1.Introduction.

One of the most remarkable abilities of human vision is that of face recognition. It develops over several of childhood, is important for several aspects of our social life, and together with related abilities, such as estimating the expression of people with which we interact, has played an important role in the course of evolution. So, humans are very good at recognizing faces and complex patterns. Even a passage of time doesn't affect this capability and therefore it would help if computers become as robust as humans in face recognition.

Face recognition system can help in many ways:

1) Checking for criminal records.

2) Enhancement of security by using surveillance cameras in conjunction with face recognition system.

3) Finding lost children's by using the images received from the cameras fitted at some public places.

4) Knowing in advance if some VIP is entering the hotel.

5) Detection of a criminal at public place.

6) Can be used in different areas of science for comparing a entity with a set of entities.

7) Pattern Recognition.

This project is a step towards developing a face recognition system which can recognize static images.

There are two traditional classes of techniques applied to the recognition of digital images of frontal views of face under roughly constant illumination. The first technique is based on the computation of a set of geometrical *features* from the picture of a face. This was the first approach toward an automated recognition of faces. The second class of techniques is based on *template matching.* We attempt here the first one of the techniques.

1.1. Motivation

The project was designed as a challenge for the student. There is a pattern recognition program called **KS 400** and the student should obtain the ***Face recognition*** using it. It is known that could be better using other systems, for example, Matlab. But the project wasn’t design that way. The program isn’t used as a programming tool but as a client tool where it’s possible writing a macro.

1.2. Objectives

- Build a database of faces
- Analyze an image finding features from the face in the image.
- Try to recognize people from the database from a different image
- Analyze the results

1.3. Procedures

The project had four diferent parts. The first one is to build a database of faces. The second one to write a macro which implement the a face recognition program, that program can measure diferent features from the face as the eyes or the mouth. The third part joins the first and second one in the same program, then can it measure and classify the faces. At last we analyze the results from the experiment and try to find some conlcusions from them.

1.3.1. Building the database

The database is built from the people who work in the laboratory, people who study at the FH München or friends. A camera was installed and a protocol was established for its use. Near to twenty people were photographed.

1.3.2. Face analysis

Before starting the face analysis, we trained with different forms. Those forms were seeds, we tried to recognize, measure and classify those seeds. When we had learned enough about the program, we started with the ***face recognition.*** At first we looked for some features. Some features were found, as the eyes or the mouth. Then, they were measured.

1.3.3. Recognizing faces

Then, every face from the database was measured. Therefore it was able to use a classifier with those faces and data. First, the features were extracted from the faces. The features extraction required a very fine tuning. When that point was reached, a work-around had to be implemented because the program doesn't allow a classifying from a complex database but from a picture. Finally when that problem was solved, the program was ready to classify the faces.

1.3.4. Analysis and results

Several times, the program was proved. We changed the procedures and try to find a better way to classify the faces. As we increased the number

of faces, we needed more features to find out who was in the picture. The final results are showed and analyzed in its own chapter.

2. State of the art

2.1. Methods of recognition

Over the past fifteen years many methods have been developed to tackle the problem of recognizing human faces. Face recognition is currently one of the most researched areas in pattern recognition. Its popularity stems from the fact that its applications are used in a variety of real life situations ranging from human - computer interaction to authentication and surveillance. Although various machine learning techniques have been developed, their success is limited because of the restrictions imposed by data acquisition systems. This literature survey will evaluate some of the methods that have been tested and also discuss the advantages of tensor analysis over traditional methods.

2.1.1 Background: Previous work

Over the past ten years, new conferences such as the International Conference on Audio and Video-Based Authentication (AVBPA) and International Conference on Automatic Face and Gesture Recognition (AFGR) and systematic empirical evaluations of face recognition techniques (FRT) have been started due to the growing interest in facial recognition among researchers in a variety of disciplines such as image processing, neural networks, computer graphics and psychology. FRT systems can be broadly classified into two groups depending on whether they make use of still images or video. In this study, I will focus only on FRT systems that make use of static images. The problem statement for facial recognition can be formulated as follows: Given an image of a person under varying conditions of illumination, pose or facial expression, verify/identify the people in the stored database of facial images.

2.1.2 Geometric Feature Matching

Brunelli and Poggio [1] in 1993 extended Kanade's [2] algorithm and used "Geometric Feature based Matching" for face recognition. The basic idea behind their algorithm was to describe the overall configuration of the face by a vector of numerical data representing the relative position and size of the main facial features: eyes and eyebrows, nose and mouth.

The classification was done using the nearest neighbour classifier on the vector corresponding to the given image with respect to the vectors corresponding to the images in the database.

The results, although impressive at the time, were not conclusive since they only considered a database of 47 people with 4 images of each person.

2.1.3 Eigenfaces

Eigenfaces proposed by Turk [3] are a set of orthonormal basis vectors computed from a collection of training face images. They provide a basis of low dimensional representation of the facial images and are optimal in the minimum least square error sense. If the training set of *N* facial images is represented by $\{z_1, z_2, \ldots, z_N\}$, Principal Component Analysis is applied to the set of training images to find the *N* eigenvectors of the covariance matrix

$$\frac{1}{N}\sum_{n=1}^{N}\left(z_n - \bar{z}\right)\left(z_n - \bar{z}^T\right)$$, where $$\bar{z} = \frac{1}{N}\sum_{n=1}^{N} z_n$$

is the average of the ensemble. The eigenvectors corresponding to the largest *k* (pre-determined) eigenvalues form the basis of an eigenface space. Classification is based on the eigen-feature vectors. The simplest classifier is based on Euclidean distance even though nearest neighbour classifier can also be used. The fact that the algorithm is fast and easy to implement makes Eigenfaces a very appealing technique. However, the main constraint is that one the frontal view of the images can be used and they are sensitive to extreme changes in pose and expression.

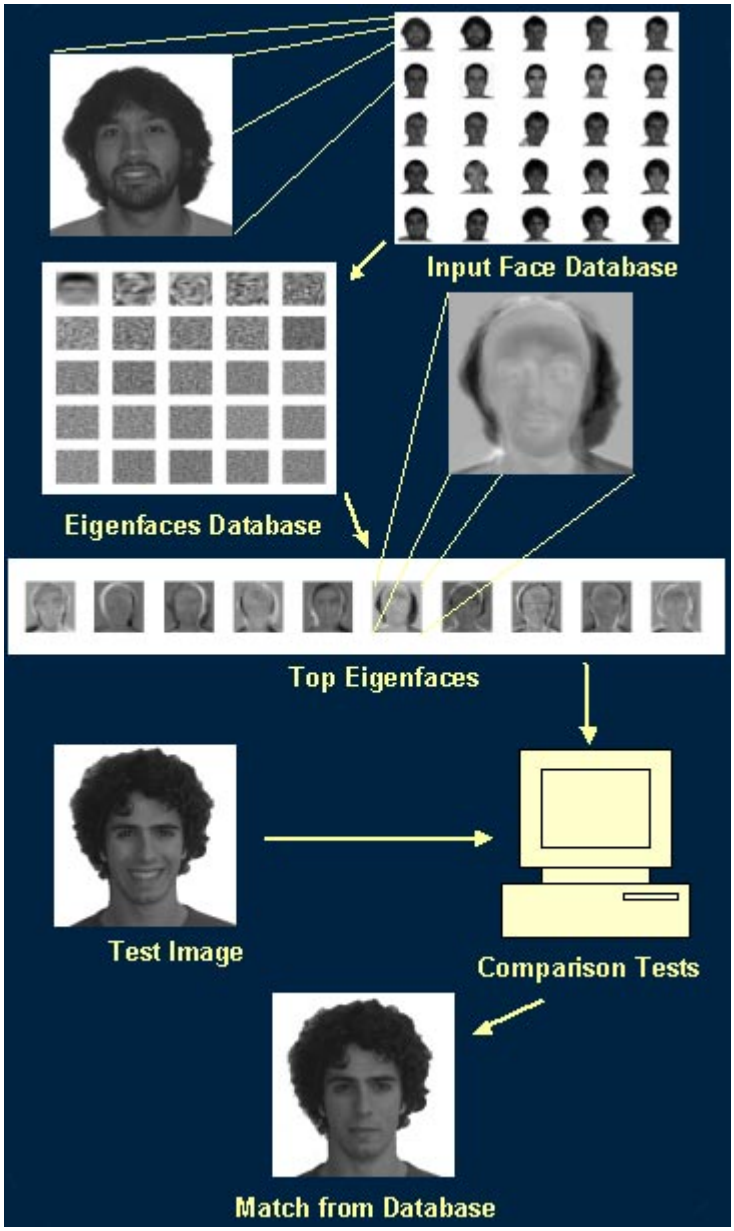


Figure 1: A robust detection system can yield correct matches when the person is feeling happy or sad.

2.1.4 Elastic Bunch Graph Matching

Another feature based facial recognition method is Elastic Bunch Graph Matching. The basic idea is to derive distance and position features from the placement of internal facial elements.

One early algorithm, written by T. Kanade [4], localized the corners of the eyes, nostrils, etc. in frontal views. His system computed parameters for each face, which were compared to the parameters of faces in the database already, using Euclidean metric.

Another algorithm, developed by L. Wiskott, J. Fellous, N. Krüger, and C. von der Malsbnurg [5], is a bit more recent. Each face is represented as a graph. Edges are represented by vectors with nodes positioned at certain outstanding points. In order to identify it, the new face has to be adapted to the size of the graph that was created. The features are compared with the edges and nodes on the graph. They are checked to see if they match (have similar size and position). It sounds like a fairly simple process, but it is actually quite complicated. The new face has to be resized and repositioned to minimize the difference between it and the original image.

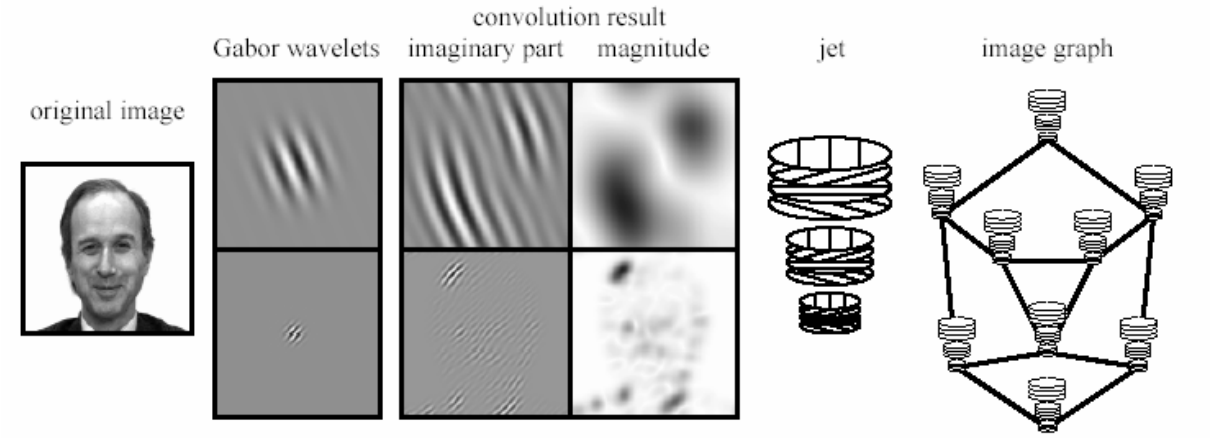


Fig. 2. The Gabor transform is used to perform the Elastic Bunch Graph Matching.

This method works well when the images are looked at straight on. If the face is tilted or turned in any way, this method has less accuracy.

Another disadvantage to this method is that it can be quite tedious for those performing it. A large amount of the grid placement has to be done manually.

Elastic Bunch Graph Matching is yet another method for facial recognition that works very well in certain situations, but not in others.

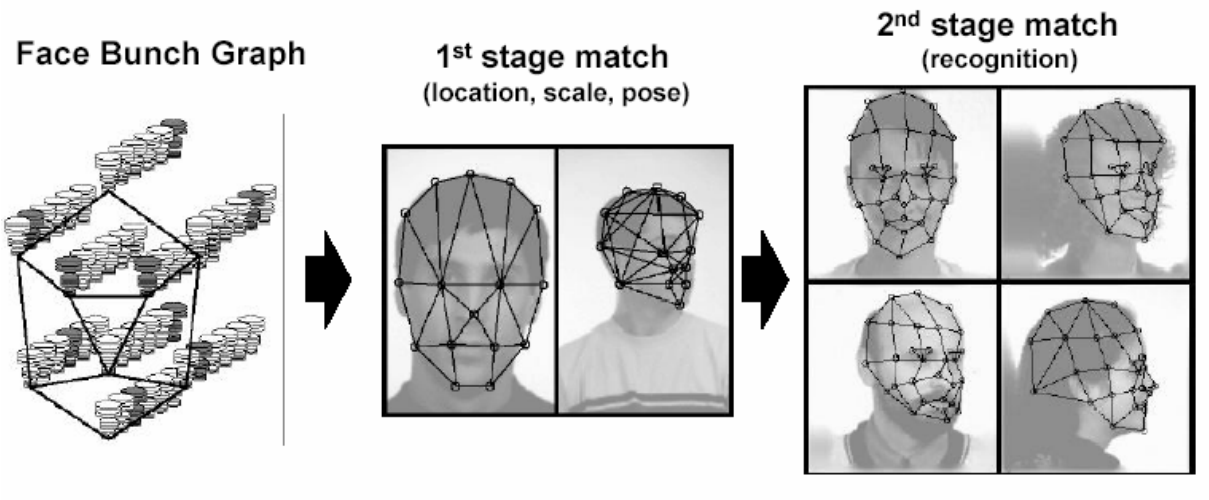


Fig. 3. The matching process in the Elastic Bunch Graph Matching

2.1.5 Support vector machines

In 2001, Guo [6] , incorporated Support Vector Machines (SVM's) with binary tree recognition for multi-class recognition. Given a set of points belonging to two classes, a traditional SVM finds a hyper-plane that separates the largest fraction of points of the same class on the same side while maximizing the distance from each class to the hyper-plane.

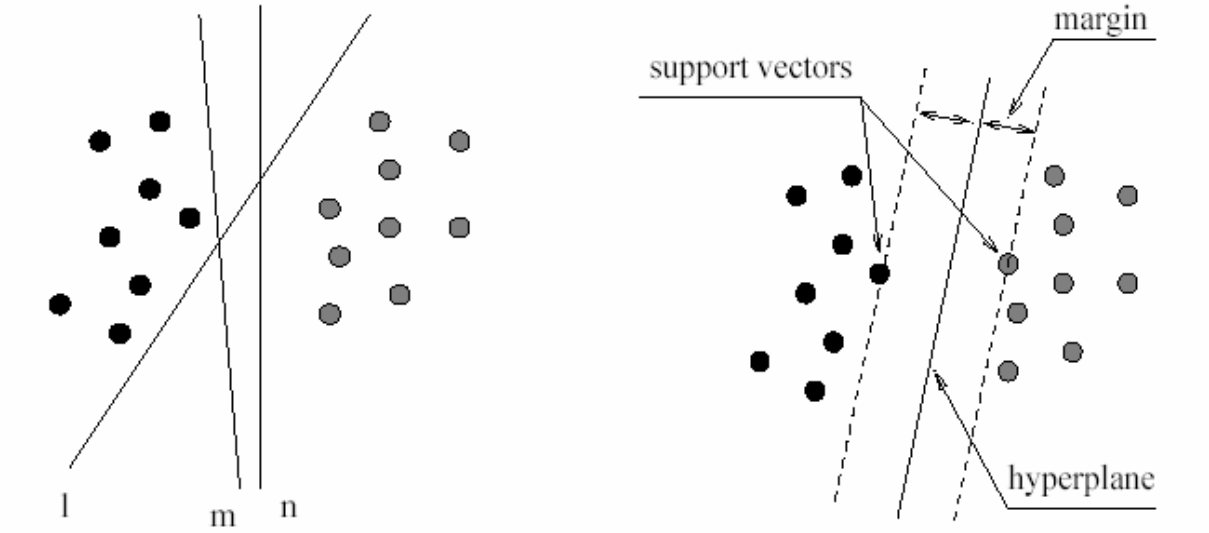


Fig 4. Classification using hyperplanes. 1. arbitrary hyperplanes, 2. optimal hyperplane.

However, in the case of Face Recognition, we have multiple classes, where each person belongs to a different class and therefore the authors had to extend SVM's so that they could be applied to the multi-class problem. They proposed a binary tree structure which is appropriate to extend the pairwise discrimination of the SVM's to a multi-class recognition scenario. In 2003, Li [7] proposed a new algorithm for face recognition over multiple views. In order to accomplish this, they divide the "view sphere" into different segments. On each segment a face detector is created and the pose of the detected face is

explicitly predicted. The algorithm was tested for face recognition over multiple views but the results obtained were unsatisfactory. For images with frontal views, the accuracy of the SVM's was found to be much higher compared to the Eigenface approach.

2.1.6 Matching inexact graphs

In 2001 Cesar [8] approached facial feature recognition as a problem of matching inexact graphs where the graphs were built from regions and relationships between regions in an image. The image where recognition has to be performed is represented as a graph G_D based on an over-segmentation performed using the watershed algorithm. Each region in the segmented image corresponds to a node in the graph. There exists a model graph G_M where each node corresponds to a facial feature and the algorithm focuses on using the inexact graph matching technique to map G_D to G_M since the number of nodes in each graph is different. Recognition is then performed by searching for a homomorphism between G_D and G_M that satisfies both structural and similarity constraints. The authors do not talk about extending their results from the inexact graph matching technique to face recognition.

However, their results suggest that it is an interesting idea to pursue, even though traditional face recognition techniques that rely on facial feature recognition have not been successful due to the lack of algorithms that can accurately extract facial features from images.

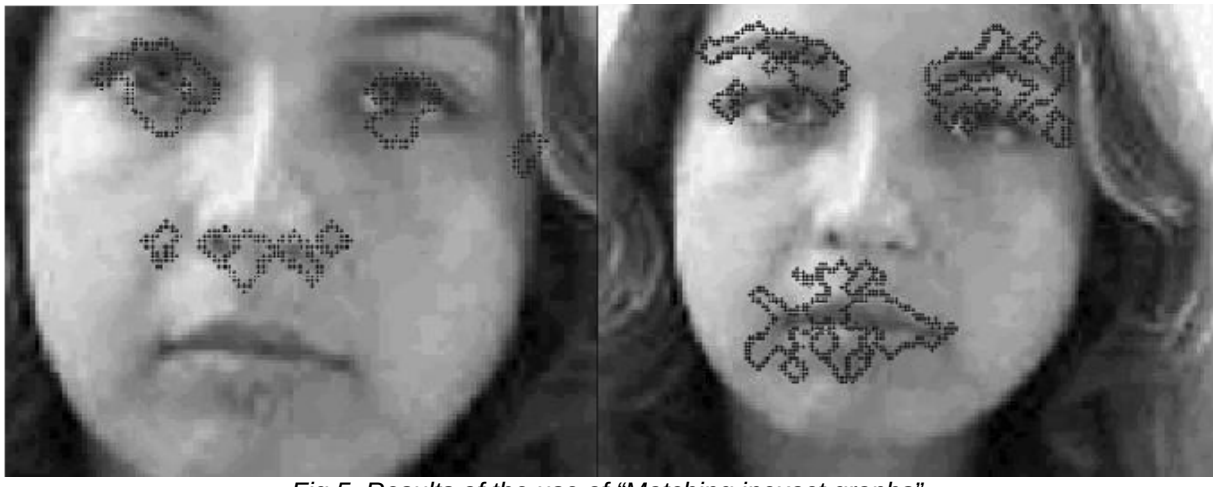


Fig 5. Results of the use of "Matching inexact graphs"

2.1.7 Depth and texture maps

Texture coding provides information about facial regions with little geometric structure like hair, forehead and eyebrows whereas a depth map provides us with information about regions with little texture such as chin, jaw line and cheeks. Considering this fact, Ben Abdelkader [9] *et al.* proposed that the accuracy of FRT systems can be improved by considering not only the texture map but also the depth map. While the results of their 3-D face recognition system are excellent, it may not always be feasible to use a structured light based 3D camera that can simultaneously capture the 3D shape and texture of the face in all applications.

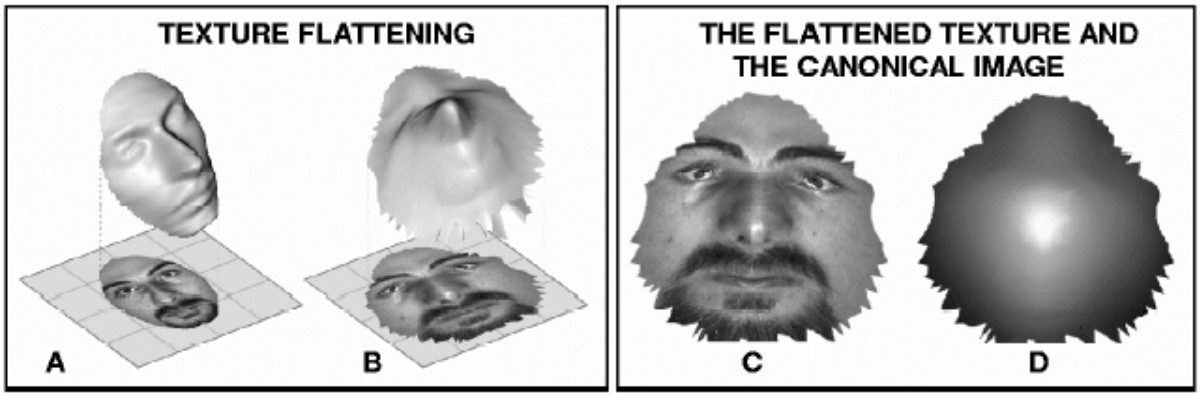


Fig 6. 3D face recognition. Texture flattening.

2.1.8 Multiresolution analysis

Ekenel and Sankur [10] proposed multiresolution facial recognition in 2005. They employ multiresolution analysis to decompose the image into its subbands prior to the subspace operations such as principal or independent component analysis. Some of the earlier techniques like Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Multidimensional Scaling (MDS) suffer from a performance drop whenever facial appearance is subject to occlusion and variations in illumination, expression, pose, accessories and aging. They applied a multiresolution technique to mitigate the loss of performance due to changes in facial appearance. A 2-D discrete wavelet transform was used to extract those components that are less sensitive to intrinsic deformations and then either PCA/ICA is performed on the vectors obtained from subband decomposition. Their algorithm obtains a significant performance gain especially against changes in facial expression. Even though multiresolution analysis addresses the issues of face recognition under varying illumination and facial expressions, it still fails to remove the constraint that the photographs need to be taken from a frontal view.

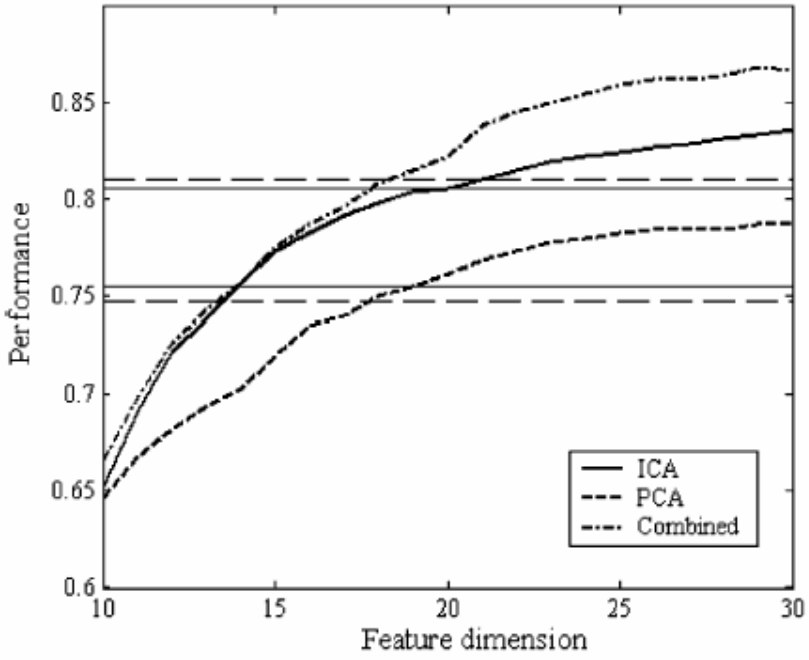


Fig 7. Improvement of using multiresolution analysis.

2.1.9 Tensor analysis

Vasilescu [11] tried to solve the problem of facial recognition using Tensor Analysis.

They identified the analysis of an ensemble of facial images resulting from the confluence of multiple factors related to scene structure, illumination, and viewpoint as a problem in multilinear algebra in which the image ensemble is represented as a higher-dimensional tensor. Using the "N-mode SVD" algorithm, a multilinear extension of conventional matrix singular value decomposition (SVD), this image data tensor is decomposed to separate and parsimoniously represent the constituent factors. The authors also propose a recognition method based on multilinear analysis which is analogous to the conventional one for linear PCA. The recognition algorithm performs TensorFaces decomposition of the Tensor containing vectorized training images and constructs the basis tensor B. The classifier uses the projection vector B-T in order to find the image with least error.

$$D = Z \; X_1 \, U_{people} \, X_2 \, U_{views} \, X_3 \, U_{illum} \, X_4 \, U_{expr} \, X_5 \, U_{pixels}$$

Fig 8. Image data tensor D.

2.2 Conclusions

The qualitative comparison of the all the techniques is shown in Table 1.

Techniques	Resistance to			Computational Efficiency	Classification Quality
	Illumination	**View**	**Expression**		
Geometric Features	Good	poor	Good	good	very poor
Eigenfaces	average	poor	Average	good	average
SVM	average	average	Average	good	very good
Depth and Texture maps	Good	good	Good	average	very good
Multiresolution Analysis	Good	good	very good	average	very good
Gabor Feature Classifier	Good	good	Good	average	very good
Tensor Analysis	very good	very good	very good	average	very good

Table 1. Results of the different methods of face recognition.

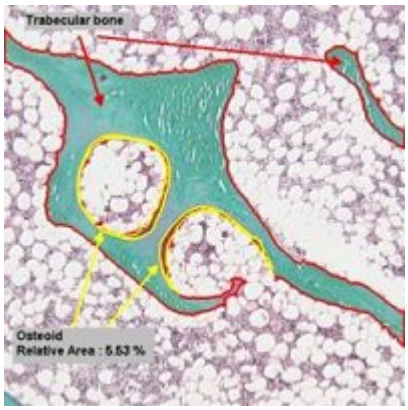
Based on preliminary examination, Tensor Analysis seems to give extremely good results for facial recognition under varying conditions of illumination, expression and pose.

3. Algorithm

3.1. Introduction to Carl Zeiss KS systems

The **CZ KS** image analysis systems are designed to provide quantitative measurement and statistical analysis of images, efficient archiving and comprehensive image enhancement to provide optimal quality images. For applications requiring high productivity many processes can be automated when combined with motorised microscope and motorised scanning stages.

CZ KS systems are available in two configurations, the entry level **CZ KS300** and advanced **CZ KS400**. Adding application-specific software modules can enhance the performance and functionality of each system thus ensuring **CZ KS300** and **CZ KS400** keep pace with changing image analysis demands. The **CZ KS300** and **CZ KS400** share several important advantages. Both are designed for an endless variety of applications from industry and research, plus they can both be expanded and enhanced to suit specific needs. A convenient user Windows™ interface is provided and all that is required is to select the desired functions and parameters from the toolbars and menus.



3.1.1. Carl Zeiss KS 400

A versatile image-processing program designed to support demanding research or development applications. This all-purpose high-end system lets solve *virtually* image processing problems quickly and reliably. The **CZ KS400** can be customised for specific applications by using additional modules.

Fig 9. Example of the KS systems. Measurement of histological sections of arteries.

3.1.2. Carl Zeiss KS 300

An extensive entry level program for image analysis and processing permitting fast, accurate measurement of morphological and densitometric parameters. The Macro-Recorder makes it easy to store frequently-used operations for automatic execution with all applications.

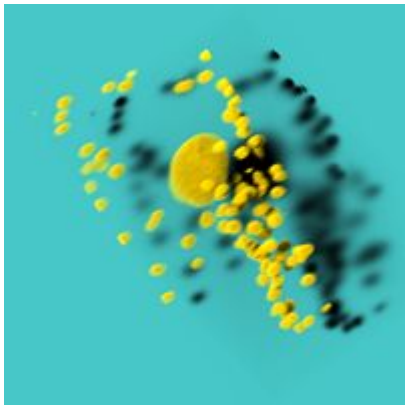


Fig 10. Example of the KS systems. 3D Reconstruction and measurement.

3.1.3. Advanced Image Analysis that KS systems support.

- 3D Reconstruction and Measurement
- FRET (Fluorescence Resonance Energy Transfer)
- Texture Analysis

- Fourier Analysis
- Classification
- Circular deagglomeration

3.2. Our work.

We worked on the **CZ KS400** system. It's installed in our laboratory in the FH München. There runs the KS400 3.0 Release. The program needs a dongle to work, so it's only in that laboratory available.

As we already know, the objective of our work is to perform **face recognition** using this program, now it could be better noticed the challenge nature of the project.

We did as a training the recognition of some seeds, this way we acquired a better and deeper knowledge on the **KS 400** system. Finally we started with the **face recognition** programming. During the programming, many versions and ways were explored. This ways are also explained on the papers, but most of them aren't the final solution.

We developed two different principal versions of the recognition.

- Classification with calculated features.
- Classification with calculated features and virtual grid.

The codes are macros, it isn't a fixed code because of its macro nature. It can be modified easily and work in another way very fast. The problem is that it needs the KS400 program to work, can't work itself. So, KS400 works as the interpreter of our codes. KS400 uses for its macros an own programming language. During the explanation, these will be explained to help the reader for a better comprehension.

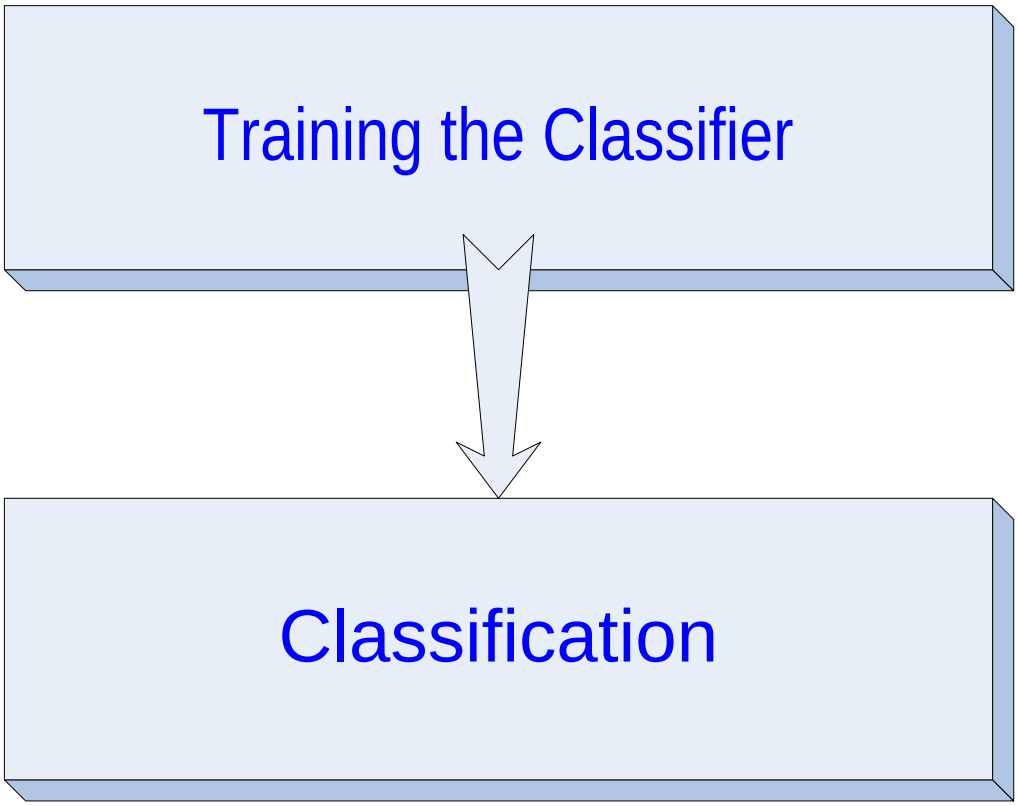
Also, many problems were found during the programming, because the KS system isn't design for this kind of works. These problems were sometimes very annoying, and sometimes they were difficult to solve, trough the explanation, they will be exposed, and also the solutions we developed for them.

Next, we talk about the structure. Both versions are very similar, so they are explained at the same time.

3.2.1. Structure

The program has two big different parts. In the first part, the program is trained. This training consists in an analysis of the images, then the features are measured and finally a classifier is trained. In the second part we used the same analysis and the same measurement but with an image that isn't in the classifier. Finally the classifier shows the results of the classification on the screen.

Fig 11. External operating of the program



In overall, this is the way the program works. In the next pages, it will be explained in detail.

3.2.2. First part 'Training the classifier'

We have divided this part in four different subparts. These subparts are:

- Images database measurement.
- Building the database "Face".
- Transformation and normalization of the database "FACE".
- Creation of virtual images and teaching the classifier.

The reason of that it's very easy. It's not difficult to understand, that this is the normal process to perform the training of a classifier. First we take the objects about we should learn, then we measure them and with the results we train a classifier. Only, the creation of virtual images is a strange step on this scheme. Later its reason will be explained.

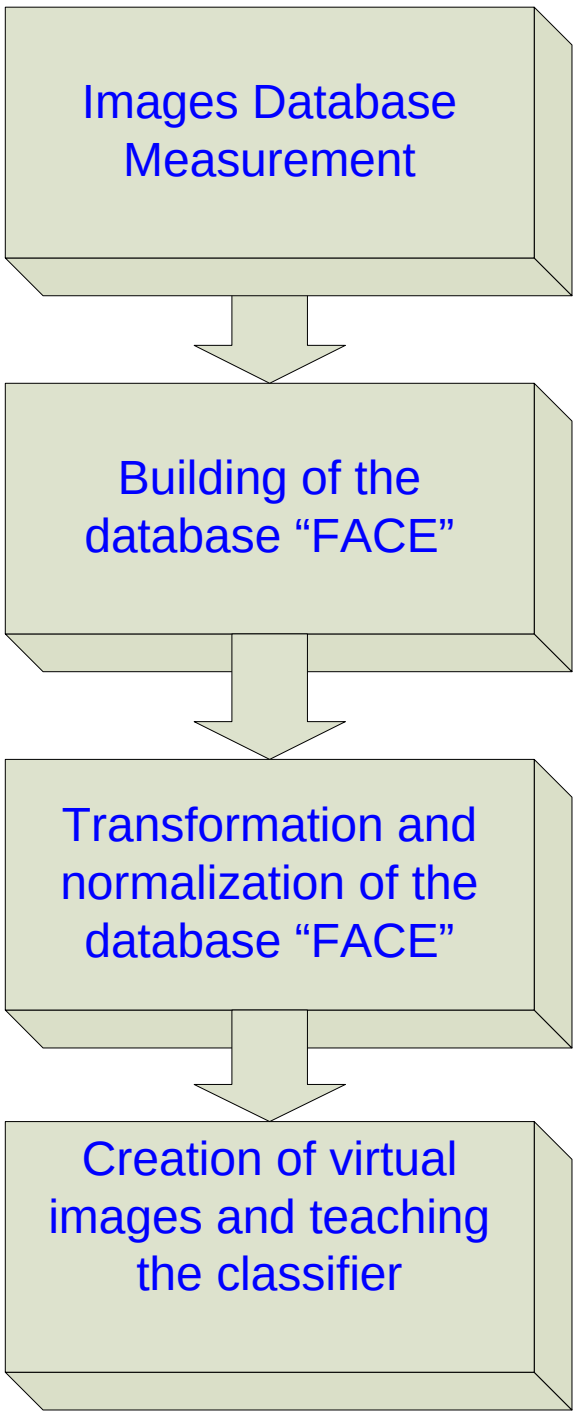


Fig 12. Structure of the Training of the Classifier part.

The image database measurement consists in loading the images of the database and measuring the face features (like eyes and mouth). To perform that, we need colour filters and other types of filters. Finally the results are saved on databases. These databases are named ‘Testmouth’ and ‘Testeyes’. The process is explained in detail in the next pages.

Next, we build a new database called ‘Face’. It’s easy to find the reason of its name. With the database from the mouth ‘Testmouth’ and the database of the eyes ‘Testeyes’, we build a new database with new features. These features are the old ones, or are calculated with the old ones.

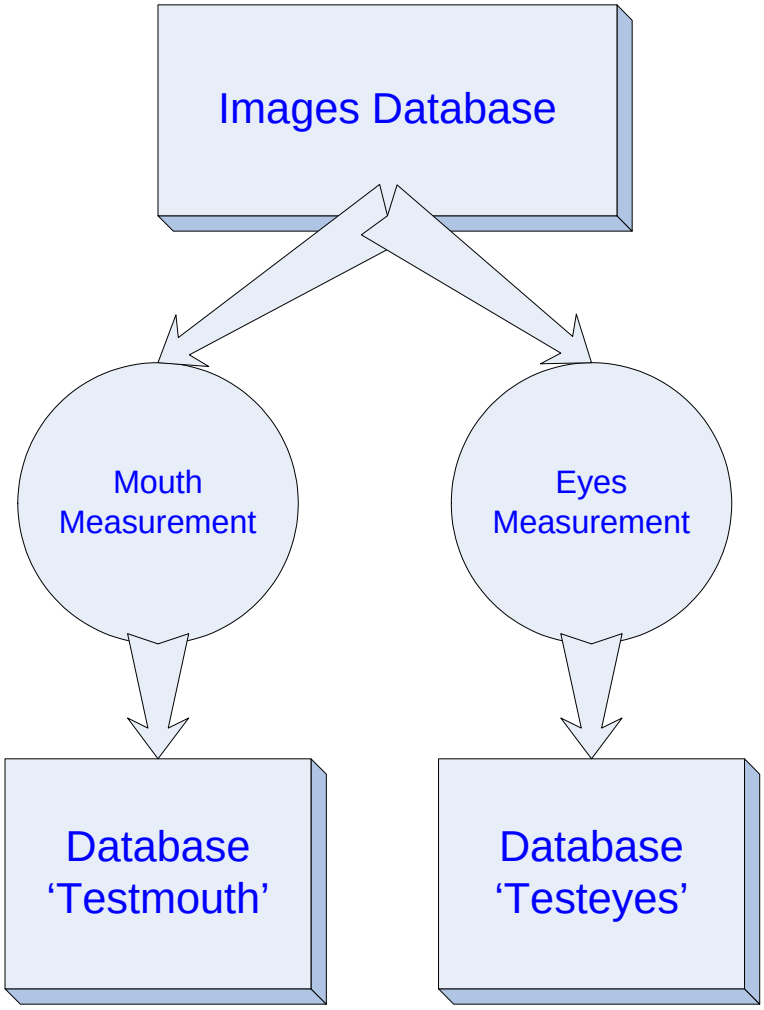
Afterwards, the database ‘Face’ is transformed. We need a normalized database. They are only with integer numbers, because in the implementation of the virtual images, real numbers aren’t allowed.

The normalization has been performed manually. The highest and lowest measurements have been listed. Through these numbers we performed the normalization of the database. The new values are from 0 to 100. But later, they are renormalized depending on the version. The reason of that is explained in the last part of this explanation. This new database is called ‘FaceNorm’.

The KS400 system doesn’t allow us to learn from a database (it sounds strange but it’s like that). Therefore, we had to perform a work-around to reach our objective. That work-around isn’t difficult. We created images, virtual images, from the database ‘Facenorm’. Then these images are measured and this way the program is able to learn from this database. Finally we create a classifier. Also the type of classifier (Mahalanobis, Bayes, Min-Dist) is defined. And it’s trained with the images that were created in the last step. It’s a quizzical method but was the only way that was found to avoid the problem.

Next, these blocks and their codes are analyzed in detail.

3.2.2.1. Images database measurement.



This part consists as we had already explained, in the loading and the measurement of the faces of the database. This block has also two different parts, the measurement of the mouth and the measurement of the eyes. It wasn't able to reach a good measurement of other parts of the face but it would be also useless due to the learning problem of the KS400 system that we later explain.

Fig 13. Structure of the Images database measurement.

The eyes and the mouth measurement are very similar on their structures. For example, the setup of the classifier, that it's going to be explained next.

In the KS400 systems, it's needed to setup a classifier to perform a measuring like that, in the classifier has to be defined:

- Classes, to perform the classification.
- Features of the region that is going to be measured.
- Conditions on these features.
- Type of classification.
- Databases.

First the different classes are defined. These classes are the people that we have to recognize. Then we defined a classifier and conditions of the classifier too. These conditions try to distinguish the useless forms from the useful ones. Here it's showed an example:

```
Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "david"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION","FERETY>50&&FERETX<20"
# MSsetcond
```

```
MSsetprop
"REGIONFEAT","CLASS,CLASSNAME,FERETY,CGRAVX,CGRAVY"
```

Finally the image is measured and in the results are saved in a database, this time “testmouth”.

```
MSmeasmask 4,2,"testmouth",0,1,1
```

- **Mouth measurement.**

Now, we talk only about the mouth measurement. First, the features that are going to be measured have to be chosen. It’s no very difficult to realize that the most interesting ones for the mouth are:

- Length
- Width
- Position of the mouth

These distances like the length or the wide are measured in pixels and the position of the mouth is the relative position in the image, and its represented by the centre of gravity points, x-axis and y-axis points. For our program they are represented this way.

- FERETY: Represent the length of the mouth.
- FERETX: Represent the wide of the mouth.
- CGRAVX: Represent the centre of gravity point in the x-axis.

- CGRAVY: Represent the centre of gravity point in the y-axis.

Afterwards, the conditions on the mouth had to be chosen. Certainly, the length and width of the mouths could be the perfect features, a mouth can't be wider as longer and its length has minimum values. So, it could be a good idea distinguishing mouths from other objects by their forms. Another way that we use to distinguish the mouths is its position. We use a mask to prevent us to take other objects which aren't the mouth.

In the next lines, the code that corresponds to the mouth measurement is explained in detail.

Here is the code:

```
Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "david"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION","FERETY>50&&FERETX<20"
# MSsetcond


MSsetprop
"REGIONFEAT","CLASS,CLASSNAME,FERETY,CGRAVX,
CGRAVY"
```

With *Gclear* in the first line we are prevented from mixing our actually images with other that come from old runnings.

Then, measurement properties are set. The first things we set are the classes, these classes distinguish the people. Each class is a different person. We defined them with their names. They are defined too with a number.

Afterwards we define a classifier, we used in this case no name ‘<none>’ for it, but for example ‘mouthclassifier’ could be used.

Finally, we set the condition of the objects that will be measured. These objects shall comply with the conditions we set. In this case the condition is "FERETY>50&&FERETX<20", which means that the object should be at least 50 pixels long in the y-axis and no longer as 20 pixels in the x-axis. This condition will be valid until another condition would be set.

With the command:

```
MSsetprop
“REGIONFEAT","CLASS,CLASSNAME,CGRAVX,CGRAVY,FERET
Y,FERETX”
```

We define the properties we want to measure; in this case (for the mouth) we measured the centre of gravity (CGRAVX for x-axis, CGRAVY for y-axis), the length of the mouth (FERETY) and the wide (FERETX).

The function MSsetprop is very important. It sets the measurement properties. Also, the properties could be set interactively; there are windows available in the KS400. In the appendix, we explain in more detail this function.

Now we load the images from the database in order to process them. We show the program a path to find them with the command:

```
imgsetpath "c:\ks400\conf\images\db"
```

Then we load them, for example we load Juan’s pictures. We load them together. They’re saved in images, 1, 5 and 10. It’s done this way in order to do it easier and logic.

```
imgload "juan1.bmp",1
imgload "juan2.bmp",5
imgload "juan3.bmp",10
```

Now, the images are converted into grey value images. It’s done with the command *imgRGB2grey* , which convert an RGB image into a standard grey value image with 256 grey tones. They’re saved on the images 2, 6, 11. Forward, we are going to talk about only one image, but it’s known that they’re three different ones, and three from each person.

```
imgRGB2grey 1, 2
```

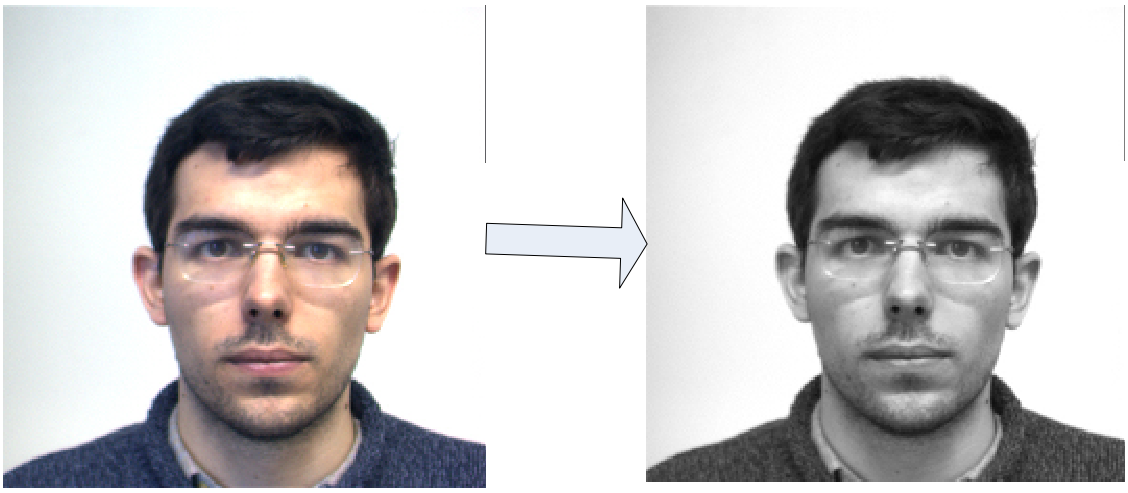


Fig 14. Transformation RGB to Grey values.

The mask, which helps us to distinguish the mouth from other objects, is also used. It has to be loaded from the memory. The command that it's used for is

```
imgload "mouth1.bmp",20
```

"`mouth1.bmp`" is the name of the file where this image is stored. And it is saved in auxiliary image number 20.

Afterwards, we use a filter. The reason of using that filter is ***very important***: **The program can't find itself the mouth in a RGB photo. It needs our help. It needs a simple mask of the face (binary image) where it could find out simple forms or objects. So, we build up an image with the original one, which helps the program.** This filter implements Grey value segmentation. It was fitted to show as clear as it is possible the mouth. This regulation depends of the light, (the photos can be lighter or darker) and this was fitted to measure our photos which have a specific light level. This is the command of the filter:

```
dislev 2,3,88,255,1
```

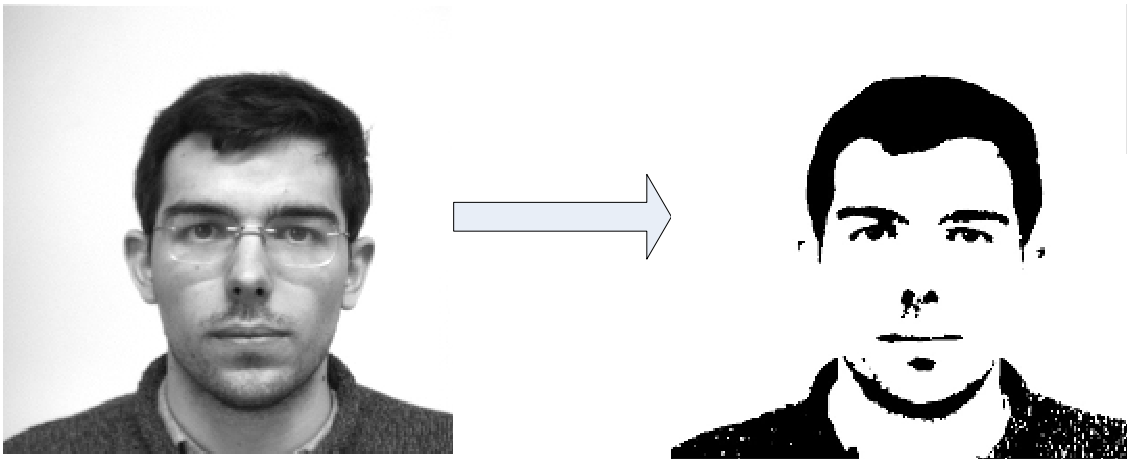


Fig 15. Application of the filter, grey value segmentation.

This image isn't the image we need, the one we need is exactly the opposite one, so we use the *not* tool to obtain the right one.

```
binnot 3,3
```

Afterwards, the mask of the mask is applied. It's simple we use the Boolean function AND. Where the mask is white, it stays and where the mask is black, becomes black.

```
binand 3,20,4
```

Finally, using the original image and the image we have created, the image is measured. The command MSmeasmask is the one that implement the measurement.

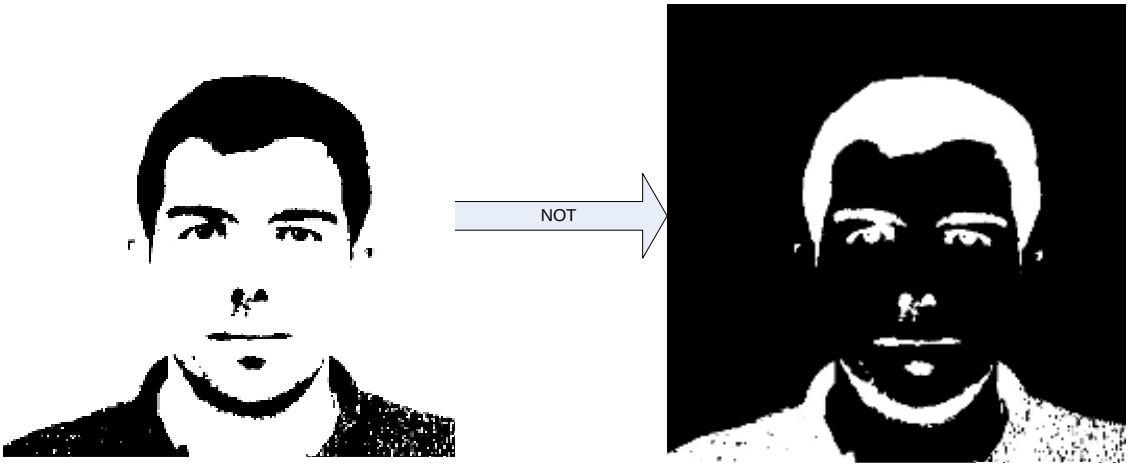


Fig 15. Application of the Boolean not function.

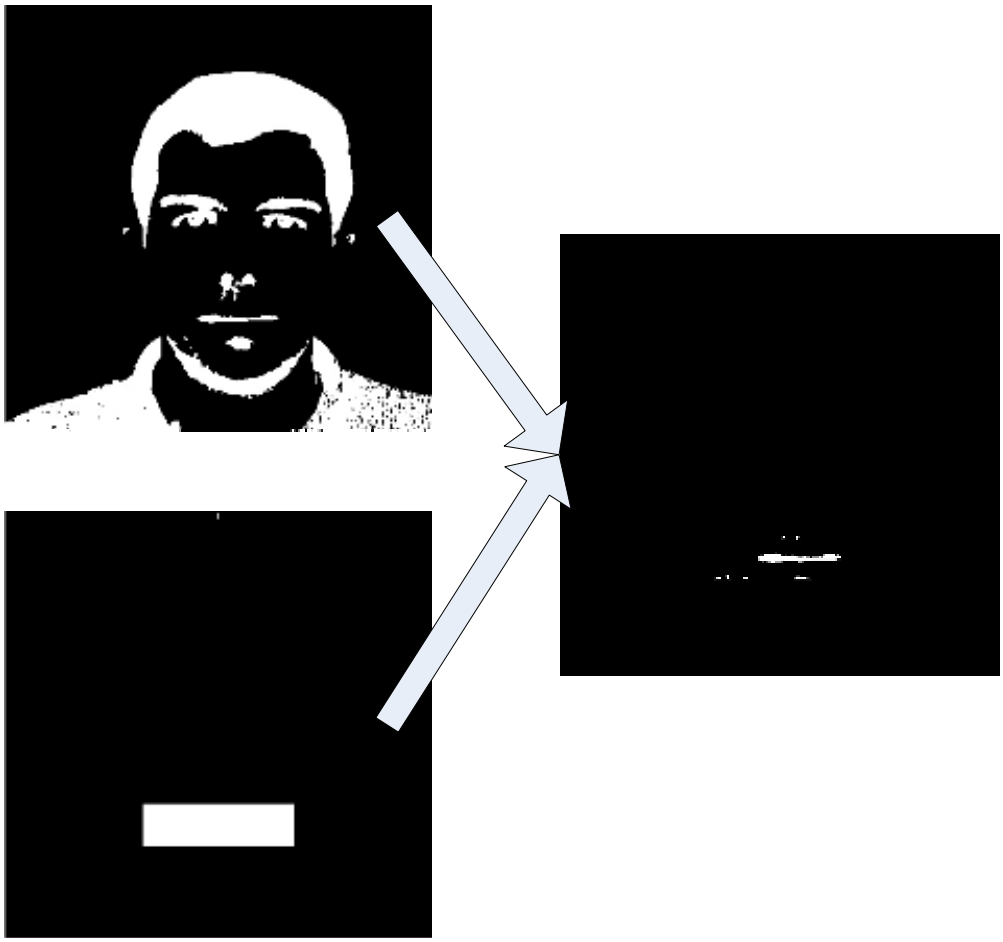


Fig 16. Application of the Mouth mask.

We define also the name of the database, this case “testmouth”.

```
MSmeasmask 4,2,"testmouth",0,1,1
```

Fig 17. Mouth measurement.

The results are showed with the command datalist:

```
datalist “testmouth”,0,0
```

- **Eyes measurement.**

Eyes measurement is our next objective, the eyes measurement procedure is very similar to the mouth one. So, explain it not as deep as we did it with the mouth.

The features that are going to be measured have to be chosen. Finally, were chosen these ones:

- Area.
- Circularity.
- Position of the eyes.
- Length
- Width.

These features are almost the same as the moth ones. We only added circularity and the area. The KS400 system represents them this way:

- DCIRCLE: Circularity of the eyes.
- AREA: Area in pixels^2.
- FERETY: Represent the length of the eye.
- FERETX: Represent the wide of the eye.
- CGRAVX: Represent the centre of gravity point in the x-axis.
- CGRAVY: Represent the centre of gravity point in the y-axis.

The conditions on the eyes were chosen. They were chosen in the process of tuning and depend on the light and distance conditions. We use also a mask like in the mouth case in order to prevent us to take other objects which aren’t the eyes.

The conditions that are now required for the objects have changed. Now they are:

```
MSsetprop "CONDITION","AREA>50 && FERETX>30
&& FERETX<80 && CGRAVX<286 && CGRAVX>206 &&
DCIRCLE > 42"
```

A positional requirement is set because sometimes the nose could be confused with an eye. These requirements were found during the tuning of the program and depend on the external conditions too.

The steps of the eye recognition are almost the same. The only difference is a filter that is applied on the photo after the *Grey value segmentation*. This filter detects regions or edges on an image. We used it because eyes are easy to distinguish but difficult to separate from eyebrows. So, we did the eye and the eyebrow an only object with this function. The command that was used is the next one:

```
dmarr 13,13,9.50,2
```

The rest of the eye detection is the same but the values of the filters or the name of the database, this time “testeyes”.

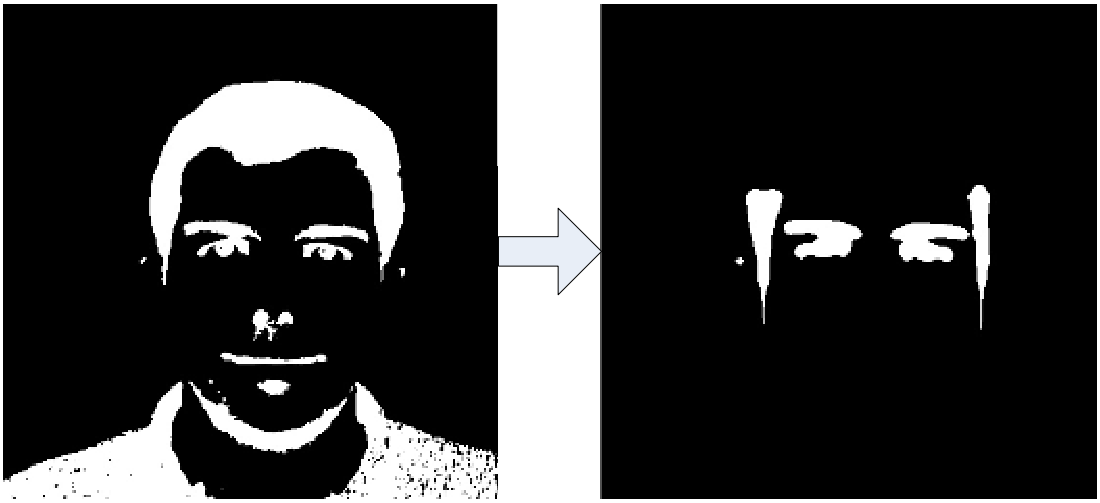


Fig 16. Image that results of after applying dmarr and the eyes mask.

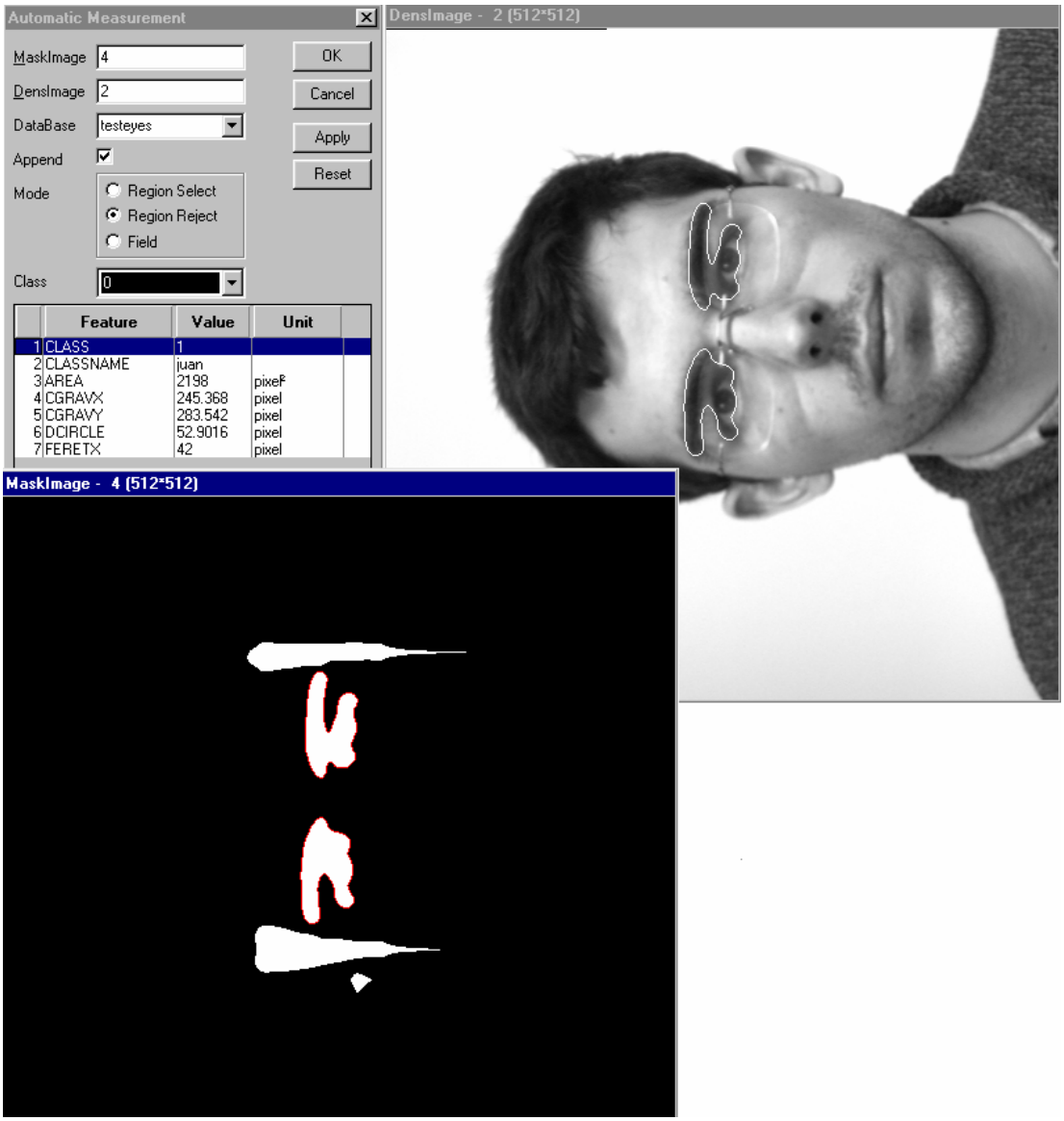


Fig 17. Eyes measurement.

3.2.2.2 Building the database 'Face'.

The database 'Face' contains the results of the calculation of new features based on the databases 'testmouth' and 'testeyes'. These features are, for example, the distance eye to eye or the angle mouth to eye.

During this process, we used the database commands. They can be easily recognized because they start with the letters DB, which correspond with the word *database*.

A new database is created, after that the new features are calculated and then inserted in the new database. The final features are:

- Mouth length.
- Eye 1 area.
- Eye 2 area.

- Eye to eye distance. <new>
- Eye 1 to mouth distance. <new>
- Eye 2 to mouth distance. <new>
- Eye 1 to mouth angle. <new>
- Eye 2 to mouth angle. <new>
- Circularity eye 1.
- Circularity eye 2.

They aren't used overall. That happens because the method we use to classify. Now, it's showed the method to create a database and to fill in it. We use in the example the database 'Face'.

First, the old databases are opened. We placed the pointer at the first line of the database with this command:

```
DBfirstline "testeyes"
DBfirstline "testmouth"
```

Afterwards the new database is created and its fields are defined.

```
DBnew "face",11
DBsetcolumn "face",1,"CLASS","Int","<none>"
DBsetcolumn "face",2,"CLASSNAME","String","<none>"
  …
```

Then we defined some auxiliary variables to help us in the transfer of data. With a 0 we define an integer number and with a 0.0 a float one.

```
mouthlength=0                  # Mouth Length
eyedistance=0.0                # Eye distance
classmouth=0
classeyes1=0
classeyes2=0
…
```

Now the data is transferred. We take the data from the old database, then new features are calculated and finally they are written in the new database. We use a while loop that finishes when we are at the end of the source databases.

```
while 1
if _STATUS == 0: break
DBgetvalue "testmouth","CLASS",classmouth
DBgetvalue "testmouth","CLASSNAME",classmouthn
DBgetvalue "testmouth","FERETY",mouthlength
…
DBnextline "testeyes"
DBgetvalue "testeyes","CLASS",classeyes2
DBgetvalue "testeyes","CGRAVX",xeyes2
…
```

The old features that we use are:

Mouth length:

Fig 18. Mouth length.

Eye area 1 and eye area 2 and circularity:

Fig 19. Measurement of eye area and circularity.

Here, the new features are calculated:
Eye-eye distance. We supposed that the eyes are at the same level.

```
eyedistance= abs( yeyes1 - yeyes2)
```

Eye-mouth distances.

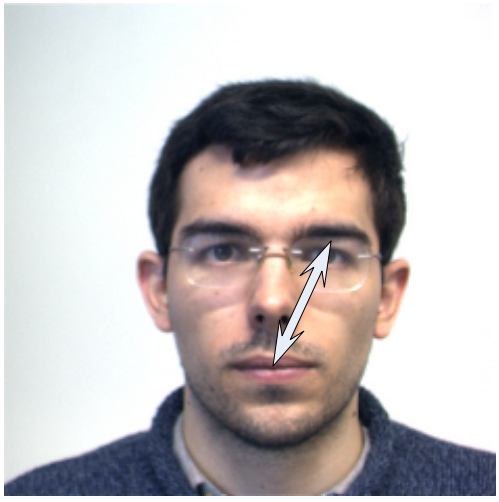


Fig 20. Eye mouth distance.

```
emdi = sqrt((yeyes1 -ymouth)*(yeyes1-
ymouth)+(xeyes1-xmouth)*(xeyes1-xmouth))

emdd = sqrt( (yeyes2 - ymouth)*(yeyes2 -
ymouth)+(xeyes2 - xmouth)*(xeyes2 - xmouth))
```

$$eyemouthdist = \sqrt{\left(y_{eye} - y_{mouth}\right)^2 - \left(x_{eye} - x_{mouth}\right)^2}$$

Eye-mouth angles in grades.

```
ema5 = asin ( (eyedistance)/(2*emdi))
ema6 = asin ( (eyedistance)/(2*emdd))
ema5 = (ema5 * 180)/3.141593
ema6 = (ema6 * 180)/3.141592
```

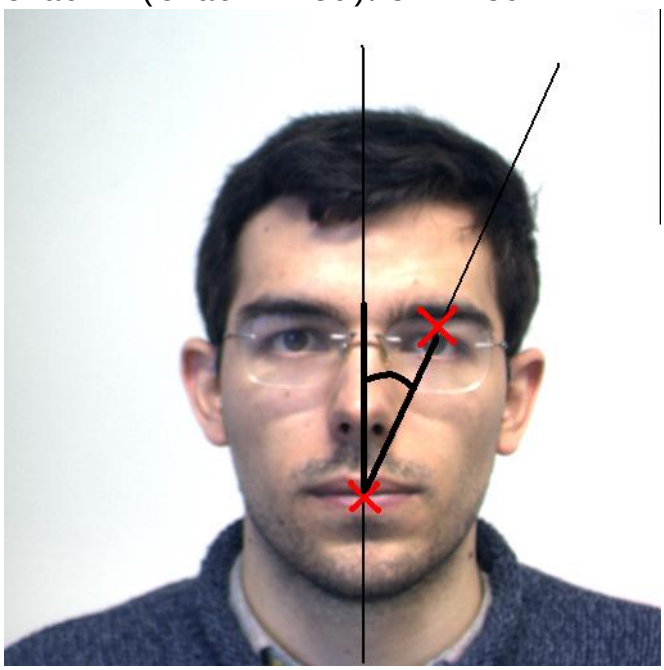


Fig 21. Eye mouth angle.

Finally the new database is filled with the new and old values.

```
DBaddline "face"
DBsetvalue "face", "CLASS",classmouth
DBsetvalue "face", "CLASSNAME",classmouthn
DBsetvalue "face", "MLength",mouthlength
…
endwhile
```

And the results are shown:

```
datalist "face",0,1
```

3.2.2.3 Transformation and normalization of the database ‘Face’.

The next step is preparing the new database for the creation of virtual images. We are going to normalize the values of the database ‘Face’.

We were able to face this normalization in two different ways. The first way was a manual normalization, in which we the values maximum and minimum were chosen. And the second one, this way was an automatic normalization, in which the program would take the maximum and the minimum values of the table and would establish them as the normalization values. The first one was chosen because the program could be very slow if also this automatic normalization were implemented.

After choosing the method, it has to be implemented. The database ‘Face’ is opened and a new face, which is called ‘Facenorm’, is built from the old one.

The normalization could be changed fast and easily. The only thing that has to be done is changing the values of the normalization constants. As we already know, these constants are extracted manually from the database ‘Face’.

These constants are defined at the beginning of the code. There are two constants for each feature. They are:

- Mouth length constants: MLA, MLB
- Eyes area constants: AEA,AEB
- Eye distance constants: EDA,EDB
- Eye mouth distance constants: EMDA, EMDB
- Eye mouth angle constants: EMAA, EMAB
- Circularity constants: CA, CB

The method we followed was:

1. Extract the maximum and the minimum value.
2. Calculate the difference.
3. We use as a safe gap, the 10% of the total distance.
4. Normalize the values.

Example:

A feature which range comes from 200 to 300.
The range is 100.
The safe gap is 10.
We normalize from 190-310 into 0-100.

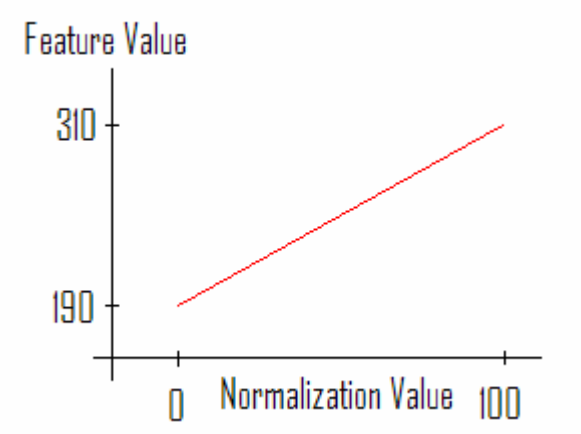


Fig 22. Normalization method.

We implemented two types of normalization. The features could be normalized in a 0-10 range or in a 0-100 range. These two types of normalization are needed for a classification using a virtual grid (on the next pages), and aren't needed for the normal grid.

The KS400 system has a strange problem. It can't transform a float value into an integer value. We need integer values because the commands we use in the next step use as input this type of values. The next lines expose the solution that we took.

```
while mouthlength > 1
integML = integML + 1
mouthlength = mouthlength - 1
endwhile
```

3.2.2.4 Creation of virtual images and teaching the classifier.

This is the part where the classifier learns. The teaching should be very simple, but due to some problems, it could be very difficult. These difficulties come from the program that isn't designed for this kind of work.

The first problem we had, was a very big problem. The problem is that our program can't learn itself from a database. The only way it can learn is from a database created from the classification of objects on an

image. So we found a work-around, this work-around consists in creating images from the database face norm and then do the classifier learn from that images.

We had also another big problem. The forms we can create with the KS400 systems are very simple. They are only rectangles and circles. We chose the rectangles because they are more difficult (they have more variables to choose) than the circles.

In the normalization we transformed the values into integer values. The reason is, that the command *Grect,* which is used in the creating of rectangles, can handle only integer values. We lost some accuracy at this step. And also in the normalization because some of the features had big values and we reduced them to a 0-100 range.

```
      Grectg eyedistance2, emdi2 , dcircle4 ,
mouthlength ,7,7
```

Fig 23. Structure of the part 'Creation of virtual images and teaching the classifier.'

In a first approximation, we were able to use only four of the features, because only four of the features were able to be introduced in the virtual image. The *Grect* command has these inputs:

Grectg *StartX, StartY, SizeX, SizeY, Colour, Mode*

We used the first four ones, which are the position and the size of the rectangles. We can combine the features that we use for these inputs and multiple results are possible.

The code is very simple. We use a 512x512 pixels image to draw the rectangles. This image is a called 'white.bmp'.

```
 Gclear
imgload "white3.bmp",1
seldisplay "Display"
imgdisplay 1

Grectg eyedistance2, emdi2 , dcircle4 , mouthlength
,7,7
```

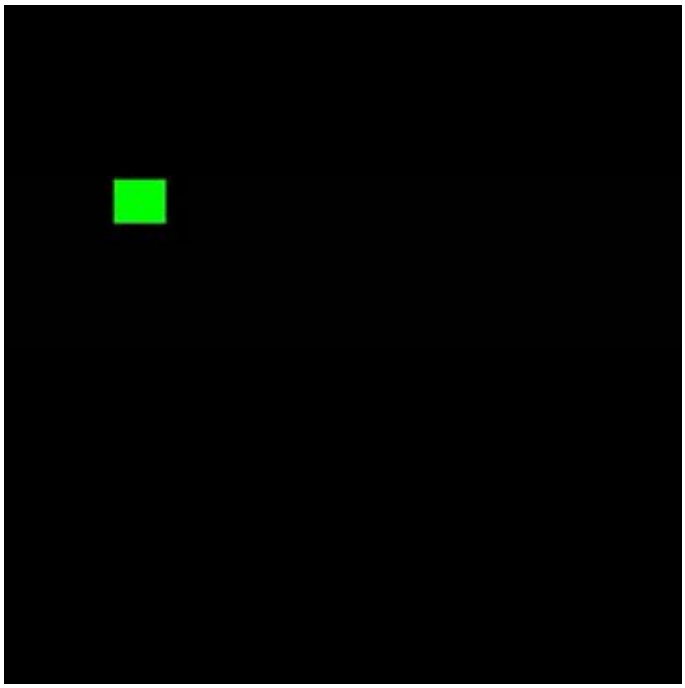


Fig 24. Example of a rectangle created.

Then the rectangle is drawn (*Grectg* command) and then fixed on the image (*Gmerge* command).

```
Gmerge
imgdisplay 1
dislev 1,2,3,252,1
imgRGB2grey 1, 1
```

It's needed also a mask image; the mask is always needed for the automatic measurement. So, we used again the filters to obtain this mask image.

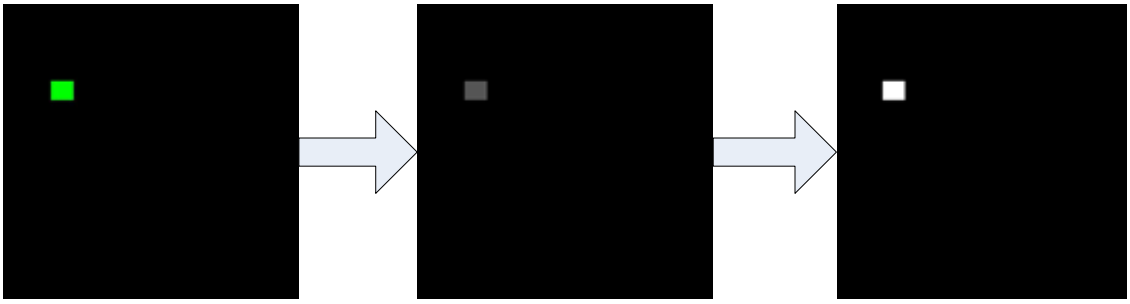


Fig 25. Obtaining the mask image.

```
if i==0
MSmeasmask 1,2,"virtualtest",0,1,classmouth
endif
if i==1
MSmeasmask 1,2,"virtualtest",1,1,classmouth
endif
i=1
```

Afterwards, the rectangles are measured and the data is saved in the database ‘virtualtest’.

```
DBnextline "facenorm"
endwhile

datalist "virtualtest"
```

Finally, the classifier learns from the database ‘virtualtest’. The kind of the classifier is also defined, in the example we have chosen a Mahalanobis classification.

```
#    teaching of the classifier
MSclassteach "VIRTUALTEST","MAHALA","MAHALA-ERGEB",1
! datalist MAHALA-ERGEB,0,0
```

So, we can only use four features to classify. It could be better to be able to use more features. So, a new way to perform the classification was developed.

This way is based in the old one. Instead of a 512x512 image, now it’s used a 2000x2000 pixels image. We use this big image as a grid. This grid had 100 squares, each one of 200x200 pixels sized. So, another two features can be used, one on the x-axis and the other on the y-axis. Their range is from 0 to 10.

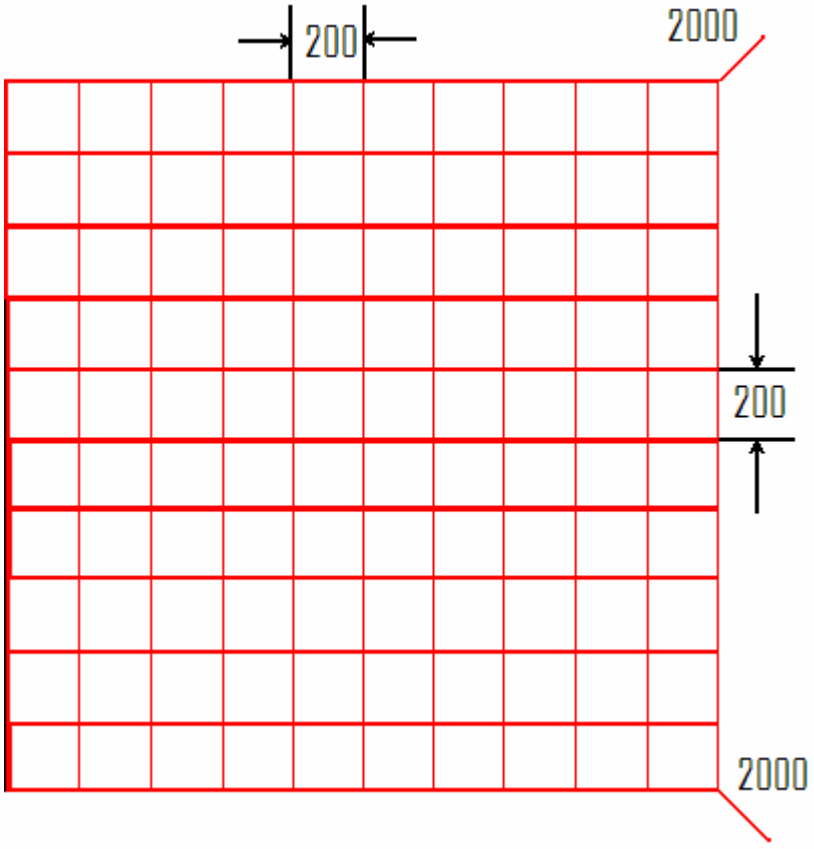


Fig 26. New grid used to perform classification.

This grid facilitates the classification because the rectangles are better distributed in accordance with the faces. Only a little change has to be implemented. Two new variables have to be added to the old ones. And they move on the grid with 200 pixels steps. The rectangle is never in two different squares. Because its maximum values are normalize from 0 to 100. The next figure shows it:

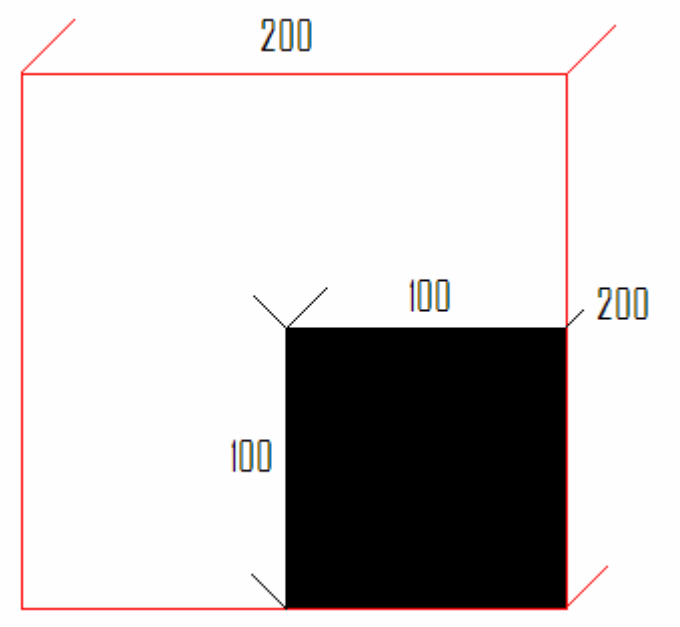


Fig 27. Structure of a square of the grid with the rectangle inside with maximum values.

These are the commands that have to be added, furthermore the image called 'white3.bmp' has to be a 2000x2000 pixels image.

```
imgload "white3.bmp",1
seldisplay "Display"
imgdisplay 1
# Grectg
mouthlength,eyedistance2,dcircle4,areaeyes1,ema7,7

eyedistance2 = eyedistance2 + 200 * areaeyes1
emdi2= emdi2 + 200 * ema7

Grectg eyedistance2, emdi2 , dcircle4 , mouthlength
,7,7
```

This case we used the area of the eyes and the eye mouth angle as the new features.

But it has also a problem. The problem is that the program has to use very big images, 2000x2000 pixels. And that makes the program too slow, besides the improvement of the classification isn't very important. In overall, the old one could be easier, faster and sometimes better.

3.2.3. Classification

When the classifier has learnt, the classification phase starts. This phase is very similar to the 'Training of the classifier' phase because in overall it has the same phases. Let's see them:

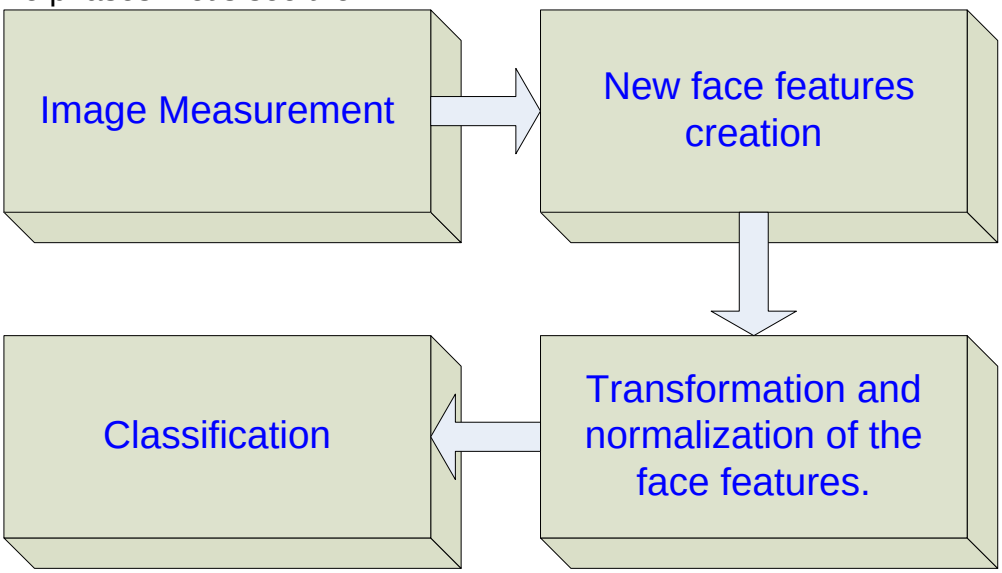


Fig 28. Structure of the classification phase.

The differences are:

- The image could be taken directly from the camera.

```
tvsetup
wait 10000
tvframeinput 1,0,0,512,512
imgsave 1,"testing.bmp"
imgload "testing.bmp",1
```

- Now only one image is processed.

- MSclassteach, the teaching command, is substituted for MSmeasmask to classify.

```
MSsetprop "CLASSIFIER","MAHALA"
MSmeasmask 1,2,"MAHALA~1",0,0,0
! datalist "MAHALA~1",0,0
```

- And finally the result is showed on the screen.

```
DBfirstline "MAHALA~1"
DBgetvalue "MAHALA~1", "CLASS",classmouth
DBgetvalue "MAHALA~1", "CLASSNAME",classmouthn

if classmouth>0
```

```
n=12
endif

if n==12
MBok "The person is "+classmouthn+"."
endif

if n==0
MBok "No person identified"
```

3.3 Classifiers

Our program can use three types of classifiers:

- Minimum distance classifier.
- Bayes classifier.
- Mahalanobis classifier.

Next these types of classifier are shortly explained.

3.3.1 Minimum distance classifier

It frequently happens that we can measure a fixed set of d features for any object or event that we want to classify. For example, we might always be able to measure

$x_1 = area$
$x_2 = perimeter$
...
$x_d = arc\_length / straight\_line\_distance$

In this case, we can think of our feature set as a **feature vector** $\boldsymbol{x}$, where $\boldsymbol{x}$ is the d-dimensional column vector

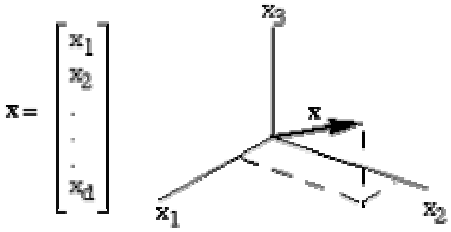


Fig 29. Feature vector.

Equivalently, we can think of $\boldsymbol{x}$ as being a point in a d-dimensional **feature space**. By this process of feature measurement, we can represent an object or event abstractly as a point in feature space.

This leads us to the following classical model for pattern recognition. A system or program called the **feature extractor** processes the raw data to determine the numerical values for a set of d features $x_1,x_2,...,x_d$, which comprise the components of a feature vector $\boldsymbol{x}$. A system or program called the **classifier** receives $\boldsymbol{x}$ and assigns it to one of c categories, *Class 1, Class 2,..., Class c*.

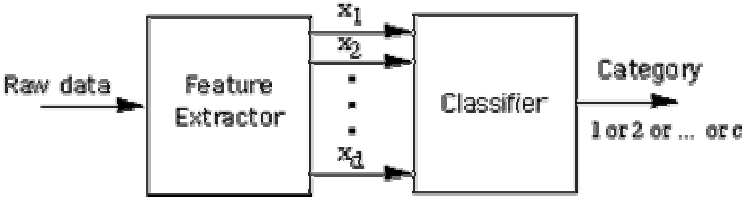


Fig30. External Scheme of the problem.

The design of the feature extractor is very problem dependent. The ideal feature extractor would produce the same feature vector $\boldsymbol{x}$ for all patterns in the same class, and different feature vectors for patterns in different classes. In practice, different inputs to the feature

extractor will always produce different feature vectors, but we hope that the within-class variability is small relative to the between-class variability.

At this point, we assume that the designer of the feature extractor has done the best job it can, and that the feature vector contains the information needed to distinguish the patterns. Given that feature set, our job is to design the classifier.

Let $\boldsymbol{x}$ be the feature vector for the unknown input, and let $\boldsymbol{m}_1,\boldsymbol{m}_2,...,\boldsymbol{m}_c$ be templates (i.e., perfect, noise-free feature vectors) for the c classes. Then the error in matching $\boldsymbol{x}$ against $\boldsymbol{m}_k$ is given by $\| \boldsymbol{x} - \boldsymbol{m}_k \|$.

Here $\| \boldsymbol{u} \|$ is called the **norm** of the vector $\boldsymbol{u}$. A minimum-error classifier computes $\| \boldsymbol{x} - \boldsymbol{m}_k \|$ for $\boldsymbol{k} = \boldsymbol{1}$ to $\boldsymbol{c}$ and chooses the class for which this error is minimum. Since $\| \boldsymbol{x} - \boldsymbol{m}_k \|$ is also the distance from $\boldsymbol{x}$ to $\boldsymbol{m}_k$, we call this a ***minimum-distance*** classifier.

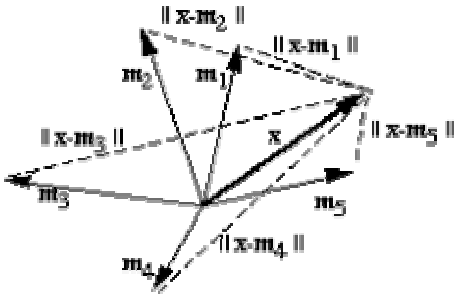


Fig 31. Minimum distance classifier.

There is more than one way to define the norm || **u** ||, and these correspond to different ways of measuring distance, i.e., to different **metrics**. Two of the most common are

1. **Euclidean** metric*:* $\| \boldsymbol{u} \| = sqrt(u_1^2 + u_2^2 + ... + u_d^2)$

2. **Manhattan** (or taxicab) metric: $\| \boldsymbol{u} \| = |u_1| + |u_2| + ... + |u_d|$

For the rest of these notes, we will use either the Euclidean distance or something called the Mahalanobis distance. We will see that

- Contours of constant Euclidean distance are circles (or spheres)
- Contours of constant Manhattan distance are squares (or boxes)
- Contours of constant Mahalanobis distance are ellipses (or ellipsoids)

Fig 32. Contours of different metrics

Recall that when we use a minimum-distance classifier to classify a feature vector $\boldsymbol{x}$, we measure the distance from $\boldsymbol{x}$ to the templates $\boldsymbol{m}_1,\boldsymbol{m}_2,...,\boldsymbol{m}_c$ and assign $\boldsymbol{x}$ to the class of the nearest template. Using the inner product to express the Euclidean distance from $\boldsymbol{x}$ to $\boldsymbol{m}_k$, we can write

$$\| \mathbf{x} - \mathbf{m}_k \|^2 = (\mathbf{x} - \mathbf{m}_k)' (\mathbf{x} - \mathbf{m}_k)$$

$$= \mathbf{x}'\mathbf{x} - \mathbf{m}_k'\mathbf{x} - \mathbf{x}'\mathbf{m}_k + \mathbf{m}_k'\mathbf{m}_k$$

$$= -2 \left[\mathbf{m}_k'\mathbf{x} - 0.5\, \mathbf{m}_k'\mathbf{m}_k \right] + \mathbf{x}'\mathbf{x}$$

Notice that the $x'x$ term is the same for every class, i.e., for every k. To find the template m_k that minimizes $\| x - m_k \|$, it is sufficient to find the m_k that maximizes the bracketed expression, $m_k' x - 0.5\, m_k' m_k$. Let us define the **linear discriminant function** $g(x)$ by

$$g(x) = m' x - 0.5 \| m \|^2 .$$

Then we can say that a minimum-Euclidean-distance classifier classifies an input feature vector x by computing c linear discriminant functions $g_1(x), g_2(x), \ldots, g_c(x)$ and assigning x to the class corresponding to the maximum discriminant function. We can also think of the linear discriminant functions as measuring the correlation between x and m_k, with the addition of a correction for the "template energy" represented by $\| m_k \|^2$. With this correction included, a minimum-Euclidean-distance classifier is equivalent to a maximum-correlation classifier.

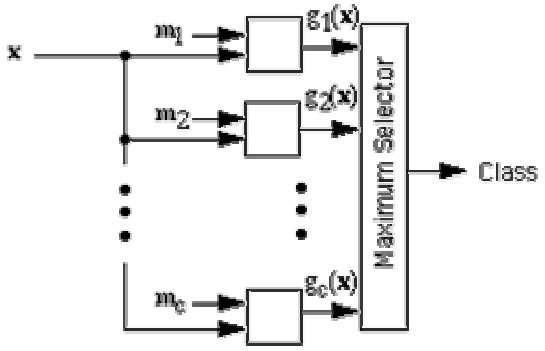


Fig 33. External Scheme of the minimum distance classifier.

3.3.2 Mahalanobis classifier

The quantity r in

$$r^2 = (\mathbf{x} - \mathbf{m}_x)' \, C_x^{-1} (\mathbf{x} - \mathbf{m}_x)$$

is called the **Mahalanobis distance** from the feature vector x to the mean vector m_x, where C_x is the covariance matrix for x. It can be shown that the surfaces on which r is constant are ellipsoids that are centered about the mean m_x. In the special case where the features are uncorrelated and the variances in all directions are the same, these surfaces are spheres, and the Mahalanobis distance becomes equivalent to the Euclidean distance.

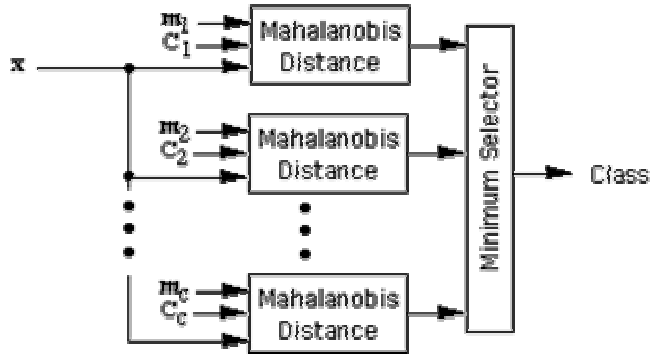


Fig 34. Mahalanobis solution.

One can use the Mahalanobis distance in a minimum-distance classifier as follows. Let $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_c$ be the means (templates) for the c classes, and let $\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_c$ be the corresponding covariance matrices. We classify a feature vector $\boldsymbol{x}$ by measuring the Mahalanobis distance from $\boldsymbol{x}$ to each of the means, and assigning $\boldsymbol{x}$ to the class for which the Mahalanobis distance is minimum.

The use of the Mahalanobis metric removes several of the limitations of the Euclidean metric:

1. It automatically accounts for the scaling of the coordinate axes

2. It corrects for correlation between the different features

3. It can provide curved as well as linear decision boundaries

However, there is a price to be paid for these advantages. The covariance matrices can be hard to determine accurately, and the memory and time requirements grow quadratically rather than linearly with the number of features. These problems may be insignificant when only a few features are needed, but they can become quite serious when the number of features becomes large.

There is an important special case in which the Mahalanobis metric again results in linear discriminant functions. This case occurs when the clusters in all c classes have the same covariance matrix $\boldsymbol{C}$. Geometrically, this situation arises when the clusters for all of the categories have the same ellipsoidal shape. Physically, this situation arises when the problem is to detect one of c different signals in the presence of additive noise, where the noise does not depend upon which of the signals is present. In that case, we can expand the squared Mahalanobis distance from the feature vector $\boldsymbol{x}$ to the mean vector $\boldsymbol{m}_k$ as

$$
\begin{aligned}
r^2 &= (\mathbf{x}-\mathbf{m}_k)' \mathbf{C}^{-1} (\mathbf{x}-\mathbf{m}_k) \\
&= \mathbf{x}' \mathbf{C}^{-1} \mathbf{x} - \mathbf{m}_k' \mathbf{C}^{-1} \mathbf{x} - \mathbf{x}' \mathbf{C}^{-1} \mathbf{m}_k + \mathbf{m}_k' \mathbf{C}^{-1} \mathbf{m}_k \\
&= -2 \left[(\mathbf{C}^{-1} \mathbf{m}_k)' \mathbf{x} - 0.5\, \mathbf{m}_k' \mathbf{C}^{-1} \mathbf{m}_k \right] + \mathbf{x}' \mathbf{C}^{-1} \mathbf{x}
\end{aligned}
$$

This should remind you of the similar expression we obtained for a minimum-Euclidean-distance classifier. Once again we can obtain linear discriminant functions by maximizing the expression in the brackets. This time we define the linear discriminant function $\boldsymbol{g(x_k)}$ by

$$g_k(\mathbf{x}) = \mathbf{w}_k' \mathbf{x} + w_{k0}$$

where

$$\mathbf{w}_k = \mathbf{C}^{-1} \mathbf{m}_k \qquad \text{and}$$

$$w_{k0} = -0.5\, \mathbf{m}_k' \mathbf{C}^{-1} \mathbf{m}_k$$

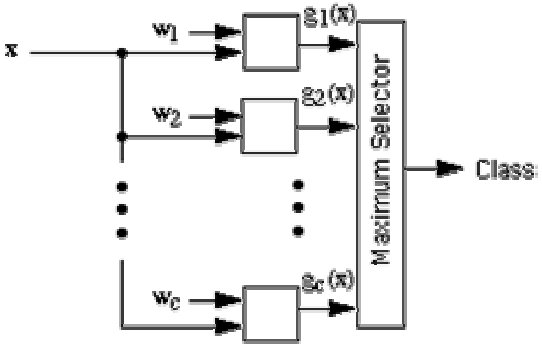


Fig 35.External Scheme of the Mahalanobis classifier.

This result is very useful. Although it gives up the advantages of having curved decision boundaries, it retains the advantages of being invariant to linear transformations. In addition, it reduces the memory requirements from the c d-by-d covariance matrices to the c d-by 1 weight vectors $\boldsymbol{w}_1,\boldsymbol{w}_2,...,\boldsymbol{w}_c$, with a corresponding speed-up in the computation of the discriminant functions. Finally, when the covariance matrices are the same for all c classes, one can pool the data from all the classes and get a much better results from a limited amount of data.

3.3.3 Bayes classifier

The basis of Bayes classification (also known Naive Bayes) is Bayes theorem. Essentially we want to classify a new record based on probabilities estimated from the training data. That is, we want to determine the probability of a hypothesis h (or specifically class membership) given the training data D (i.e., $P(h \mid D)$) Bayes theorem gives us the clue to calculating this value:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

Ignoring the detail we can work with $P(D|h)P(h)$ to identify a hypothesis that best matches our data (since $P(D)$ is constant for a particular D). Both of these quantities we can estimate from the training data D. $P(h)$ is simply the proportion of the database consistent with hypothesis h (i.e., the proportion of records with class c_i-- c_i is regarded as the hypothesis).

Calculation of $P(D|h)$ still poses some challenges and this is where the naive part of naive Bayes comes in. The naive assumption is that the features (each record D is describe by features a_1, a_2…, a_n) are conditionally independent given the class (i.e., the hypothesis that the classification is c_j, which could be the class soft in our example data) of the record. That is, given that a patient has a soft contact lens, to use our example, the probability of the patient being all of young, myopic, non-astigmatic and with a normal-tear-rate is the same as the product of the individual probabilities. The naive assumption, more concretely, is that being astigmatic or not, for example, does not affect the relationship between being young given the use of soft lenses. Mathematically we write this as:

$$P(a_1, a_2, ..., a_n \mid c_j) = \prod_i P(a_i \mid c_j)$$

Empirically determining the values of the joint probability on the left of this equation is a problem. To estimate it from the data we need to have available in the data every possible combination of values and examples of their classification so we can then use these frequencies to estimate the probabilities. This is usually not feasible. However, the collection of probabilities on the right poses little difficulty. We can easily obtain the estimates of these probabilities by counting their occurrence in the database.

For example, we can count the number of patients with *Age* being young and belonging to class soft (perhaps there are only two) and divide this

by the number of patients overall belonging to class soft (perhaps there are five). The resulting estimate of the probability (i.e.,$P(young|soft)$) is `0.4`.

The Naive Bayes algorithm is then quite simple. From the training data we estimate the probability of each class, $P(c_j)$, by the proportions exhibited in the database. Similarly the probabilities of each feature's value, given a particular class ($P(a_i|c_j)$), is simply the proportion of those training records with that class having the particular feature value.

Now to place a new record (d) into a class we simply assign it to the class with the highest probability. That is, we choose the c_j which maximises:

$$P(c_j)\prod_{a_i \in d} P(a_i \mid c_j)$$

4. Results

4.1. Images database

4.1.1. Protocol

The protocol that we followed was very simple. We installed a photographic stand in the laboratory. The stuff we used were collected from the lab. No objects were bought to perform the project.

Then, some rules about the photos and about the procedures were set. The most important rules come from the light and position problems because of their decisive role in the classification.

- A standard distance from the camera to the object was set.

Fig 36. Camera and screen.

That's the photo stand. The distance from the camera to the person is constant. The camera is placed on the photo stand and the height is controlled with a handle and it depends on the height of the person. The height can be observed in the side of the column.

- A standard position was also set.

Fig 37. Standard position of feet.

People should adjust their feet to these crosses to avoid problems with the position. Logically, the were previous adjusted.

- A standard light.

Fig 38. Standard Light in the lab.

4.1.2. Images

4.1.2.1. Database

Our database is an 11 people database. Each one has at least 3 photos. Some of them are taken in the same session but other are from different sessions.

Fig 39. Example of Images

The photos are 512x512 pixels photos. Light and distance are always like the ones on this example.

4.1.2.2. Previous classification

Not every photo is useful and also, not every face is. A previous classification of the photos has to be done. The program can't itself classify the useful photos from the useless ones. But when it uses them, it fails. So it's very easy to find which photos are the wrong ones.

Fig 40. Example of a useless photo due to the hair, light, size and position.

Next we list some examples of photos the program can't classify. Photos which light aren't correct.

- Photos which size isn't correct.
- Photos which position isn't correct.
- Photos in which parts of face can't be messed.
- Photos which have the features are the same as others have.

If every photo of a class has the same features the program is unable to perform a classifier. It's logic because it can't calculate statistical values from a class that hasn't deviation.

4.1.3. Material

4.1.3.1. Lights

The lights that were used were:

- Standard lights of the lab. See figure x.
- A Spotlight. Primalux 2500. 2500W max.

Fig 41. Back sight of the spotlight .

4.1.3.2. Camera

We used a SONY XC-003/003P CAMERA. The XC-003/003P is a 3-CCD RGB color camera module designed primarily for process control and image processing applications. The new 3 CCD C-mount optical prism block lets to achieve three times higher resolution compared to a 1-CCD system. Moreover acceptance the standard C-mount lens system (bayonet-mount is not required) allows customers to use a wide variety of lenses. In combination with a frame grabber the camera can do a long time integration (on the ccd) for max. 4 seconds. this makes that the camera can be used for fluoresent recording The camera is also equipped with a newly developed E-Donpisha external trigger shutter which assures accurate capturing of random fast moving objects. Such a feature is useful in high speed color inspection systems. Camera XC-003 (XC-003P) works in accordance with EIA (CCIR) video norm, using NTSC (PAL) color coding system.

Features

- High resolution: 768×494 (NTSC), 752×582 (PAL)
- Cosite sampling
- E-Donpisha Asynchronous Reset function
- Long term, low-noise integration max 4 sec.

- RGB, composite and Y/C outputs
- External sync
- Compact design: 50×56×128 mm, 440 g
- Computer controled

Fig 42. SONY XC-003/003P CAMERA.

4.2. Results

4.2.1. Experiment results

4.2.1.1. First experiment

The first part of the experiments was to try to recognize people from the database which photos were already introduced in the classifier. Logically, that is easy and every photo that could be introduced on the classifier was able to be recognized. It was 100 percent successful. Because of that it was impossible to differentiate the classifiers.

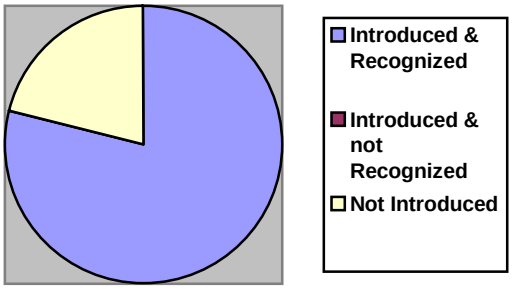


Fig 43. Results of the experiment.

The 'not introduced' photos are the photos that weren't able to be used by the program because of some of the reasons exposed on the previous classification.

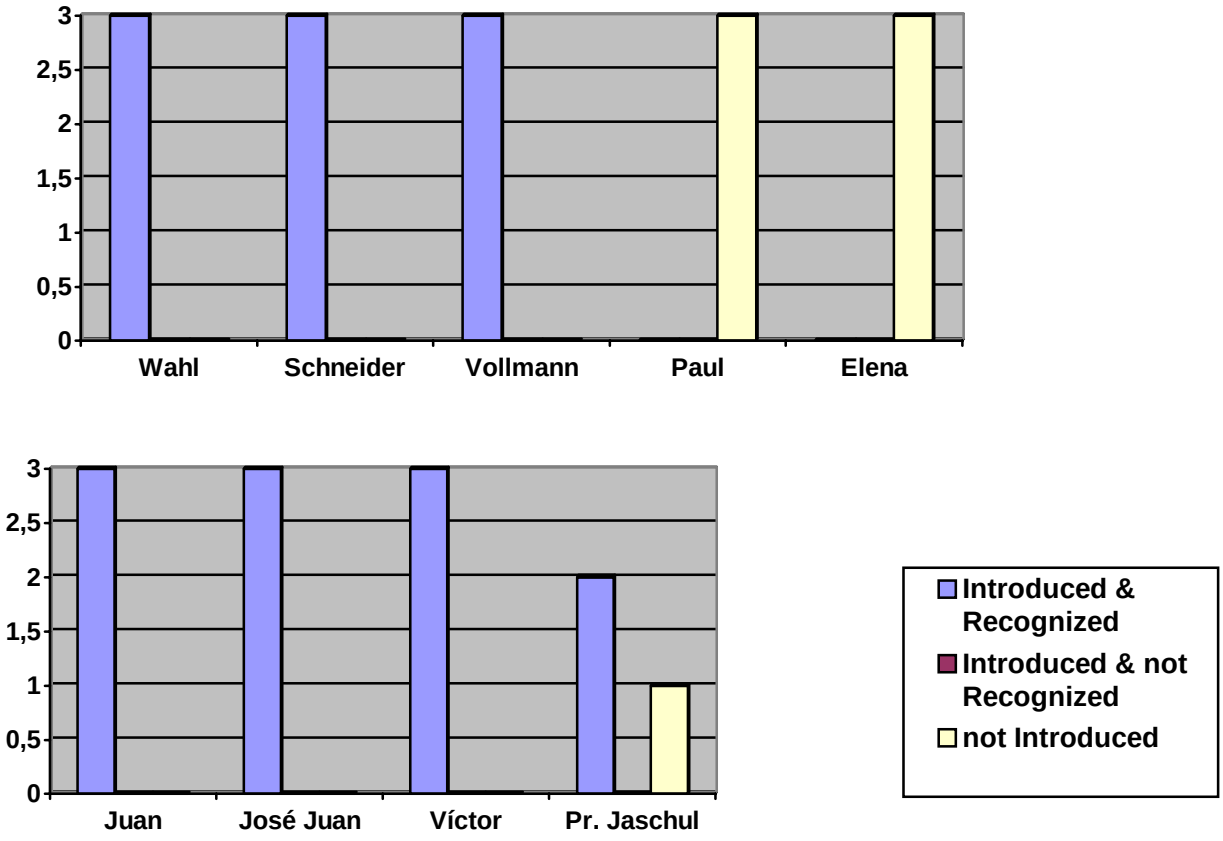


Fig 44. Results of the first experiment depending on person

4.2.1.2 Second experiment

Then the second experiment started. We tried to recognized photos that weren't on the database and the program hasn't work with. That's more difficult at first sight. Here we had a big problem: Our database isn't big enough to perform a good test. Some of the people have a lot of photos and some of them only three. With the last ones we had to train with only two photos and then they were tested with the last one.

On the left we show the results of this experiment:

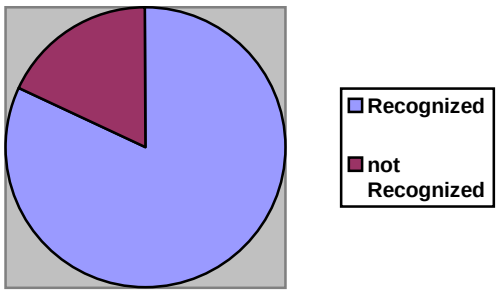


Fig 45. Results of the second experiment.

The photos we used to perform this test are photos that we know the program tolerate. The others that weren't able to be introduced in the classifier in the first experiment were rejected.

The results are good. But it has to be said that the results are from a little number of photos (32). Not enough to perform a good test.

We had also another problem. Because of the normalization the values that are measured are integers and don't change much if the photos are similar. If one feature repeats the value in the photos the program can't perform the classification and fails. That's especially important on the version where we didn't use the grid. So, our test on this version has little number of samples.

The classifiers had the same behaviour. Some factors could be reason:

- The database is little and their behaviour should be similar.
- The test was also little. Perhaps with a bigger test bench the differences could be appreciated.
- The accuracy was very good. If the classifier doesn't fail no differences could be messed.

4.2.2. Tables

Next the results of the process are shown in tables. The main process is always the same. It only changes in the normalization of the values. It's easy to notice only with the tables the differences between the features of the people. These results are only for 26 photos, the typical number of images we used.

No	CLASS	CLASSNAME	FERETY/pixel	CGRAVX/pixel	CGRAVY/pixel
1	1	juan	98	376	238,13
2	1	juan	92	389,09	253,11
3	1	juan	101	372,69	266,07
4	2	josejuan	82	389,15	251,87
5	2	josejuan	79	372,79	236,77
6	2	josejuan	79	395,87	280,23
7	3	thomas	97	370,9	215,15
8	3	thomas	96	362,93	218,87
9	3	thomas	94	364,47	214,48
10	4	victor	90	338,38	229,64
11	4	victor	90	333,75	218,43
12	4	victor	88	330,96	220,39
13	5	pr.jaschul	86	371,55	248,78
14	5	pr.jaschul	84	370,09	249,13
15	6	david	86	368,45	231,76
16	6	david	86	369,62	228,98
17	6	David	85	370,21	234,77
18	7	Schneider	69	360,82	231,98
19	7	Schneider	66	365,29	236,11
20	7	Schneider	65	367,31	237,3
21	8	Vollmann	93	360,77	235,33
22	8	Vollmann	85	362,32	236,91
23	8	Vollmann	91	358,7	240,37

24	9	Wahl	83	345,59	233,5
25	9	Wahl	86	342,08	243,47
26	9	Wahl	87	341,93	231,46

Table 2. Measurement of the mouth.

This table corresponds to the mouth measurement. That is the first step on the face recognition. It is easier to study the tables watching the process in the third chapter.

No	CLASS	CLASSNAME	AREA/pixel^2	CGRAVX/pixel	CGRAVY/pixel	DCIRCLE/píxel	FERETX/pixel	FERETY/pixel
1	1	Juan	2179	264,83	204,39	52,67	44	79
2	1	Juan	1797	260,31	301,78	47,83	42	76
3	1	Juan	2212	268,26	200,81	53,07	43	80
4	1	Juan	1827	263,61	298,14	48,23	42	77
5	1	Juan	2316	250,27	218,99	54,3	41	81
6	1	Juan	2161	249,07	317,59	52,45	39	81
7	2	Josejuan	2017	281,89	208,91	50,68	34	67
8	2	Josejuan	1796	273,57	292,1	47,82	41	61
9	2	Josejuan	1988	253,54	190,28	50,31	37	63
10	2	Josejuan	2219	248,13	280,88	53,15	42	67
11	2	Josejuan	1858	283,08	233,91	48,64	34	64
12	2	Josejuan	1937	278,07	319,94	49,66	39	65
13	3	Thomas	2293	263,39	162,91	54,03	41	75
14	3	Thomas	2644	256,7	254,68	58,02	55	84
15	3	Thomas	2208	258,78	166,74	53,02	38	73
16	3	Thomas	2475	251,29	256,36	56,14	54	80
17	3	Thomas	2236	260,53	160,62	53,36	38	74
18	3	Thomas	2483	253,17	251,01	56,23	54	81
19	4	Victor	2728	222,88	175,4	60,38	48	77
20	4	Victor	1811	217,48	277,94	48,02	41	59
21	4	Victor	2644	221,2	169,38	58,02	45	75
22	4	Victor	1822	218,13	269,93	48,16	38	58
23	4	Victor	2615	218,4	170,65	57,7	45	74
24	4	Victor	1830	215,39	271,09	48,27	38	58
25	5	pr.jaschul	1605	268,07	209,26	45,21	35	69
26	5	pr.jaschul	1548	270,36	292,87	44,4	35	72
27	5	pr.jaschul	1640	268,7	207,99	45,7	31	69
28	5	pr.jaschul	1482	270,43	293,39	43,44	32	67
29	6	david	2506	243,79	185,24	56,49	41	80
30	6	david	2046	244,88	279,5	51,04	39	71
31	6	david	2501	245	183,71	56,43	41	81
32	6	david	2037	246,82	277,53	50,93	39	71
33	6	david	2489	245,95	189,07	56,29	41	81
34	6	david	2020	247,75	282,77	50,71	39	71
35	7	schneider	1965	250,82	215,21	50,02	38	80
36	7	schneider	1793	260,39	305,67	47,78	42	69
37	7	schneider	1978	253,59	218,97	50,18	37	81

38	7	schneider	1811	264,11	309,65	48,02	42	69
39	7	schneider	2016	255,7	218,98	50,66	39	82
40	7	schneider	1855	266,24	308,25	48,6	42	69
41	8	vollmann	2895	246,77	192,88	60,71	45	83
42	8	vollmann	2613	246,63	288,31	57,68	44	77
43	8	vollmann	2846	252,6	197,29	60,2	45	81
44	8	vollmann	2613	252,32	291,95	57,68	43	77
45	8	vollmann	2747	246,83	200,96	59,14	44	80
46	8	vollmann	2526	247,22	294,67	56,71	43	76
47	9	wahl	1855	229,68	211,2	48,6	63	60
48	9	wahl	1721	232,68	289,98	46,81	45	60
49	9	wahl	1845	229,59	217,81	48,47	63	62
50	9	wahl	1652	231,88	296,69	45,86	46	60
51	9	wahl	1815	230,75	206,56	48,07	62	61
52	9	wahl	1645	233,57	285,73	45,77	45	60

Table 3. This one correspond to the eye measurement.

No	CLASS	CLASSNAME	MLength	Eye1Area	Eye2Area	EyeDistance	EMD1	EMD2	EMA1	EMA2	Circle
1	1	juan	98	2179	1797	97,39	116,17	132,04	24,78	21,64	50,25
2	1	juan	92	2212	1827	97,33	131,66	133,31	21,69	21,41	50,65
3	1	juan	101	2316	2161	98,59	131,16	133,93	22,08	21,6	53,38
4	2	josejuan	82	2017	1796	83,19	115,54	122,38	21,1	19,87	49,25
5	2	josejuan	79	1988	2219	90,6	127,99	132,24	20,73	20,03	51,73
6	2	josejuan	79	1858	1937	86,03	121,93	124,31	20,66	20,25	49,15
7	3	thomas	97	2293	2644	91,76	119,53	120,84	22,57	22,31	56,03
8	3	thomas	96	2208	2475	89,62	116,47	117,77	22,63	22,36	54,58
9	3	thomas	94	2236	2483	90,38	117,07	117,14	22,71	22,69	54,79
10	4	victor	90	2728	1811	102,54	127,6	130,19	23,69	23,19	54,2
11	4	victor	90	2644	1822	100,55	122,77	126,57	24,17	23,4	53,09
12	4	victor	88	2615	1830	100,43	123,07	126,2	24,08	23,45	52,99
13	5	Pr.jaschul	86	1605	1548	83,62	110,77	110,37	22,17	22,26	44,8
14	5	Pr.jaschul	84	1640	1482	85,4	109,41	109,05	22,97	23,05	44,57
15	6	david	86	2506	2046	94,26	133,05	132,47	20,75	20,84	53,76
16	6	david	86	2501	2037	93,81	132,59	132,05	20,72	20,81	53,68
17	6	david	85	2489	2020	93,7	132,39	131,53	20,72	20,87	53,5
18	7	schneider	69	1965	1793	90,46	111,27	124,57	23,99	21,29	48,9
19	7	schneider	66	1978	1811	90,68	113	125,08	23,66	21,25	49,1
20	7	schneider	65	2016	1855	89,27	113,1	123,49	23,25	21,19	49,63
21	8	vollmann	93	2895	2613	95,43	121,65	125,84	23,09	22,28	59,2
22	8	vollmann	85	2846	2613	94,65	116,65	123	23,94	22,63	58,94
23	8	vollmann	91	2747	2526	93,71	118,61	124	23,27	22,2	57,93
24	9	wahl	83	1855	1721	78,78	118,03	126,25	19,49	18,18	47,7
25	9	wahl	86	1845	1652	78,87	115,38	122,38	19,99	18,8	47,17
26	9	wahl	87	1815	1645	79,17	113,93	121,19	20,33	19,07	46,92

Table 4. Face features measured.

No	CLASS	CLASSNAME	MLength	EyeArea	EyeDistance	EMD	EMA	CIRCLE
1	1	juan	84	4	73	61	8	40

2	1	juan	70	4	73	90	5	42
3	1	juan	91	5	77	90	5	58
4	2	josejuan	47	3	23	42	3	34
5	2	josejuan	40	5	49	82	3	49
6	2	josejuan	40	3	33	57	3	34
7	3	thomas	82	7	53	47	6	73
8	3	thomas	80	6	46	36	6	65
9	3	thomas	75	6	49	36	7	66
10	4	victor	66	6	91	77	8	63
11	4	victor	66	5	84	63	9	56
12	4	victor	61	5	84	62	9	56
13	5	pr.jaschul	56	1	25	13	6	9
14	5	pr.jaschul	52	1	31	8	7	8
15	6	David	56	6	62	91	4	60
16	6	David	56	6	61	90	4	60
17	6	David	54	6	60	88	4	59
18	7	Schneider	17	3	49	39	7	32
19	7	Schneider	10	3	50	43	6	34
20	7	Schneider	8	3	45	40	6	37
21	8	Vollmann	73	9	66	59	7	91
22	8	Vollmann	54	9	64	45	8	90
23	8	Vollmann	68	8	60	51	7	84
24	9	Wahl	49	3	8	54	0	26
25	9	Wahl	56	2	8	42	1	23
26	9	Wahl	59	2	9	37	2	21

Table 5. Face features after normalization. Grid version.

No	CLASS	CLASSNAME	MLength	EyeArea	EyeDistance	EMD	EMA	CIRCLE
1	1	Juan	216	85	188	156	209	104
2	1	Juan	181	90	187	232	138	109
3	1	Juan	234	127	199	232	150	149
4	2	Josejuan	122	72	60	109	92	89
5	2	Josejuan	104	104	127	210	87	125
6	2	Josejuan	104	70	86	147	90	88
7	3	Thomas	210	165	137	120	176	188
8	3	Thomas	205	144	118	92	179	167
9	3	Thomas	193	147	125	92	187	170
10	4	Victor	169	132	234	199	219	161
11	4	Victor	169	126	216	161	234	145
12	4	Victor	157	124	215	160	233	144
13	5	pr.jaschul	145	17	64	33	167	24
14	5	pr.jaschul	133	15	80	21	201	21
15	6	David	145	133	160	234	105	155
16	6	David	145	132	156	230	104	154
17	6	David	139	129	155	227	105	151
18	7	Schneider	45	67	126	100	185	84
19	7	Schneider	27	70	128	110	177	87
20	7	Schneider	21	76	115	103	167	95

21	8	Vollmann	187	212	170	152	187	234
22	8	Vollmann	139	208	163	117	213	230
23	8	Vollmann	175	192	155	130	189	216
24	9	Wahl	127	52	21	138	21	67
25	9	Wahl	145	46	22	108	45	59
26	9	Wahl	151	43	24	96	58	55

Table 6. Face features after normalization. Normal version.

No	CLASS	CLASSNAME	FERETY/pixel	FERETX/pixel	XMIN/pixel	YMIN/pixel
1	1	juan	84	40	873	1661
2	1	juan	70	42	873	1090
3	1	juan	91	58	1077	1090
4	2	josejuan	47	34	623	642
5	2	josejuan	40	49	1049	682
6	2	josejuan	40	34	633	657
7	3	thomas	82	73	1453	1247
8	3	thomas	80	65	1246	1236
9	3	thomas	75	66	1249	1436
10	4	victor	66	63	1291	1677
11	4	victor	66	56	1084	1863
12	4	victor	61	56	1084	1862
13	5	pr.jaschul	56	9	225	1213
14	5	pr.jaschul	52	8	231	1408
15	6	david	56	60	1262	891
16	6	david	56	60	1261	890
17	6	david	54	59	1260	888
18	7	schneider	17	32	649	1439
19	7	schneider	10	34	650	1243
20	7	schneider	8	37	645	1240
21	8	vollmann	73	91	1866	1459
22	8	vollmann	54	90	1864	1645
23	8	vollmann	68	84	1660	1451
24	9	wahl	49	26	608	54
25	9	wahl	56	23	408	242
26	9	wahl	59	21	409	437

Table 7. Measurement in the virtual images. Grid version.

No	CLASS	CLASSNAME	FERETY/pixel	FERETX/pixel	XMIN/pixel	YMIN/pixel
1	1	Juan	216	104	188	156
2	1	Juan	181	109	187	232
3	1	Juan	234	149	199	232
4	2	josejuan	122	89	60	109
5	2	josejuan	104	125	127	210
6	2	josejuan	104	88	86	147
7	3	thomas	210	188	137	120
8	3	thomas	205	167	118	92
9	3	thomas	193	170	125	92

10	4	victor	169	161	234	199
11	4	victor	169	145	216	161
12	4	victor	157	144	215	160
13	5	pr.jaschul	145	24	64	33
14	5	pr.jaschul	133	21	80	21
15	6	david	145	155	160	234
16	6	david	145	154	156	230
17	6	david	139	151	155	227
18	7	schneider	45	84	126	100
19	7	schneider	27	87	128	110
20	7	schneider	21	95	115	103
21	8	vollmann	187	234	170	152
22	8	vollmann	139	230	163	117
23	8	vollmann	175	216	155	130
24	9	Wahl	127	67	21	138
25	9	Wahl	145	59	22	108
26	9	Wahl	151	55	24	96

Table 8. Measurement in the virtual images. Normal version.

5. Conclusions and future lines.

5.1. KS400 system

Working with this system was a good experience. It's a powerful system of image processing and has a simple interface for its use. Finally, the goals that we had arranged with the KS400 systems were achieved.

- The features from the faces were measured.
- A classification with a normal success was obtained.

The experiments showed that a face recognition system could be implemented in the KS400 system. The lines we had explored could be final system or a final version.

We had also problems with it. It wasn't easy to do this image processing system work as a face recognition system. We had lot of problems that have to be solved about the KS400 system:

- It's too slow, and sometimes its behaviour is very annoying.
- Sometimes it has problems due to unknown circumstances.
- The features treatment isn't the ideal for our problem.
- It can't learn from a normal database.

If the work wants to be continued, these problems have to be revised.

Anyway, this program isn't the way to perform a face recognition system, so we want to perform a very efficient face recognition system we need other systems. A good example of system to perform it is Matlab.

5.2 Database

The database creation was also an interesting process. We obtained a little image database which we used for the classification. It's a little database with 11 people. But at this moment, time of previous works with the classification system, a bigger database wasn't needed. Maybe, more photos from each person could be much better.

It's known that to build up a face database isn't easy. We didn't have the ways to congregate much people and I think that it will be also difficult in the future. So, I suggest a new strategy. This new strategy is to use an open database like *FERET* database in order to have the possibility to probe with a lot of standard images. These 'public' databases are useful because we could compare our results with the other people.

Maybe with a bigger database, the classifiers could be differentiated. Furthermore, we could draw maps of features where each image has its own point in the space and the classification could be directly observed. That could be an interesting way to proceed.

Also, the results of the classifier could be compared with the results from other programs like Matlab.

5.3 Features extraction and processing

First, we achieved to find features. These features were extracted from eyes and mouth. This part is very reliable. Almost every photo is processed and its features correctly measured, also new images taken from the camera. It worked always properly when the images were correctly taken (see chapter 4).

This features extraction is simple but isn't efficient. New images have to be introduced manually and in different parts of the code. So, it could be a good idea to perform a loop where every person independent from the name were processed. We suggest the idea of creating a 'Names' file where every person is included. Naturally, then this file has to be opened and then used in this loop to make easier and more efficient the introduction of new people.

Finding also new features could be interesting, but at this moment it impossible to use more features. The nose and the form of the face could be the next objectives.

The classification process has also to be improved. The work around we used isn't a final solution. Because if more features want to be used this system has to be changed. Therefore a new system of classification has to be performed in the next steps.

6. Bibliography

[1] R. Brunelli and T. Poggio. *Face recognition: Features versus templates*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(10), 1993

[2] Kanade, T. *Computer Recognition of Human Faces*. Birkhuser Verlag, Stuttgart, 1977.

[3] M.Turk and A.Pentland, "*Eigenfaces for Recognition*", Journal of Cognitive Neuroscience, vol. 3, no.1, pp. 71-86, 1991.

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

[5] WISKOTT, L., FELLOUS, J., KRÜGER, N., AND VON DER MALSBNURG, C. 1997. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern analysis and Machine Intelligence* 19: 775-779.

[6] G-D. Guo, S. Li, and K. Chan, Face Recognition by Support Vector Machines, *Image and Vision Computing*, Special Issue on Artificial Neural Networks for Image Analysis and Computer Vision, Vol. 19, Issue 9-10, 631-638, 2001.

[7] S. Z Li, L. Zhu, Z. Q. Zhang, A. Blake, H. J. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *Proc. ECCV*, 2003.

[8] Roberto M. Cesar, Endika Bengoetxea, Isabelle Bloch: Inexact Graph Matching Using Stochastic Optimization Techniques for Facial Feature Recognition. ICPR (2) 2002: 465-468

[9] Abdelkader, C., Cutler, R., and Davis, L., 2002, *Motion-based recognition of people in eigengait space*, Proc. of the 5th Int. Conference on Automatic Face and Gesture Recognition.

[10 H.K. Ekenel, B. Sankur, Multiresolution Face Recognition, Image and Vision Computing, (under review), 2004

[11] "Multilinear Independent Component Analysis", M. A. O. Vasilescu and D. Terzopoulos, *Learning 2004* Snowbird, UT, April, 2004.

[12] KS400 user guide. Carl Zeiss Vision GmbH

7. Appendix

7.1. Code

7.1.1. Classification with calculated features, training part.

```
#  Normalization constants.
MLA=61.4
MLB=43.2
EAA=2952
EAB=3096
EDA=76.404
EDB=28.512
EMDA=213.754
EMDB=56.472
EMAA=36.68
EMAB=11.88
CA=43.107
CB=17.556

Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "david"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION","FERETY>50&&FERETX<20"
# MSsetcond


MSsetprop "REGIONFEAT","CLASS,CLASSNAME,FERETY,CGRAVX,CGRAVY"


# MSsetfeat "MOUTH"
imgsetpath "c:\ks400\conf\images\db"


imgload "juan1.bmp",1
imgload "juan12.bmp",5
imgload "juan14.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11

imgload "mouth1.bmp",20


dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1
```

```
binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13


MSmeasmask 4,2,"testmouth",0,1,1
MSmeasmask 8,6,"testmouth",1,1,1
MSmeasmask 13,11,"testmouth",1,1,1

imgload "jose11.bmp",1
imgload "jose1.bmp",5
imgload "jose3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,2
MSmeasmask 8,6,"testmouth",1,1,2
MSmeasmask 13,11,"testmouth",1,1,2

imgload "thomas1.bmp",1
imgload "thomas2.bmp",5
imgload "thomas3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13


MSmeasmask 4,2,"testmouth",1,1,3
MSmeasmask 8,6,"testmouth",1,1,3
MSmeasmask 13,11,"testmouth",1,1,3
```

```
imgload "victor1.bmp",1
imgload "victor2.bmp",5
imgload "victor3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11


dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13


MSmeasmask 4,2,"TESTMOUTH",1,1,4
MSmeasmask 8,6,"testmouth",1,1,4
MSmeasmask 13,11,"testmouth",1,1,4

imgload "jaschul1.bmp",1
imgload "jaschul2.bmp",5
imgload "jaschul3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12


binand 3,20,4
binand 7,20,8
binand 12,20,13


# MSmeasmask 4,2,"testmouth",1,1,5
MSmeasmask 8,6,"testmouth",1,1,5
MSmeasmask 13,11,"testmouth",1,1,5

imgload "david1.bmp",1
imgload "david2.bmp",5
imgload "david3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1
```

```
binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,6
MSmeasmask 8,6,"testmouth",1,1,6
MSmeasmask 13,11,"testmouth",1,1,6


imgload "schneider1.bmp",1
imgload "schneider2.bmp",5
imgload "schneider3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,7
MSmeasmask 8,6,"testmouth",1,1,7
MSmeasmask 13,11,"testmouth",1,1,7

imgload "vollmann1.bmp",1
imgload "vollmann2.bmp",5
imgload "vollmann3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,8
MSmeasmask 8,6,"testmouth",1,1,8
MSmeasmask 13,11,"testmouth",1,1,8
```

```
imgload "wahl1.bmp",1
imgload "wahl2.bmp",5
imgload "wahl3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,9
MSmeasmask 8,6,"testmouth",1,1,9
MSmeasmask 13,11,"testmouth",1,1,9
#
# imgload "elena1.bmp",1
# imgload "elena2.bmp",5
# imgload "elena3.bmp",10
# imgRGB2grey 1, 2
# imgRGB2grey 5, 6
# imgRGB2grey 10, 11
# dislev 2,3,90,255,1
# dislev 6,7,90,255,1
# dislev 11,12,90,255,1
#
# binnot 3,3
# binnot 7,7
# binnot 12,12
#
#
#
# binand 3,20,4
# binand 7,20,8
# binand 12,20,13
 #
# ! MSmeasmask 4,2,"testmouth",1,1,6
# ! MSmeasmask 8,6,"testmouth",1,1,6
# ! MSmeasmask 13,11,"testmouth",1,1,6
datalist "testmouth",0,0


# MSsetfeat "EYES"
MSsetprop
"REGIONFEAT","CLASS,CLASSNAME,AREA,CGRAVX,CGRAVY,DCIRCLE,FERETX,F
ERETY"
MSsetprop "CONDITION","AREA>50 && FERETX>30 && FERETX<80 &&
CGRAVX<286 && CGRAVX>206 && DCIRCLE > 42"

imgload "eyes.bmp",20

imgload "juan11.bmp",1
imgload "juan12.bmp",5
imgload "juan14.bmp",10
imgRGB2grey 1, 2
```

```
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",0,1,1
MSmeasmask 8,6,"testeyes",1,1,1
MSmeasmask 13,11,"testeyes",1,1,1

imgload "jose11.bmp",1
imgload "jose1.bmp",5
imgload "jose3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,2
MSmeasmask 8,6,"testeyes",1,1,2
MSmeasmask 13,11,"testeyes",1,1,2

imgload "thomas1.bmp",1
imgload "thomas2.bmp",5
imgload "thomas3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2
```

```
MSmeasmask 4,2,"testeyes",1,1,3
MSmeasmask 8,6,"testeyes",1,1,3
MSmeasmask 13,11,"testeyes",1,1,3

imgload "victor1.bmp",1
imgload "victor2.bmp",5
imgload "victor3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,4
MSmeasmask 8,6,"testeyes",1,1,4
MSmeasmask 13,11,"testeyes",1,1,4

imgload "jaschul1.bmp",1
imgload "jaschul2.bmp",5
imgload "jaschul3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2
# MSmeasmask 4,2,"testeyes",1,1,5
MSmeasmask 8,6,"testeyes",1,1,5
MSmeasmask 13,11,"testeyes",1,1,5

imgload "david1.bmp",1
imgload "david2.bmp",5
imgload "david3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
```

```
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,6
MSmeasmask 8,6,"testeyes",1,1,6
MSmeasmask 13,11,"testeyes",1,1,6

imgload "schneider1.bmp",1
imgload "schneider2.bmp",5
imgload "schneider3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,7
MSmeasmask 8,6,"testeyes",1,1,7
MSmeasmask 13,11,"testeyes",1,1,7


imgload "vollmann1.bmp",1
imgload "vollmann2.bmp",5
imgload "vollmann3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,8
MSmeasmask 8,6,"testeyes",1,1,8
MSmeasmask 13,11,"testeyes",1,1,8

imgload "wahl1.bmp",1
```

```
imgload "wahl2.bmp",5
imgload "wahl3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,9
MSmeasmask 8,6,"testeyes",1,1,9
MSmeasmask 13,11,"testeyes",1,1,9

# imgload "elena1.bmp",1
# imgload "elena2.bmp",5
# imgload "elena3.bmp",10
# imgRGB2grey 1, 2
# imgRGB2grey 5, 6
# imgRGB2grey 10, 11
# dislev 2,3,95,255,1
# dislev 6,7,95,255,1
# dislev 11,12,95,255,1
#
# binnot 3,3
# binnot 7,7
# binnot 12,12
# binand 3,20,4
# binand 7,20,8
# binand 12,20,13
# dmarr 13,13,9.50,2
# dmarr 4,4,9.50,2
# dmarr 8,8,9.50,2
 #
# ! MSmeasmask 4,2,"testeyes",1,1,9
# ! MSmeasmask 8,6,"testeyes",1,1,9
# ! MSmeasmask 13,11,"testeyes",1,1,9


datalist "TESTEYES",0,0



# Generation of new features
DBfirstline "testeyes"
DBfirstline "testmouth"

DBnew "face",11
DBsetcolumn "face",1,"CLASS","Int","<none>"
DBsetcolumn "face",2,"CLASSNAME","String","<none>"
DBsetcolumn "face",3,"MLength","Float","<none>"
DBsetcolumn "face",4,"Eye1Area","Float","<none>"
DBsetcolumn "face",5,"Eye2Area","Float","<none>"
DBsetcolumn "face",6,"EyeDistance","Float","<none>"
```

```
DBsetcolumn "face",7,"EMD1","Float","<none>"
DBsetcolumn "face",8,"EMD2","Float","<none>"
DBsetcolumn "face",9,"EMA1","Float","<none>"
DBsetcolumn "face",10,"EMA2","Float","<none>"
DBsetcolumn "face",11,"Circle","Float","<none>"



mouthlength=0          # Mouth Length
eyedistance=0.0        # Eye distance
classmouth=0           # Classes
classeyes1=0
classeyes2=0
dcircle1= 0.0          #Circularity
dcircle2= 0.0
areaeyes1=0            # Area eyes1
areaeyes2=0            # Area eyes2
xeyes1=0               # Eye Position 1
yeyes1=0
xeyes2=0               # Eye Position 2
yeyes2=0
xmouth=0               # Mouth Position
ymouth=0
emdi=0.0               # Eye mouth distance
emdd=0.0

while 1
if _STATUS == 0: break
DBgetvalue "testmouth","CLASS",classmouth
DBgetvalue "testmouth","CLASSNAME",classmouthn
DBgetvalue "testmouth","FERETY",mouthlength
DBgetvalue "testmouth","CGRAVX",xmouth
DBgetvalue "testmouth","CGRAVY",ymouth
DBgetvalue "testeyes","CLASS",classeyes1
DBgetvalue "testeyes","CGRAVX",xeyes1
DBgetvalue "testeyes","CGRAVY",yeyes1
DBgetvalue "testeyes","AREA",areaeyes1
DBgetvalue "testeyes","DCIRCLE",dcircle1

DBnextline "testeyes"
DBgetvalue "testeyes","CLASS",classeyes2
DBgetvalue "testeyes","CGRAVX",xeyes2
DBgetvalue "testeyes","CGRAVY",yeyes2
DBgetvalue "testeyes","AREA",areaeyes2
DBgetvalue "testeyes","DCIRCLE",dcircle2

eyedistance= abs( yeyes1 - yeyes2)
emdi = sqrt((yeyes1 -ymouth)*(yeyes1-ymouth)+(xeyes1-
xmouth)*(xeyes1-xmouth))
emdd = sqrt( (yeyes2 - ymouth)*(yeyes2 - ymouth)+(xeyes2 -
xmouth)*(xeyes2 - xmouth))
ema5 = asin ( (eyedistance)/(2*emdi))
ema6 = asin ( (eyedistance)/(2*emdd))
ema5 = (ema5 * 180)/3.141593
ema6 = (ema6 * 180)/3.141592
dcircle1 = (dcircle1 + dcircle2)/2

DBaddline "face"
DBsetvalue "face", "CLASS",classmouth
DBsetvalue "face", "CLASSNAME",classmouthn
DBsetvalue "face", "MLength",mouthlength
DBsetvalue "face", "Eye1Area",areaeyes1
DBsetvalue "face", "Eye2Area",areaeyes2
```

```
DBsetvalue "face", "EyeDistance", eyedistance
DBsetvalue "face", "EMD1", emdi
DBsetvalue "face", "EMD2",emdd
DBsetvalue "face","EMA1",ema5
DBsetvalue "face","EMA2",ema6
DBsetvalue "face","Circle",dcircle1

DBnextline "testmouth"
DBnextline "testeyes"
endwhile

datalist "face",0,1

#  Norlmalisation of the DB "face"
DBfirstline "face"

DBnew "facenorm",8
DBsetcolumn "facenorm",1,"CLASS","Int","<none>"
DBsetcolumn "facenorm",2,"CLASSNAME","String","<none>"
DBsetcolumn "facenorm",3,"MLength","Int","<none>"
DBsetcolumn "facenorm",4,"EyeArea","Int","<none>"
DBsetcolumn "facenorm",5,"EyeDistance","Int","<none>"
DBsetcolumn "facenorm",6,"EMD","Int","<none>"
DBsetcolumn "facenorm",7,"EMA","Int","<none>"
DBsetcolumn "facenorm",8,"CIRCLE","Int","<none>"


dcircle3= 0.0
mouthlength=0.0            # Mouth Length
eyedistance1=0.0           # Eye distance
classmouth=0
areaeyes1=0.0              # Area1
areaeyes2=0.0              # Area2
emdi=0.0
emdd=0.0

while 1
if _STATUS == 0: break
DBgetvalue "face", "CLASS",classmouth
DBgetvalue "face", "CLASSNAME",classmouthn
DBgetvalue "face", "MLength",mouthlength
DBgetvalue "face", "Eye1Area",areaeyes1
DBgetvalue "face", "Eye2Area",areaeyes2
DBgetvalue "face", "EyeDistance", eyedistance1
DBgetvalue "face", "EMD1", emdi
DBgetvalue "face", "EMD2",emdd
DBgetvalue "face","EMA1",ema5
DBgetvalue "face","EMA2",ema6
DBgetvalue "face","Circle",dcircle3


mouthlength = ((mouthlength - MLA)*256/ MLB)
areaeyes1 = (((areaeyes1 + areaeyes2 - EAA)/EAB)*256)
eyedistance1= (256*(eyedistance1-EDA)/EDB)
emdi =  (((emdi + emdd-EMDA)/EMDB)*256)
ema5 =  (((ema5 + ema6 - EMAA)/EMAB)*256)
dcircle3 = ((dcircle3 - CA)*256  / CB)

integCIR= 0        # Integer values
integAE=1
integML=0
integED=0
integEMD=0
```

```
integEMA=0

while mouthlength > 1
integML = integML + 1
mouthlength = mouthlength - 1
endwhile

while dcircle3 > 1
integCIR = integCIR + 1
dcircle3 = dcircle3 - 1
endwhile

while areaeyes1 > 1
integAE = integAE + 1
areaeyes1 = areaeyes1 - 1
endwhile


while eyedistance1 > 1
integED = integED + 1
eyedistance1 = eyedistance1 - 1
endwhile


while emdi > 1
integEMD = integEMD + 1
emdi = emdi - 1
endwhile


while ema5 > 1
integEMA = integEMA + 1
ema5 = ema5 - 1
endwhile

DBaddline "facenorm"
DBsetvalue "facenorm", "CLASS",classmouth
DBsetvalue "facenorm", "CLASSNAME",classmouthn
DBsetvalue "facenorm", "MLength",integML
DBsetvalue "facenorm", "EyeArea",integAE
DBsetvalue "facenorm", "EyeDistance", integED
DBsetvalue "facenorm", "EMD", integEMD
DBsetvalue "facenorm","EMA", integEMA
DBsetvalue "facenorm","CIRCLE", integCIR

DBnextline "face"
endwhile

! datalist "facenorm",0,1



#  Creation of virtual images
Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
```

```
MSsetprop "CLASSNAME6", "david"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION",1
# MSsetcond


MSsetprop "REGIONFEAT","CLASS,CLASSNAME,FERETY,FERETX,XMIN,YMIN"
DBfirstline "facenorm"

mouthlength=0            # Mouth Length
eyedistance2=0           # Eye distance
classmouth=0
dcircle4 =0
areaeyes1=0              # Area1
emdi2=0
ema7=0
color=0
i=0
while 1
if _STATUS == 0: break
DBgetvalue "facenorm", "CLASS",classmouth
DBgetvalue "facenorm", "CLASSNAME",classmouthn
DBgetvalue "facenorm", "MLength",mouthlength
DBgetvalue "facenorm", "EyeArea",areaeyes1
DBgetvalue "facenorm", "EyeDistance", eyedistance2
DBgetvalue "facenorm", "EMD", emdi2
DBgetvalue "facenorm","EMA",ema7
DBgetvalue "facenorm","CIRCLE",dcircle4

Gclear
imgload "white.bmp",1
seldisplay "Display"
imgdisplay 1
Grectg eyedistance2, emdi2 , dcircle4 , mouthlength ,7,7
Gmerge
imgdisplay 1
dislev 1,2,3,252,1
imgRGB2grey 1, 1

if i==0
MSmeasmask 1,2,"virtualtest",0,1,classmouth
endif
if i==1
MSmeasmask 1,2,"virtualtest",1,1,classmouth
endif
i=1

DBnextline "facenorm"
endwhile

datalist "virtualtest"



#    teaching of the classifier
MSclassteach "VIRTUALTEST","BAYES","BAYES-ERGEB",1
! datalist BAYES-ERGEB,0,0
```

7.1.2. Classification with calculated features, test part.

```
# test of the classifier



Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "david"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION","FERETY>50&&FERETX<20"
# MSsetcond
n=0

MSsetprop "REGIONFEAT","CLASS,CLASSNAME,FERETY,CGRAVX,CGRAVY"


# MSsetfeat "MOUTH"
imgsetpath "c:\ks400\conf\images\db"

# Image from camera
# tvsetup
# wait 10000
# tvframeinput 1,0,0,512,512
# imgsave 1,"testing.bmp"

# imgload "testing.bmp",1

# Image from database
imgload "david1.bmp",1
imgRGB2grey 1, 2

imgload "mouth1.bmp",20

dislev 2,3,90,255,1
binnot 3,3
binand 3,20,4
MSmeasmask 4,2,"TESTMOUTH",0,1,0
datalist "testmouth",0,0


# MSsetfeat "EYES"
MSsetprop
"REGIONFEAT","CLASS,CLASSNAME,AREA,CGRAVX,CGRAVY,DCIRCLE,FERETX,F
ERETY"
MSsetprop "CONDITION","AREA>50 && FERETX>30 && FERETX<80 &&
CGRAVX<286 && CGRAVX>206 && DCIRCLE > 42"

imgload "eyes.bmp",20

imgRGB2grey 1, 2
dislev 2,3,95,255,1
binnot 3,3
```

```
binand 3,20,4
dmarr 4,4,9.50,2

MSmeasmask 4,2,"TESTEYES",0,1,0
datalist "testeyes",0,0

DBfirstline "testeyes"
DBfirstline "testmouth"

DBnew "face",11
DBsetcolumn "face",1,"CLASS","Int","<none>"
DBsetcolumn "face",2,"CLASSNAME","String","<none>"
DBsetcolumn "face",3,"MLength","Float","<none>"
DBsetcolumn "face",4,"Eye1Area","Float","<none>"
DBsetcolumn "face",5,"Eye2Area","Float","<none>"
DBsetcolumn "face",6,"EyeDistance","Float","<none>"
DBsetcolumn "face",7,"EMD1","Float","<none>"
DBsetcolumn "face",8,"EMD2","Float","<none>"
DBsetcolumn "face",9,"EMA1","Float","<none>"
DBsetcolumn "face",10,"EMA2","Float","<none>"
DBsetcolumn "face",11,"Circle","Float","<none>"



mouthlength=0                  # Mouth Length
eyedistance=0.0               # Eye distance
classmouth=0
classeyes1=0
classeyes2=0
dcircle1= 0.0
dcircle2= 0.0
areaeyes1=0                 # Area1
areaeyes2=0                 # Area2
xeyes1=0                 # Eye Position 1
yeyes1=0
xeyes2=0                 # Eye Position 2
yeyes2=0
xmouth=0                 # Mouth Position
ymouth=0
emdi=0.0
emdd=0.0

while 1
if _STATUS == 0: break
DBgetvalue "testmouth","CLASS",classmouth
DBgetvalue "testmouth","CLASSNAME",classmouthn
DBgetvalue "testmouth","FERETY",mouthlength
DBgetvalue "testmouth","CGRAVX",xmouth
DBgetvalue "testmouth","CGRAVY",ymouth
DBgetvalue "testeyes","CLASS",classeyes1
DBgetvalue "testeyes","CGRAVX",xeyes1
DBgetvalue "testeyes","CGRAVY",yeyes1
DBgetvalue "testeyes","AREA",areaeyes1
DBgetvalue "testeyes","DCIRCLE",dcircle1

DBnextline "testeyes"
DBgetvalue "testeyes","CLASS",classeyes2
DBgetvalue "testeyes","CGRAVX",xeyes2
DBgetvalue "testeyes","CGRAVY",yeyes2
DBgetvalue "testeyes","AREA",areaeyes2
DBgetvalue "testeyes","DCIRCLE",dcircle2

eyedistance= abs( yeyes1 - yeyes2)
```

```
emdi = sqrt((yeyes1 -ymouth)*(yeyes1-ymouth)+(xeyes1-
xmouth)*(xeyes1-xmouth))
emdd = sqrt( (yeyes2 - ymouth)*(yeyes2 - ymouth)+(xeyes2 -
xmouth)*(xeyes2 - xmouth))
ema5 = asin ( (eyedistance)/(2*emdi))
ema6 = asin ( (eyedistance)/(2*emdd))
ema5 = (ema5 * 180)/3.141593
ema6 = (ema6 * 180)/3.141592
dcircle1 = (dcircle1 + dcircle2)/2

DBaddline "face"
DBsetvalue "face", "CLASS",classmouth
DBsetvalue "face", "CLASSNAME",classmouthn
DBsetvalue "face", "MLength",mouthlength
DBsetvalue "face", "Eye1Area",areaeyes1
DBsetvalue "face", "Eye2Area",areaeyes2
DBsetvalue "face", "EyeDistance", eyedistance
DBsetvalue "face", "EMD1", emdi
DBsetvalue "face", "EMD2",emdd
DBsetvalue "face","EMA1",ema5
DBsetvalue "face","EMA2",ema6
DBsetvalue "face","Circle",dcircle1

DBnextline "testmouth"
DBnextline "testeyes"
endwhile

datalist "face",0,1

#  Norlmalisation of the DB "face"
DBfirstline "face"

DBnew "facenorm",8
DBsetcolumn "facenorm",1,"CLASS","Int","<none>"
DBsetcolumn "facenorm",2,"CLASSNAME","String","<none>"
DBsetcolumn "facenorm",3,"MLength","Int","<none>"
DBsetcolumn "facenorm",4,"EyeArea","Int","<none>"
DBsetcolumn "facenorm",5,"EyeDistance","Int","<none>"
DBsetcolumn "facenorm",6,"EMD","Int","<none>"
DBsetcolumn "facenorm",7,"EMA","Int","<none>"
DBsetcolumn "facenorm",8,"CIRCLE","Int","<none>"


dcircle3= 0.0
mouthlength=0.0                  # Mouth Length
eyedistance1=0.0               # Eye distance
classmouth=0
areaeyes1=0.0              # Area1
areaeyes2=0.0              # Area2
emdi=0.0
emdd=0.0

while 1
if _STATUS == 0: break
DBgetvalue "face", "CLASS",classmouth
DBgetvalue "face", "CLASSNAME",classmouthn
DBgetvalue "face", "MLength",mouthlength
DBgetvalue "face", "Eye1Area",areaeyes1
DBgetvalue "face", "Eye2Area",areaeyes2
DBgetvalue "face", "EyeDistance", eyedistance1
DBgetvalue "face", "EMD1", emdi
DBgetvalue "face", "EMD2",emdd
DBgetvalue "face","EMA1",ema5
```

```
DBgetvalue "face","EMA2",ema6
DBgetvalue "face","Circle",dcircle3

mouthlength = ((mouthlength - MLA)*256  / MLB)
areaeyes1 = (((areaeyes1 + areaeyes2 - EAA)/EAB)*256)
eyedistance1= (256*(eyedistance1-EDA)/EDB)
emdi =  (((emdi + emdd-EMDA)/EMDB)*256)
ema5 =  (((ema5 + ema6 - EMAA)/EMAB)*256)
dcircle3 = ((dcircle3 - CA)*256  / CB)
integCIR= 0
integAE=1
integML=0
integED=0
integEMD=0
integEMA=0

while mouthlength > 1
integML = integML + 1
mouthlength = mouthlength - 1
endwhile

while dcircle3 > 1
integCIR = integCIR + 1
dcircle3 = dcircle3 - 1
endwhile

while areaeyes1 > 1
integAE = integAE + 1
areaeyes1 = areaeyes1 - 1
endwhile


while eyedistance1 > 1
integED = integED + 1
eyedistance1 = eyedistance1 - 1
endwhile


while emdi > 1
integEMD = integEMD + 1
emdi = emdi - 1
endwhile


while ema5 > 1
integEMA = integEMA + 1
ema5 = ema5 - 1
endwhile

DBaddline "facenorm"
DBsetvalue "facenorm", "CLASS",classmouth
DBsetvalue "facenorm", "CLASSNAME",classmouthn
DBsetvalue "facenorm", "MLength",integML
DBsetvalue "facenorm", "EyeArea",integAE
DBsetvalue "facenorm", "EyeDistance", integED
DBsetvalue "facenorm", "EMD", integEMD
DBsetvalue "facenorm","EMA", integEMA
DBsetvalue "facenorm","CIRCLE", integCIR

DBnextline "face"
endwhile

datalist "facenorm",0,1
```

```
#  Creation of virtual images
Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "david"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION",1
# MSsetcond


MSsetprop "REGIONFEAT","CLASS,CLASSNAME,FERETY,FERETX,XMIN,YMIN"
DBfirstline "facenorm"

mouthlength=0                  # Mouth Length
eyedistance2=0               # Eye distance
classmouth=0
dcircle4 =0
areaeyes1=0              # Area1
emdi2=0
ema7=0
color=0
i=0
while 1
if _STATUS == 0: break
DBgetvalue "facenorm", "CLASS",classmouth
DBgetvalue "facenorm", "CLASSNAME",classmouthn
DBgetvalue "facenorm", "MLength",mouthlength
DBgetvalue "facenorm", "EyeArea",areaeyes1
DBgetvalue "facenorm", "EyeDistance", eyedistance2
DBgetvalue "facenorm", "EMD", emdi2
DBgetvalue "facenorm","EMA",ema7
DBgetvalue "facenorm","CIRCLE",dcircle4

Gclear
imgload "white.bmp",1
seldisplay "Display"
imgdisplay 1

Grectg eyedistance2, emdi2 , dcircle4 , mouthlength ,7,7
Gmerge
imgdisplay 1
dislev 1,2,3,252,1
imgRGB2grey 1, 1

MSsetprop "CLASSIFIER","BAYES"
MSmeasmask 1,2,"BAYES-~1",0,0,0
! datalist "BAYES-~1",0,0
```

```
DBfirstline "BAYES-~1"
DBgetvalue "BAYES-~1", "CLASS",classmouth
DBgetvalue "BAYES-~1", "CLASSNAME",classmouthn

if classmouth>0
n=12
endif

if n==12
MBok "The person is "+classmouthn+"."
endif

if n==0
MBok "No person identified"
```

7.1.3. Classification with calculated features and virtual grid, training part.

```
#  Normalization constants.
MLA=61.4
MLB=43.2
EAA=2952
EAB=3096
EDA=76.404
EDB=28.512
EMDA=213.754
EMDB=56.472
EMAA=36.68
EMAB=11.88
CA=43.107
CB=17.556

Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "david"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION","FERETY>50&&FERETX<20"
# MSsetcond


MSsetprop "REGIONFEAT","CLASS,CLASSNAME,FERETY,CGRAVX,CGRAVY"


# MSsetfeat "MOUTH"
imgsetpath "c:\ks400\conf\images\db"


imgload "juan1.bmp",1
imgload "juan12.bmp",5
imgload "juan14.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11

imgload "mouth1.bmp",20

dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12
```

```
binand 3,20,4
binand 7,20,8
binand 12,20,13


MSmeasmask 4,2,"testmouth",0,1,1
MSmeasmask 8,6,"testmouth",1,1,1
MSmeasmask 13,11,"testmouth",1,1,1

imgload "jose11.bmp",1
imgload "jose1.bmp",5
imgload "jose3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,2
MSmeasmask 8,6,"testmouth",1,1,2
MSmeasmask 13,11,"testmouth",1,1,2

imgload "thomas1.bmp",1
imgload "thomas2.bmp",5
imgload "thomas3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13


MSmeasmask 4,2,"testmouth",1,1,3
MSmeasmask 8,6,"testmouth",1,1,3
MSmeasmask 13,11,"testmouth",1,1,3


imgload "victor1.bmp",1
imgload "victor2.bmp",5
imgload "victor3.bmp",10
```

```
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11

imgload "mouth1.bmp",20


dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13


MSmeasmask 4,2,"TESTMOUTH",1,1,4
MSmeasmask 8,6,"testmouth",1,1,4
MSmeasmask 13,11,"testmouth",1,1,4
#
imgload "jaschul1.bmp",1
imgload "jaschul2.bmp",5
imgload "jaschul3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13


# MSmeasmask 4,2,"testmouth",1,1,5
MSmeasmask 8,6,"testmouth",1,1,5
MSmeasmask 13,11,"testmouth",1,1,5

imgload "david1.bmp",1
imgload "david2.bmp",5
imgload "david3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
```

```
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,6
MSmeasmask 8,6,"testmouth",1,1,6
MSmeasmask 13,11,"testmouth",1,1,6


imgload "schneider1.bmp",1
imgload "schneider2.bmp",5
imgload "schneider3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,7
MSmeasmask 8,6,"testmouth",1,1,7
MSmeasmask 13,11,"testmouth",1,1,7

imgload "vollmann1.bmp",1
imgload "vollmann2.bmp",5
imgload "vollmann3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,8
MSmeasmask 8,6,"testmouth",1,1,8
MSmeasmask 13,11,"testmouth",1,1,8

imgload "wahl1.bmp",1
imgload "wahl2.bmp",5
```

```
imgload "wahl3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,90,255,1
dislev 6,7,90,255,1
dislev 11,12,90,255,1

binnot 3,3
binnot 7,7
binnot 12,12



binand 3,20,4
binand 7,20,8
binand 12,20,13

MSmeasmask 4,2,"testmouth",1,1,9
MSmeasmask 8,6,"testmouth",1,1,9
MSmeasmask 13,11,"testmouth",1,1,9
#
# imgload "elena1.bmp",1
# imgload "elena2.bmp",5
# imgload "elena3.bmp",10
# imgRGB2grey 1, 2
# imgRGB2grey 5, 6
# imgRGB2grey 10, 11
# dislev 2,3,90,255,1
# dislev 6,7,90,255,1
# dislev 11,12,90,255,1
#
# binnot 3,3
# binnot 7,7
# binnot 12,12
#
#
#
# binand 3,20,4
# binand 7,20,8
# binand 12,20,13

# ! MSmeasmask 4,2,"testmouth",1,1,6
# ! MSmeasmask 8,6,"testmouth",1,1,6
# ! MSmeasmask 13,11,"testmouth",1,1,6
datalist "testmouth",0,0


# MSsetfeat "EYES"
MSsetprop
"REGIONFEAT","CLASS,CLASSNAME,AREA,CGRAVX,CGRAVY,DCIRCLE,FERETX,F
ERETY"
MSsetprop "CONDITION","AREA>50 && FERETX>30 && FERETX<80 &&
CGRAVX<286 && CGRAVX>206 && DCIRCLE > 42"

imgload "eyes.bmp",20

imgload "juan11.bmp",1
imgload "juan12.bmp",5
imgload "juan14.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
```

```
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",0,1,1
MSmeasmask 8,6,"testeyes",1,1,1
MSmeasmask 13,11,"testeyes",1,1,1

imgload "jose11.bmp",1
imgload "jose1.bmp",5
imgload "jose3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,2
MSmeasmask 8,6,"testeyes",1,1,2
MSmeasmask 13,11,"testeyes",1,1,2

imgload "thomas1.bmp",1
imgload "thomas2.bmp",5
imgload "thomas3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,3
```

```
MSmeasmask 8,6,"testeyes",1,1,3
MSmeasmask 13,11,"testeyes",1,1,3

imgload "victor1.bmp",1
imgload "victor2.bmp",5
imgload "victor3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,4
MSmeasmask 8,6,"testeyes",1,1,4
MSmeasmask 13,11,"testeyes",1,1,4

imgload "jaschul1.bmp",1
imgload "jaschul2.bmp",5
imgload "jaschul3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2
# MSmeasmask 4,2,"testeyes",1,1,5
MSmeasmask 8,6,"testeyes",1,1,5
MSmeasmask 13,11,"testeyes",1,1,5

imgload "david1.bmp",1
imgload "david2.bmp",5
imgload "david3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
```

```
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,6
MSmeasmask 8,6,"testeyes",1,1,6
MSmeasmask 13,11,"testeyes",1,1,6

imgload "schneider1.bmp",1
imgload "schneider2.bmp",5
imgload "schneider3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,7
MSmeasmask 8,6,"testeyes",1,1,7
MSmeasmask 13,11,"testeyes",1,1,7


imgload "vollmann1.bmp",1
imgload "vollmann2.bmp",5
imgload "vollmann3.bmp",10
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,8
MSmeasmask 8,6,"testeyes",1,1,8
MSmeasmask 13,11,"testeyes",1,1,8

imgload "wahl1.bmp",1
imgload "wahl2.bmp",5
imgload "wahl3.bmp",10
```

```
imgRGB2grey 1, 2
imgRGB2grey 5, 6
imgRGB2grey 10, 11
dislev 2,3,95,255,1
dislev 6,7,95,255,1
dislev 11,12,95,255,1

binnot 3,3
binnot 7,7
binnot 12,12
binand 3,20,4
binand 7,20,8
binand 12,20,13
dmarr 13,13,9.50,2
dmarr 4,4,9.50,2
dmarr 8,8,9.50,2

MSmeasmask 4,2,"testeyes",1,1,9
MSmeasmask 8,6,"testeyes",1,1,9
MSmeasmask 13,11,"testeyes",1,1,9

# imgload "elena1.bmp",1
# imgload "elena2.bmp",5
# imgload "elena3.bmp",10
# imgRGB2grey 1, 2
# imgRGB2grey 5, 6
# imgRGB2grey 10, 11
# dislev 2,3,95,255,1
# dislev 6,7,95,255,1
# dislev 11,12,95,255,1
#
# binnot 3,3
# binnot 7,7
# binnot 12,12
# binand 3,20,4
# binand 7,20,8
# binand 12,20,13
# dmarr 13,13,9.50,2
# dmarr 4,4,9.50,2
# dmarr 8,8,9.50,2
 #
# ! MSmeasmask 4,2,"testeyes",1,1,9
# ! MSmeasmask 8,6,"testeyes",1,1,9
# ! MSmeasmask 13,11,"testeyes",1,1,9


datalist "TESTEYES",0,0



# Generation of new features
DBfirstline "testeyes"
DBfirstline "testmouth"

DBnew "face",11
DBsetcolumn "face",1,"CLASS","Int","<none>"
DBsetcolumn "face",2,"CLASSNAME","String","<none>"
DBsetcolumn "face",3,"MLength","Float","<none>"
DBsetcolumn "face",4,"Eye1Area","Float","<none>"
DBsetcolumn "face",5,"Eye2Area","Float","<none>"
DBsetcolumn "face",6,"EyeDistance","Float","<none>"
DBsetcolumn "face",7,"EMD1","Float","<none>"
DBsetcolumn "face",8,"EMD2","Float","<none>"
```

```
DBsetcolumn "face",9,"EMA1","Float","<none>"
DBsetcolumn "face",10,"EMA2","Float","<none>"
DBsetcolumn "face",11,"Circle","Float","<none>"



mouthlength=0          # Mouth Length
eyedistance=0.0        # Eye distance
classmouth=0           # Classes
classeyes1=0
classeyes2=0
dcircle1= 0.0          #Circularity
dcircle2= 0.0
areaeyes1=0            # Area eyes1
areaeyes2=0            # Area eyes2
xeyes1=0               # Eye Position 1
yeyes1=0
xeyes2=0               # Eye Position 2
yeyes2=0
xmouth=0               # Mouth Position
ymouth=0
emdi=0.0               # Eye mouth distance
emdd=0.0

while 1
if _STATUS == 0: break
DBgetvalue "testmouth","CLASS",classmouth
DBgetvalue "testmouth","CLASSNAME",classmouthn
DBgetvalue "testmouth","FERETY",mouthlength
DBgetvalue "testmouth","CGRAVX",xmouth
DBgetvalue "testmouth","CGRAVY",ymouth
DBgetvalue "testeyes","CLASS",classeyes1
DBgetvalue "testeyes","CGRAVX",xeyes1
DBgetvalue "testeyes","CGRAVY",yeyes1
DBgetvalue "testeyes","AREA",areaeyes1
DBgetvalue "testeyes","DCIRCLE",dcircle1

DBnextline "testeyes"
DBgetvalue "testeyes","CLASS",classeyes2
DBgetvalue "testeyes","CGRAVX",xeyes2
DBgetvalue "testeyes","CGRAVY",yeyes2
DBgetvalue "testeyes","AREA",areaeyes2
DBgetvalue "testeyes","DCIRCLE",dcircle2

eyedistance= abs( yeyes1 - yeyes2)
emdi = sqrt((yeyes1 -ymouth)*(yeyes1-ymouth)+(xeyes1-
xmouth)*(xeyes1-xmouth))
emdd = sqrt( (yeyes2 - ymouth)*(yeyes2 - ymouth)+(xeyes2 -
xmouth)*(xeyes2 - xmouth))
ema5 = asin ( (eyedistance)/(2*emdi))
ema6 = asin ( (eyedistance)/(2*emdd))
ema5 = (ema5 * 180)/3.141593
ema6 = (ema6 * 180)/3.141592
dcircle1 = (dcircle1 + dcircle2)/2

DBaddline "face"
DBsetvalue "face", "CLASS",classmouth
DBsetvalue "face", "CLASSNAME",classmouthn
DBsetvalue "face", "MLength",mouthlength
DBsetvalue "face", "Eye1Area",areaeyes1
DBsetvalue "face", "Eye2Area",areaeyes2
DBsetvalue "face", "EyeDistance", eyedistance
DBsetvalue "face", "EMD1", emdi
```

```
DBsetvalue "face", "EMD2",emdd
DBsetvalue "face","EMA1",ema5
DBsetvalue "face","EMA2",ema6
DBsetvalue "face","Circle",dcircle1

DBnextline "testmouth"
DBnextline "testeyes"
endwhile

datalist "face",0,1

#  Norlmalisation of the DB "face"
DBfirstline "face"

DBnew "facenorm",8
DBsetcolumn "facenorm",1,"CLASS","Int","<none>"
DBsetcolumn "facenorm",2,"CLASSNAME","String","<none>"
DBsetcolumn "facenorm",3,"MLength","Int","<none>"
DBsetcolumn "facenorm",4,"EyeArea","Int","<none>"
DBsetcolumn "facenorm",5,"EyeDistance","Int","<none>"
DBsetcolumn "facenorm",6,"EMD","Int","<none>"
DBsetcolumn "facenorm",7,"EMA","Int","<none>"
DBsetcolumn "facenorm",8,"CIRCLE","Int","<none>"


dcircle3= 0.0
mouthlength=0.0                  # Mouth Length
eyedistance1=0.0                # Eye distance
classmouth=0
areaeyes1=0.0              # Area1
areaeyes2=0.0              # Area2
emdi=0.0
emdd=0.0

while 1
if _STATUS == 0: break
DBgetvalue "face", "CLASS",classmouth
DBgetvalue "face", "CLASSNAME",classmouthn
DBgetvalue "face", "MLength",mouthlength
DBgetvalue "face", "Eye1Area",areaeyes1
DBgetvalue "face", "Eye2Area",areaeyes2
DBgetvalue "face", "EyeDistance", eyedistance1
DBgetvalue "face", "EMD1", emdi
DBgetvalue "face", "EMD2",emdd
DBgetvalue "face","EMA1",ema5
DBgetvalue "face","EMA2",ema6
DBgetvalue "face","Circle",dcircle3


mouthlength = ((mouthlength - MLA)*100  / MLB)
areaeyes1 = (((areaeyes1 + areaeyes2 - EAA)/EAB)*10)
eyedistance1= (100*(eyedistance1-EDA)/EDB)
emdi =  (((emdi + emdd-EMDA)/EMDB)*100)
ema5 =  (((ema5 + ema6 - EMAA)/EMAB)*10)
dcircle3 = ((dcircle3 - CA)*100  / CB)

# Integer values
integCIR= 0
integAE=1
integML=0
integED=0
integEMD=0
integEMA=0
```

```
while mouthlength > 1
integML = integML + 1
mouthlength = mouthlength - 1
endwhile

while dcircle3 > 1
integCIR = integCIR + 1
dcircle3 = dcircle3 - 1
endwhile

while areaeyes1 > 1
integAE = integAE + 1
areaeyes1 = areaeyes1 - 1
endwhile


while eyedistance1 > 1
integED = integED + 1
eyedistance1 = eyedistance1 - 1
endwhile


while emdi > 1
integEMD = integEMD + 1
emdi = emdi - 1
endwhile


while ema5 > 1
integEMA = integEMA + 1
ema5 = ema5 - 1
endwhile

DBaddline "facenorm"
DBsetvalue "facenorm", "CLASS",classmouth
DBsetvalue "facenorm", "CLASSNAME",classmouthn
DBsetvalue "facenorm", "MLength",integML
DBsetvalue "facenorm", "EyeArea",integAE
DBsetvalue "facenorm", "EyeDistance", integED
DBsetvalue "facenorm", "EMD", integEMD
DBsetvalue "facenorm","EMA", integEMA
DBsetvalue "facenorm","CIRCLE", integCIR

DBnextline "face"
endwhile

! datalist "facenorm",0,1




#  Creation of virtual images
Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "david"
```

```
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION",1
# MSsetcond


MSsetprop "REGIONFEAT","CLASS,CLASSNAME,FERETY,FERETX,XMIN,YMIN"
DBfirstline "facenorm"

mouthlength=0                  # Mouth Length
eyedistance2=0               # Eye distance
classmouth=0
dcircle4 =0
areaeyes1=0              # Area1
emdi2=0
ema7=0
color=0
i=0
while 1
if _STATUS == 0: break
DBgetvalue "facenorm", "CLASS",classmouth
DBgetvalue "facenorm", "CLASSNAME",classmouthn
DBgetvalue "facenorm", "MLength",mouthlength
DBgetvalue "facenorm", "EyeArea",areaeyes1
DBgetvalue "facenorm", "EyeDistance", eyedistance2
DBgetvalue "facenorm", "EMD", emdi2
DBgetvalue "facenorm","EMA",ema7
DBgetvalue "facenorm","CIRCLE",dcircle4

Gclear
imgload "white3.bmp",1
seldisplay "Display"
imgdisplay 1
eyedistance2 = eyedistance2 + 200 * areaeyes1
emdi2= emdi2 + 200 * ema7
Grectg eyedistance2, emdi2 , dcircle4 , mouthlength ,7,7
Gmerge
imgdisplay 1
dislev 1,2,3,252,1
imgRGB2grey 1, 1

if i==0
MSmeasmask 1,2,"virtualtest",0,1,classmouth
endif
if i==1
MSmeasmask 1,2,"virtualtest",1,1,classmouth
endif
i=1

DBnextline "facenorm"
endwhile

datalist "virtualtest"



#    teaching of the classifier
MSclassteach "VIRTUALTEST","BAYES","BAYES-ERGEB",1
! datalist BAYES-ERGEB,0,0
```

7.1.4. Classification with calculated features, test part.

```
# test of the classifier


Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "beli"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION","FERETY>50&&FERETX<20"
# MSsetcond
n=0

MSsetprop "REGIONFEAT","CLASS,CLASSNAME,FERETY,CGRAVX,CGRAVY"


# MSsetfeat "MOUTH"
imgsetpath "c:\ks400\conf\images\db"

# Images from camera
# tvsetup
# wait 10000
# tvframeinput 1,0,0,512,512
# imgsave 1,"testing.bmp"
# imgload "testing.bmp",1


# Images from database
imgload "jose2.bmp",1
imgRGB2grey 1, 2

imgload "mouth1.bmp",20

dislev 2,3,90,255,1
binnot 3,3
binand 3,20,4
MSmeasmask 4,2,"TESTMOUTH",0,1,0
datalist "testmouth",0,0


# MSsetfeat "EYES"
MSsetprop
"REGIONFEAT","CLASS,CLASSNAME,AREA,CGRAVX,CGRAVY,DCIRCLE,FERETX,F
ERETY"
MSsetprop "CONDITION","AREA>50 && FERETX>30 && FERETX<80 &&
CGRAVX<286 && CGRAVX>206 && DCIRCLE > 42"

imgload "eyes.bmp",20

imgRGB2grey 1, 2
dislev 2,3,95,255,1
```

```
binnot 3,3
binand 3,20,4
dmarr 4,4,9.50,2

! MSmeasmask 4,2,"TESTEYES",0,1,0
! datalist "testeyes",0,0

DBfirstline "testeyes"
DBfirstline "testmouth"

DBnew "face",11
DBsetcolumn "face",1,"CLASS","Int","<none>"
DBsetcolumn "face",2,"CLASSNAME","String","<none>"
DBsetcolumn "face",3,"MLength","Float","<none>"
DBsetcolumn "face",4,"Eye1Area","Float","<none>"
DBsetcolumn "face",5,"Eye2Area","Float","<none>"
DBsetcolumn "face",6,"EyeDistance","Float","<none>"
DBsetcolumn "face",7,"EMD1","Float","<none>"
DBsetcolumn "face",8,"EMD2","Float","<none>"
DBsetcolumn "face",9,"EMA1","Float","<none>"
DBsetcolumn "face",10,"EMA2","Float","<none>"
DBsetcolumn "face",11,"Circle","Float","<none>"



mouthlength=0                  # Mouth Length
eyedistance=0.0               # Eye distance
classmouth=0
classeyes1=0
classeyes2=0
dcircle1= 0.0
dcircle2= 0.0
areaeyes1=0                 # Area1
areaeyes2=0                 # Area2
xeyes1=0                  # Eye Position 1
yeyes1=0
xeyes2=0                  # Eye Position 2
yeyes2=0
xmouth=0                  # Mouth Position
ymouth=0
emdi=0.0
emdd=0.0

while 1
if _STATUS == 0: break
DBgetvalue "testmouth","CLASS",classmouth
DBgetvalue "testmouth","CLASSNAME",classmouthn
DBgetvalue "testmouth","FERETY",mouthlength
DBgetvalue "testmouth","CGRAVX",xmouth
DBgetvalue "testmouth","CGRAVY",ymouth
DBgetvalue "testeyes","CLASS",classeyes1
DBgetvalue "testeyes","CGRAVX",xeyes1
DBgetvalue "testeyes","CGRAVY",yeyes1
DBgetvalue "testeyes","AREA",areaeyes1
DBgetvalue "testeyes","DCIRCLE",dcircle1

DBnextline "testeyes"
DBgetvalue "testeyes","CLASS",classeyes2
DBgetvalue "testeyes","CGRAVX",xeyes2
DBgetvalue "testeyes","CGRAVY",yeyes2
DBgetvalue "testeyes","AREA",areaeyes2
DBgetvalue "testeyes","DCIRCLE",dcircle2
```

```
eyedistance= abs( yeyes1 - yeyes2)
emdi = sqrt((yeyes1 -ymouth)*(yeyes1-ymouth)+(xeyes1-
xmouth)*(xeyes1-xmouth))
emdd = sqrt( (yeyes2 - ymouth)*(yeyes2 - ymouth)+(xeyes2 -
xmouth)*(xeyes2 - xmouth))
ema5 = asin ( (eyedistance)/(2*emdi))
ema6 = asin ( (eyedistance)/(2*emdd))
ema5 = (ema5 * 180)/3.141593
ema6 = (ema6 * 180)/3.141592
dcircle1 = (dcircle1 + dcircle2)/2

DBaddline "face"
DBsetvalue "face", "CLASS",classmouth
DBsetvalue "face", "CLASSNAME",classmouthn
DBsetvalue "face", "MLength",mouthlength
DBsetvalue "face", "Eye1Area",areaeyes1
DBsetvalue "face", "Eye2Area",areaeyes2
DBsetvalue "face", "EyeDistance", eyedistance
DBsetvalue "face", "EMD1", emdi
DBsetvalue "face", "EMD2",emdd
DBsetvalue "face","EMA1",ema5
DBsetvalue "face","EMA2",ema6
DBsetvalue "face","Circle",dcircle1

DBnextline "testmouth"
DBnextline "testeyes"
endwhile

datalist "face",0,1

#  Norlmalisation of the DB "face"
DBfirstline "face"

DBnew "facenorm",8
DBsetcolumn "facenorm",1,"CLASS","Int","<none>"
DBsetcolumn "facenorm",2,"CLASSNAME","String","<none>"
DBsetcolumn "facenorm",3,"MLength","Int","<none>"
DBsetcolumn "facenorm",4,"EyeArea","Int","<none>"
DBsetcolumn "facenorm",5,"EyeDistance","Int","<none>"
DBsetcolumn "facenorm",6,"EMD","Int","<none>"
DBsetcolumn "facenorm",7,"EMA","Int","<none>"
DBsetcolumn "facenorm",8,"CIRCLE","Int","<none>"


dcircle3= 0.0
mouthlength=0.0                  # Mouth Length
eyedistance1=0.0               # Eye distance
classmouth=0
areaeyes1=0.0              # Area1
areaeyes2=0.0              # Area2
emdi=0.0
emdd=0.0

while 1
if _STATUS == 0: break
DBgetvalue "face", "CLASS",classmouth
DBgetvalue "face", "CLASSNAME",classmouthn
DBgetvalue "face", "MLength",mouthlength
DBgetvalue "face", "Eye1Area",areaeyes1
DBgetvalue "face", "Eye2Area",areaeyes2
DBgetvalue "face", "EyeDistance", eyedistance1
DBgetvalue "face", "EMD1", emdi
DBgetvalue "face", "EMD2",emdd
```

```
DBgetvalue "face","EMA1",ema5
DBgetvalue "face","EMA2",ema6
DBgetvalue "face","Circle",dcircle3

mouthlength = ((mouthlength - MLA)*100  / MLB)
areaeyes1 = (((areaeyes1 + areaeyes2 - EAA)/EAB)*10)
eyedistance1= (100*(eyedistance1-EDA)/EDB)
emdi =  (((emdi + emdd-EMDA)/EMDB)*100)
ema5 =  (((ema5 + ema6 - EMAA)/EMAB)*10)
dcircle3 = ((dcircle3 - CA)*100  / CB)
integCIR= 0
integAE=1
integML=0
integED=0
integEMD=0
integEMA=0

while mouthlength > 1
integML = integML + 1
mouthlength = mouthlength - 1
endwhile

while dcircle3 > 1
integCIR = integCIR + 1
dcircle3 = dcircle3 - 1
endwhile

while areaeyes1 > 1
integAE = integAE + 1
areaeyes1 = areaeyes1 - 1
endwhile


while eyedistance1 > 1
integED = integED + 1
eyedistance1 = eyedistance1 - 1
endwhile


while emdi > 1
integEMD = integEMD + 1
emdi = emdi - 1
endwhile


while ema5 > 1
integEMA = integEMA + 1
ema5 = ema5 - 1
endwhile

DBaddline "facenorm"
DBsetvalue "facenorm", "CLASS",classmouth
DBsetvalue "facenorm", "CLASSNAME",classmouthn
DBsetvalue "facenorm", "MLength",integML
DBsetvalue "facenorm", "EyeArea",integAE
DBsetvalue "facenorm", "EyeDistance", integED
DBsetvalue "facenorm", "EMD", integEMD
DBsetvalue "facenorm","EMA", integEMA
DBsetvalue "facenorm","CIRCLE", integCIR

DBnextline "face"
endwhile
```

```
datalist "facenorm",0,1



#  Creation of virtual images
Gclear 0

#  generation of learn-set
MSsetprop "CLASSNAME0","unknown"
MSsetprop "CLASSNAME1", "juan"
MSsetprop "CLASSNAME2", "jose juan"
MSsetprop "CLASSNAME3", "thomas"
MSsetprop "CLASSNAME4", "victor"
MSsetprop "CLASSNAME5", "pr.jaschul"
MSsetprop "CLASSNAME6", "david"
MSsetprop "CLASSNAME7","schneider"
MSsetprop "CLASSNAME8", "vollmann"
MSsetprop "CLASSNAME9", "wahl"
MSsetprop "CLASSNAME10","elena"
MSsetprop "CLASSIFIER","<none>"
MSsetprop "CONDITION",1
# MSsetcond


MSsetprop "REGIONFEAT","CLASS,CLASSNAME,FERETY,FERETX,XMIN,YMIN"
DBfirstline "facenorm"

mouthlength=0                  # Mouth Length
eyedistance2=0               # Eye distance
classmouth=0
dcircle4 =0
areaeyes1=0              # Area1
emdi2=0
ema7=0
color=0
i=0
while 1
if _STATUS == 0: break
DBgetvalue "facenorm", "CLASS",classmouth
DBgetvalue "facenorm", "CLASSNAME",classmouthn
DBgetvalue "facenorm", "MLength",mouthlength
DBgetvalue "facenorm", "EyeArea",areaeyes1
DBgetvalue "facenorm", "EyeDistance", eyedistance2
DBgetvalue "facenorm", "EMD", emdi2
DBgetvalue "facenorm","EMA",ema7
DBgetvalue "facenorm","CIRCLE",dcircle4

Gclear
imgload "white3.bmp",1
seldisplay "Display"
imgdisplay 1

eyedistance2 = eyedistance2 + 200 * areaeyes1
emdi2= emdi2 + 200 * ema7
Grectg eyedistance2, emdi2 , dcircle4 , mouthlength ,7,7
Gmerge
imgdisplay 1
dislev 1,2,3,252,1
imgRGB2grey 1, 1

MSsetprop "CLASSIFIER","BAYES"
```

```
MSmeasmask 1,2,"BAYES-~1",0,0,0
! datalist "BAYES-~1",0,0


DBfirstline "BAYES-~1"
DBgetvalue "BAYES-~1", "CLASS",classmouth
DBgetvalue "BAYES-~1", "CLASSNAME",classmouthn

if classmouth>0
n=12
endif

if n==12
MBok "The person is "+classmouthn+"."
endif

if n==0
MBok "No person identified"
```

7.2. Commands

Gclear

This function sets all graphics plane pixels to one colour. The graphics plane is filled with the selected Colour. The display device (Display or DisplayMDI window) can be changed with the function *seldisplay.*

Parameters:

Colour: Colour for drawing the graphics plane, 0...15

0 - Transparent, clear graphics plane
1 - Blue 9 - Light blue
2 - Green 10 - Light green
3 - Cyan 11 - Light cyan
4 - Red 12 - Light red
5 - Magenta 13 - Light magenta
6 - Yellow 14 - Light yellow
7 - Light grey 15 - White
8 - Dark grey

Macro Command:

Gclear Colour

MSsetprop

This function sets a single measurement property. Before measuring an image the properties have to be defined. It is a good idea to define always the complete set to guarantee that all properties have the needed values.

A predefined set with all properties can be read or written with the functions Load/Save Measurement File As (macro commands MSload and MSsave). Single properties or values can be read with the function Get Property (macro command MSgetprop).

There are also functions to set a group of properties interactively (Geometric Calibration, Densitometric Calibration, Set Frame, Set Condition, and Set Features (macro commands MSsetgeom, MSsetdens, MSsetframe, MSsetcond, and MSsetfeat).

It is possible to read the value of the selected property from a predefined property set (file). To do so select the desired file from the list box From File. When a new value is defined it is always written to the <current> property set. All measurements are done with the properties defined under <current>.

When the function is used interactively all properties can be changed (select a property and enter a new value). When the function is used in a macro only one property can be changed at a time.

To extend a current property value use the following syntax: MSsetprop "REGIONFEAT",",,PERIM" (double comma at the beginning of the value string).
Parameters:

Property	SCALEX	Horizontal scaling factor
	SCALEY	Vertical scaling factor
	UNIT	Geometric unit
	DISTANCEX	Horizontal scaling distance
	DISTANCEY	Vertical scaling distance
	DENSTABLE	Densitometric transformation table
	DENSFACTOR	Densitometric transformation factor
	DENSOFFSET	Densitometric transformation offset
	DENSUNIT	Densitometric unit
	FRAMEMODE	Frame condition mode (0...6)
	FRAMESTARTX	Upper left corner of rectangular frame, X-coordinate (in pixel)
	FRAMESTARTY	Upper left corner of rectangular frame, Y-coordinate (in pixel)
	FRAMESIZEX	Width of frame (in pixel)
	FRAMESIZEY	Height of frame (in pixel)
	FRAMECENTERX	Center of circular frame, X-coordinate (in pixel)
	FRAMECENTERY	Center of circular frame, Y-coordinate (in pixel)
	FRAMERADIUS	Radius of circular frame (in pixel)
	MINAREA	Minimum area of regions (in pixel)
	CONNECT	Connectivity of the regions (4 or 8)
	CONDITION	Measurement condition
	REGIONFEAT	Region features
	FIELDFEAT	Field features
	POINTFEAT	Point features
	DRAWFEAT	Draw features
	NSPACE	Line spacing for chord measurement (1...)
	NFERETS	Accuracy of feret based parameters (2...128)
	POLYDIST	Accuracy of polygon approximation (0...)
	WINDOW	Derivative window size
	PROFILES	Number of parallel, averaged profiles
	THRESHOLD	Threshold in % (0 ... 100)
	SPACE	Space between parallel, averaged profiles
	MODE	Gradient Mode: 0 - all 1 - positive 2 - negative

	3 - valleys 4 - hills
CLASSIFIER	Name of the classifier
CLASSNAME0	Name of the class 0
CLASSNAME1	Name of the class 1
CLASSNAME2	Name of the class 2
CLASSNAME3	Name of the class 3
CLASSNAME4	Name of the class 4
CLASSNAME5	Name of the class 5
CLASSNAME6	Name of the class 6
CLASSNAME7	Name of the class 7
CLASSNAME8	Name of the class 8
CLASSNAME9	Name of the class 9
CLASSNAME10	Name of the class 10
CLASSNAME11	Name of the class 11
CLASSNAME12	Name of the class 12
CLASSNAME13	Name of the class 13
CLASSNAME14	Name of the class 14
CLASSNAME15	Name of the class 15
NFD	Number of Fourier descriptors (1...20)
NTEXLEVEL	Grey resolution for texture measurement (2...64)
TEXDISPL	Displacement for texture measurement (1...64)
DEAGSTEP	Circular deagglomeration step

Value Value for selected property

Macro Command:

MSsetprop Property, Value

MSsetcond

This function defines the measurement conditions. Only regions, which meet the condition, are measured. All measurement functions test the condition. The conditions can be as simple as in this example:

AREA1 > 1000 (only regions greater or equal then 1000 units are measured).

This condition has the same effect as setting the property MINAREA to the value 1000 (MSsetprop "MINAREA",1000) or as complex as:

FCIRCLE>0.7&&SQRT(CGRAVXCGRAVX+CGRAVYCGRAVY)<400.

These conditions evaluate round regions whose center of gravity is less than 400 units from the coordinate origin.

The formula can be entered using the keyboard or constructed by clicking on the appropriate elements. The syntax is explained at the beginning of this section. To read a predefined condition select a file from the list box From File. When a new condition is defined, it is always written to the <current> property set.

The definition can also be done using the function Set Property (macro command MSsetprop). The property name is Condition. To store the settings use the function Save Measurement File As (macro command MSsave).

Use the Check button to test whether the formula syntax is correct. Moving the cursor onto the column containing the features will show a short description below the columns for some seconds.

The very left column of the dialog box shows the names of the feature groups. The features of the selected group are displayed in the second column. The third column shows mathematical functions, which can be included. The next column presents mathematical and Boolean operations. The last column shows numerical digits and constants. Also user defined global variables may be included.

The function works in the interactive mode only (the dialog box is always opened).

Parameters:

none

Macro Command:

MSsetcond

Set image path

This function sets the path for the image directory.

The directory entered must already exist. A root directory cannot be assigned ("C:\"). The functions Load (macro command *imgload*), Save As (macro command *imgsave*), and *imgenum* use this directory. The function *imggetpath* returns the setting of the current directory.

Parameters:

ImagePath Image drive and directory

Macro Command:

imgsetpath ImagePath

Load

This function loads an image file from a disk or network drive.

All image files in the current directory with the selected image format are displayed in the File Name list box. The image directory can be set with the function imgsetpath. The default path for images is C:\KS400\CONF\IMAGES. In this path the delivered images are located. The directories on the current drive are shown in the Directories list box. Another drive may be selected from the Drives list box.

The image format is selected in the list box List Files of Type. Some of them store also the look-up table (LUT). The type work will list the content for the Gallery. The following types are supported:

Image Type	LUT	Extension
Work (Gallery)	yes	WOR
CZV (previous Kontron Elektronik)	yes	IMG
BMP, Bitmap	yes	BMP
TIFF, Tagged Image File Format	yes	TIF
JPEG, Joint Photography Expert Group (works with true colour images only)	---	JPG
MAC, Mac Point	no	MAC
PCX, Zsoft Paintbrush	no	PCX

If Preview is activated, the content of the selected image is displayed on the left side. The right image shows the content of the image in which the selected image is to be loaded. The name of the selected image is displayed above it. Information about the image is given in the lower right section of the dialog box.

Parameters:

Input	File name of the input image
Output	Name of the output image

Macro Command:

imgload *Input, Output*

RGB to Grey

This function converts a true colour image to a grey value image.
The intensities of the three colour channels of the image Input are combined and written to the image Output.

Parameters:

Input	Input image (colour image)
Output	Output image (grey image)

Macro Command:

imgRGB2grey *Input, Output*

Threshold interactive

This function performs grey value segmentation.

The Interactive radio button of the Function list in the Threshold dialog box must be selected.

Segmentation is especially used to generate binary regions. They are needed for measurement.

Two threshold values determine which grey value range of the image Input is retained or deleted in the image Output.

The threshold values Low and High are determined either by moving the borders in the grey value histogram or by using the scroll bars below. In addition, the Low, Centre and High values can be set by making entries in the appropriate fields.

If the low (L) and high (H) values are to be moved together, the horizontal line in the histogram can be moved, or the center (C) scroll bar can be adjusted.

The parameter Colour radio buttons Green and Blue/Red determine whether the pixels inside (Green) or outside (Blue/Red) of the grey value range [Low, High] are displayed in the appropriate colour:

- If Green is selected, the pixels within the chosen range are overlaid with green, while the rest of the image retains its original grey values. The pixels with the grey values Low and Low+1 are defined in blue. The pixels with the grey values High and High1 are defined in red.
- If Blue/Red is selected, the pixels with grey values within the range Low, High remain unchanged. Pixels with grey values lower than Low are overlaid with blue, those with grey values higher than High with red.

- If the Invert option is selected, the area outside the selected range is chosen.

If the Binary option is selected all grey values within the range Low to High are set to white (grey value 255) in the image Output, the rest are set to 0. If the option is not selected, the grey values in the selected range remain unchanged and those outside are set to black (grey value 0).

If the image Input is a true colour image, the lightness channel is displayed in the histogram (the image is internally converted from RGB to HLS). For the segmentation of true colour images use the optional function ThresholdRGB (macro command dislevrgb).

Parameters:

Input	Input image
Output	Output image
L	Lower grey value threshold
H	Upper grey value threshold
Binary	0 - Selected pixels retain their original grey value 1 - Selected pixels are set to the grey value 255, the remainder to grey value 0

Macro Command:

dislev *Input, Output, L, H, Binary*

Boolean Not

This function carries out the bitwise negation of an image.

The NOT radio button of the Function list in the Boolean dialog box must be selected.

This function provides the same result as the function Grey transform in the menu Enhance (macro command greytran) with the table "inverse".

Parameters:

Input1	Input image
Output	Output image

Macro Command:

binnot *Input1, Output*

Automatic Measurement

This function carries out an automatic or a selective measurement.
All regions are tested against the defined conditions before they are measured. Conditions can be set with the functions Set Property, Set Frame, and Set Condition (macro commands MSsetprop, MSsetframe, and MSsetcond).
The regions are defined with a MaskImage (regions must be separated by black pixels with grey value 0). The DensImage is needed for the measurement of densitometric features. The results are written to a DataBase file. It is possible to Append data if the selected features are the same as those in the existing DataBase. The Mode defines if the region features (property REGIONFEAT) or the field features (property FIELDFEAT) are to be measured.

The difference of Region Select and Region Reject takes place in the interactive Mode only. Region Select means, no regions are selected when the

dialog is started (the user has to select regions). To select all regions at the beginning choose the Mode Region Reject. Executing the function in the non-interactive mode will measure in both terms all regions.

Field will calculate the field specific features of the selected regions. Selection or deselection of regions is not possible in this Mode.

Class defines to which class the selected regions should belong. The class selection is only useful while defining a new classifier (refer to Teach Classifier, macro command MSclassteach). To define the class members the feature CLASS must be included in the property REGIONFEAT.

The table shows all measurement values if the cursor is placed on a region. The left image (MaskImage) shows the valid regions, marked with a contour. A red coloured contour means, that the region is not selected. A selected region is marked by a green contour or the Class colour if the feature CLASS is a member of the feature list. The right image (DensImage) shows the selected regions in the corresponding class colour. To display the images in full size double click at them or click with the right mouse button on them and select Big Display from the context menu. Clicking on a region will change its status (selected, not selected).

Parameters:

MaskImage	Mask image defining the regions
DensImage	Image for densitometric measurement
Database	Name of database
Append	0 - Values are written to a new database 1 - Values are appended to the database
Mode	0 - Region Select: No regions are selected in the interactive mode. Features defined in the property REGIONFEAT are measured. 1 - Region Reject: All regions are selected originally in the interactive mode. 2 - Field: All selected features defined in the property FIELDFEAT are measured
Class	Membership of selected regions

Macro Command:

MSmeasmask	*MaskImage, DensImage, DataBase, Append, Mode, Class*

List

This function displays the data from databases as a data listing.

A file is selected from the list box DataBase. If the check box Print is activated, the values are also printed. When the check box Clipboard is activated, the values are copied to the clipboard. By pressing OK or Apply a window like the following is displayed:

The file name (in this case database), database and size are shown in the title bar of the window. The file has three measurement parameters and 322 records. The first nine records are shown. Using the scroll bars you can move between the data records (vertical scroll bar) and the parameters (horizontal scroll bar). Using the short horizontal scroll bar you can move up and down in the file.

Parameters:

DataBase	Name of the database to be displayed
Print	0 - Do not print window
	1 - Print window
Clipboard	0 - Do not copy to the clipboard
	1 - Copy to the clipboard

Macro Command:

datalist	*DataBase, Print, Clipboard*

Contours – Marr

This function detects edges or regions in an image. The Marr radio button of the Function list in the Contours dialog box must be selected.

In contrast to Valleys and Canny, a Laplace filter, following smoothing with a Gauss filter, is calculated and the edges (Mode - 1 Edges) or the regions (Mode - 2 Regions) in the image are detected.

The amount of smoothing in the image Input is determined by the parameter Sigma (0.01 means no smoothing).

Parameters:

Input	Input image
Output	Output image
Sigma	Smoothing factor
Mode	1 - Edges: Edge detection
	2 - Regions: Region detection

Macro Command:

dmarr	*Input, Output, Sigma, Mode*

Draw Rectangle

This function draws a rectangle in the graphics plane.

The start position and size of the rectangle area is defined by StartX, StartY, SizeX, and SizeY. The Colour and the filling pattern (Mode) can also be defined. If the rectangle is larger than the currently being displayed image, only the visible part is drawn.

While the dialog box is open, the rectangle can be defined interactively in the Display window.

The graphics plane can be selected with the function seldisplay.

Parameters:

StartX Starting point of the rectangle, X-coordinate
StartY Starting point of the rectangle, Y-coordinate
SizeX Width of the rectangle
SizeY Height of the rectangle

Colour Colour for drawing the graphics plane, 0...15

0 - Transparent, clear graphics plane
1 - Blue
2 - Green
3 - Cyan
4 - Red
5 - Magenta
6 - Yellow
7 - Light grey
8 - Dark grey
9 - Light blue
10 - Light green
11 - Light cyan
12 - Light red
13 - Light magenta
14 - Light yellow
15 - White

Mode

0 Empty: No filling
1 Dots: Dot filling
2 / Lines: 135 degree diagonal lines
3 \ Lines: 45 degree diagonal lines
4 | Lines: Vertical lines
5 - Lines: Horizontal lines
7 Filled: Completely filled

Macro Command:

Grectg *StartX, StartY, SizeX, SizeY, Colour, Mode*

Teach classifier

This function generates a classifier.

This function is needed for automatic class assignment of the measurement functions Automatic Measurement and Interactive Measurement (macro command MSmeasmask and MSmeasint).

The following steps will show the necessary procedure to define a classifier:

- Load a typical image to define a classifier.
- Segment the regions in the image.
- Define the measurement features, which will distinguish the regions for the different classes. The feature CLASS must be included.

Including the feature CLASSNAME will show the defined name instead of numbers.

- Optionally rename the classes. Use the function Set Property (macro command MSsetprop) and assign new names to the properties CLASSNAMEx.

- Deselect a probably defined classifier. Use the function Set Property (macro command MSsetprop) and set the property CLASSIFIER to "<none>".
- Perform a selective or interactive measurement with the function Automatic Measurement or Interactive Measurement (macro command MSmeasmask or MSmeasint). Select a number of member regions for each class. The minimum number depends mainly on the number of features and feature classes.

- Teach the classifier.
- Set the classifier for all measurements. Use the function Set Property (macro command MSsetprop) and assign new classifier name to the property CLASSIFIER.
- Start with classified measurement.

The Teach Classifier dialog offers the possibility to check which features are really needed for the classification. To do so open the List window from the Evaluate menu (macro command datalist) before entering the Classifier dialog. Switch on the entry Last Modified Database in the Options menu of this window.

The parameter DataBase in the Classifier dialog must name the one, which was used during the selective or interactive measurement. The file defined with the parameter Classifier will hold the data, which are necessary for classification. The Exchange database holds information about the quality of classification. Press the Apply button to check the settings.

Each line holds the information of the automatic classification of the different classes and an additional one for all classes (Total).

The column Class shows the selected classes (or class names) and a Total at the last line. Correct shows the quality in percent, all numbers should be 100 for a faultless classification. The following columns show the number of regions per class which were automatically classified. The rows show to which class the regions were originally defined during the selective measurement.

The list box Classes in the dialog specifies, which of them are valid for the classification. The list box on the right side shows the features, which were taken. The Classifier will take the selected features only into account.

The button Save will write the Classifier to a file with the extension CLS to the directory C:\KS400\CONF\MS. When the file already exists an overwrite prompt will appear. The button Delete will delete the selected Classifier file from the disk. When the option Split Data is chosen one half of the database

entries are taken for the classifier calculation and the other half for testing the classifier.

Parameters:

DataBase	Database name with teach data
Classifier	Classifier name
Exchange	Database name of exchange matrix
Type	Type of classification algorithm

0 - Bayes
1 - Mahalanobis
2 - Minimum Distance

Macro Command:

MSclassteach	DataBase, Classifier, Exchange, Type

After the teaching of the classification the measurement features may be redefined without any changes for the classification.