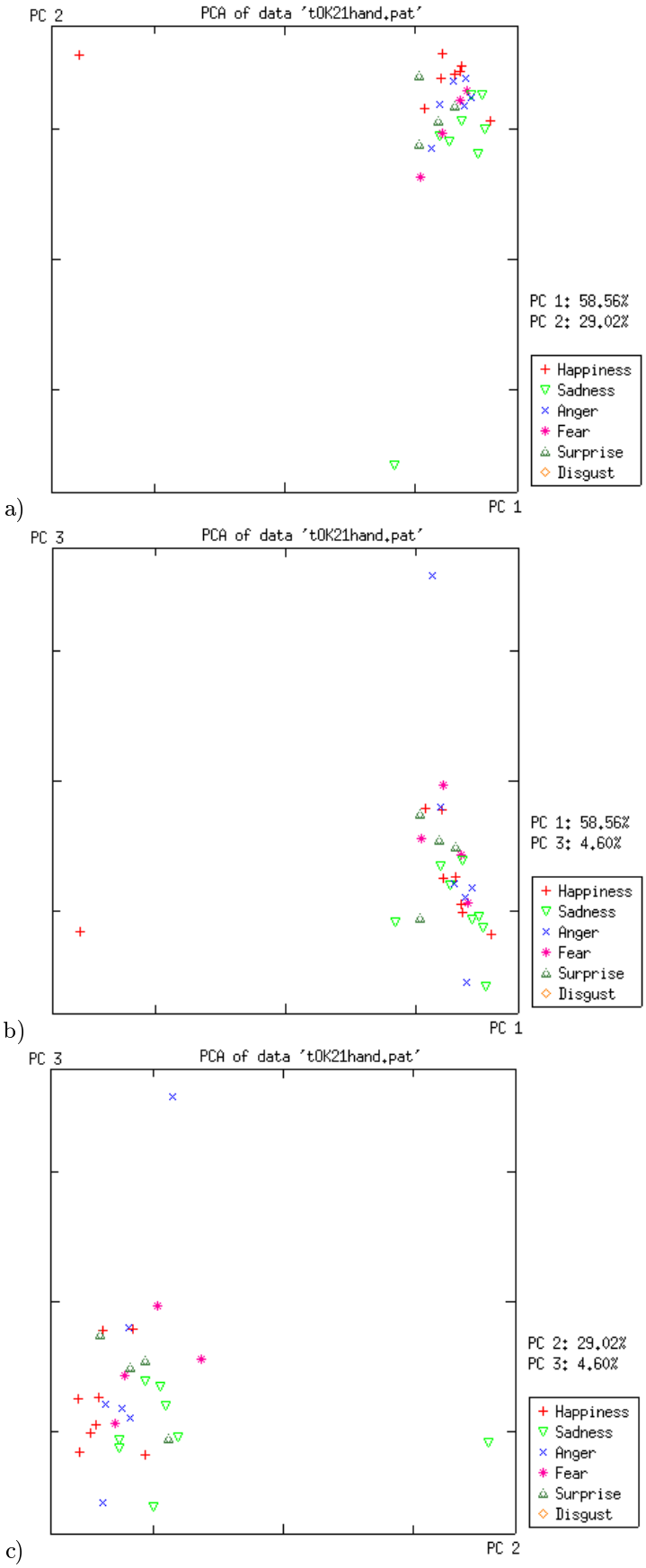


Figure A.3: 2D plotting of results of PCA for handmade linedraws a) components 1&2 b) components 1&3 c) components 2&3

Appendix B

Applet data

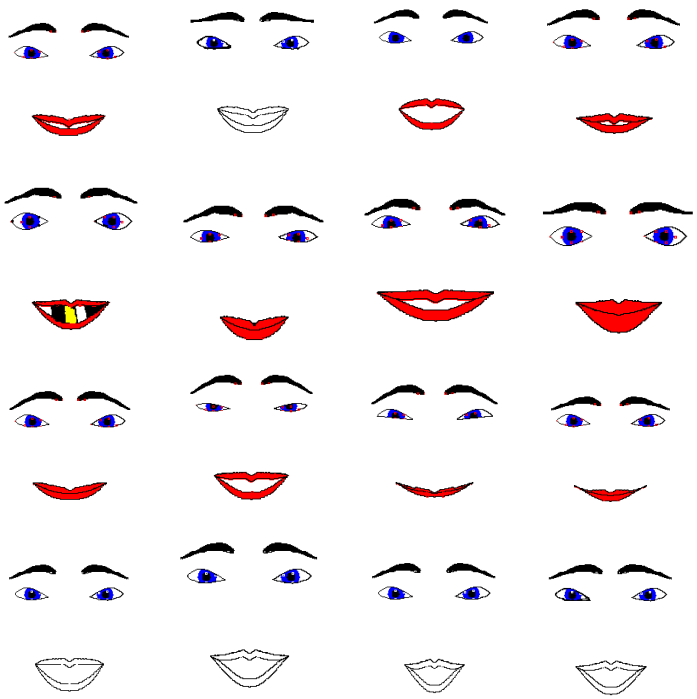


Figure B.1: Java applet: Happiness

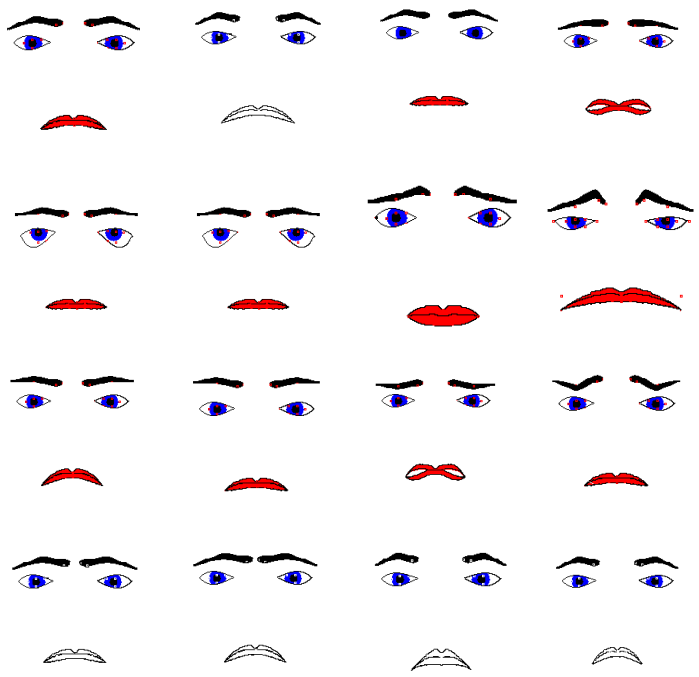


Figure B.2: Java applet: Sadness

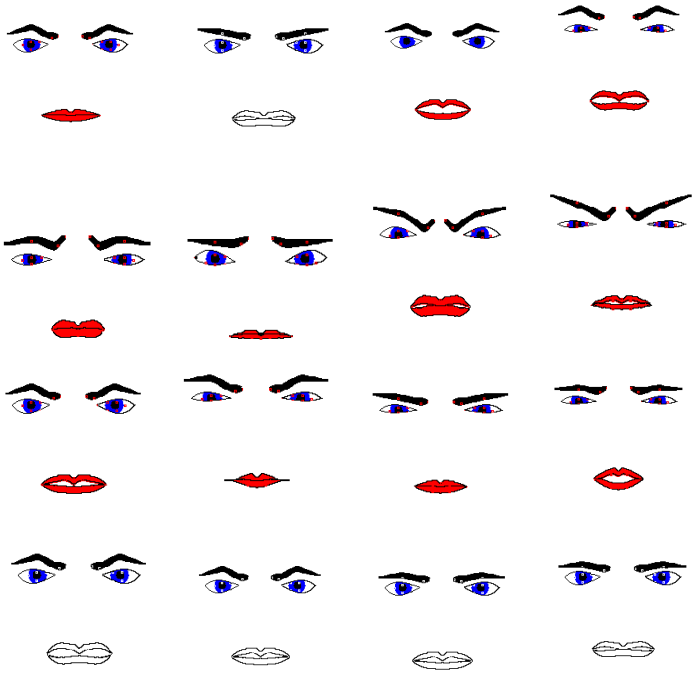


Figure B.3: Java applet: Anger

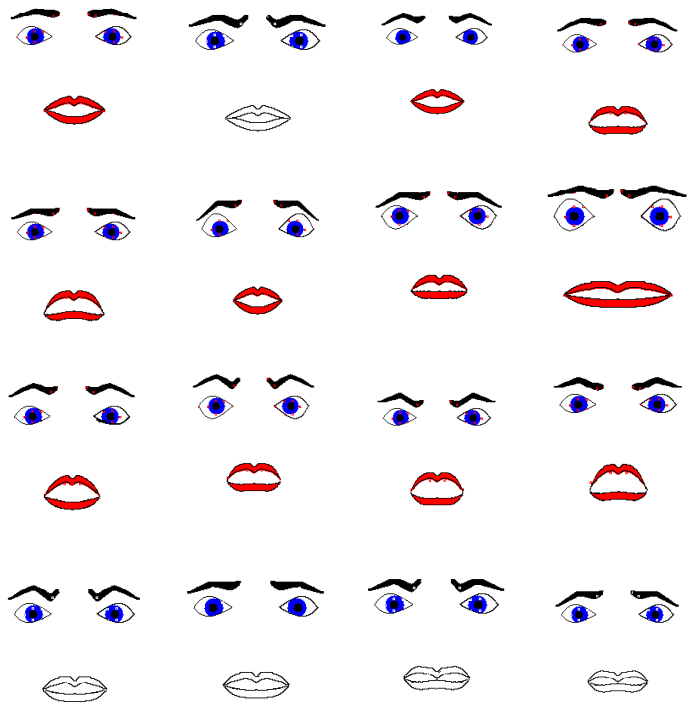


Figure B.4: Java applet: Fear

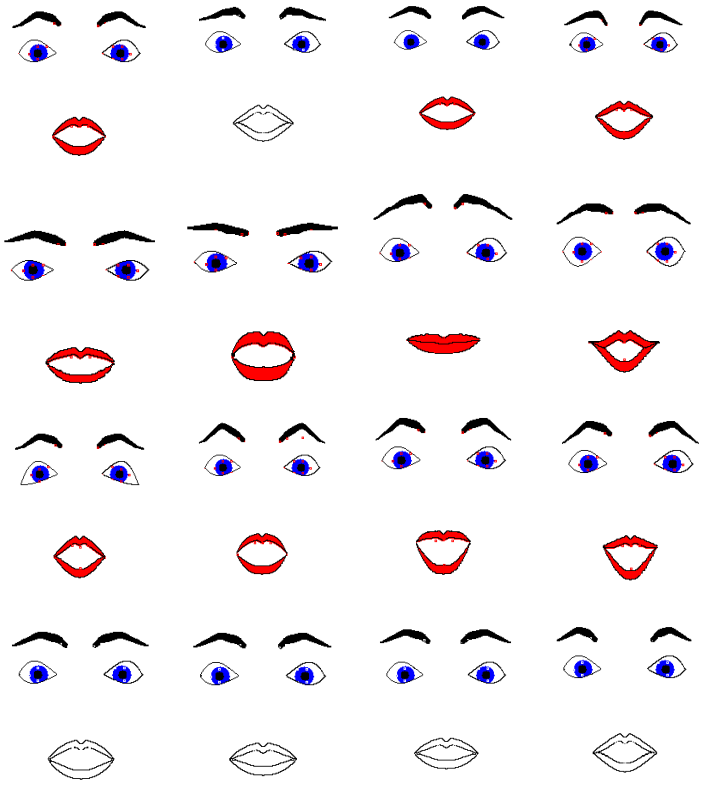


Figure B.5: Java applet: Surprise

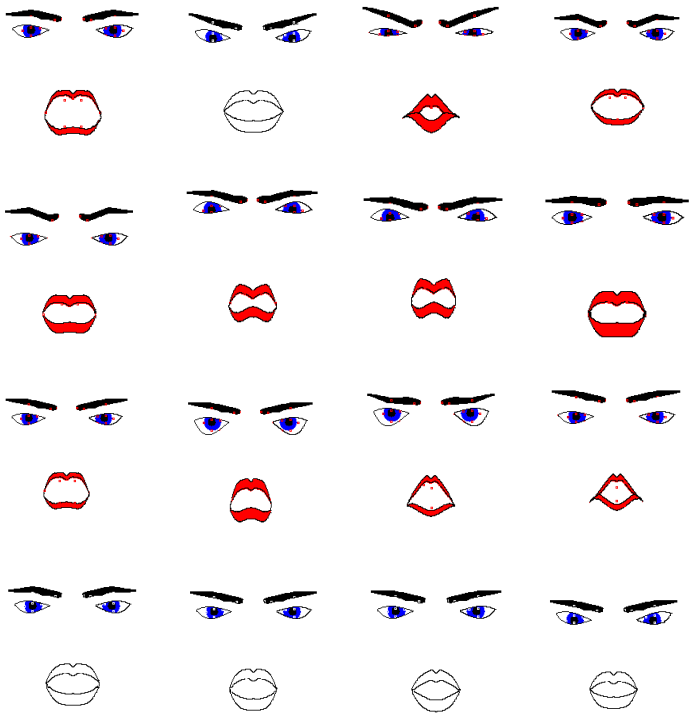


Figure B.6: Java applet: Disgust

Appendix C

Handmade data

Data which we choosed for testing (corresponding with emotions), are marked by frames.

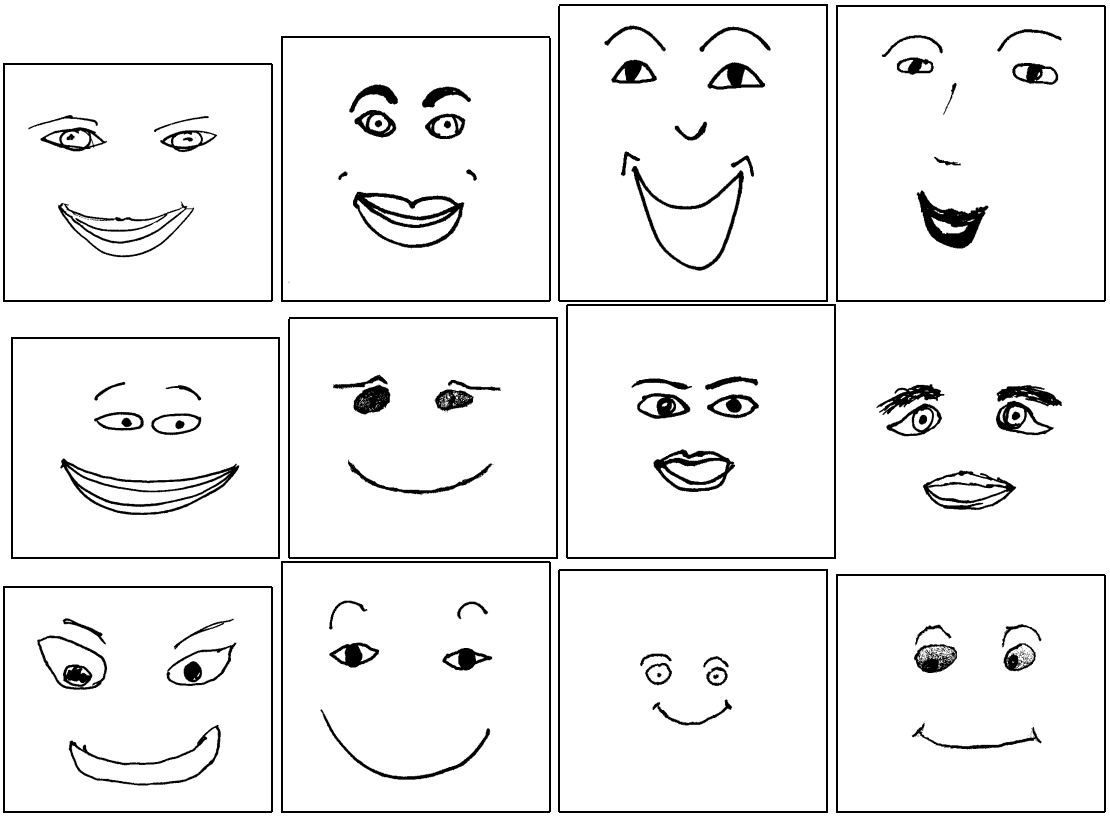


Figure C.1: Happiness

Figure C.2: Sadness

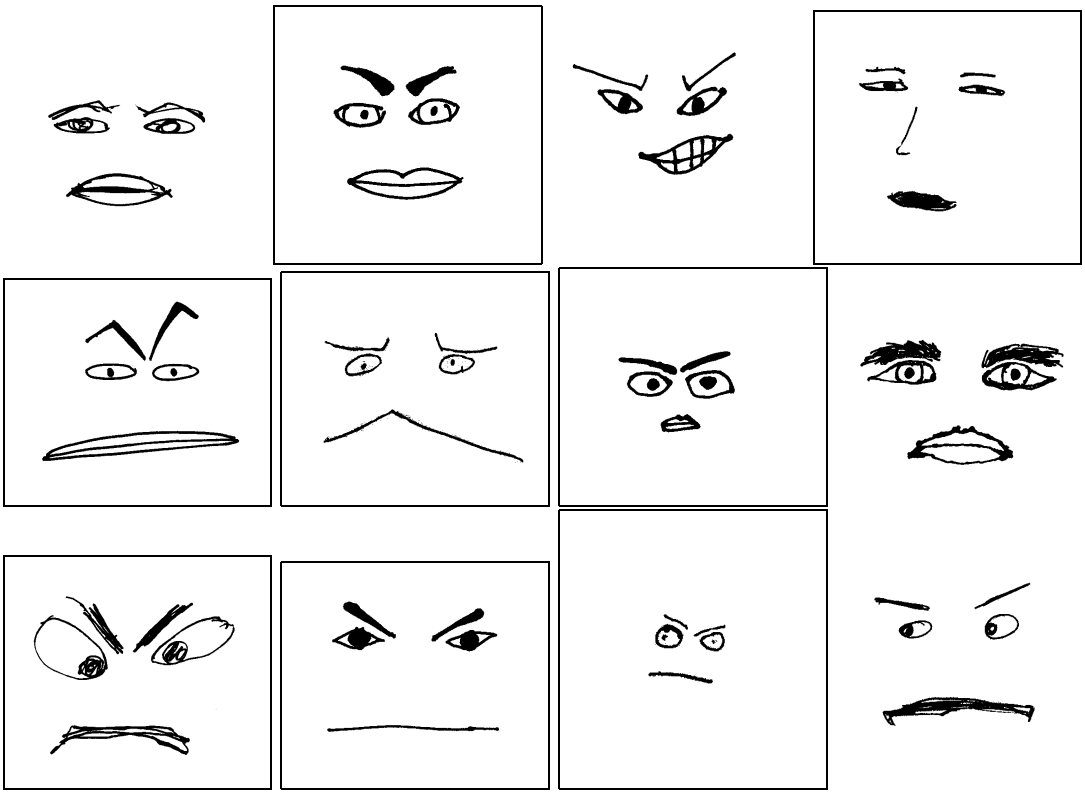


Figure C.3: Anger

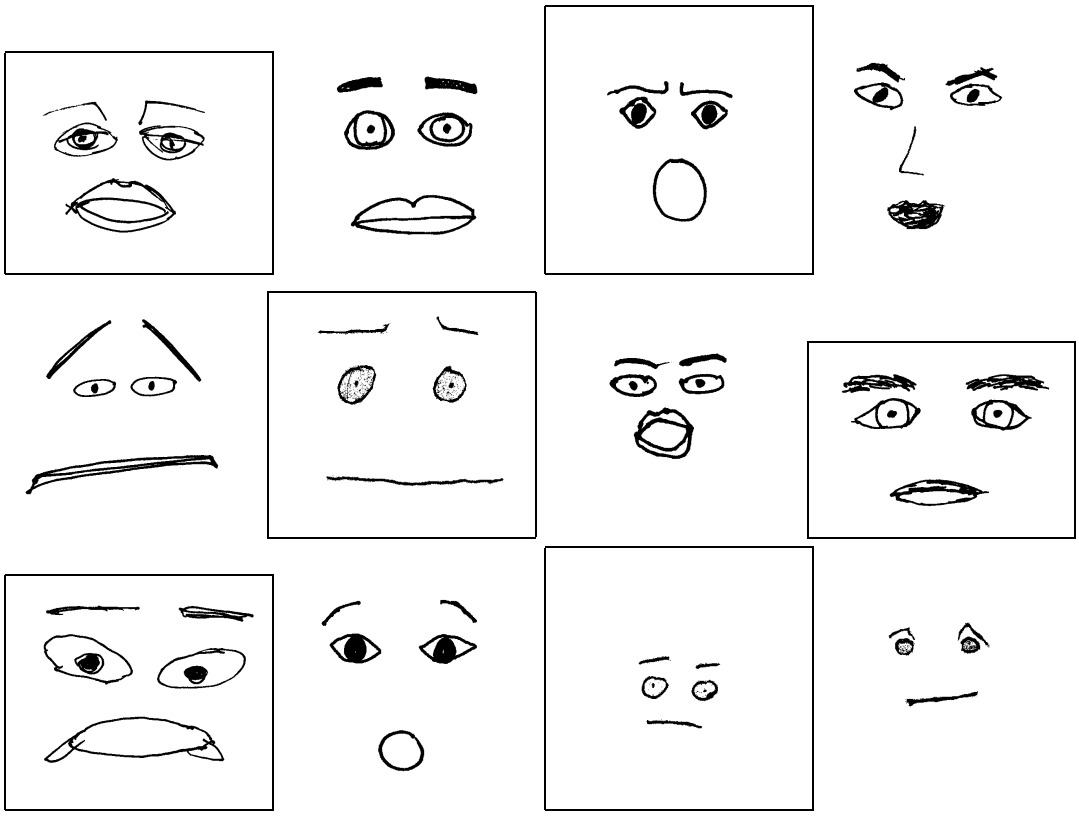


Figure C.4: Fear

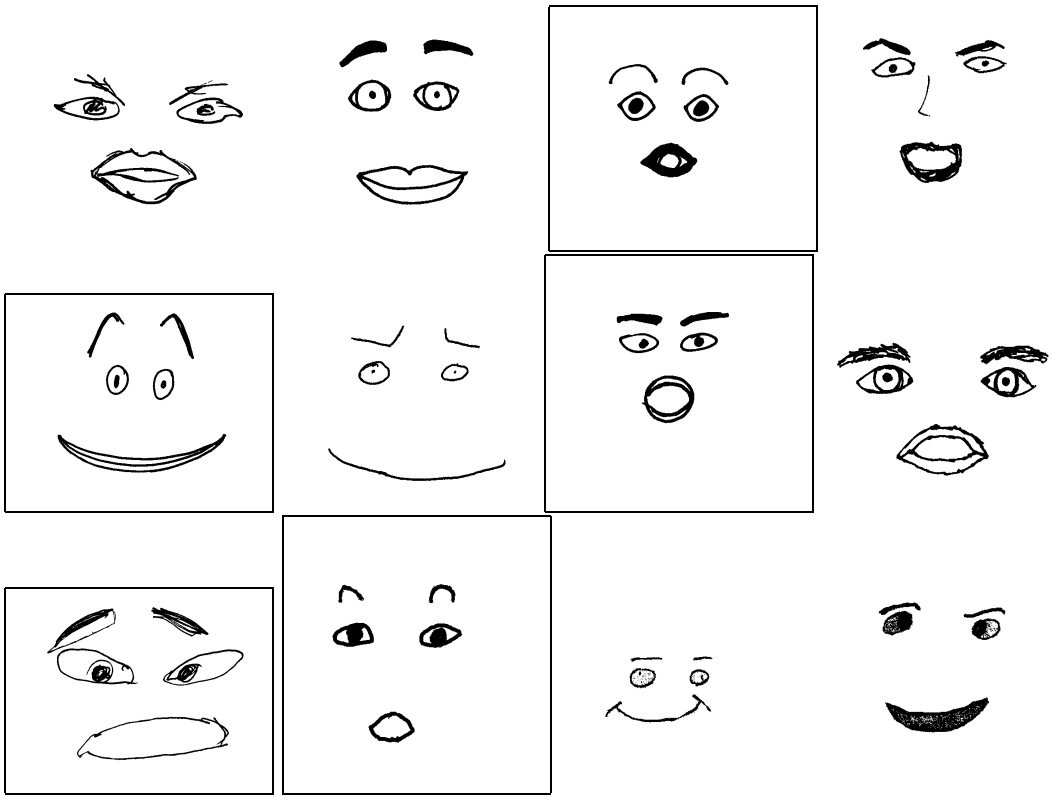


Figure C.5: Surprise

Figure C.6: Disgust

Appendix D

Data reduction from 60 to 21 dimensions

Transformation from 60-dimensional feature space to 21-dimensional.

$$X_0 = \frac{X_1+X_2}{2}$$
$$Y_0 = \frac{Y_1+Y_2}{2}$$
$$\theta = arctan\left(\frac{X_0-X_{26}}{Y_0-Y_{26}}\right)$$
$$BASE = (X_2 - X1) \cdot cos(\theta)$$

$$D_1 = X_3 - X_4$$
$$D_2 = Y_7 - Y_5 + Y_8 - Y_6$$
$$D_3 = 2 \cdot (Y_7 - Y_1 + Y_8 - Y_2)$$
$$D_4 = 2 \cdot (Y_1 - Y_5 + Y_2 - Y_6)$$
$$D_5 = 2 \cdot (Y_{13} - Y_9 + Y_{14} - Y_{10})$$
$$D_6 = 2 \cdot (Y_{15} - Y_{11} + Y_{16} - Y_{12})$$
$$D_7 = 2 \cdot (Y_1 - Y_9 + Y_2 - Y_{10})$$
$$D_8 = 2 \cdot (Y_{13} - Y_1 + Y_{14} - Y_2)$$
$$D_9 = 2 \cdot (Y_1 - Y_{11} + Y_2 - Y_{12})$$
$$D_{10} = 2 \cdot (Y_{15} - Y_1 + Y_{16} - Y_2)$$
$$D_{11} = X_{20} - X_{19}$$
$$D_{12} = Y_{17} - Y_7 + Y_{18} - Y_8$$
$$D_{13} = Y_{19} - Y_1 + Y_{20} - Y_2$$
$$D_{14} = 2 \cdot (Y_{17} - Y_{21} + Y_{18} - Y_{22})$$
$$D_{15} = Y_{21} - Y_1 + Y_{22} - Y_2$$
$$D_{16} = 0.5 \cdot (X_{24} - X_{23})$$
$$D_{17} = Y_{26} - Y_{25}$$
$$D_{18} = 0.5 \cdot (Y_{26} - Y_{24} + Y_{26} - Y_{23})$$
$$D_{19} = 1.5 \cdot (Y_{23} - Y_{27} + Y_{24} - Y_{28})$$
$$D_{20} = 1.5 \cdot (Y_{29} - Y_{23} + Y_{30} - Y_{24})$$
$$D_{21} = Y_{29} - Y_{27} + Y_{30} - Y_{28}$$

$$DATA_i = \frac{D_i}{BASE \cdot cos(\theta)} \qquad i = (1, \ldots, 21)$$

Where X_r,Y_s $r, s \in (1, \ldots, 30)$ are the 60-dimensional input data and $DATA_i$ 21-dimensional data are result we have needed.

Appendix E

Developed software programs

- Standart and special Libraries we used (C++)

  Standart stdio.h, alloc.h, stdarg.h, string.h.

  QT 1.44 ctype.h, qimage.h, qpainter.h, qapplication.h, qevent.h, qwidget.h, qobject.h, qwindowdefs.h, qmessagebox.h, qpushbutton.h, qdatetime.h, qscrollbar.h, qapplication.h, qpushbutton.h, qfont.h, qimage.h, qpainter.h, qstring.h, qpushbutton.h, qscrollbar.h, qlcdnumber.h,

  matrix.hpp, svddec.hpp, location:**/PROJECT/matclass/**

- Feature Extracting (C++), location:**/PROJECT/EXTRACTOR/**

  cmdline.cpp Our library for the processing of the command line. It means switches, input output and file names.

  process.cpp Our library which consist of Loading image, Face cutting, Face settlement and Additional processing.

  svd.cpp Our library for a computing the least square error solution for Fitting a parabola to points.

  edgeagent.cpp Our library for Edge following, Eye corners extracting, Extract characteristic points and Data transformation.

  prvni.cpp An application which uses our library functions with or without a graphics user interface.

  prvni Name of executable file for the application.

- Data processing (C++), location:**/PROJECT/LIN_COMB/**

  cmdline.cpp See above.

  PCA_NeN.cpp Our library for the loading files in the format compatible with SNNS to matrix and for saving into this format.

  lincomb.cpp An application which uses our library functions and produces linear combination of input data. Output is file compatible with SNNS.

  lincomb Name of executable file for the application.

- Data analysis (C++), location:**/PROJECT/PCA/**

  cmdline.cpp See above.

  PCA_NeN.cpp see above.

PCA.cpp Our library for the computing of the Principal Component Analysis.

plotPCA.cpp An application which uses our library functions and reduces feature space. An input and an output is a file compatible with SNNS. The application has optional graphics user interface.

plotPCA Name of executable file for the application.

- Utilities (C++), location:**/PROJECT/ERRCOUNT/ and /PROJECT/SPLITPAT/**

count.cpp Our library for the counting of absolute error. An input is result file from SNNS. An output is value which specifies classification error. Error is computed: $\frac{correctpatterns}{allpatterns}$.

counterror Name of executable file for the utility.

split.cpp Our library for the counting of absolute error. An input and an output it result file from SNNS.

splitdata Name of executable file for the utility.

- Data location:**/PROJECT/**

/DATAHAND/ Pictures drawn by hand in format GIF.

/DATAAPPLET/ Pictures grabed from Java Applet in format GIF.

/DATATRAIN/ All data in pattern file. SNNS compatible.

All developed software programs are presented on enclosed CD.